

# Collaborative Surveillance of Large Geographical Area by Fleet of Drones

DESIGN DOCUMENT

Team 50

Client / Advisers: Professor Goce Trajcevski, Prabin Giri

Team Members: Marcus Jakubowsky, Rowan Collins,  
Joseph Edeker, Jacob Houts, Jaden Forde, Thomas Glass

Team Email: [sdmay23-50@iastate.edu](mailto:sdmay23-50@iastate.edu)

Team Website: <https://sdmay23-50.sd.ece.iastate.edu/#>

Revised: 12/02/2022

# Executive Summary

## Development Standards & Practices Used

In order to successfully complete this project, there are multiple software development practices and engineering principles that will be used which can be seen below:

- Scrum Methodology
- IEEE Standard 1063: Standard for Software User
- IEEE Standard 1012: Standard for Software Verification and Validation
- IEEE 610.12: Standard Glossary of Software Engineering Terminology
- Software Engineering Code of Ethics and Professional Practice (Principle 2 & 3)

## Summary of Requirements

Functional Requirements:

- The software shall allow the user to login to a personal account profile.
- The software shall give the user the ability to choose which phenomena (e.g. fire, explosion, ...) to apply to the drone flight simulation.
- The software shall give the user the ability to choose which drone algorithm(s) to apply to the drone flight simulation.
- The software shall accept input combinations through selected files.
- The software shall provide a visualization of the drone flight.
  - The software shall have a 2d grid layout of the geographical area.
- The software shall calculate drone statistics and values over time and record them to storage.
- The software shall provide a view of statistics around the drones (battery life, location, speed, etc).
- The software shall save previously run simulations for input combinations that will be accessible for other users.
- If the user selects an input combination that has been simulated before then the software shall return back the already run simulation data.
- If the user selects an input combination that has not been simulated previously then the software shall queue the simulation to be run on the backend.

- If a simulation is run from a queued input combination then the software shall notify the requesting user when the simulation is completed (push-based notification).
- The software shall accept user input (file) which is given in source code which is ready to run (compiled or interpreted, etc).
- The software shall store the simulation output in a format containing: starting location, starting time, destination location, arrival time, trajectory.
- The software shall have 3 UI components: list of algorithms to pick, list of event phenomena input, visualization screen of simulation output.
  - Possibly a 4th component for notifications.
- 

#### Non-Functional Requirements:

- The software should be easy to use and understandable since not all of our users have a technical background
- UI elements of the software shall be intuitive and clearly labeled or documented.
- The software shall handle errors in the input gracefully.
- The software shall combine concurrent access for already executed input combinations and will run push notifications for users with new input simulations.
- The software shall be compatible with Windows, MacOS, and Linux.
- The software shall be developed in a manner that is supportable & maintainable after our team leaves.

#### Constrains:

- The software development process shall not cost more than \$300.
- If the runtime of the simulation exceeds a time period of 30 min, then the software shall terminate the simulation and notify the user of the termination.

### Applicable Courses from Iowa State University Curriculum

- Com S 227: Object Oriented Programming
- Com S 228: Introduction to Data Structures
- Com S 311: Introduction to the Design and Analysis of Algorithms
- S E 309: Software Development Practices
- Com S 252: Linux Operating System Essentials
- Com S 363: Introduction to Database Management Systems

- S E 409: Software Requirements Engineering
- S E 339: Software Architecture and Design
- Engl 314: Technical Communication

### New Skills/Knowledge acquired that was not taught in courses

- Learn Specific Approaches for Managing Collaborative Fleet of Drones
- New Applications for Drone Fleets
- Simulations
- Visualization of Motion
- Specific Integration of Frontend and Backend

# Table of Contents

<b>1 Team</b>	<b>5</b>
<b>2 Introduction</b>	<b>5</b>
2.1 PROBLEM STATEMENT	5
2.2 INTENDED USERS AND USES	6
<b>3 Project Plan</b>	<b>9</b>
3.1 Project Management/Tracking Procedures	9
3.2 Task Decomposition	9
3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria	10
3.4 Project Timeline/Schedule	12
3.5 Risks And Risk Management/Mitigation	13
3.6 Personnel Effort Requirements	14
3.7 Other Resource Requirements	16
<b>4 Design</b>	<b>17</b>
4.1 Design Context	17
4.1.1 Broader Context	17
4.1.2 Prior Work/Solutions	18
4.1.3 Technical Complexity	19
4.2 Design Exploration	20
4.2.1 Design Decisions	20
4.2.2 Ideation	21
4.2.3 Decision-Making and Trade-Off	22
4.3 Proposed Design	23
4.3.1 Overview	23
4.3.2 Detailed Design and Visual(s)	24
4.3.3 Functionality	25
4.3.4 Areas of Concern and Development	31
4.4 Technology Considerations	32

4.5 Design Analysis	33
<b>5 Testing</b>	<b>33</b>
5.1 Unit Testing	34
5.2 Interface Testing	35
5.3 Integration Testing	36
5.4 System Testing	36
5.5 Regression Testing	37
5.6 Acceptance Testing	37
5.7 Security Testing (if applicable)	39
5.8 Results	40
<b>6 Implementation</b>	<b>40</b>
<b>7 Professional Responsibility</b>	<b>40</b>
7.1 Areas of Responsibility	40
7.2 Project Specific Professional Responsibility Areas	40
7.3 Most Applicable Professional Responsibility Area	41
<b>8 Closing Material</b>	<b>41</b>
8.1 Discussion	41
8.2 Conclusion	41
8.3 References	41
8.4 Appendices	41
8.4.1 Team Contract	41

# 1 Team

## 1.1 TEAM MEMBERS

Marcus Jakubowsky	<i>Software Engineering, Mathematics &amp; Cyber Security Minor</i>
Rowan Collins	<i>Software Engineering</i>
Joseph Edeker	<i>Software Engineering, Mathematics Minor</i>
Jacob Houts	<i>Software Engineering</i>
Jaden Forde	<i>Software Engineering, Cyber Security Minor</i>
Thomas Glass	<i>Cyber Security Engineering</i>

## 1.2 REQUIRED SKILL SETS FOR YOUR PROJECT

- Version control (Gitlab, Git)
- Database knowledge (login, store user data)
- Web development (hosted website)
- User interface design/development (web client fronted)
- Web server hosting/deployment (hosted website)
- Data structures (queue system)
- Operating Systems (hosting & caching optimization)
- APIs (Communication between website and backend)
- Knowledge of coding standards (Maintainability)

## 1.3 SKILL SETS COVERED BY THE TEAM

<b>Team Member</b>	<b>Skills, Expertise, and Unique Perspectives</b>
Marcus	Languages: Java, C, Python, Typescript Frameworks: Angular, Springboot, SQL Technologies: AWS, UNIX, Git, CI/CD, Docker, Database
Thomas	Languages: Java, C#, Python, C Technologies: Splunk, Wireshark, pfSense Experience: Firewalls, Network Security, Threat Analysis
Jaden	Languages: Java, C, Python, Typescript, SQL Frameworks: React, Angular, Spring API Technologies: Computer vision, web server deployment, Git
Joe	Languages: Java, HTML Frameworks: React Other: Graphic Design, Video Editing
Rowan	Languages: Java, C, Python, SQL, noSQL, Javascript, HTML Frameworks: Springboot, Flask Technologies: Git, UNIX, AWS

Jacob	Languages: C, C++, Java, Javascript, HTML Frameworks: React, OpenGL, Springboot. Technologies: Git, UNIX
-------	--

#### 1.4 PROJECT MANAGEMENT STYLE ADOPTED BY THE TEAM

During the design phase the team used the Waterfall project management style to complete the design doc and prepare for the implementation phase. Once the implementation phase begins, the team will transition to the Agile Development methodology for completing features layed out in this document.

#### 1.5 INITIAL PROJECT MANAGEMENT ROLES

Marcus Jakubowsky	<i>Team Lead</i>
Rowan Collins	<i>Presentations</i>
Joseph Edeker	<i>UI/UX Design</i>
Jacob Houts	<i>Testing</i>
Jaden Forde	<i>Client Interaction</i>
Thomas Glass	<i>Standards &amp; Security Validation</i>

\* All team members have the role of “*Developer*” alongside these listed roles.

## 2 Introduction

### 2.1 PROBLEM STATEMENT

Problem we are trying to solve: Simulating Drone flight patterns in a controlled environment

**WHO** has the problem? Research & Development people working on drone fleets.

**WHAT** is the problem? Testing drone fleets in real life is very expensive and time consuming.

**WHERE** is the problem occurring? All around the world.

**WHEN** is the problem occurring? When researchers need visual examples on how their algorithms operate.

**WHY** is it important? By simulating drone flight paths and how they react in different environments following different algorithms, production and testing costs are cut tremendously

**HOW** will it be solved? Our project will utilize software to present data and a simulation to mimic actual flight data based on given algorithms.



Testing drones in real life is hard. Geographic surveyors of different professions all over the world are looking for visual and statistical data on how drone fleets react to different phenomena. However, running real world tests for these drones can be risky and expensive. This is why many researchers in companies and academia resort to simulation. They can be used to test the battery/energy expenditures under various conditions, as well as explore the quality of coverage (in terms of average arrival to a location of an “interesting event”), etc... However, most of the simulators are “custom-made” and not easy to generalize for comparative studies.

Our project aims to overcome this kind of restriction and provide an environment where different routing/dispatching algorithms for a single drone or a fleet of drones can be compared against each other in terms of desired metrics (e.g., average or worst-case arrival time or coverage). In addition, our project will provide a visualization tool to show the motion plan of the drones executing a mission over an area of interest. This front-end functionality will be supported by a back-end host that will generate the actual data based on a selected algorithm and an input phenomenon-dataset.

## 2.2 INTENDED USERS AND USES

### Researchers:

- A. Persona
  - a. Hobbies/Interests: Researching, learning
  - b. Motivations: Pushing technology’s limits, achieving recognition, making new discoveries
  - c. Personality/Emotions: Smart, hard-working
  - d. Values: Getting results, accuracy, efficiency
- B. User Needs
  - a. Researchers need a way to simulate drone fleet algorithms because testing drones in real life is expensive and time consuming.

### Farmers:

- A. Persona
  - a. Demographics: Varies (e.g., free-range cattle; crop monitoring; soil property survey)
  - b. Motivations: Higher crop yields, lower costs, using technology for saving money
  - c. Personality/Emotions: Hard working
  - d. Values: cost-effectiveness, environmental conservation, family
- B. User Needs
  - a. Farmers need a way to automatically evaluate geographical areas because it helps them determine the state of crops and potential plots of land much faster and more accurately than a manual approach.

### Companies:

- A. Persona
  - a. Demographics: Technology companies interested in drone development
  - b. Motivations: Profit

- c. Values: Optimize design in terms of profit as well as combination of structure with battery types.
- B. User Needs
  - a. Companies need a way to achieve this degree of flexibility because they want to be able to cover larger market-share (i.e., larger pool of potential customers) .

### First Responders:

- A. Persona
  - a. Demographics: Varies (e.g., firefighters; police; emergency workers)
  - b. Motivations: Save lives
  - c. Personality/Emotions: Caring, motivated
  - d. Values: Helping others
- B. User Needs
  - a. First responders need a way to map the terrain efficiently, because it helps them act faster in an emergency situation.

## 2.3 REQUIREMENTS & CONSTRAINTS

### Functional Requirements:

- The software shall allow the user to login to a personal account profile.
- The software shall give the user the ability to choose which phenomena (e.g. fire, explosion, ...) to apply to the drone flight simulation.
- The software shall give the user the ability to choose which drone algorithm(s) to apply to the drone flight simulation.
- The software shall accept input combinations through selected files.
- The software shall provide a visualization of the drone flight.
  - The software shall have a 2d grid layout of the geographical area.
- The software shall calculate drone statistics and values over time and record them to storage.
- The software shall provide a view of statistics around the drones (battery life, location, speed, etc).
- The software shall save previously run simulations for input combinations that will be accessible for other users.
- If the user selects an input combination that has been simulated before then the software shall return back the already run simulation data.
- If the user selects an input combination that has not been simulated previously then the software shall queue the simulation to be run on the backend.
- If a simulation is run from a queued input combination then the software shall notify the requesting user when the simulation is completed (push-based notification).
- The software shall accept user input (file) which is given in source code which is ready to run (compiled or interpreted, etc).
- The software shall store the simulation output in a format containing: starting location, starting time, destination location, arrival time, trajectory.
- The software shall have 3 UI components: list of algorithms to pick, list of event phenomena input, visualization screen of simulation output.
  - Possibly a 4th component for notifications.

#### Non-Functional Requirements:

- The software should be easy to use and understandable since not all of our users have a technical background
- UI elements of the software shall be intuitive and clearly labeled or documented.
- The software shall handle errors in the input gracefully.
- The software shall combine concurrent access for already executed input combinations and will run push notifications for users with new input simulations.
- The software shall be compatible with Windows, MacOS, and Linux.
- The software shall be developed in a manner that is supportable & maintainable after our team leaves.

#### Constrains:

- The software development process shall not cost more than \$300.
- If the runtime of the simulation exceeds a time period of 30 min, then the software shall terminate the simulation and notify the user of the termination.

## 2.4 ENGINEERING STANDARDS

What Engineering standards are likely to apply to your project? Some standards might be built into your requirements (Use 802.11 ac wifi standard) and many others might fall out of design. For each standard listed, also provide a brief justification.

*Table 1: Engineering Standards*

<b>Engineering Standards</b>	<b>Justification</b>
Scrum Methodology	We need to have structure on how we develop the project. We will work in 2 week sprints.
IEEE Standard 1063: Standard for Software User	We will make sure that our project is documented correctly for future users and developers.
IEEE Standard 1012: Standard for Software Verification and Validation	We must make sure that our project meets the clients requirement completely and thoroughly.
IEEE 610.12: Standard Glossary of Software Engineering Terminology	Make sure we use language similar to our peers.
Software Engineering Code of Ethics and Professional Practice (Principle 3)	We will make sure that our product meets the highest possible standards.
Software Engineering Code of Ethics and Professional Practice (Principle 2)	We will meet the expectations of our client as well as the potential users of our product.

## 3 Project Plan

### 3.1 PROJECT MANAGEMENT/TRACKING PROCEDURES

Which of agile, waterfall or waterfall+agile project management style are you adopting. Justify it with respect to the project goals.

- As there are two distinct portions of the project - design document and actual implementation, our team will adopt the dual-style of waterfall (for the sequential development of the parts of the design document) + agile (for the implementation). Utilizing the agile approach to project management during the implementation phase will enable us to work on multiple parts of the project simultaneously, with work that may be broken down into small tasks which may be completed in a 2 week sprint period. In addition, the framework's flexibility will allow us to make changes later on as necessary, with emphasis on a modular design for the different parts of the project.

As specific examples:

- While part of the team is implementing the frontend visualization of the drone simulation, another part of the team will be implementing the backend queue system for simulation with new input combinations.
- Complementary, when working on the UI, one sprint would be implementing the drop-down menu functionality, followed by the next one implementing the button-like selectors.

What will your group use to track progress throughout the course of this and the next semester. This could include Git, Github, Trello, Slack or any other tools helpful in project management.

- Git (for code source control)
- Gitlab (for code repository)
- Gitlab Issues & Issue Boards (for task decomposition/user stories)
- Discord (communication)

### 3.2 TASK DECOMPOSITION

In order to solve the problem at hand, it helps to decompose it into multiple tasks and subtasks and to understand interdependence among tasks. This step might be useful even if you adopt agile methodology. If you are agile, you can also provide a linear progression of completed requirements aligned with your sprints for the entire project.

1. User Authentication System
  - a. Frontend create new user screen
  - b. Backend create new user endpoint
  - c. Backend login endpoint
  - d. Frontend login screen
  - e. Reset password functionality
2. Users can choose an event phenomena file.

- a. Frontend screen to choose event phenomena file.
  - b. Backend saves event input. (standardized format?)
- 3. A system which can save previous simulations
  - a. Database which saves run data for specified input combinations.
  - b. Utilizes previous input combinations to output old saved data rather than running a new simulation
- 4. Running simulations
  - a. Simulation is ran on the backend
    - i. Queue system for new input combinations.
      - 1. FCFS -> SRTF -> MLFQ
    - ii. Push Notification for when simulations are complete.
    - iii. 30 min time limit on simulation runtime.
    - iv. Finished simulations will return a file with information on the ran simulation
  - b. Simulation on the Frontend
    - i. Screen with input options for setting up the simulation.
      - 1. List of options for the algorithms
      - 2. List of options for event phenomena
    - ii. Screen with input options for viewing the simulation
      - 1. List of options for which simulation results to display

### 3.3 PROJECT PROPOSED MILESTONES, METRICS, AND EVALUATION CRITERIA

What are some key milestones in your proposed project? It may be helpful to develop these milestones for each task and subtask from 2.2. How do you measure progress on a given task? These metrics, preferably quantifiable, should be developed for each task. The milestones should be stated in terms of these metrics: Machine learning algorithm XYZ will classify with 80% accuracy; the pattern recognition logic on FPGA will recognize a pattern every 1 ms (at 1K patterns/sec throughput). ML accuracy target might go up to 90% from 80%.

In an agile development process, these milestones can be refined with successive iterations/sprints (perhaps a subset of your requirements applicable to those sprint).

Fall 2022

- User Needs & Problem Statement (+ literature survey)
- Functional & Non-functional requirements
- Project Planning
- Project Design
- Tool & Platform Considerations
- Testing
- Professionalism
- Final Document & Presentation

*Progress on Fall 2022 milestones will be measured by the feedback received on the submitted documents.*

Spring 2023

- Platform Setup
  - Databases
  - URLs
  - Code base template

*Progress will be measured by direct checks for each sprint in terms of functionality of the respective components/subcomponents.*

- Authentication
  - Create new user (front & back)
  - Login user (front & back)

*Progress will be measured by completing unit testing for each applicable outcome. This will include failed login/registration attempts, along with successful ones as well.*

There are multiple milestones under a common “umbrella” of algorithm execution (running simulation):

- Different source code language support
  - Python
  - C++
- System that takes event input, translates it to be passed given algorithm, executes algorithm with event input.
  - Different grid-sizes and phenomena distribution
  - Different (algorithms, input) pairs

*Progress will be measured by ensuring the correct parameters are sent by using mocking calls to the server and verifying results with unit testing.*

- Takes output of algorithm running, translates output into simulation data file.
- Simulation will be terminated if it exceeds 30 minutes

*Progress will be measured by ensuring functionality that algorithm runs and output is as expected using unit testing.*

- Simulation Setup/Request
  - Event phenomena selection
  - Algorithm selection

*Progress will be measured by observing the execution of a specific algorithm over different datasets.*

- Simulation data response
  - w/Repeated input combination
    - Requested input combination will return previously run simulation files.
  - w/New input combination
    - Input combinations will be put into queue
    - Data of completed simulations will be stored based on the input combination



### 3.5 RISKS AND RISK MANAGEMENT/MITIGATION

Table 1: Risk Management and Mitigation Strategies

TASK/COMPONENT	RISK	RISK PROBABILITY	MITIGATION
Design	All requirements were not properly defined	0.2	None
Development	Workers not completing assigned tasks	0.5	Discipline for missing deadlines will be increased.
Planning	Client size unknown	0.5	Conduct research to figure out how many people will use the product.
Maintenance	Users not knowing how to use the application	0.6	Develop tutorials to help the users.
Testing	Tests do not cover all possible combinations of environments and events	0.7	Researching into testing tools that can help.
Planning	Cannot find proper frameworks to create application	0.2	None
Design	Use cases may not properly describe how a user uses the application	0.6	Conduct proper research to understand how different users will interact with application



Development	Going over budget	0.4	None
Design	User Information Privacy	0.6	Design code to be safe from sql injection attacks etc.

### 3.6 PERSONNEL EFFORT REQUIREMENTS

Table 2: Personal Effort Requirements

TASK NAME	DETAIL	NOTES	HOURS REQUIRED
<b>PLATFORM SETUP</b>			
Frontend	Establish a baseline template for the frontend	Should be accessible from a web browser	8
Backend	Establish a baseline template for the backend	Should be reachable via HTTP requests	8
Database	Set up a data storage medium	Should be reachable via IP or other communication protocol	8
<b>USER AUTHENTICATION</b>			
User Creation	Create a sign up page where users can create a new user with their chose information	Information Needed for New User: Username, Password, Email	16
Reset Password	Functionality to allow users to change passwords		4
User Login	Create a login page where users can sign in using previously created users	Login with: Username or Email and Password	8
<b>SIMULATION SETUP/REQUEST</b>			
Event Phenomena Selection	Create a series of drop down menus and fields/sliders for users to select phenomena to be applied to the algorithm	Possible Phenomena: Wind, Fire, Explosion	8
Algorithm Selection	Create a dropdown of possible algorithms to use or the ability to add their own	Algorithms come from Professor Goce and his team	8

	algorithm		
<b>ALGORITHM EXECUTION</b>			
Algorithm Source Code Translation Layer	Ability to decode different algorithms into usable programs for the backend		32
Algorithm Execution	Ability to run selected algorithms on backend		16
Exporting Algorithm Data	1. Define file format for algorithm output 2. Encode algorithm output to file		8
Algorithm Notification System	1a. Define notification events 1b. Backend recognizes events 2. Backend communicates events to Frontend 3. Frontend displays notifications	Sample events: simulation request received, simulation executing, simulation execution complete, simulation runtime error	24
Simulation Queue System	Ability to queue multiple algorithms. Notifications to be sent upon completions of queued algorithms.	Possible Queue Systems: FCFS -> SRTF -> MLFQ	12
<b>SIMULATION DATA RESPONSE</b>			
Simulation File Transfer (between Front & Back End)	Ability to send stored data (back end) to the UI (front end)	Transfer data in an efficient format using JSON	4
Storage for Previously Completed Simulations	Upon completion of algorithms, store data into database	Store datasets into a database on our preferred cloud service provider	12
<b>SIMULATION VISUALIZATION</b>			
Display Grid/Coordinate System	Create a 2D grid with coordinates built in		8
Display Drones	Display drones on 2D grid based on position data from algorithm		8
Display Phenomena	Define indicators and display phenomena on the 2D grid	Possible Phenomena: Wind, Fire, Explosion	24

Indicators for Drone Diagnostic Data	Hover, Overlay, or Panel for Drone information	Examples: Battery Life, Position (grid coordinates), Speed, Elevation	16
Media Controls for Drone Simulations	Play/Pause/Fast Forward/Rewind		16
Simulation Menu Selection	Selection for which simulation to play and loading screen when waiting for simulation to start		12
Total			260

### 3.7 OTHER RESOURCE REQUIREMENTS

The only other resources required for this project to aid in completion, are the following:

- Algorithms provided by Prof Goce & Prabin.
- Platform to host the application.

## 4 Design

### 4.1 Design Context

We now proceed with the details of the design context and in the sequel, we discuss broader context, prior work/solutions and technical complexity.

#### 4.1.1 Broader Context

Our broad context is to provide a platform for drone users to simulate flight patterns from a variety of routing and scheduling algorithms. By providing a frontend web application to users, simulation becomes a much easier process for customers of our project, with no downloads being required. Drone enthusiasts will be able to visualize different flight patterns for drones cutting down costs on real world testing, which can be expensive. In addition, they can estimate certain values such as min/max/avg service time when a fleet of drones is tasked with surveillance of an area of interest.

Table 3: Broader Context

Area	Description	Examples
Public health, safety, and welfare	Customers- 1. By providing a virtual service that emulates how drones follow example flight patterns, we mitigate potential safety concerns when testing with physical drones.	Customers- Fire Fighters (In terms of getting reports of how the fire is spreading.) Delivery and Pickup Drones

	<p>2. Completely eliminating the need for actual drones and allowing virtual simulation negates the risk to physical property and people.</p>	
Global, cultural, and social	<p>Some of our groups of interest include emergency personnel, agriculture companies, and geographical surveyors. These groups typically value robustness, reliability, and accuracy above all. Our project will reflect these aims by providing a platform which allows for faster testing of drones resulting in a fleet which is more reliable, accurate, and adaptable than one that had not used our system.</p>	<p>Testing algorithms to guide police personnel to critical areas.</p> <p>Testing algorithms to survey land efficiently.</p>
Environmental	<p>1. By reducing the need for use of physical drones while testing flight algorithms, we effectively reduce the carbon footprint the drones create. Tests can be run tens to hundreds of times requiring tons of power to run these drones and allow them to fly in the air.</p> <p>2. While running the server for our web application will also take a significant amount of power, compared to flying drones, it saves energy in the long run. By saving test results with specific parameters, we reduce the time the server needs to be computing flight data and instead we can display previous runs containing the exact same parameters.</p>	<p>One can simulate a spreading of a disease in both urban/human settings as well as agricultural settings - and the project will help investigate surveillance approaches for efficient monitoring.</p>
Economic	<p>Customers-</p> <p>1. When providing a virtual service we completely eliminate the need for physical drones. This provides thousands of dollars in savings for the users of our application. Simulating different phenomena also allows users to better prepare their drones for different environments, making them more cost-effective and reliable.</p>	<p>Any profession in which a fleet of drones could assist in detecting, monitoring and responding to phenomena of interest.</p>

#### 4.1.2 Prior Work/Solutions

We looked at previous work from the last semester's group, which has a related project, and in addition we consulted other references.

More specifically, in [1] we went through an algorithm that optimizes energy consumption of multiple drones working on a collaborative task. We will use this algorithm to build our web application.

In [2], we learn about typical problems fleet drones need to solve, we will use this information to help us build some events to put up against the fleet drone algorithm. The types of problems explained in this paper are, including wireless communication support, monitoring targets of interest, serving a wireless sensor network, and collaborating with ground robots. In particular, an overview of the existing publications on the coverage problem, connectivity of flying robots, energy capacity limitation, target searching, path planning, flying robot navigation with collision avoidance, etc.

In [3], displays the benefits of using a Capstone Project for the professional development of students. We used this information as a basis for our learning goals from this assignment.

The previous student led group's design document can be found [here](https://sdmay22-33.sd.ece.iastate.edu/docs/33FinalDesignDoc.pdf) (https://sdmay22-33.sd.ece.iastate.edu/docs/33FinalDesignDoc.pdf)

Most of the simulators focus on examining the parameters of a single drone during service, or a fixed setting of collaboration among a fleet of drones. Our project aims to be more flexible/extensible in terms of combining not only multiple collaborative algorithms but also generation of (spatio-temporal) occurrences of phenomena of interest.

#### Pros of our Solution

- The main thing that our project will do differently from competitors is that it will be more accessible than other solutions.
- This application will be web based, so anybody with an internet connection will be able to access it.
- Previous solutions are custom built and do not offer similar accessibility.

#### Cons of our Solution

- The visualization will most likely not be as advanced as our competitors.
- To use our application you will have to adhere to our version of the input files.

### 4.1.3 Technical Complexity

From the technical side, our project has three components, backend, frontend and a database. We will also have an API, implemented through Spring Boot, which will allow the components to communicate with each other. The frontend will be a web client, and will communicate with our backend to store and retrieve information from the database as needed.

1. Backend: The backend will have to retrieve data from the database and be able to update the information within the database as needed. The backend will also be able to run the simulations and return a file with the simulation results to the frontend. The backend will also be used to implement user authentication.

2. Frontend: The frontend will have 3 main functionalities
  - a. User Authentication: Login screen for the user, processed through the backend. Allows user to save personal simulations for a later use.
  - b. Event Selection: Screen with input options for setting up the simulation.
    - i. List of options for the algorithms
    - ii. List of options for event phenomena
  - c. Event Visualization: Screen with input options for viewing the simulation
    - i. List of options for which simulation results to display
    - ii. Visualizing drone flight
3. Database: Database will have different tables according to the information we need stored
  - a. Users: Username and Password
  - b. Simulations: Run files of previous simulations to reduce overhead on duplicate simulations.

### Internal Complexity

1. Four interfaces between components
  - a. Frontend to Backend: Providing User Data, Providing Event Conditions
  - b. Backend to Frontend: Providing visualization file after simulation is run, sending notifications.
  - c. Backend to Database: sending visualization files to be stored.
  - d. Database to Backend: returning visualization files when requested.
2. Research of new technologies
  - a. Many people within the team will need to learn technologies that they have never used before.

### External Complexity

The goal of this project was to exceed the industry standards of drone simulation technology. Currently most drone simulation applications are custom made and this application is being developed to be more accessible to more people. This puts us on the forefront of new technologies within the field. This project will also have components of most industry projects such as, error handling, user authentication, notification system, optimization, Input/Output Processing, etc.

## 4.2 Design Exploration

We now proceed with the details of the design context and in the sequel, we discuss design decisions, ideation, decision-making and tradeoffs.

### 4.2.1 Design Decisions

List key design decisions (at least three) that you have made or will need to make in relation to your proposed solution. These can include, but are not limited to, materials, subsystems, physical components, sensors/chips/devices, physical layout, features, etc. Describe why these decisions are important to project success.

*Table 5: Design Decisions*

Frontend		
Problem	Description	Options
User Interface	Determine which library or framework to use (if at all) for UI components of the application.	React Angular Plain HTML/CSS/JS
Simulation Visualization	Determine if a 2D Plotting Library is more effective for simulation visualization, and if so which.	Plotly.js Chart.js Desmos API None
	Determine if a 2D Game Engine is more effective for simulation visualization, and if so which.	PixiJS MelonJS None
Architecture Pattern(s)	Determine which design philosophies are most appropriate for the frontend.	MVC
Backend		
Framework; Client-Server Communication	Determine which language/framework for the server to use to communicate with the frontend.	Spring Boot (Java) Node.js (JavaScript) Flask (Python) Django (Python)
Architecture Pattern(s)	Determine which design philosophies are most appropriate for the backend.	MVC
Database		
Database Manager	Determine which type of database to use and which database manager.	MySQL Oracle MongoDB
Simulation File		
File Structure	Determine how data should be stored in a file.	Binary JSON
File Transfer	Determine which protocols are necessary for file transfer.	HTTP Requests Socket.IO

#### 4.2.2 Ideation

Figure 2: Lotus Blossom Diagram

Plotly.js	Chart.js	Desmos API	Matplotlib	Seaborn	Plotly	PixiJS	MelonJS	
	2D Plotting Library		Bokeh	3D Plotting Library	PyGal		2D Game Engine	
			2D Plotting Library	3D Plotting Library	2D Game Engine	Unity	Unreal Engine	Godot
				Frontend Visualization of the Simulation	3D Game Engine		3D Game Engine	
					Manual			
						HTML/CSS/Javascript	beginPath() moveTo() lineTo()	
							Manual	

In order to decide how to implement the frontend visualization for the simulation, we debated many possible solutions for this problem. We began by identifying the problem at hand and started brainstorming possible solutions for the problem. Then we utilized previous parts of the design document to help develop a potential list of ideas. Using various resources such as the requirements or user needs we were able to develop a list of potential ways to solve the problem. Further research on the internet helped to expand our list of options and allowed us to settle on a final set of possible solutions.

After our research and brainstorming we landed on using a 2D plotting library, 3D plotting library, 2D game engine, 3D game engine, or to draw it manually. After having our overall approaches figured out we then did further research into each category for more specific approaches into our problem. This led us to a list of libraries or applications that can help accomplish our task and enabled us to further evaluate our choices. We then weighed these options against each other in the following section to help make our final decision.

#### 4.2.3 Decision-Making and Trade-Off

Table 6: Weighted Decision Matrix

Options	Ease of Use	Experience	Platform Maturity	Performance	Total
Weight	30%	20%	20%	30%	100%



2D Plotting Library	4	3	3	3	3.3
3D Plotting Library	2	3	3	3	2.7
2D Game Engine	3	1	3	5	3.2
3D Game Engine	2	1	4	4	2.8
Manual	1	4	4	4	3.1
Final Decision	2D Plotting Library				

Our team settled on using a 2D Plotting Library due to its ease of use, and performance. With such a large scale project, it is vital we deliver on time and by using a tool that performs well we can expedite the process requiring less time learning a new framework. Other options would take too much time to learn and become proficient with.

### 4.3 PROPOSED DESIGN

We now proceed with the details of the proposed design and in the sequel, after providing an overview we discuss functionalities, areas of concern and development, technology considerations, and design analyses.

#### 4.3.1 Overview

A user of our product will be able to log-on to our website and begin by setting up a user account. After logging in with those credentials, the user will have the option to view previously run simulations as well as set up new simulations to test. For new simulations, the user will be able to select the simulation setup, including number of drones, drone location, and time and location of event phenomena, as well as the drone fleet algorithm to evaluate with. Once inputs are selected, the user can run the simulation which may take up to 30 min to run in the background on the server. After a simulation is completed, it is added to the list of simulations that can be viewed by the user. (*see figure 3 for context diagram*)

#### 4.3.2 Detailed Design and Visual(s)

Our design for the web application will be implemented by using layered architecture. From the diagram below you can see the four layers of our application and some more information about what functionalities they offer.

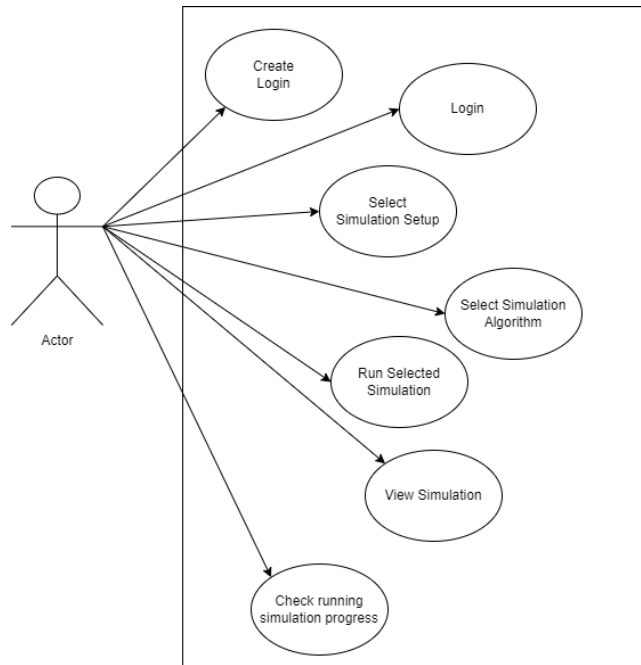


Figure 3: Context Diagram

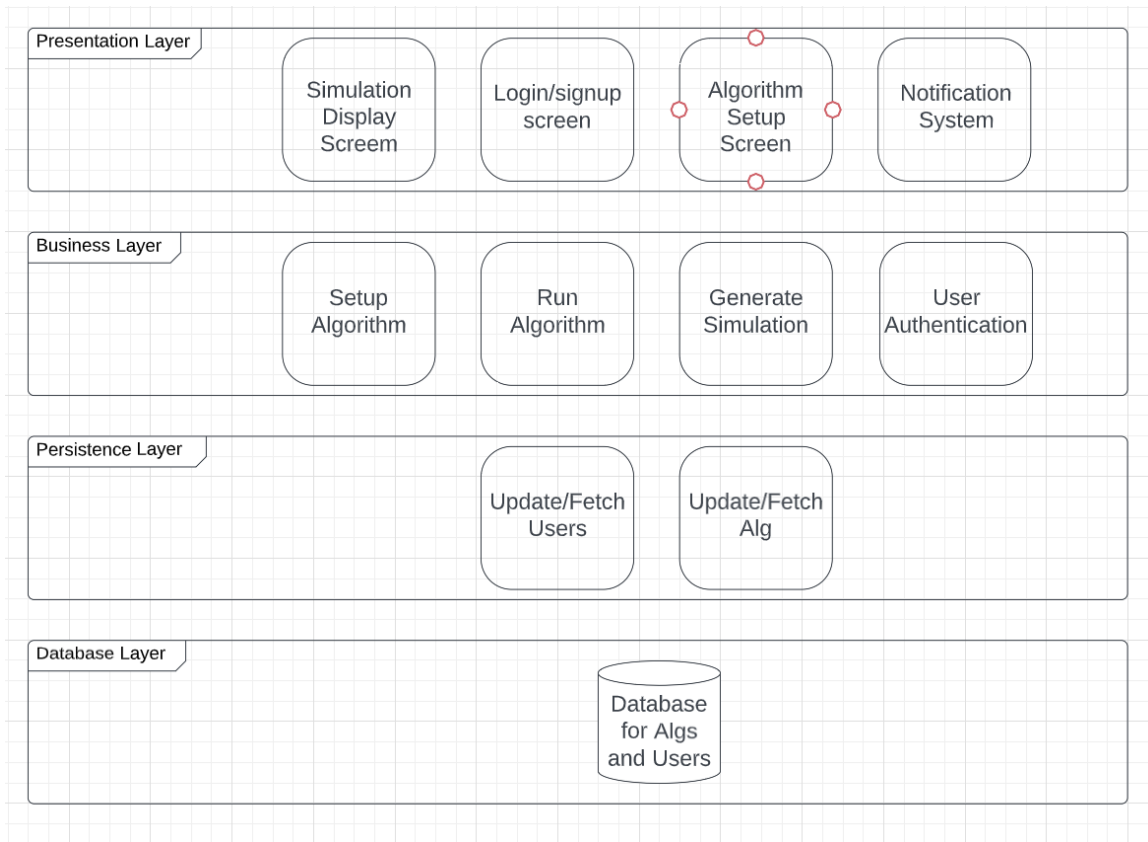


Figure 4: Layered Architecture Proposal

Presentation Layer - This is the layer seen by the users of our system. This layer contains the UI, which displays all the functionality our program offers. From the diagram you can see this layer includes, simulation display, login screen, algorithm input screen, and notification system.

Business Layer - This is the layer that contains all of the “behind the scenes calculations. The user will not have access to this layer. Within this layer, algorithms are run and converted into display files for the frontend to use. This layer also handles the logic for the notification system and user authentication.

Persistence Layer - This layer contains all the functionality for updating and fetching data from the database. Two main tables - Users and Algorithms

Database Layer - This layer just contains the database and all data associated with it.

#### Frontend Sub-systems

1. User Login Form
2. Simulation Setup
3. Simulation Fetch
  - a. Server → Client
  - b. Database → Server → Client (Stretch Goal)
4. Simulation Parsing/Loading
5. Simulation Rendering
6. Algorithm Notification Handler

#### Backend Sub-systems

1. User Login/Authentication
2. API request layer
3. Simulation Queuing System
4. Algorithm Translation Layer (secondary concern)
  - a. Python → Java
  - b. C → Java?
5. Algorithm Execution
6. Algorithm Export
7. Algorithm Notification System

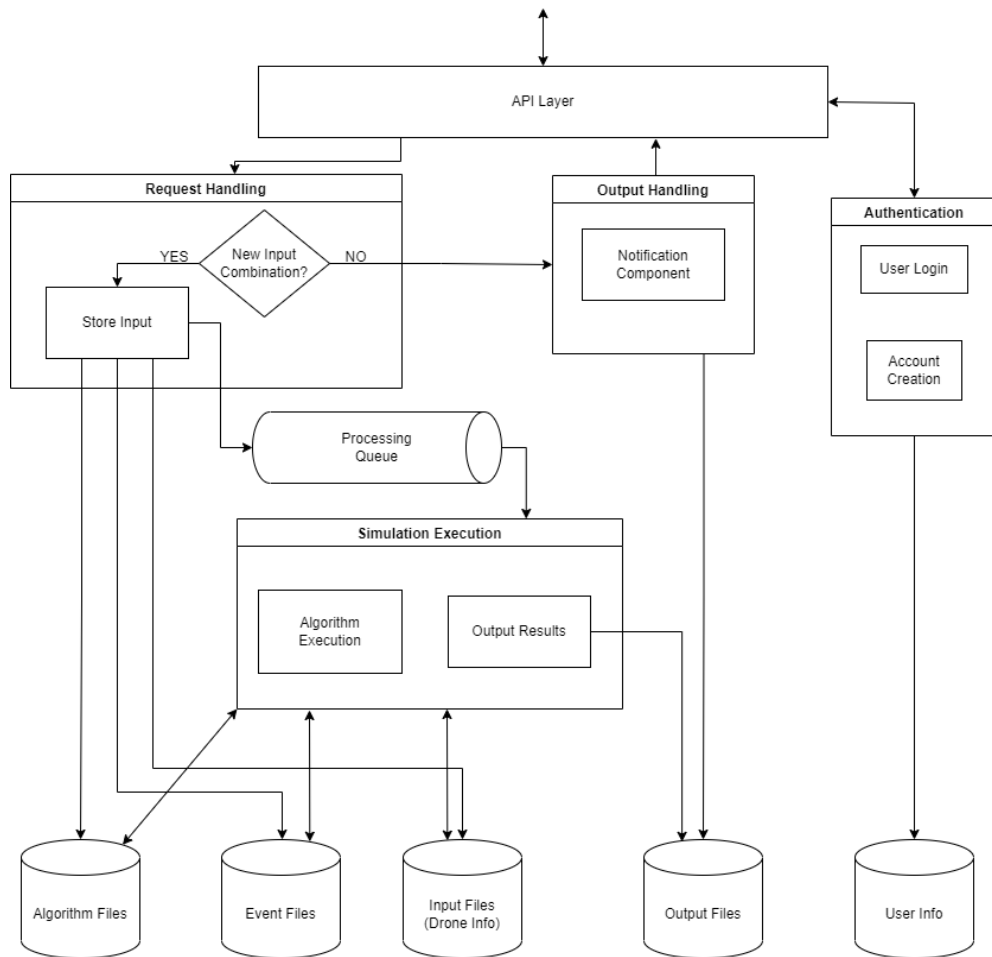


Figure 5: System Architecture Diagram

### 4.3.3 Functionality

Following is a description of how the desired functionality is achieved.

The first step would be for a user to log in, or register for an account( *Figure 9*). Once authenticated, the user has the option to load up previously run simulations or create their own and add it to the queue. The queue will constantly be running in the background, popping off the oldest urban simulations. If the user requests to run a simulation with the same parameters as a previous run simulation, then visualization data will be returned and displayed on their screen ( *Figure 10*).

Below is a sequence diagram depicting our solution at a very high level. The diagram depicts what would happen if a user were to request a simulation with unique parameters, and an empty queue. Once the data points are computed the visualization would be returned to be displayed on the user's screen.

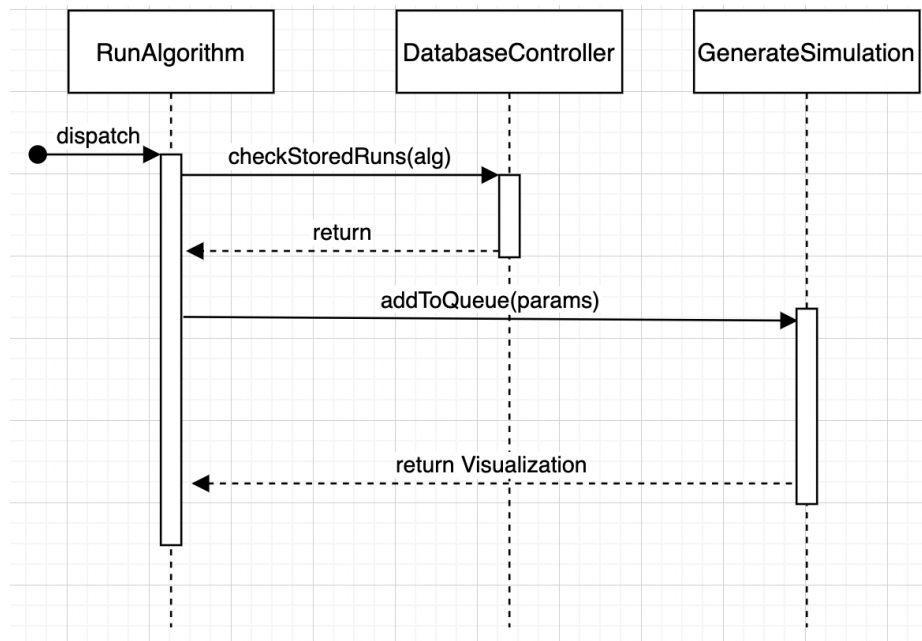


Figure 6: Sequence Diagram

In the case of long run times, we could implement a notification system, featured in our design overview (Figure 5). This would allow us to send out an email to the user requesting the simulation to be run. In the database layer for each user we will save the user inside the simulation data once added to the queue. Once run through the selected algorithm, we will notify the user with an email.

### Frontend UI screens and elements

Each of the following screens contains functionality which allows the user to complete each respective task: Login, authenticate the user and determine the associated simulations; Sign-Up, create a new user; Simulation Setup, define inputs for a simulation and request the server to execute the simulation; Simulation Selection, select which simulation to display; Simulation Visualization, display the results of the simulation.

Below each screen is listed with the required UI components and a sample mockup to visualize what the user would see from the application.

#### Screen 1: Login

1. Textfield: Username
2. Textfield: Password
3. Button: Login
4. Button: Sign-Up
5. Button: Forgot Password
6. Notification: Login Success/Failure

#### Screen 2: Sign-Up

1. Textfield: Username
2. Textfield: Password
3. Textfield: Confirm Password
4. Button: Sign-Up
5. Notification: Sign-Up Success/Failure

Figure 7: Login / Account Creation Mockup

Project Title Placeholder

Username

Password

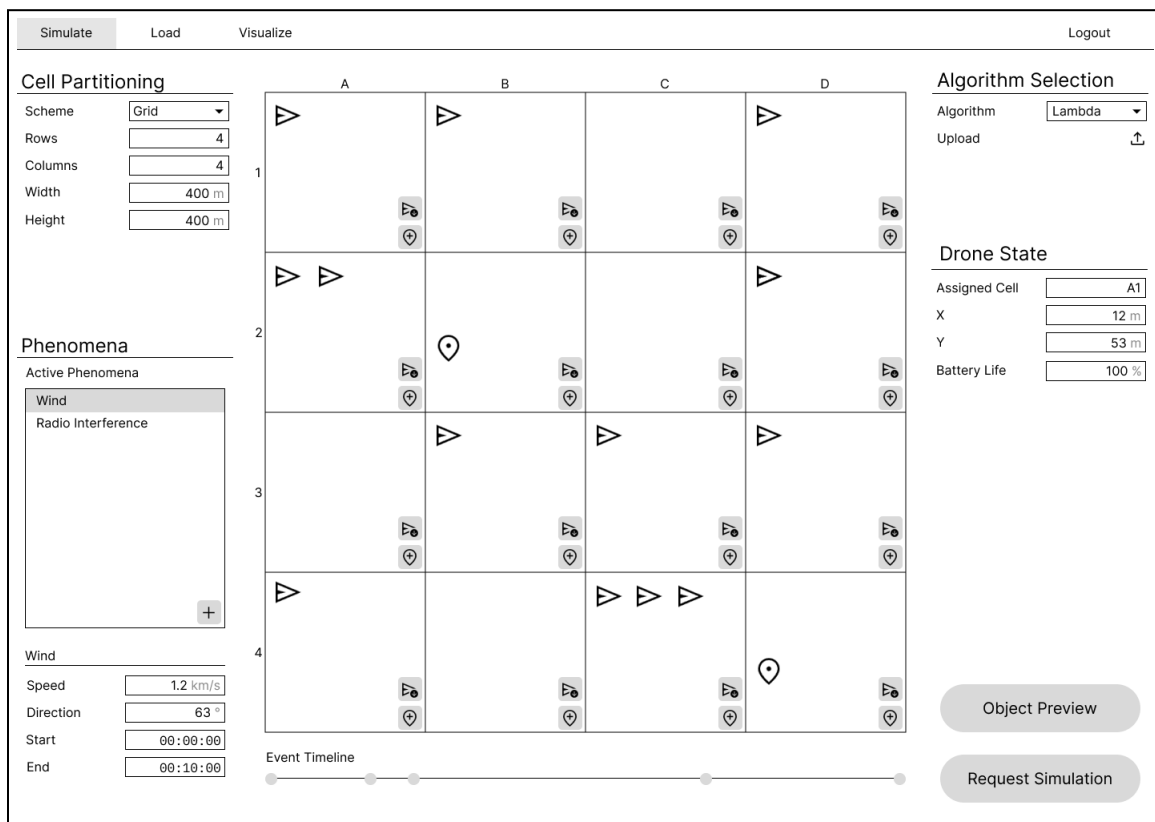
Login

Sign-Up

### Screen 3: Simulation Setup

1. Navigation: Screen Selection
2. Dropdown Menu: Algorithm Selection
3. Upload Button: Algorithm file
4. Cell Distribution
  - a. Textfield: Area Width
  - b. Textfield: Area Height
  - c. Textfield: Rows
  - d. Textfield: Columns
5. Drones
  - a. Button: Add Drone
  - b. Textfield: x-position
  - c. Textfield: y-position
  - d. Textfield: Assigned Cell Number
6. Areas of Interest
  - a. Button: Add New Area of Interest
  - b. Textfield: Event Name
  - c. Textfield: x-position
  - d. Textfield: y-position
  - e. Textfield: Start Time
  - f. Textfield: End Time
7. Additional Phenomena
  - a. Dropdown Menu: Phenomenon Selection
8. Canvas: Object Grid Display
9. Timeline: Event Timeline
10. Button: Add Keyframe
11. Button: Request Simulation

Figure 8: Simulation Setup Mockup



The mockup shows a simulation setup interface with the following components:

- Navigation:** 'Simulate' (active), 'Load', 'Visualize', and 'Logout' buttons.
- Cell Partitioning:** A sidebar with 'Scheme' (Grid), 'Rows' (4), 'Columns' (4), 'Width' (400 m), and 'Height' (400 m) controls.
- Phenomena:** A sidebar with 'Active Phenomena' (Wind, Radio Interference) and 'Wind' controls (Speed: 1.2 km/s, Direction: 63°, Start: 00:00:00, End: 00:10:00).
- Algorithm Selection:** 'Algorithm' (Lambda) and 'Upload' button.
- Drone State:** 'Assigned Cell' (A1), 'X' (12 m), 'Y' (53 m), and 'Battery Life' (100%) controls.
- Canvas:** A 4x4 grid with columns A-D and rows 1-4. It contains drone icons, a central event icon, and various other symbols.
- Event Timeline:** A horizontal timeline at the bottom with a play button.
- Buttons:** 'Object Preview' and 'Request Simulation' buttons at the bottom right.

#### Screen 4: Simulation Selection

1. Navigation: Screen Selection
2. List: Previous Simulations
3. Menu: Sort By
  - a. Date
  - b. Algorithm
  - c. Alphabetical
4. Button: Ascending/Descending
5. Button: Run (Simulation)
6. Button: Rename (Simulation)
7. Button: Delete (Simulation)

Figure 9: Load Simulation Mockup

The mockup shows a web interface for loading simulations. At the top, there are three tabs: 'Simulate', 'Load' (which is active), and 'Visualize'. A 'Logout' link is located in the top right corner. The main content area is divided into two sections. On the left is a table with three columns: 'File Name', 'Date', and 'Algorithm'. Each column has a double-headed arrow icon indicating it is sortable. The table contains three rows of simulation data. On the right is a 'Preview' section, which currently displays a solid blue square. Below the preview is a section labeled 'Simulation Parameters' which is currently empty. At the bottom right of the main content area is a large, rounded rectangular button labeled 'Load Simulation'.

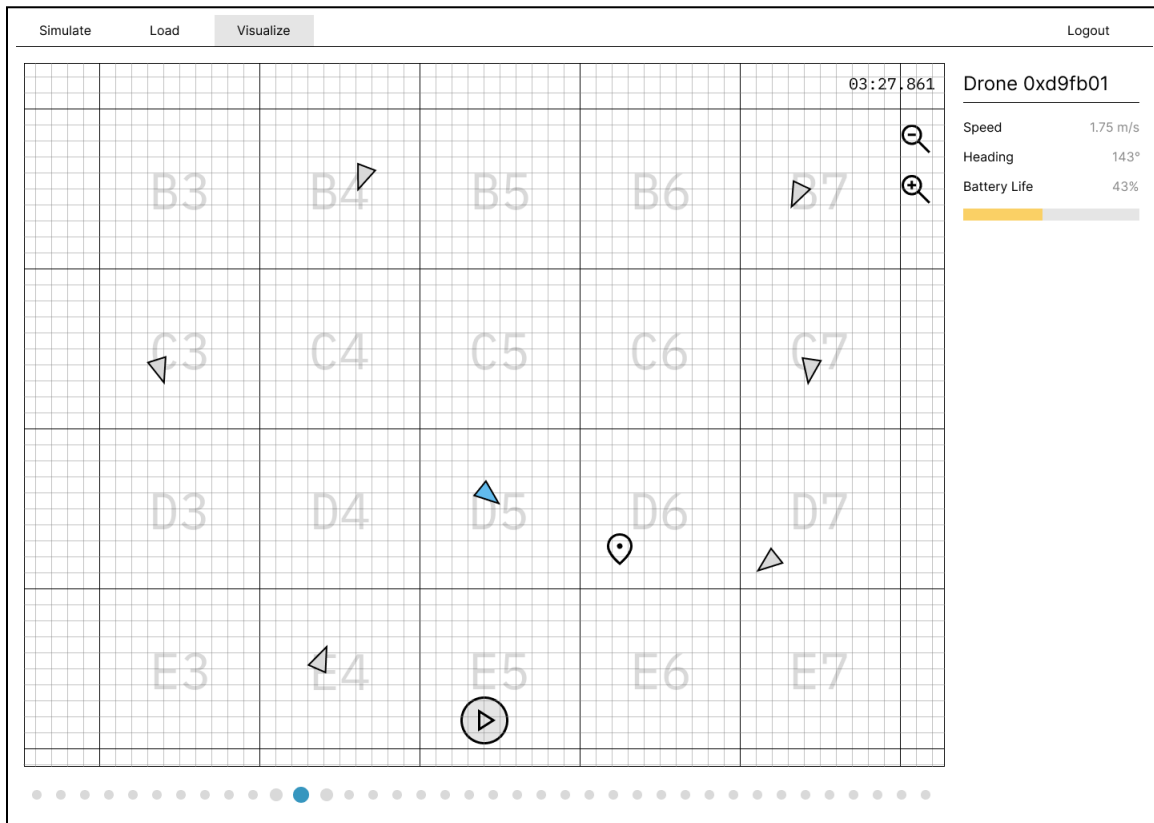
File Name	Date	Algorithm
simulation-01.sim	10-12-2022 07:53	alpha-v1.0.3
simulation-02.sim	10-14-2022 13:42	delta-v2.4.0
simulation-03.sim	10-15-2022 12:04	lambda-v1.5.2



### Screen 5: Simulation Visualization

1. Navigation: Screen Selection
2. Ribbon Menu:
3. Canvas: Simulation Canvas
4. Button: Zoom In/Out
5. Button: Play/Pause
6. Slider: Timeline
7. Side Menu: View Drone Diagnostic Data
  - a. Speed/Heading Data
  - b. Battery Life Indicator

Figure 10: Visualization Mockup



#### 4.3.4 Areas of Concern and Development

We have had multiple meetings with our client to refine requirements and thus the system design. These meetings will continue over the course of the year to continually iterate to meet user needs/expectations.

Primary Concerns:

- Usability, how user friendly the application is
  - By utilizing a clean front end look we will ensure it is simple and easy to use. We will also consult with other students to get feedback on how easily they were able to understand and use the software.
- Scalability of the web application to allow for multiple users
  - By utilizing a queue we can ensure users are able to line up their algorithms to run when the backend becomes available for use. This allows for multiple users to prepare their algorithms to run without having to necessarily wait at their computer to run it when the backend becomes available.
- How long it will take for algorithms to run
  - This will be taken care of by implementing a cutoff on long running algorithms and will notify the user of such an occurrence through a push notification.
- Familiarity with technologies/frameworks used to develop our design for our web application.
  - By assigning tasks to individual members who have experience with those languages/technologies.

Secondary Concerns:

- Securing user data
  - By utilizing modern-day encryption algorithms we will ensure user data is stored privately. If necessary we will consult with fellow students/faculty to ensure our systems are well protected from cyber-attacks.
- Long term support for the project after we graduate
  - By documenting and commenting our code as much as possible, we will ensure future students/faculty will be able to continue to support our project after we graduate.
- The ability for users to upload their own algorithms in other coding languages
  - By implementing a translation layer in our backend we will allow users to submit their own algorithms for use even if it is in a few other supported languages.

We envision two basic categories of questions for the client/advisor.

- 1) How do we develop integration testing scenarios that will enable checking how implementation in the future satisfies our current design?
- 2) What would be a good scenario for checking multiple algorithms?
  - a) How many algorithms should be incorporated in the final deliverable.

## 4.4 TECHNOLOGY CONSIDERATIONS

Table 7: Technology Considerations

Technology	Pros	Cons	Alternatives
React (Presentation)	+ Very popular, lots of documentation + Flexible and easy to adapt to our needs	- Can be resource intensive - May require additional work to adapt to different screens	Angular, Vue
Spring (Business/Persistence)	+ Very popular, great documentation and support + Group members have used it before + Quick setup and little boilerplate	- Can be slow for startup - More specific/ complicated use cases can be tedious	Quarkus
MySQL (Database)	+ Mature and reliable technology + Group members have experience with it	- Requires strict setup for data schema - Not as flexible or easy to set up	PostGres, MongoDB, Oracle
GitLab (Deployment/CICD)	+Group members have experience with it +Flexible and easy to set up	- Not as many features as some other options	Jenkins, Github Workflows

## 4.5 DESIGN ANALYSIS

So far we have only familiarized ourselves with the scientific fundamentals and we explored the frameworks and technologies/tools that are most suitable for the implementation stage of the project. The actual development of the individual modules as well as APIs and their integration is planned to take place during the Spring semester and the envisioned activities and milestones were illustrated in Figure 5.

**We have thus far not implemented any proposed design and do not plan to until the second half of the course. We will begin to plan out work for implementing the above design at the beginning of the spring semester.**

## 5 Testing

In this section we discuss our plan for conducting various testing procedures to validate the implementation of the project. We note that the requirements (functional and non-functional)

were already translated into the architectural components discussed in Section 3 of this document, and we refer to corresponding figures from that section here.

Given that the project involves large (simulation based) datasets, along with execution of algorithms and visualization of the output, there are certain distinct characteristics of our testing.

We note that the main objective of this project is to develop a proof-of-concept implementation, not a complete commercially available prototype. As such, the cost component is not a major testing factor.

## 5.1 UNIT TESTING

There are a handful of aspects in our proposed design that require unit testing to ensure correct functionality of all systems. Unit testing is essential to scale functionality of our project. Our team will be testing in parallel with new code development in accordance with agile methodologies. Below is listed our proposed ideas for unit testing:

- We need to ensure consistent algorithm output. By running JUnit tests on an algorithm given specific parameters, we can ensure that output is always in our generalized file format. While we can't test the output, as algorithms being used may vary, we can test that an output file is formatted correctly.
- We need to make sure the system can detect a simulation with parameters that have already been run by running JUnit tests to grab data from an output file and make sure it matches with preset parameters.
- Our frontend system needs to function properly and translate the proper data to the server for drone visualization. In order to ensure correct functionality we need to ensure:
  - parameters are passed as selected by the client
  - the account responsible for creating the request is passed with the parameters
- Our notification system needs to function properly in order to notify the correct users that the simulation data is up. In order to verify the functionality we will need to make sure that:
  - the notification system correctly identifies the user who submitted the request
  - the notification email contains the formatted file attached
- Our user registration system will need to be tested in order to be certain that duplicate users cannot be created.

While this list isn't completed, it's an outline for what's to come. Unit testing will be vital in order to provide a functioning and reliable system, so new items will be added in the future. To provide a scalable system our group will aim for near perfect line and case coverage of our application. Test input will be small in comparison when compared to data being computed in production. Using small sample sizes when testing will help us predict with higher accuracy when unit testing.

Tools: JUnit, Mockito

## 5.2 INTERFACE TESTING

To communicate within the application, the following interfaces are defined: simulation setup, simulation parsing, and algorithm notification. The simulation setup interface interprets information from the frontend and creates a simulation. The simulation parsing interface converts JSON into objects to display on screen. And finally, the notification system alerts the frontend on the status of simulations. Similar to unit testing, our team will be testing in parallel with new code development in accordance with agile methodologies.

Testing user selection of simulation setup with backend:

To test the simulation setup interface:

- Sample JSON files given to the interface and results will be compared to other known results of the algorithm.
- Invalid arguments will be given to the interface to ensure the interface rejects invalid data.

To test simulation parsing interface:

- Partial simulation results in the form of JSON objects will be passed to the interface to display, verify that each drone's location, and phenomena display individually.
- Complete JSON files will also be passed to the interface to verify that all objects are able to display at once.

To test the notification interface:

- Sample notifications will be created and passed to the interface which the frontend must display.

None of the success of each the setup, parsing, and notification interfacing test shall depend on each other, and thus each may be developed and verified in parallel. This also supports the Agile methodology by further compartmentalizing functionality.

Tools: Jest, Mockito, JUnit

## 5.3 INTEGRATION TESTING

There are a couple critical integration paths that need to be tested. The first path is between Spring Boot and the database, and the second path is between Spring Boot and the React frontend. These paths are critical because we pass algorithm information, simulation information and user information between these paths. The data is sent along these paths by using the REST API that we are implementing using Spring Boot. These API Calls need to be tested to ensure correctness and efficiency. Our primary way of testing these API calls will be to send specific data to a known location, then ensuring that this data arrives at said location unmodified. These tests will be conducted with input data on a smaller scale than production data, and done parallel to new code additions.

Examples:

- Test that we can send a user inputted algorithm and store it on the database.
- Test that simulation information gets processed to the frontend correctly.
- Test that the notification system on the front end works when the notification is triggered in Spring Boot.

Tools: Mockito, Postman

Many of these tests will be conducted in parallel in order to reduce the amount of time and resources this testing will take. This also is in accordance with Agile methodology, which we are using for our development strategy.

## 5.4 SYSTEM TESTING

For system testing we will use the functional requirements as criteria. Refer to section 2.1.1 of this document. Key requirements that need system testing.

- The software shall provide a visualization of the drone flight.
  - The software shall have a 2d grid layout of the geographical area.
- The software shall save previously run simulations for input combinations that will be accessible for other users.
- The software shall allow the user to login to a personal account profile.
- etc.

Each of these requirements require the entire system to be operational in order to test. These tests will be conducted using a combination of unit, interface, and integration tests. In order for these tests to be deemed successful, they will be evaluated by the client. The basic metrics will be:

- Tests output with the correct result
- Server responses come within a timely manner
- Database storage efficiently resource usage.

The tools used throughout this process may include (but are not limited to): Postman, Mockito, Jtest, Junit.

## 5.5 REGRESSION TESTING

Regression testing ensures that existing functionality still works when new additions or updates are made to the software.

Regardless of any changes that are made to the application, it is vital that the web interface remains accessible online. Of course we also need to ensure that the simulation is accessible and functional, and any other functionality should remain unless it is intended to be removed by the update.

We will manage this principally by developing tests for components as part of their development. This is an important part of our agile approach to this project's development, and will allow us to ensure all desired functionality is present before anything reaches deployment. For example, if I was creating a component to import new drone algorithms, I would also write unit and integration tests to demonstrate proper operation before any code is reviewed. Similarly, if I was updating a component, I would need to write new tests showing the new functionality. For these tests to be meaningful, they must of course completely demonstrate the behavior our application should exhibit.

These tests will remain, and each test set will verify proper operation of their respective components. Therefore before any new changes and tests are deployed, all existing unit, integration, and system tests must be run to ensure that existing functionality is retained.

These tests will run via our CI/CD pipeline once code is pushed to our remote version control system.

These tests should also be run when non code changes are made, such as the addition of new algorithms and phenomena.

## 5.6 ACCEPTANCE TESTING

The primary purpose of testing is to show that our functional and nonfunctional requirements are being met. Thus, our testing must be clear in what acceptance means for each test.

### Functional Requirements:

- The software shall give the user the ability to choose which phenomena (e.g. fire, explosion, ...) to apply to the drone flight simulation.
  - **Acceptance criteria: User is able to select a file to be used to describe the event phenomena.**
- The software shall give the user the ability to choose which drone algorithm(s) to apply to the drone flight simulation.
  - **Acceptance criteria: User is able to select a file to be used for the fleet algorithm.**
- The software shall accept input combinations through selected files.
  - **Acceptance criteria: User selected files are used in the running of the simulation.**
- The software shall provide a visualization of the drone flight.
  - **Acceptance criteria: The software shall have a 2d grid layout of the geographical area. There should be identifiable icons for the drones and the events across a timeline of running.**
- The software shall calculate drone statistics and values over time and record them to storage.
  - **Acceptance criteria: Output of running the simulation should have drone statics in its output file. The output file should be stored in a location to be retrieved at a later date (ie database).**
- The software shall provide a view of statistics around the drones (battery life, location, speed, etc).
  - **Acceptance criteria: Battery life, location, speed and other client specified data points (to be documented) should be in the output file of the simulation.**
- The software shall save previously run simulations for input combinations that will be accessible for other users.
  - **Acceptance criteria: Output file should be stored in a database.**
- If the user selects an input combination that has been simulated before then the software shall return back the already run simulation data.
  - **Acceptance criteria: Repeated input combinations should retrieve previously run simulation output files, instead of re-running the simulation. ie Repeat input should not result in a new simulation output file in the database.**

- If the user selects an input combination that has not been simulated previously then the software shall queue the simulation to be run on the backend.
  - **Acceptance criteria: New input combinations should result in another output simulation file in the database.**
- If a simulation is run from a queued input combination then the software shall notify the requesting user when the simulation is completed (push-based notification).
  - **Acceptance criteria: For new input combinations, the user should receive a notification (via the web page or possibly email), after the simulation is run.**
- The software shall accept user input (file) which is given in source code which is ready to run (compiled or interpreted, etc).
  - **Acceptance criteria: Python files should be accepted as valid algorithm files.**
- The software shall store the simulation output in a format containing: starting location, starting time, destination location, arrival time, trajectory.
  - **Acceptance criteria: The output simulation file should have starting location, starting time, destination location, arrival time, and trajectory.**
- The software shall have 3 UI components: list of algorithms to pick, list of event phenomena input, visualization screen of simulation output.
  - Possibly a 4th component for notifications.
  - **Acceptance criteria: Self-evident.**
- 

#### Non-Functional Requirements:

- The software should be easy to use and understandable since not all of our users have a technical background
  - **Acceptance criteria: A new user, given an example event phenomena file and algorithm file, should be able to run a simulation and view the results within 30 min of being introduced to the website.**
- UI elements of the software shall be intuitive and clearly labeled or documented.
  - **Acceptance criteria: A new user should be able to correctly identify the purpose of each UI element (buttons, drop-downs, etc), within 30 min of being introduced to the website.**
- The software shall handle errors in the input gracefully.
  - **Acceptance criteria: For a selected file that is not correctly formatted for its selected use, there should be a red colored warning message about the selected file being invalid, and a short description for the reason behind the error.**
- The software shall combine concurrent access for already executed input combinations and will run push notifications for users with new input simulations.
  - **Acceptance criteria: The user should be able to see and select the result of previously run simulations.**
- The software shall be compatible with Windows, MacOS, and Linux.
  - **Acceptance criteria: A user on all 3 types of machines should be able to access and operate the website.**
- The software shall be developed in a manner that is supportable & maintainable after our team leaves.



- **Acceptance criteria: Each function, and non-self-evident piece of code, shall have a single/multi-line comment to explain its purpose. The software should be developed in a manner that allows for modularity and the possibility of adding additional features.**

## 5.7 SECURITY TESTING (IF APPLICABLE)

Despite security not being implicitly stated in our project description, we feel it is our responsibility to uphold a certain level of scrutiny when it comes to protecting sensitive data. Since users of our app will be giving us usernames, emails, and a password we have a duty to make sure this information is not shared with unintended audiences. We will ensure the safety of our users by using the latest encryption techniques, such as AES, to secure passwords and other sensitive information.

Asking for help from other security oriented students, and our own team, to scan for vulnerabilities in our database or possible flaws in our web application would be beneficial. However, since security is not a primary concern of our project, we believe no detailed plans for security testing would be necessary for the scope of our design.

- The software shall allow the user to login to a personal account profile. (Functional requirement)
  - **Acceptance criteria: User with credentials is able to login.**

## 5.8 RESULTS

**At this point we do not have a product to test against, and thus do not have any results to report.**

# 6 Implementation

**No implementation work has been done thus far.**

# 7 Professional Responsibility

This discussion is with respect to the paper titled “ Contextualizing Professionalism in Capstone Projects Using the IDEALS Professional Responsibility Assessment”, *International Journal of Engineering Education* Vol. 28, No. 2, pp. 416–424, 2012

In this section we present an overview of different aspects of professional responsibility based on the materials presented in [3].

## 7.1 AREAS OF RESPONSIBILITY

*Table 8a: Responsibility NSPE*

<b>Area of Responsibility</b>	<b>Definition</b>	<b>NSPE Canon</b>
Work Competence	Perform work of high quality, integrity, timeliness, and professional competence.	Perform services only in areas of their competence; Avoid deceptive acts
Financial Responsibility	Deliver products and services of realizable value at reasonable costs	Act for each employer or client as faithful agents or trustees.
Communication Honesty	Report work truthfully without deception and understandable to stakeholders	Issue public statements only in an objective and truthful manner; Avoid deceptive acts
Healthy, Safety, Well-Being	Minimize risks to safety, health, and well-being of stakeholders	Hold paramount the safety, health, and welfare of the public
Property Ownership	Respect property, ideas, information of clients and others.	Act for each employer or client as faithful agents or trustees.
Sustainability	Protect the environment and natural resources locally and globally	Hold paramount the safety, health, and welfare of the public
Social Responsibility	Produce products and services that benefit society and communities.	Conduct themselves honorably, responsibly, ethically, and lawfully so as to enhance the honor, reputation, and usefulness of the profession.

*Table 8b: Responsibility IEEE*

<b>Area of Responsibility</b>	<b>IEEE Code of Ethics</b>
Work Competence	<b>1.6.</b> to maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations;
Financial Responsibility	<b>1.4.</b> to avoid unlawful conduct in professional activities, and to reject bribery in all its forms;
Communication Honesty	<b>1.5.</b> to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, to be honest and realistic in stating claims or estimates based on available data, and to

	credit properly the contributions of others;
Healthy, Safety, Well-Being	1.1. to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, to protect the privacy of others, and to disclose promptly factors that might endanger the public or the environment;
Property Ownership	1.5. to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, to be honest and realistic in stating claims or estimates based on available data, and to credit properly the contributions of others;
Sustainability	1.1. to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, to protect the privacy of others, and to disclose promptly factors that might endanger the public or the environment;
Social Responsibility	1.2. to improve the understanding by individuals and society of the capabilities and societal implications of conventional and emerging technologies, including intelligent systems;

## 7.2 PROJECT SPECIFIC PROFESSIONAL RESPONSIBILITY AREAS

**Work Competence:** Work Competence is vital to our project. Senior Design's whole goal is to simulate a real work environment. We need to use the skills we have learned over the years in order to produce a product, and without the ability to adapt to project specifications and hit deadlines, we will be jeopardizing our ability to produce our application. As the next semester starts, it'll be more essential than ever to hit milestones and finish strong. Our team is performing high in this area of responsibility.

**Financial Responsibility:** Main objective is to develop a prototype, not a working product; therefore, we did not have a large budget to manage. As a consequence, the component of financial responsibility is not a major issue of this project. Therefore, our team is performing N/A in this aspect.

**Communication Honesty:** Communication honesty is also another area of responsibility that has high importance to this project. We need to be open and honest when communicating with our advisors, Professor Goce Trajcevski and Prabin Giri, in order to keep progressing. Our team is performing high in this area of responsibility.

**Healthy, Safety, Well-Being:** This level of responsibility has little relation to our project. Since our product will be a web application, there is no safety risk associated with it. There is little risk involved with our project to the health and safety of both the developers and future consumers. Because of the little application to our project, our team is performing N/A in this aspect.

**Property Ownership:** Property Ownership is something our team is performing highly in. We are in constant communication with our advisor, and make changes to our design as he sees fit. As a team we are excited to continue work on implementation in the next semester.

**Sustainability:** Sustainability is extremely important to our project, along with any web application. We aim to provide a service that will be scalable if additions are to be made to it in the future. For these reasons, our team is performing high in this area of responsibility at this point.

**Social Responsibility:** Social Responsibility is the second most important area of responsibility when it comes to our project. We aim to provide a platform to allow consumers to emulate drone flight patterns, using a variety of algorithms. We are actively trying to improve the understanding of this technology by making it easier to access. For this reason, our team is performing high in this aspect.

### 7.3 MOST APPLICABLE PROFESSIONAL RESPONSIBILITY AREA

Work Competence is the most applicable professional responsibility in relation to our project. In order to complete the plan we have laid out above, it's important to be timely. We have hit every deadline for this semester and we will continue to do so into the next semester. As a group we are responsible for understanding the skills required to achieve our goals, and without that knowledge timelines will be missed. Our group so far has displayed a high level of work competence, and this is because of our dedication to the project, and our willingness to dive deep and understand the components required to complete this outline.

## 8 Closing Material

### 8.1 DISCUSSION

From this semester we learned a lot about following design patterns, and how to use our resources. We have created a document that will guide us towards a successful end goal, being a web application. At this point we do not have any results to put on display yet. Next semester we will put the above plan into action, following the requirements we have set inside the document.

### 8.2 CONCLUSION

Over the course of this semester we have spent hours using our advisor's input in order to create this design document. Originally we set out this semester to create a plan to help us succeed in the following semester when we utilize our design. Our goal for the next semester will focus on creating the actual application, utilizing this design document to keep development on track.

### 8.3 REFERENCES

[1] Z. Fu, Y. Mao, D. He, J. Yu, and G. Xie. 2019. Secure Multi-UAV Collaborative Task Allocation. IEEE Access 7 (2019), 35579–35587. <https://doi.org/10.1109/ACCESS.2019.2902221>

[2] Hailong Huang and Andrey V. Savkin. 2018. Towards the Internet of Flying Robots: A Survey. Sensors (Basel, Switzerland) 18, 11 (19 Nov 2018), 4038. <https://doi.org/10.3390/s18114038>

[3] McCormack et al., "Contextualizing Professionalism in Capstone Projects Using the IDEALS Professional Responsibility Assessment", *International Journal of Engineering Education (IJEE)*, 28(2), 2012

## 8.4 APPENDICES

The following entries were additional materials our team produced during the course of project development and planning.

### 8.4.1 Team Contract

Team Members:

- |                      |                  |
|----------------------|------------------|
| 1) Marcus Jakubowsky | 2) Joe Edeker    |
| 3) Thomas Glass      | 4) Rowan Collins |
| 5) Jaden Forde       | 6) Jacob Houts   |

### Team Procedures

1. Day, time, and location (face-to-face or virtual) for regular team meetings:
  - a. Mondays 3-4/4:30pm Library (or other location TBD) for a regular team meet time
  - b. Weekly meeting time with Prof Goce - Wednesdays 6:30-7:30pm (In-person) - Pending
2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):
  - a. Discord - for quick communication & reminders
  - b. Face-to-face - for longer discussions & decision making
3. Decision-making policy (e.g., consensus, majority vote):
  - a. Majority vote (since we have an even number, we can discuss further and sort it out. In extreme cases, rock paper scissors may be applied. Or let Goce decide)
4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):
  - a. We will be rotating the minute leader responsibilities weekly.
  - b. Minutes will be recorded on the "Meeting Minutes" document in our shared drive.

### Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:
  - a. Everyone should be in attendance by the agreed upon time. If someone cannot make it then they will be responsible for letting the team know in advance two hours ahead of time.
  - b. If the meeting is scheduled to be in-person, each team member will do their best to be there in-person; however, if a situation arises and an individual cannot make

the meeting in-person they will inform the rest of the team and do their best to participate in the meeting online.

2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:
  - a. If a team member is assigned a task with a specific deadline, it is expected that the task will be completed by that deadline with a level of quality that would be approved by a majority of the rest of the team.
  - b. If a team member knows they will not be able to meet a scheduled deadline, they should notify the rest of the team ideally 24+ hours beforehand so the rest of the team can help cover that assignment.
    - i. More leeway if the work is close completion, conversely less if it would put a significant burden on other team members.
  - c. At any point, an individual can ask for additional support from the team on any task. The earlier the better for getting help.
    - i. Each team member is accountable for providing help to other team members on all respective skills and expertise listed in the table.
3. Expected level of communication with other team members:
  - a. We expect everybody to check the discord chat once per day for any new messages or information concerning the project.
  - b. If any urgent messages are sent to the chat responses are expected within 3 hours.
  - c. We will offer more leeway on message responding on weekends. On Saturdays we won't expect anyone to respond to messages, urgent or not, and will expect a response by Sunday.
4. Expected level of commitment to team decisions and tasks:
  - a. Every member of the group is expected to vote on motions we bring up. All members have the opportunity to bring something up to a vote.
  - b. Everyone is responsible for any work assigned to them, and will be expected to deliver on time. See consequences for not adhering to team contract on page four.
  - c. All team members are expected to follow the guidelines laid out in this contract.

## Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):

<b>Team Organizer</b>	Marcus Jakubowsky
<b>Minute Master</b>	Rotates weekly, outlined in the "Meeting Minutes" document
<b>Client Interaction</b>	Jaden Forde
<b>UI/UX Design</b>	Joe Edeker
<b>Inter-Component Tester</b>	Jake Houts
<b>Standards &amp; Security Validation</b>	Thomas Glass

<b>Team Presenter</b>	Rowan Collins
-----------------------	---------------

2. Strategies for supporting and guiding the work of all team members:
  - a. All tasks will have a clearly defined person(s) assigned to it.
  - b. All tasks will be given enough time to complete them (based on team majority opinion).
  - c. Give clear and agreed upon guidelines/expectations for a task.
    - i. If you are assigned a task and you don't understand the expectations, it is your responsibility to get clarification.
3. Strategies for recognizing the contributions of all team members:
  - a. The team will acknowledge when an individual has done good quality work verbally (or through emojis).
  - b. If you work on something, or help someone else with theirs, your name will be on that (author tags).

### Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team:

<b>Team Member</b>	<b>Skills, Expertise, and Unique Perspectives</b>
Marcus	Languages: Java, C, Python, Typescript Frameworks: Angular, Springboot, SQL Technologies: AWS, UNIX, Git, CI/CD, Docker, Database
Thomas	Languages: Java, C#, Python, C Technologies: Splunk, Wireshark, pfSense Experience: Firewalls, Network Security, Threat Analysis
Jaden	Java, Spring APIs, Python, Angular, working with data at a large scale, basic robotics/computer vision principles. Internships at NISC and Garmin doing Java backend server work, Angular frontend design/building.
Joe	Languages: Java, HTML Frameworks: React Other: Graphic Design, Video Editing
Rowan	Languages: Java, C, Python, SQL, noSQL, Javascript, HTML Frameworks: Springboot, Flask Technologies: Git, UNIX, AWS
Jacob	Languages: C, C++, Java, Javascript, HTML, swift, Xcode Frameworks: React, OpenGL, Springboot. Technologies: Git, UNIX

2. Strategies for encouraging and supporting contributions and ideas from all team members:
  - a. Asking for input often, showing gratitude to everyone who contributes.

- b. If there is a discussion on a topic, all members will be given the opportunity to speak their thoughts.
- 3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their opportunity or ability to contribute?):
  - a. Discuss at the next team meeting. If urgent, discuss in discord immediately. If this concerns a single individual, bring it to their attention first.

### **Goal-Setting, Planning, and Execution**

- 1. Team goals for this semester:
  - a. As a team we are looking to deliver a viable project design by the end of the semester. Besides delivering what we are tasked with, our group hopes to learn more about the process that goes into designing a final product.
- 2. Strategies for planning and assigning individual and team work:
  - a. Once deliverables have been determined for the week, two members will be given first priority (rotating each week) to choose which deliverables to work on. A 12 hour time limit may be enforced to select deliverables.
  - b. If someone were to want to work on an already assigned task, they would need to speak to the person who has taken that assignment. From there they can work out an agreement, if both parties are willing, to switch assignments.
  - c. Team Organizer/Team Planner will be in charge of knowing what assignments are due & what the deadlines.
- 3. Strategies for keeping on task:
  - a. As a team we will be openly communicating amongst each other in order to understand where each individual member is at with their work. When the team meets, the conversation will only be on project work.
  - b. Team Organizer is responsible for keeping meetings on track.

### **Consequences for Not Adhering to Team Contract**

- 1. How will you handle infractions of any of the obligations of this team contract?
  - a. If there are infractions of the team contract somebody within the team can bring the issue forward to the rest of the team members. Every time there is an infraction we will record it in the Meeting Minutes doc. The rest of the team can then decide how to handle the situation. Most of the time a conversation will be enough to put the teammate back on track. If we have two conversations with the same team member, we will escalate it accordingly.
  - b. If an individual is not fully comfortable with directly bringing that issue up to the team, they should at least bring it up to another individual and they can bring it before the team together.
- 2. What will your team do if the infractions continue?



- a. If the infractions continue to incur after the first two of conversations, then we will bring this to the attention of Professor Goce. From here we can make a decision to dock the team member points in the peer evaluations if necessary.

\*\*\*\*\*

- a) I participated in formulating the standards, roles, and procedures as stated in this contract.
- b) I understand that I am obligated to abide by these terms and conditions.
- c) I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.

- |                      |              |
|----------------------|--------------|
| 1) Rowan Collins     | DATE 9/19/22 |
| 2) Thomas Glass      | DATE 9/19/22 |
| 3) Jacob Houts       | DATE 9/19/22 |
| 4) Jaden Forde       | DATE 9/19/22 |
| 5) Marcus Jakubowsky | DATE 9/19/22 |
| 6) Joe Edeker        | DATE 9/19/22 |