

# Collaborative Surveillance of Large Geographical Area by Fleet of Drones

FINAL REPORT

ECpE 492 - Senior Design May '23 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/50>

Revised: 4/27/2023

# Table of Contents

|   |           |
|---|-----------|
| <b>List of Tables &amp; Figures</b>                   | <b>3</b>  |
| <b>1 Project Design</b>                               | <b>4</b>  |
| 1.1 Acknowledgement                                   | 4         |
| 1.2 Problem and Project Statement                     | 4         |
| 1.3 Design Evolution                                  | 4         |
| 1.4 Original Design                                   | 5         |
| 1.5 New Design  | 7         |
| 1.6 Requirements                                      | 10        |
| 1.7 Standards   | 11        |
| Table 1.1: Engineering Standards Applications         | 11        |
| 1.8 Constraints                                       | 11        |
| 1.9 Security Concerns and Countermeasures             | 12        |
| <b>2 Implementation</b>                               | <b>13</b> |
| 2.1 Implementation Methodology                        | 13        |
| 2.2 Server Side Implementation                        | 13        |
| 2.3 Client Side Implementation                        | 15        |
| <b>3 Testing</b>                                      | <b>16</b> |
| 3.1 Unit Testing                                      | 16        |
| 3.2 Interface Testing                                 | 16        |
| 3.3 Python Backend Testing                            | 17        |
| 3.4 Acceptance Testing                                | 17        |
| <b>4 Work Context</b>                                 | <b>19</b> |
| 4.1 Related Projects                                  | 19        |
| 4.1.1 Microsoft Airsim                                | 19        |
| 4.1.2 Flightreader                                    | 19        |
| 4.2 Related Literature                                | 20        |
| 4.2.1 Towards The Internet of Flying Robots: A Survey | 20        |
| <b>5 Conclusion</b>                                   | <b>21</b> |
| <b>6 References</b>                                   | <b>21</b> |
| <b>Appendix A - Operation Manual</b>                  | <b>22</b> |
| A.1 Signing up and Logging in                         | 22        |
| A.2 Dashboard   | 22        |
| A.3 Running Your First Simulation                     | 23        |
| A.4 Visualizing Your Simulations                      | 24        |
| A.5 Comparing Simulations                             | 25        |
| <b>Appendix B - Original Design Considerations</b>    | <b>27</b> |
| B.1 Original Design Overview                          | 27        |
| B.2 Original Backend Design                           | 27        |
| B.3 Original Frontend Design                          | 28        |

## List of Tables & Figures

- Figure 1.1 System Architecture Diagram .... Page 5
- Figure 1.2 Original Simulation Setup Mockup .... Page 6
- Figure 1.3 Redesigned System Architecture Diagram .... Page 7
- Figure 1.4 Redesigned Simulation Setup .... Page 8
- Figure 1.5 Dashboard Page .... Page 9
- Figure 1.6 Redesigned Simulation Visualization .... Page 9
- Figure 1.7 Simulation Comparison Page .... Page 10
- Table 1.1 Engineering Standards Applications .... Page 4
- Figure 2.1 System Architecture Design .... Page 14

# 1 Project Design

## 1.1 ACKNOWLEDGEMENT

We would like to thank our project advisor Dr. Goce Trajcevski and graduate student Prabin Giri for their guidance throughout the past year. They have been essential to our understanding and development of the project.

## 1.2 PROBLEM AND PROJECT STATEMENT

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.

## 1.3 DESIGN EVOLUTION

Since we began implementation from our proposed design we developed in CPRE491 our project has undergone change. This can be attributed to a variety of factors including security concerns, time constraints, increased understanding of the project, and technological issues.

Displayed below in section 1.5 is our revised backend design. As depicted in the diagram, we’ve kept a majority of the design intact. The main differences between the two architecture diagrams are the removal of output handling, and the removal of searching for previously ran simulations. Upon the start of the semester, it quickly became apparent that the simulations took much less time than originally planned. Our group thought it would take upwards of thirty minutes to calculate drone paths.

Because of the aforementioned time constraints, our group thought it best to notify the user who submitted the visualization request through email. Since our discovery we have completely removed the idea, along with reading for already ran simulations. The time to simulate is so low, that an email being sent to the user seemed redundant. Though in the future we would prefer for this application to implement this. The flexibility our application provides comes with unforeseen high runtimes of user created algorithms, and this notification system could be beneficial for users submitting long runtime algorithms.

Our group has also removed the ability to add ground phenomena that could impact drone flight patterns, and removed the ability to see individual drone performance stats. The implementation of ground phenomena required too much time, and was unfortunately cut. Viewing individual drone statistics was optimistic, and we had no consistent way to accurately predict the drone battery or the drone's velocity. Due to the time constraints of this project, it has since been removed from our list of requirements.

## 1.4 ORIGINAL DESIGN

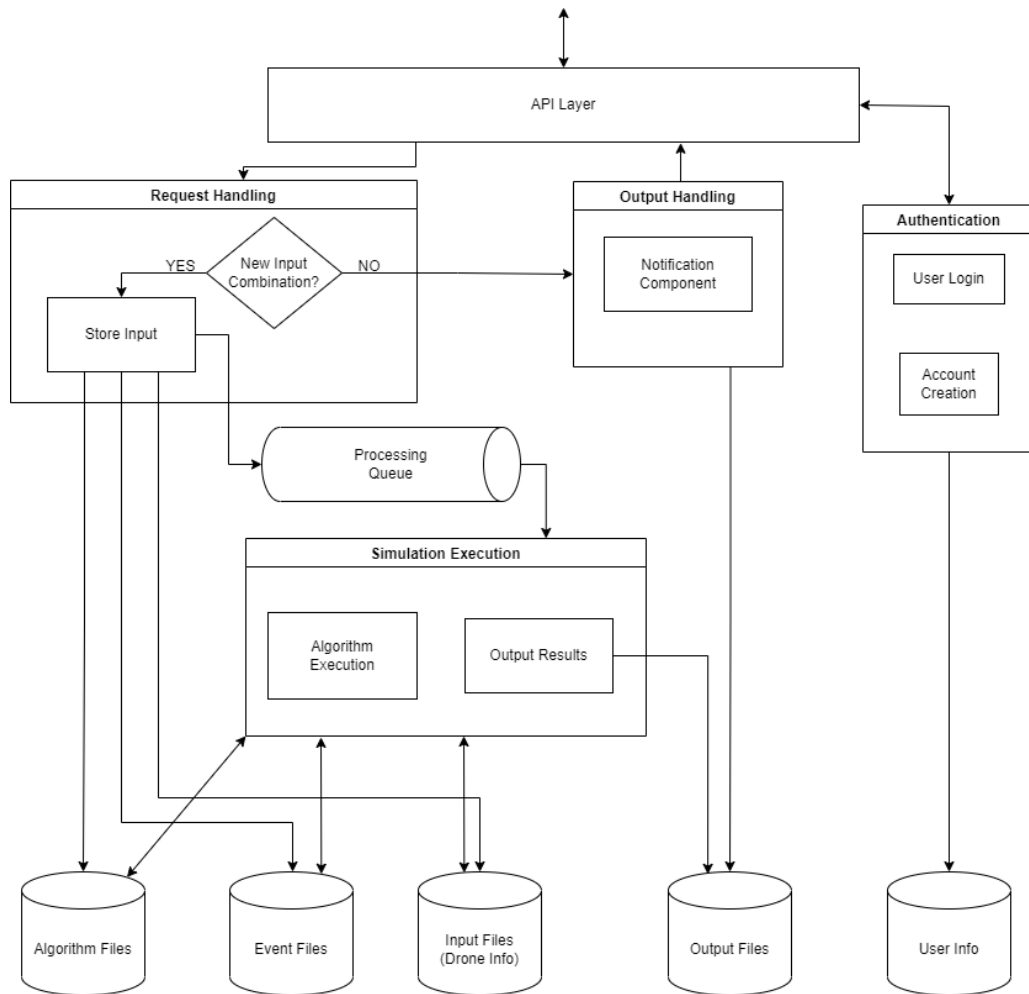


Figure 1.1: System Architecture Diagram

Figure 1.1 depicts our group's original design based on the client's specifications. The reasoning behind our original design choices can be found in our original Design Document in Section 4. This design was overly ambitious and was blind to the challenges we would encounter in the future. Despite our overly optimistic viewpoint when drafting this design, a majority of the design is intact. With the removal of both the notification component and the ability to search for previously run simulations, our design has largely remained the same.

Another aspect of the design that had to be removed is the ability to upload custom Algorithms. Our team thought long and hard about this decision and it wasn't easy to come to, but due to security concerns we agreed that the removal of custom algorithm usage was for the best. More detailed reasoning can be found below in section 1.9.

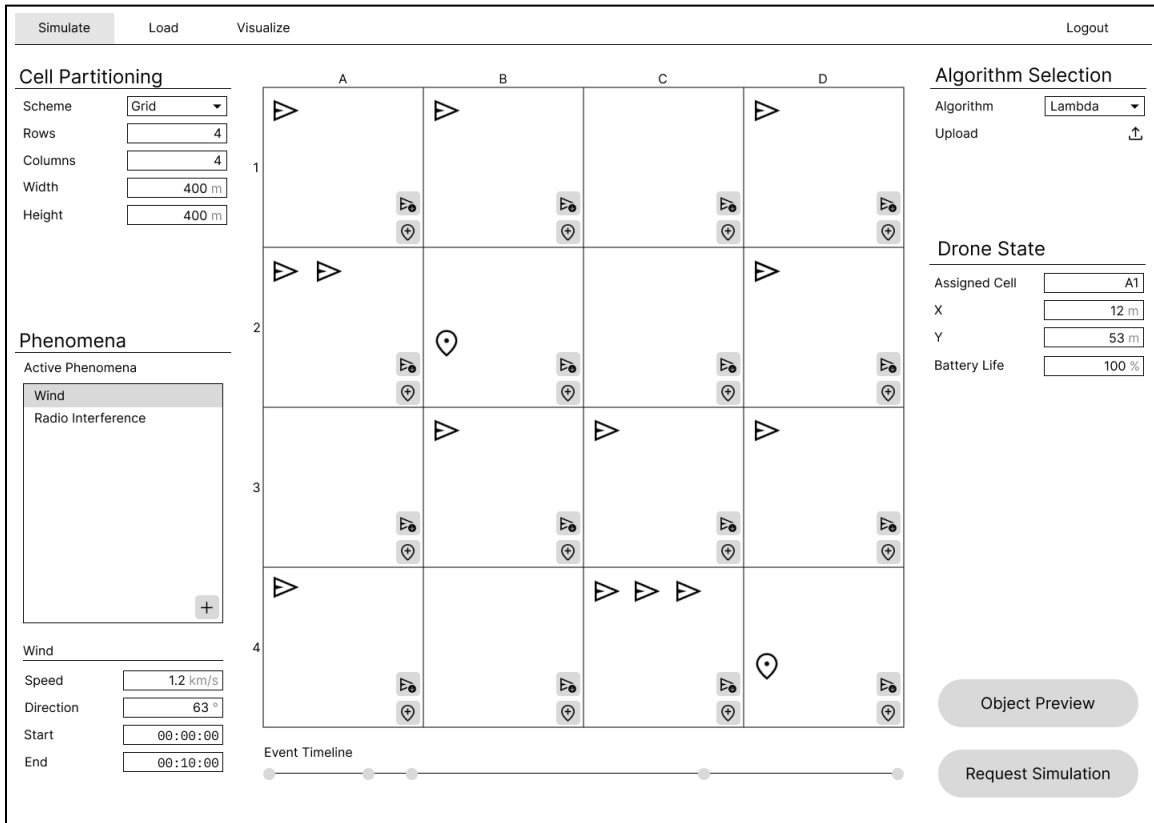


Figure 1.2: Original Simulation Setup Mockup

Displayed above in figure 1.2 is the original design our group had envisioned for simulation setup. This design quickly became outdated once implementation began, and several original requirements became unreasonable. On the sides of our design, you can see the following sections that have since been removed: phenomena and drone state. As aforementioned, these two components became quickly obsolete once work began and it became apparent that it was an unrealistic goal.

Calculating the impact of active phenomena on drone performance and pathfinding was outside our scope, and it required more specifications inside the pathfinding algorithms. Our group agreed that putting too many constraints on algorithm creation lowered the accessibility of our program, which was the most important non functional requirement. Calculating the drone state was unpredictable, and without statistics on the specific drones the user were to be using in the field, it was completely impossible.

With the removal of phenomena and drone state fields, we have condensed the simulation setup page. These changes are discussed below.

## 1.5 NEW DESIGN

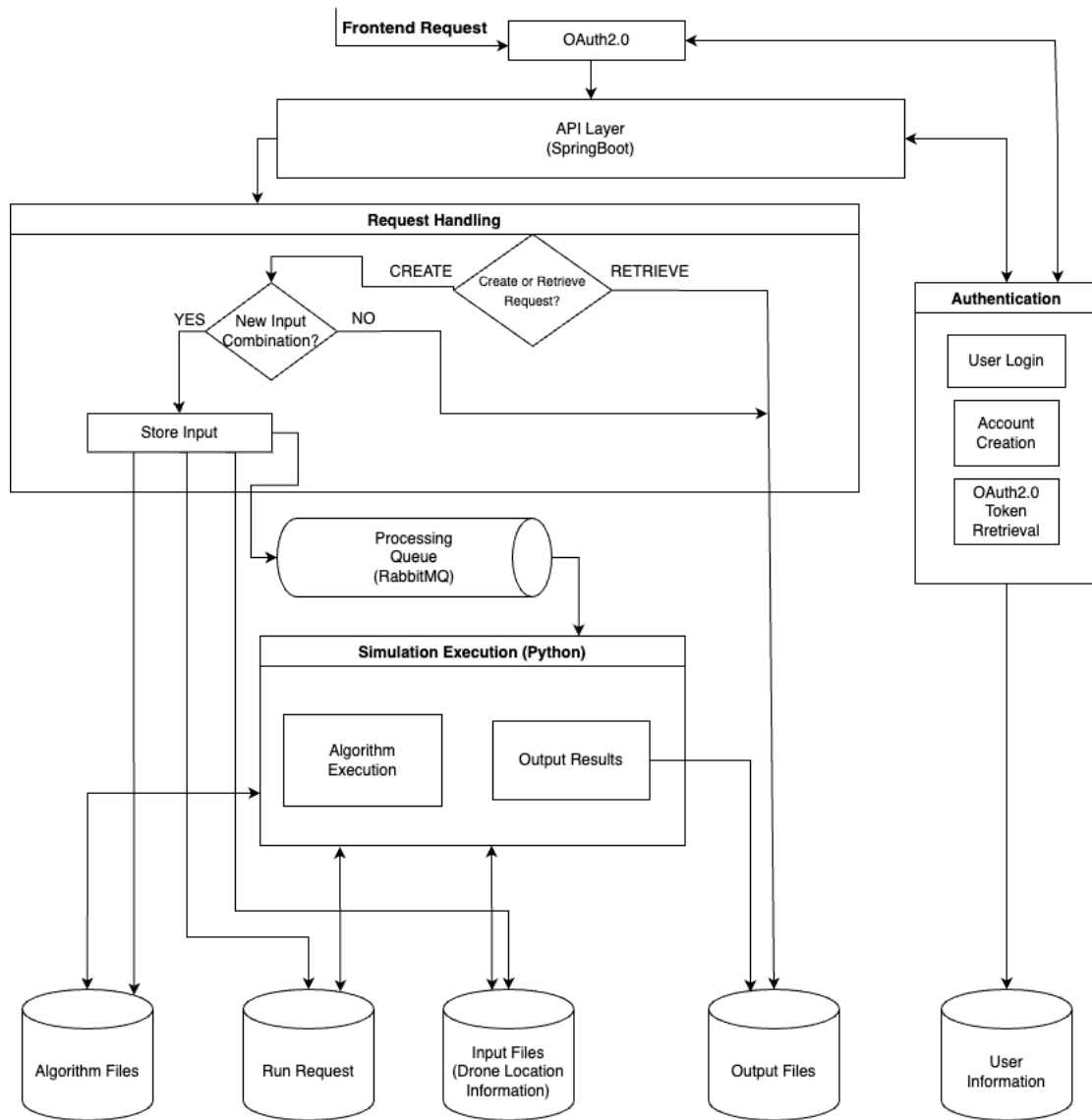


Figure 1.3: Redesigned System Architecture Diagram

Figure 1.3 depicts the new system architecture diagram we've made to illustrate how our backend systems now work. Depicted in this diagram, is the removal of the "Event Files" database and addition of the "Run Request" database. This is due to the limitations we encountered when building the system, and is noted in Section 5 as future work. Algorithm and input files are stored separately and are connected to each other through the "Run Request" database.

In order for our application to be easily accessible to new users, our application provides two different algorithms that are available for use. Uploading custom algorithms makes that algorithm available for the user who uploaded it. Storing the algorithm files separately allowed our group to display this list on the frontend, giving the users of our application more customization when creating a simulation request. Storing input files and algorithm files separately lays the groundwork for future additions. By storing input

files separately it requires minimum modifications to add the ability to search for previously run simulation requests. Storing algorithm files separately will ease the transition between our algorithms and user created ones.

Finally, the most vital component to our application is the “Run Request” database. Run requests are created when a user dispatches a request to simulate drone movement. This is the bridge between the algorithm file being used for the request, and the input file. Our application gives the user the ability to upload drone position data from either a .csv file, or manually. A request uses both the algorithm and input data in order to determine drone positions at specific times. The new entry in the run request database is then sent to our queue for our python service to process.

When a run request is dequeued by the python service, the algorithm is run using the input file data. Once all the calculations have been made by the uploaded algorithm, and information is sent back to our database with the run request is marked completed. Users can then make requests to retrieve the output results.

Along with the backend changes, there have been some changes to the user interface as well. These changes, the reasoning behind these changes, and pictures of our website can be seen below.

|    | x | y |
|----|---|---|
| 1  |   |   |
| 2  |   |   |
| 3  |   |   |
| 4  |   |   |
| 5  |   |   |
| 6  |   |   |
| 7  |   |   |
| 8  |   |   |
| 9  |   |   |
| 10 |   |   |

Figure 1.4: Redesigned Simulation Setup

Our group thought it would be best to simplify the view for our users. Our driving factor for this design is simplicity. With the removal of phenomena, our group consolidated our simulation setup page in order to boost accessibility. Below you can read about the functionality of each component:

- **Grid Size:** Grid size is responsible for changing the size of the environment for the simulation. In this section a user is able to customize the number of rows and columns used for simulating drone movement. Since our application aims to appeal to a broad audience, it was important to our group to introduce flexibility to our system to allow for unique experiences for all of our users.
- **Partition:** In partition, the user may provide the number of partitions the algorithm should use. The effect of the partitions is dependent on the algorithm, for example some algorithms may assign a drone to each partition.



- **Algorithm:** In the algorithm section, users need to upload either our algorithms Naive.py or CPH.py. While we wished more flexibility was allowed with this component, unfortunately due to security concerns we couldn't allow custom algorithm execution on our servers.
- **Event Data:** Event data is the most essential aspect of the simulation. Here users can select the simulation duration. Users can then scroll through the time and choose to either manually input the locations of drones at the specified time, or utilize a .csv file to load their drone's location automatically.

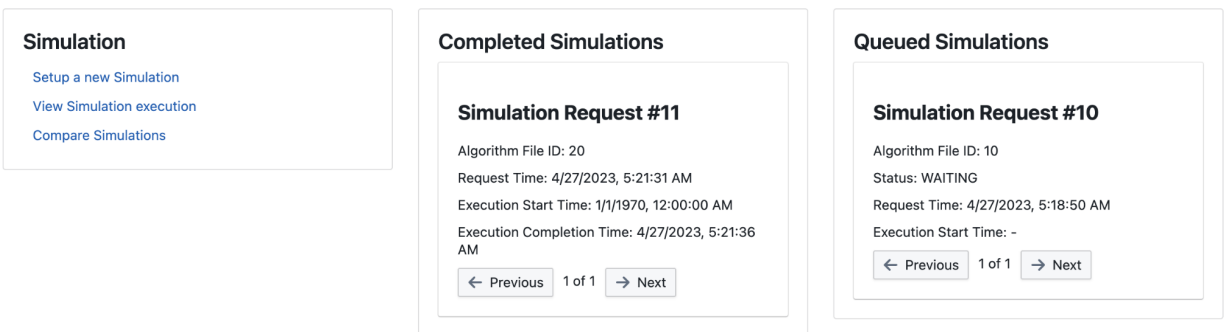


Figure 1.5: Dashboard Page

Our dashboard is a new addition to our original design. Here users can see both their completed simulations along with simulations still waiting in the queue. Each simulation is given a unique number to identify them, and users can use these numbers to pull up their simulation in both the "Compare a Simulation" section of our application, and visualizer under the "View Simulation Execution".

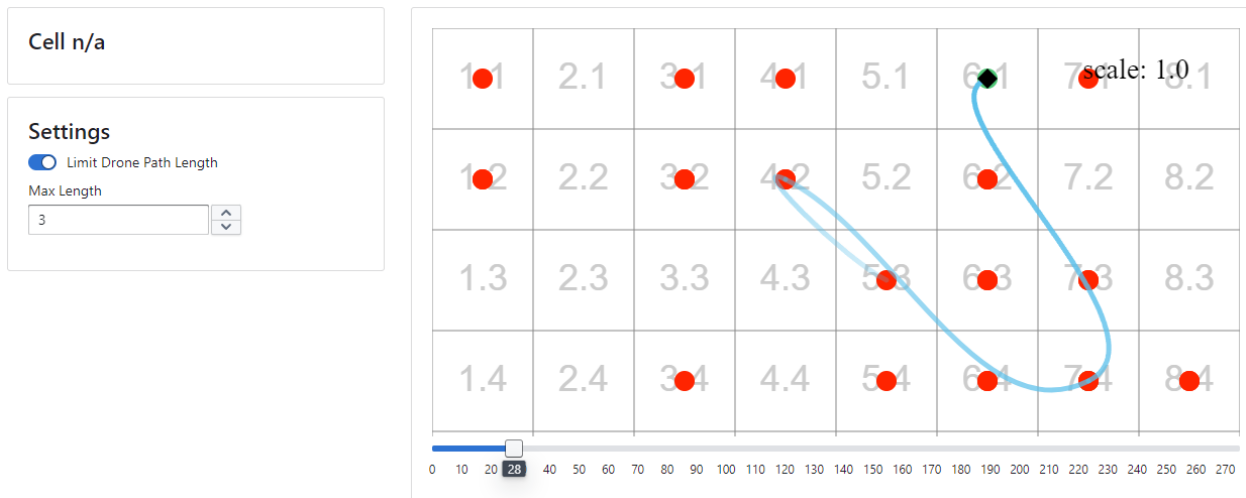


Figure 1.6: Redesigned Simulation Visualization

The grid page provides a visualization of the drone’s response to events over time. The visualization supports both zoom and pan controls and allows the user to step through time increments of the simulation. This allows the user to view which events are live at a given time (pictured in red), which events have been recently responded to (pictured in green), the location of the drone (pictured as a black diamond), and the path the drone has followed thus far. For clarity, the user can also limit the number of paths that appear on the screen, as to not cover the visualization.

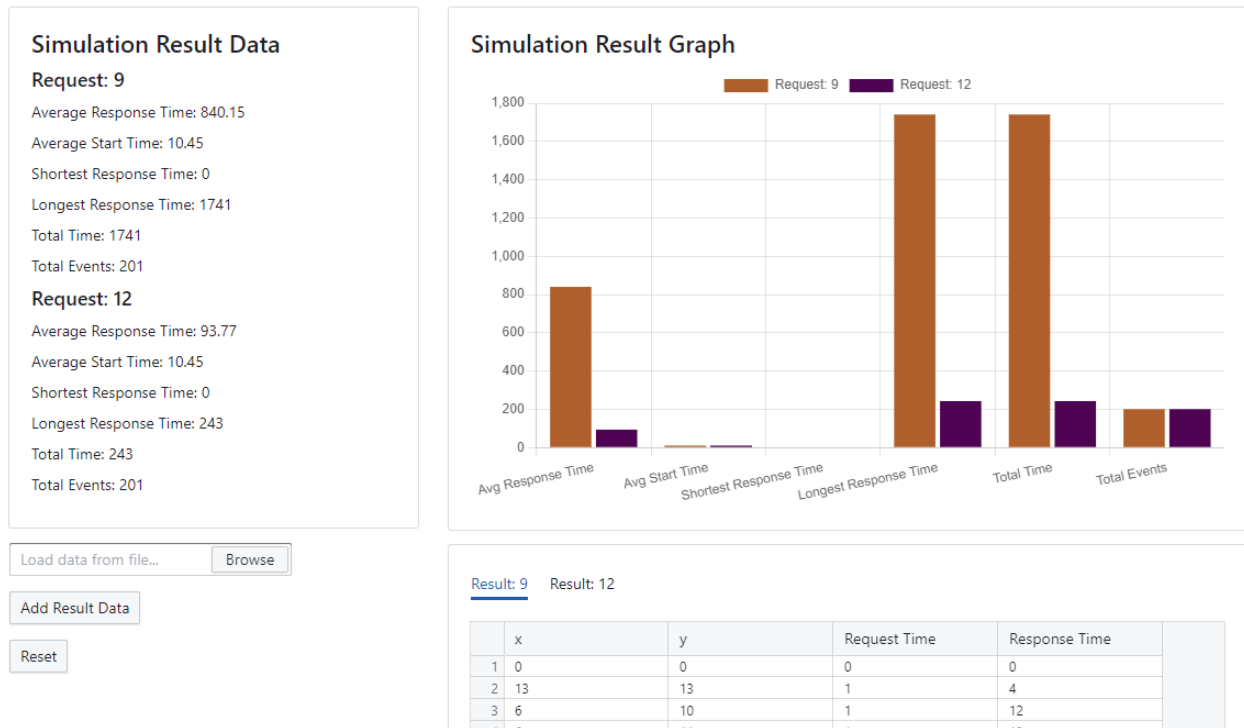


Figure 1.7: Simulation Comparison Page

The comparison page summarizes the results of each simulation and displays a visual comparison between them using a bar graph. The user may display and compare any number of simulations. Six metrics for each result are displayed: average, min, and max response time; average start time; total time; and total events. Additionally, data corresponding to each event is provided in the table below.

## 1.6 REQUIREMENTS

Functional Requirements:

- The software allows the user to login to a personal account profile.
- The software allows the user to choose which drone algorithm(s) to apply to the drone flight simulation.
- The software accepts simulation setup configuration through selected files.
- The software provides a visualization of the drone flight using a 2d grid of the geographical area.
- The software saves previously run simulations for input combinations that will be accessible for other users.
- Simulation requests are placed into a queue ensuring that every simulation is ran.

- The software stores the simulation output in a format containing: starting location, starting time, destination location, arrival time.

Non-Functional Requirements:

- The software is easy to use and understandable since not all of our users have a technical background
- UI elements of the software are intuitive and clearly labeled or documented.
- The software handles errors in the input gracefully.
- The software is compatible with Windows, MacOS, and Linux.
- The software was developed in a manner that is supportable & maintainable after our team leaves.

## 1.7 STANDARDS

We now list all engineering standards we identified last semester which are applicable to our project. Here in table 1.1 the IEEE engineering standards that were applicable to our project. These are the standards that we kept true to throughout development ensuring that our group was following best practices, with the needs of our client in mind.

| Engineering Standards   | Reasoning  |
|---|--|
| Scrum Methodology   | We utilized a structure on how we developed the project working in 2 week sprints.             |
| IEEE Standard 1063: Standard for Software User                              | 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          | We ensured that our group used language similar to our peers.                                  |
| Software Engineering Code of Ethics and Professional Practice (Principle 3) | Our project follows a professional structure making it easy to understand, and open for change |
| Software Engineering Code of Ethics and Professional Practice (Principle 2) | We've met the expectations of our client as well as the potential users of our product.        |

Table 1.1: Engineering Standards Applications

## 1.8 CONSTRAINTS

Upon being approached by our advisor about the initial project information, we had a large amount of flexibility when designing our project. There was only one constraint set that our group was responsible to follow. We had to ensure development costs weren't more than \$300, which we can proudly say was never

a problem. Using Iowa State's servers to host our application we were able to develop our project at no cost. Since that discussion our group outlined more constraints we were able to hold our project accountable for. These constraints are listed below:

- 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.
- User data will be kept minimal to ensure the least amount of personal information will be stored

## 1.9 SECURITY CONCERNS AND COUNTERMEASURES

Despite security not being implicitly stated in our project description, we felt responsible 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.

The web application is being hosted on Iowa State's servers which adds a layer of protection from outside attacks utilizing Iowa State's network security. We've also implemented another countermeasure which was not outlined in our previous design document. Adding Token Authentication to our API layer ensured bad actors can't maliciously manipulate the system to access data they don't own. This addition to our web security secures user data, and is another defense against cyber attacks overloading our backend services.

Our application's goal was to allow users to compare and contrast different algorithms and their respective run times. This meant users being able to upload and test their algorithms on our servers. Unfortunately this feature was a huge security risk, and had to be removed from the original design. Allowing user created programs to run with file system access was a risk that hadn't been properly looked at until this semester, so this functionality couldn't be implemented in a safe manner in time.

## 2 Implementation

The implementation of this project was challenging yet rewarding. Our group was tasked with finding a way to handle user input and for it to be stored in a responsible manner. Listed below is a look inside our implementation, with an in depth look into both our client and server side implementation.

### 2.1 IMPLEMENTATION METHODOLOGY

Our group spent the whole first semester devising a plan to properly implement the outlined architecture. This comes from months of research to not only fully understand the problem, but the technology we will be using too.

Throughout the semester our group adopted an agile-scrum schedule. Weekly meetings were held on Mondays where the group spent time demoing work, speaking on the challenges we were facing, and work moving forward. Utilizing an agile approach allowed our group to quickly progress without falling behind. Members of our group were able to overcome challenges when they would arise, while others would continue development of another feature of our application. This ensured continuous progress was made.

We made it a priority to try and meet with our advisor bimonthly to demo the progress we had made, and discuss any challenges we would run into. By having these discussions our group was able to maximize the little time we had for this project. If we were to run into problems we couldn't solve, we could continue development of a different feature and backlog the issue until we had the opportunity to bring it up with our advisor.

### 2.2 SERVER SIDE IMPLEMENTATION

Our group's server side logical implementation design can be seen in Figure 1.3, which gives a high level look at our implementation. In this section we will go in depth about each key component that allows our API to work fluidly and quickly, and some of the technology that was responsible for making it all come together so quickly.

Our Java backend was responsible for distributing any API calls made. Our group considered several options, but inevitably opted to use Spring Boot. Spring Boot is an open source tool that enabled our group to make microservices using Java. Previous group experience with the framework and the flexibility of Spring Boot made using this framework an easy decision. More information regarding Spring Boot can be found in our references.

The spring boot side of things was responsible for storing all data in our database, the key pieces of data we were storing was the RunRequests, AlgorithmFiles, and InputData. Each user has one to many RunRequests, algorithms files, and input data associated with their account. We implemented it this way to allow users to re-use input and algorithm files they had used in the past. As of now the only algorithm files that work while uploading are the Naive and CPH algorithms. The framework is there to allow the uploading of user created algorithms, but we would still need to specify input and output requirements for the user created files. To access the information we are storing we defined API calls which allowed us to

display user data throughout the website. For each of these API calls the user needs to provide a temporary security token for the request which we then use to authenticate the user to prevent accidental release of information. This security token gets created upon Login, or User Registration. The most important request that was implemented was the create run request endpoint. Which pulls a user uploaded algorithm, user uploaded input data, and creates a new run request. The endpoint then takes the request and pushes it into our RabbitMQ queue.

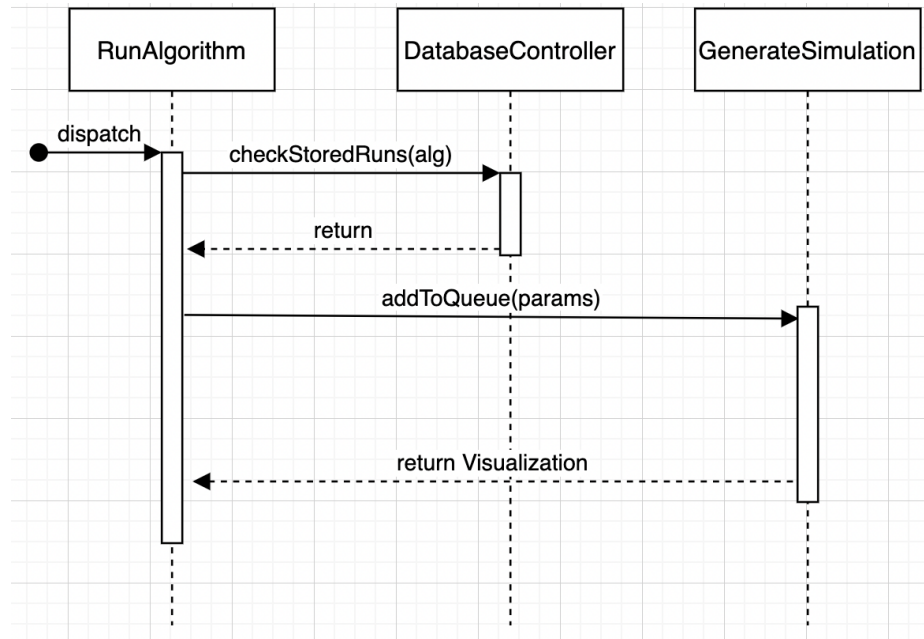


Figure 2.1: Run Requests Sequence Diagram

When run requests are present in the queue, the listening Python service pulls down the run request ID and uses it to query the database for the remaining run request data. At this time, the Run Request's status is also updated to IN\_PROGRESS, and a timestamp for the algorithm's beginning is added. The queried request data is used to load the input file and algorithm information. The algorithm information is used to determine which of the provided algorithms (CPH or Naive) should be used to run the simulation. We originally planned for users to be able to upload custom algorithms, but this would involve running custom user uploaded code on our server, which poses a major security risk. Instead, requests using a CPH algorithm will use our CPH algorithm, while other requests use our Naive algorithm. Additional user customization remains a stretch goal for the future, perhaps through the use of specific parameters rather than user Python files.

Once the algorithm is determined, the input file data consisting of grid size, partition count, and event information is passed into the algorithm. The selected algorithm executes with the specific inputs, and saves the drone movement data to a CSV file. This data is then used to calculate the particular drone path information, which is also saved to a CSV. Finally, both CSV file paths are saved to the Run Request in the database, along with a COMPLETED status update and completion timestamp.

## 2.3 CLIENT SIDE IMPLEMENTATION

The client side application provides an interface for the user to request, view, and manage drone simulations. The application was created in the Vite development environment. It is implemented as a multi-page React application, with each page being responsible for different functionality of the application. These pages include:

- Login/Register, login or register a user
- Dashboard, view pending and completed simulations, and navigate to other parts of the application
- Request, create and run a new simulation
- View, compare results of simulations
- Grid, view simulation execution

The application uses server-side routing to navigate the user to requested pages, using vite-plugin-ssr. For each page, the interfaces were built using React Components and uses the component library Blueprint.js for UI elements where applicable. These components also make API calls to the backend to send and retrieve up to date information.

In addition to modularizing code based on page, the frontend application also defines Layouts and shared Components. Layouts provide uniform structure to the page (such as determining placement of content and the navigation bar), and determines how the application should be rendered at different screen sizes. Components which are common across multiple pages are also placed into a separate folder.

The Visualization Grid uses an HTML canvas. This is wrapped in a React component which is only reinitialized when a new file is displayed to the grid. The grid stores information such as translation and scale internally, which support panning and zooming. To update the visualization, the grid makes use of its own draw loop, allowing for more efficiency and responsiveness, bypassing React overhead.

## 3 Testing

Testing for the project was split up into two groups, the front end and the backend. Front-end testing was done through manual testing. Backend testing was completed with the use of JUnit, Postman, and Mockito; using predetermined inputs and comparing it to our API's output.

### 3.1 UNIT TESTING

There are a handful of aspects in our finalized design that required unit testing to ensure correct functionality of all systems. Unit testing is essential to scale functionality of our project. Our team has been testing in parallel with new code development in accordance with agile methodologies. Listed below are the areas in which we used unit testing for the backend:

- All backend service functionality that didn't face the users has had tests created for them. These tests ensured that each respective repository returned the correct information, errors in input were handled gracefully, and backend errors were treated correctly
- Our frontend system needed to function properly and translate the proper data to the server for drone visualization. In order to ensure correct functionality our group tested that:
  - parameters are passed as selected by the client
  - the account responsible for creating the request is passed within the parameters
- Our user registration system has been tested in order to be certain that duplicate users cannot be created.
- Simulations queued for input, with incorrectly formatted parameters are thrown out and discarded. These tests ensured that incorrectly formatted requests don't halt the queue from moving forward.

Unit testing was vital in order to provide a functioning and reliable system. To provide a scalable system our group has achieved near perfect line and case coverage of our backend system. Test input was small in comparison when compared to data being computed in production. Our group used small sample sizes when testing, helping us predict with higher accuracy when unit testing. In order to achieve full backend unit testing our group used both JUnit and Mockito.

### 3.2 INTERFACE TESTING

Interface testing was performed on utilizing our backend through the Postman application and manual testing. Postman was used to ensure the correct data would be sent back to the frontend and manual testing was utilized throughout the semester. Every week during demos when our group came together we ensured all possible types of inputs could be handled.



### 3.3 PYTHON BACKEND TESTING

The Python portion of our backend system was unique in that the provided algorithms were to be used as-is, with only minor adjustments to ensure proper handling of input parameters and result output. Therefore, testing was performed to ensure required functionality was implemented correctly while the provided algorithms maintained their original accuracy.

To this end, a multitude of scenarios were tested to enshrine the following behaviors:

- Run Requests are received from the queue in the proper order in a timely manner
- The correct Run Request information is retrieved from the database
- The correct input and algorithm information is retrieved from the database and file system
- Necessary input parameters are passed in and used during execution of the algorithm
- Result output is saved to the filesystem, with the correct file path returned
- Drone path information is properly calculated using the results
- File path, status, and timestamp information is reflected in the database each time updates occur

Correct algorithm execution was measured at all steps. If execution output changes for a given input after a code change occurs, the code change is reverted and corrected. Through these methods, we were able to test for and ensure our system functions according to our selected specification.

### 3.4 ACCEPTANCE TESTING

Acceptance testing was performed on a weekly basis at our weekly standups. During these meetings team members would demo work done throughout the week, and would be followed up with a discussion. Weekly meetings allowed our group to progress, and adjust to new challenges that would arise. Throughout the semester we also held monthly meetings with our advisor Professor Goce Trajcevski, who checked our progress and helped clarify our steps moving forward.

The primary purpose of testing is to show that our functional and nonfunctional requirements are being met. Below is a list of our functional and nonfunctional requirements along with the acceptance criteria our group has determined:

#### **Functional Requirements:**

- 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 utilizes a 2d grid layout of the geographical area. There are identifiable icons for the drones and the events across a timeline of running.**
- The software shall save previously run simulations for input combinations that will be accessible for other users.
  - **Acceptance criteria: Output files are stored in a database.**

- Once the user has selected an input combination for simulation then the software shall queue the simulation to be run on the backend.
  - **Acceptance criteria: Any queued simulation results in another output simulation file in the database.**
- 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 are 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 has a starting location, starting time, destination location, and arrival time.**

#### 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, is able to run a simulation and view the results within 10 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 is able to correctly identify the purpose of each UI element (buttons, drop-downs, etc), within 10 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, a message window displays a warning message about the selected file being invalid, and a short description for the reason behind the error.**
- The software shall be compatible with Windows, MacOS, and Linux.
  - **Acceptance criteria: A user on all 3 types of machines is 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, uses self describing variable names. The software has been developed in a manner that allows for modularity and the possibility of adding additional features.**

## 4 Work Context

Drones use is becoming increasingly more prevalent. In order to ensure public safety isn't at risk it is imperative to test these drones to make certain flight paths are correct. Our group's project aims to eliminate these expensive testing fees, by virtualizing the environment. Virtual drone simulation programs exist, but our project when compared to those below don't solve the same issue. Below is a list of these programs that inspired the creation of this project.

### 4.1 RELATED PROJECTS

Below is a list of related projects. Here we discuss the differences between these projects and our own.

#### 4.1.1 MICROSOFT AIRSIM

Microsoft AirSim is an open source simulation platform for AI research and experimentation. With the ability to not only simulate drone movement, it's possible to simulate cars and more. The application itself is designed to test deep learning algorithms, and expand the abilities of self autonomous drones and cars. It's a very powerful application that already a lot has come from. This is a downloadable file that can be run in either Unity or Unreal Engine.

The main difference between our application is that it doesn't aim to improve the understanding of deep learning algorithms. Primarily from a surveillance perspective, it's important to understand the paths the drones are taking, and the fastest way to maneuver while monitoring the most ground. The goals of each project are different. While Microsoft's AirSim aims to enhance the understanding of AI systems, our application is a tool to visualize drone pathing.

<https://microsoft.github.io/AirSim>

#### 4.1.2 FLIGHTREADER

Flightreader is a subscription based service that allows users to read and visualize drone path data from flight logs. Flightreader allows its users to look at in depth statistics about specific flights their drone has flown, these statistics include: battery life, velocity, altitude, flight mode, and more. Flightreader is a tool used for individual drone analysis.

Our project is focused around the movement of a fleet of drones, and autonomous drone movement. Flightreader is a drone path visualizer for a singular drone, while our project focuses on the interactions between drones. These applications are similar, but just like Microsoft AirSim both aim to accomplish different goals.

<https://www.flightreader.com/>

## 4.2 RELATED LITERATURE

Below is a list of related literature that helped inspire the creation of this project. Here we discuss the importance of the paper to our project.

### 4.2.1 TOWARDS THE INTERNET OF FLYING ROBOTS: A SURVEY

*Towards The Internet of Flying Robots: A Survey* is a research report written by Hailong Huang and Andrey Savkin, and discusses the importance of drone surveillance in the future. The authors speak on the importance of these systems development, but that little to no progress has been made in this field. Upon the completion of their survey, they found a concerning lack of resources being put to flying robot navigation.

Our project aims to help in the efforts towards a more open and accessible world for flying drones. Without this paper it's possible the idea for our project's might have never come to fruition. By providing a web application allowing users to simulate drone movement, our group hopes that our project can lay the groundwork for more development in this field. With the flexibility and ease of use our application provides, we hope that it can open the eyes of our audience to the possibilities of drone usage in the future.

<https://doi.org/10.3390/s18114038>

## 5 Conclusion

Over the past year our group has designed and implemented a website that allows users to create a simulation of a fleet of drones. At the project's current state users are able to:

- Access the application from any device with a web browser
- Simulate drone flight paths from a selection of provided drone movement algorithms
- View a visualization of the drones' paths
- View saved run information at any point in the future, and make edits to past runs to create new ones
- Compare results of different simulations side-by-side

However the project has room for improvement; listed below are the next steps for our project:

- Further customization for custom algorithms
  - Users should have more flexibility with input and output specifications
  - Potential options:
    - Run user-uploaded code in sandboxed environment (to alleviate security concerns)
    - Allow users to "build" algorithms according to specific parameters using a custom interface
- Email system to notify users of long queue times, and when simulations are completed
- Hashing of sensitive information

Our group is grateful to have had the opportunity to work on this project. Working alongside Professor Goce Trajcevski's guidance we've accomplished more than we could have originally anticipated. Our group is proud to display our finished product, and pleased with the finished product we were able to complete. All members of our team have worked very hard to make this project a reality, and we appreciate the opportunity to contribute to this area of research.

## 6 References

- [1] - "Welcome to AirSim." *Home - AirSim*, Microsoft, <https://microsoft.github.io/AirSim>.
- [2] - "DJI Drone Flight Log Viewer: Flight Reader." *DJI Drone Flight Log Viewer | Flight Reader*, FlightReader, <https://www.flightreader.com/>.
- [3] - "Spring Boot Home." *Spring Boot*, Spring, <https://spring.io/>.
- [4] - 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>

## Appendix A - Operation Manual

Below is an in depth operation manual for our application. Using this guide should clear any confusion about our group's application. Following through with this operation manual you should be able to: create an account, request your first simulation, visualize the simulation, and learn how to compare simulations.

### A.1 SIGNING UP AND LOGGING IN

Using the link above you will be greeted with our project's landing page. On the left side of your screen you will see a "Login" button as displayed below.



Figure B.1: Landing Page

Click Login and register a new user with our system. The application will ask for a name, email address, and password. Upon registration you will be greeted with your personal dashboard.

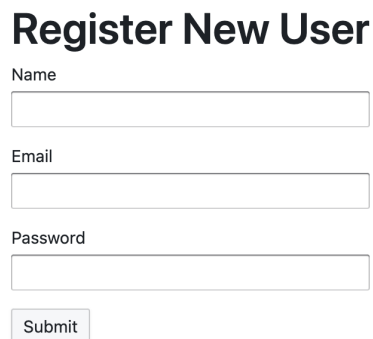
The image shows a registration form titled 'Register New User'. It includes three input fields labeled 'Name', 'Email', and 'Password', and a 'Submit' button at the bottom.

Figure B.2: Account Registration Page

## A.2 DASHBOARD

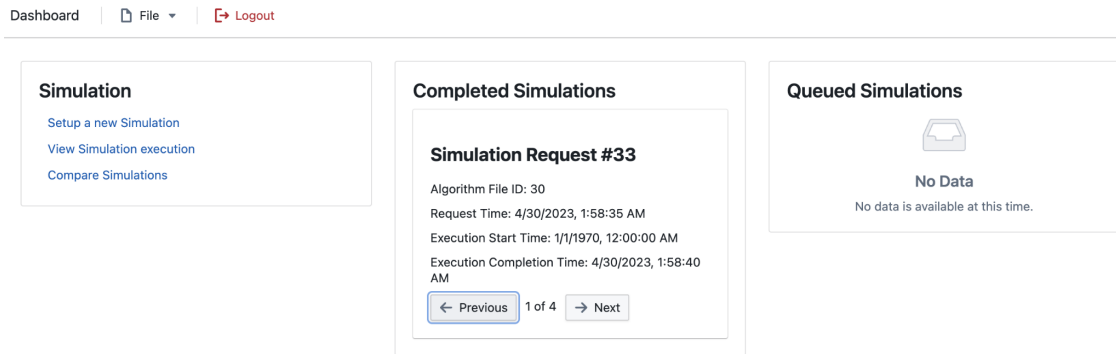


Figure B.3: Account Dashboard

Welcome to the dashboard! Here you can see a list of completed and queued simulations your account was responsible for queueing. On the left side of your screen you will see the “Simulation” header that contains three links. Here you can:

- **Setup a new Simulation:** Here on this page you will be requesting your simulation. This page can be challenging to navigate your first time, so we recommend that you follow our guide in section A.3
- **View Simulation Execution:** Viewing simulation execution allows you to watch the simulation as you move time forward. More information about this can be found in section A.4
- **Compare Simulations:** Here you can load up previously run simulations and compare miscellaneous information such as run times, response times, event count, and more. More information about this can be found in section A.5

Your account won't have anything inside “Queued Simulations” nor “Completed Simulations”. These headers contain basic information about simulations that have either been queued to run, or have already finished being visualized.

## A.3 RUNNING YOUR FIRST SIMULATION

Welcome to the simulation setup page! This page can be challenging to maneuver your first time. Listed below are the necessary fields for you to enter, and how they change the simulation:

- **Grid Size:** Grid size is responsible for changing the size of the environment for the simulation. In this section a user is able to customize the number of rows and columns used for simulating drone movement. Since our application aims to appeal to a broad audience, it was important to our group to introduce flexibility to our system to allow for unique experiences for all of our users.

- **Partition:** In partition, the user may provide the number of partitions the algorithm should use. The effect of the partitions is dependent on the algorithm, for example some algorithms may assign a drone to each partition.
- **Algorithm:** In the algorithm section, users need to upload either our algorithms Naive.py or CPH.py.
- **Event Data:** Event data is the most essential aspect of the simulation. Here users can select the simulation duration. Users can then scroll through the time and choose to either manually input the locations of drones at the specified time, or utilize a .csv file to load their drone's location automatically.

**Grid Size**

Rows (required)

Columns (required)

**Partition**

Number of Partitions

**Algorithm**

CPH\_alg\_file.py

**Event Data**

Automatically Generate Data

Simulation Duration (required)

Current Time

|    | x | y |
|----|---|---|
| 1  |   |   |
| 2  |   |   |
| 3  |   |   |
| 4  |   |   |
| 5  |   |   |
| 6  |   |   |
| 7  |   |   |
| 8  |   |   |
| 9  |   |   |
| 10 |   |   |

Figure B.4: Simulation Setup

Once all of the fields are in place, you can now click request simulation and move on to viewing it run in real time, or comparing the simulation data to another request!

## A.4 VISUALIZING YOUR SIMULATIONS

Here on the simulation visualization page you will be greeted with a blank canvas. Once you have a completed simulation you can load it and watch as your fleet of drones begin to move as you move the time bar on the bottom of the screen. At the top left of your screen you will see a dropdown menu labeled “File”, and then click “Open”:



Figure B.5: File Drop Down Menu

Once you have clicked this button, you will see a list of files uploaded to our system. An example can be seen below.



| Files                 |            |
|-----------------------|------------|
| Algorithms            |            |
| CPH_alg_file.py       | 2023-04-29 |
| OLD_Naive_alg_file.py | 2023-04-29 |
| Naive_alg_file.py     | 2023-04-29 |
| Requests              |            |
| Request: 30           | 2023-04-29 |
| Request: 31           | 2023-04-29 |
| Request: 32           | 2023-04-29 |
| Request: 33           | 2023-04-29 |
| Results               |            |

Figure B.6: Uploaded Files Page

Click on the request you wish to load, and by clicking “Open” your fleet of drones will be visualized. Red dots represent future drone locations and each unique colored line represents a specific drone’s movement. By default the drone path’s length isn’t limited so turn on that option on the left if the drone’s paths begin to clutter your screen. Moving the time bar on the bottom you will begin to see drones moving from cell to cell in the order that the algorithm decided upon. Below is an example of the visualization page:

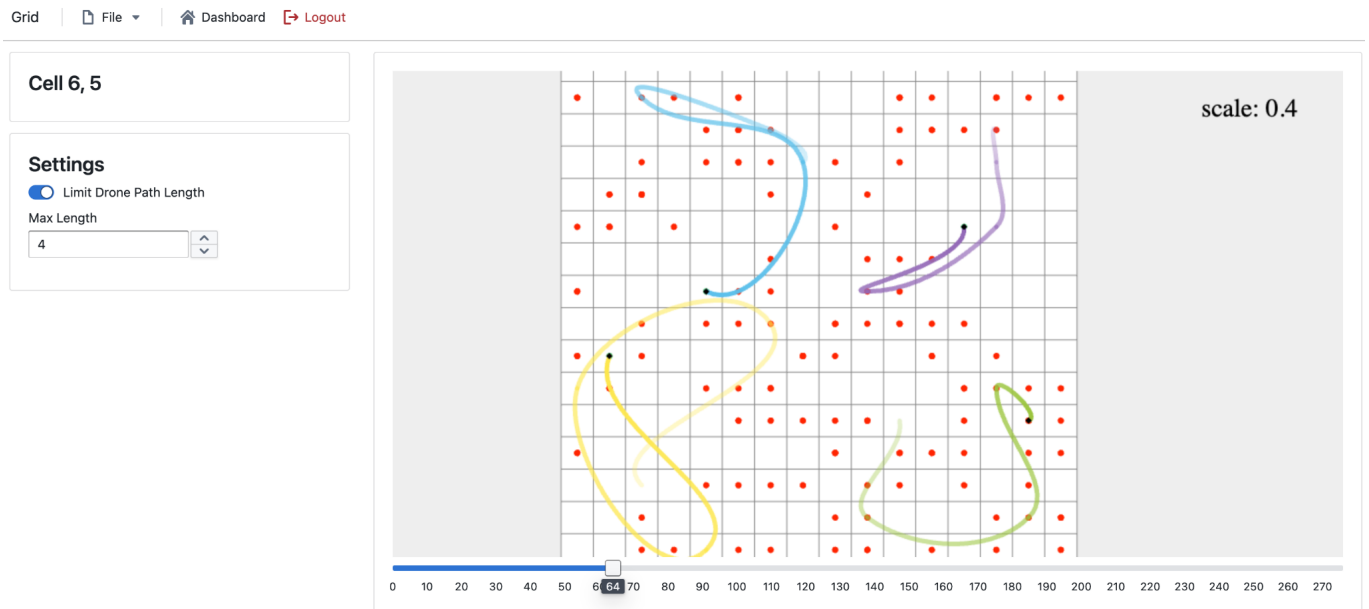


Figure B.7: Drone Visualization Page

## A.5 COMPARING SIMULATIONS

Here on the compare simulations page you will be greeted with a blank graph. Once you have a completed simulation you can load it in and compare miscellaneous stats about the algorithm's execution. At the top left of your screen you will see a dropdown menu labeled “File”, and then click “Open”:



Figure B.8: File Drop Down Menu

Once you have clicked this button, you will see a list of files uploaded to our system. An example can be seen below.

| Files                 |            |
|-----------------------|------------|
| Algorithms            |            |
| CPH_alg_file.py       | 2023-04-29 |
| OLD_Naive_alg_file.py | 2023-04-29 |
| Naive_alg_file.py     | 2023-04-29 |
| Requests              |            |
| Request: 30           | 2023-04-29 |
| Request: 31           | 2023-04-29 |
| Request: 32           | 2023-04-29 |
| Request: 33           | 2023-04-29 |
| Results               |            |

Figure B.9: Uploaded Files Page

Open two different simulation requests to compare simulation data such as response times, number of events, and total time. As seen in the picture below you can also manually input output files that our system creates. If you have your own drone simulation output file you can compare that data with our system's as well.

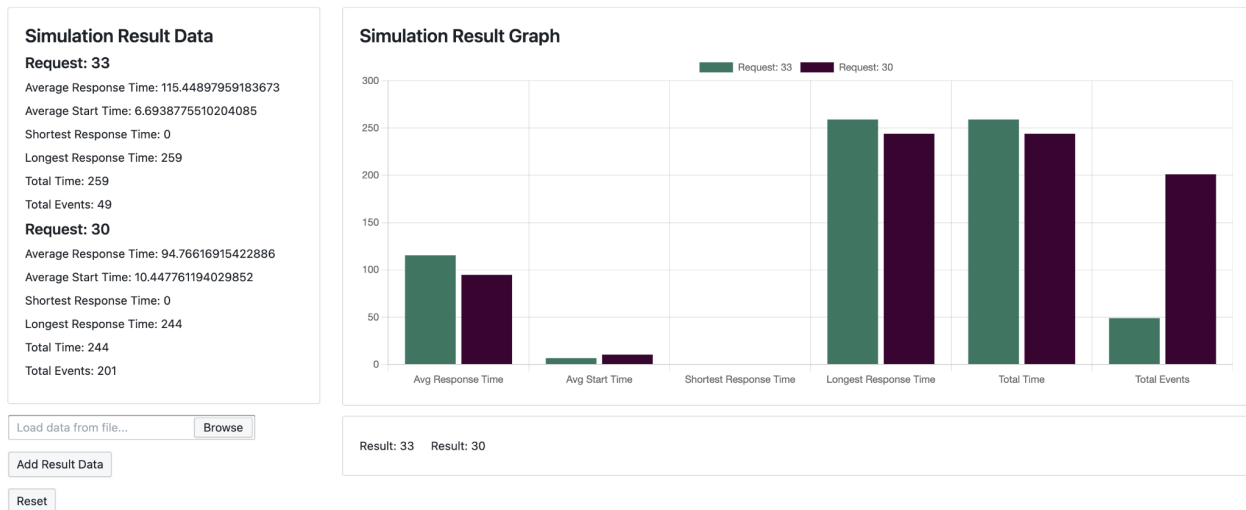


Figure B.6: Run Request Execution Comparison Page

## Appendix B - Original Design Considerations

All contents inside Appendix B is directly taken from our design document our group made in 491. While in section 1.4 we gave an overview of the original design, and our decisions to deviate from it, this section contains the original design and reasoning.

### B.1 ORIGINAL DESIGN 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)

### B.2 ORIGINAL BACKEND DESIGN

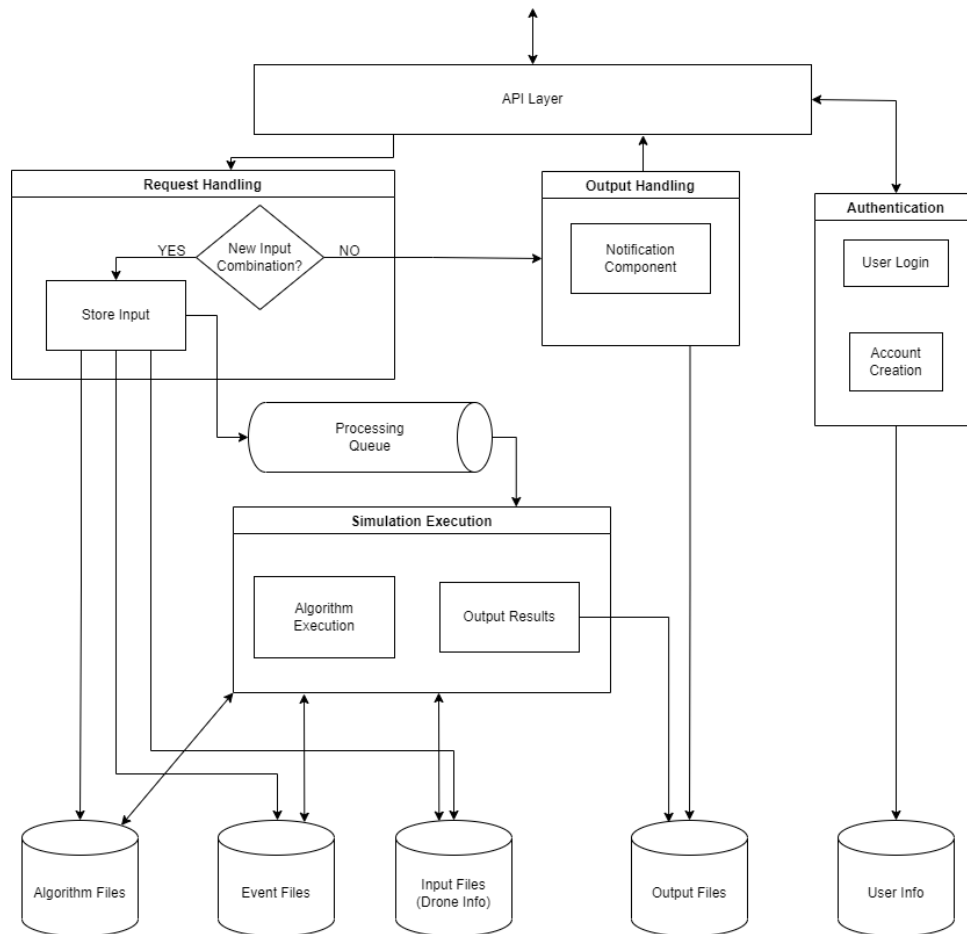


Figure B.1: Original Architecture Diagram

## Backend Sub-systems

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

## B.3 ORIGINAL FRONTEND DESIGN

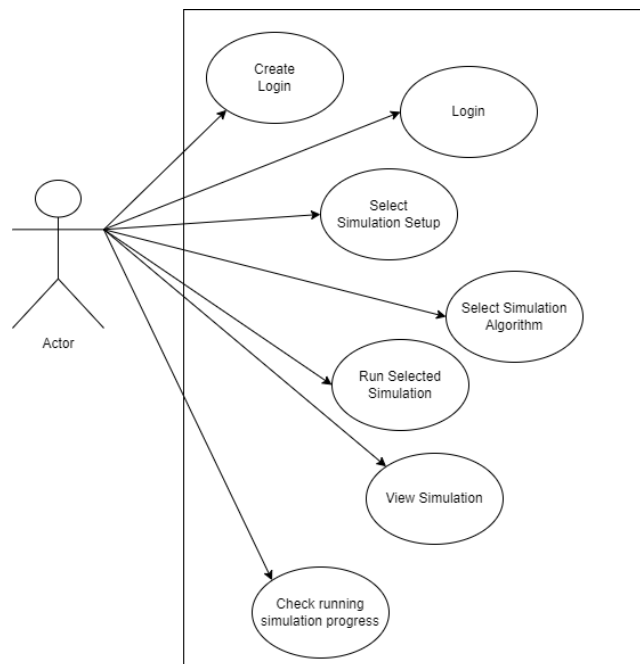


Figure B.2: Use Case Diagram

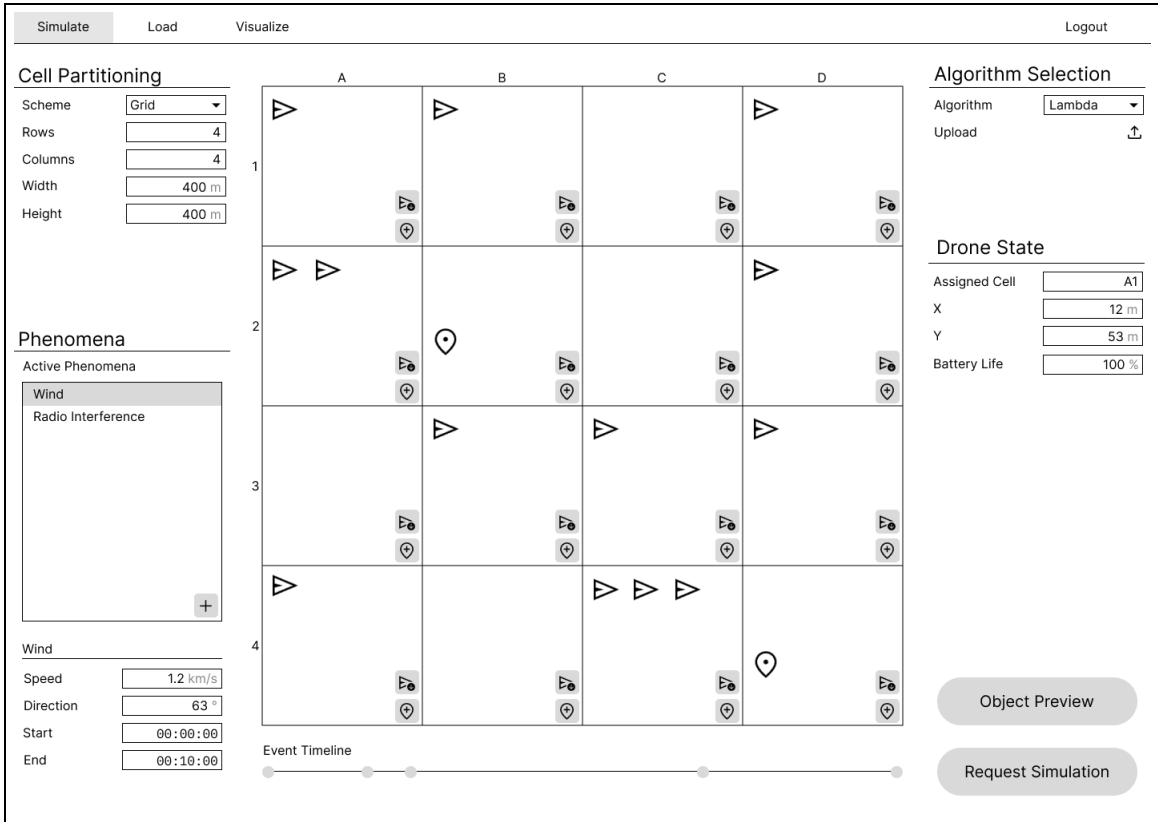


Figure B.3: Drone Simulation Screen

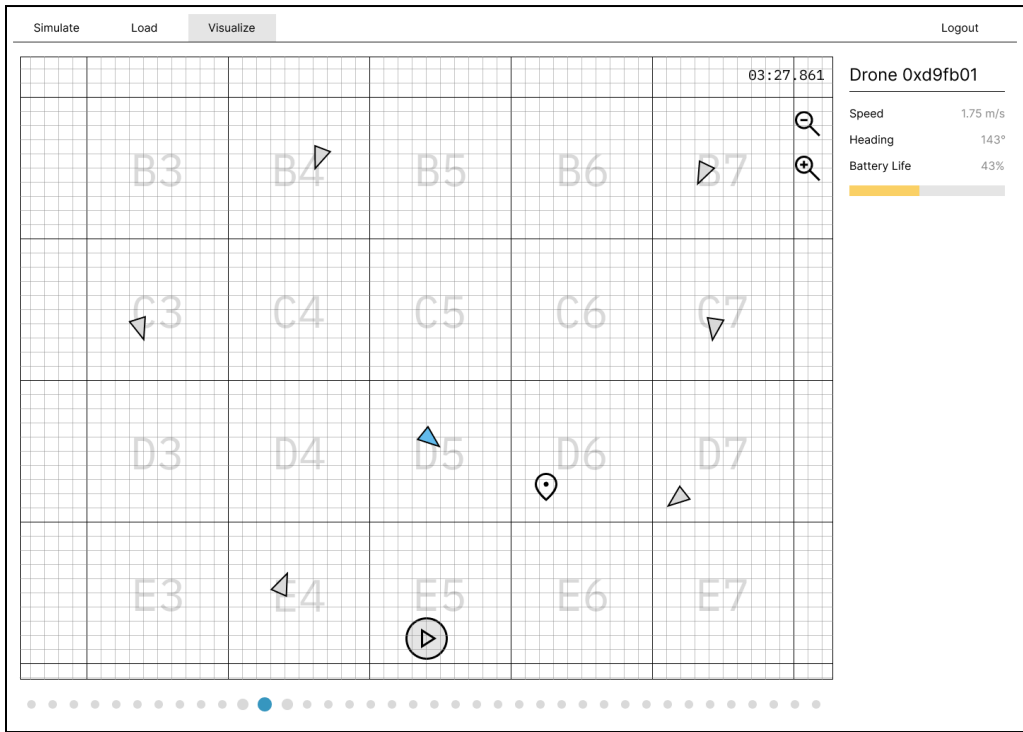


Figure B.4: Drone Visualization Screen

## Appendix C - Project Code

The entirety of our project's code can be found at our github. (<https://git.ece.iastate.edu/sd/sdmay23-50>)