4.3 Proposed Design

4.3.1 Overview

Our design for the web app 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.

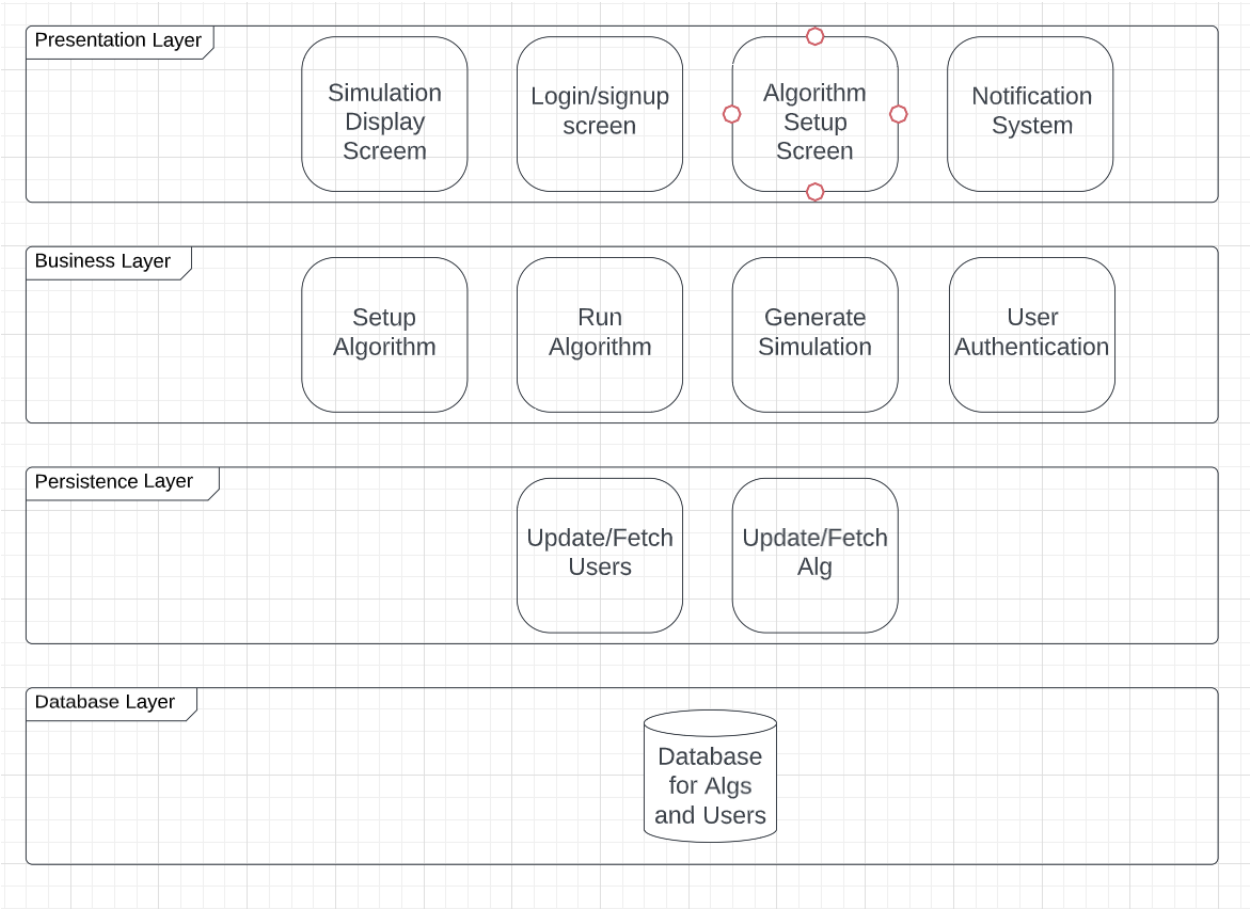

Figure 1: Layered Architecture Proposal

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 ran 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 layers 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.

4.3.2 Detailed Design and Visual(s)

Provide a detailed, technical description of your design, aided by visualizations. This description should be understandable to peer engineers. In other words, it should be clearly written and sufficiently detail such that another senior design team can look through it and implement it.

The description should include a high-level overview written for peer engineers. This should list all sub-systems or components, their role in the whole system, and how they will be integrated or interconnected. A visual should accompany this description. Typically, a detailed block diagram will suffice, but other visual forms can be acceptable.

The description should also include more specific descriptions of subsystems and components (e.g., their internal operations). Once again, a good rule of thumb is: could another engineer with similar expertise build the component/sub-system based on your description? Use visualizations to support your descriptions. Different visual types may be relevant to different types of projects, components, or subsystems. You may include, but are not limited to: block diagrams, circuit diagrams, sketches/pictures of physical components and their operation, wireframes, etc.

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

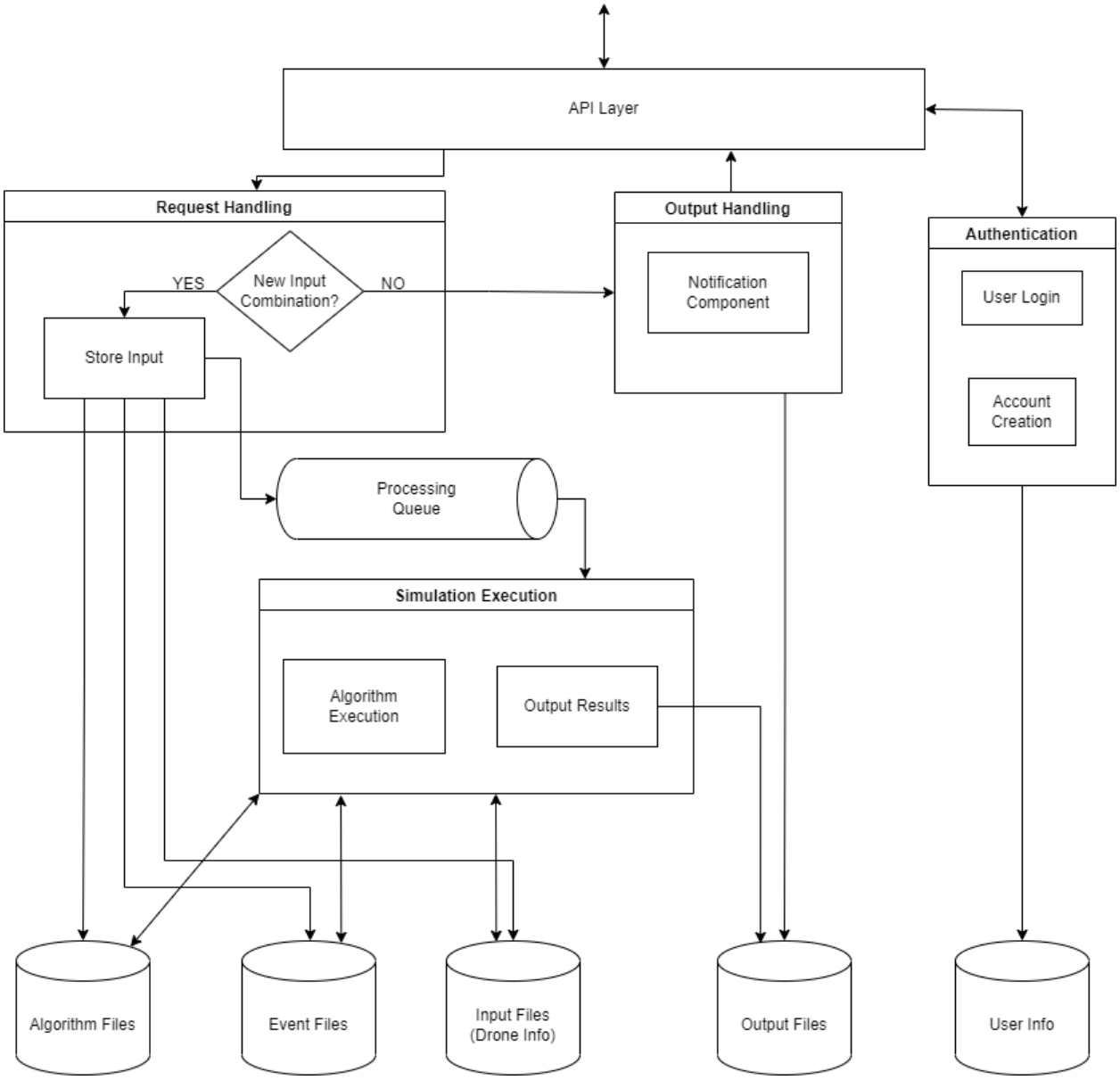

Figure 2: Design Visualization

4.3.3 Functionality

Describe how your design is intended to operate in its user and/or real-world context. What would a user do? How would the device/system/etc. respond? This description can be supplemented by a visual, such as a timeline, storyboard, or sketch.

User diagram, sequence diagram

The first step would be for a user to log in, or register for an account(*Figure 2)*. 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 4)*.

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.

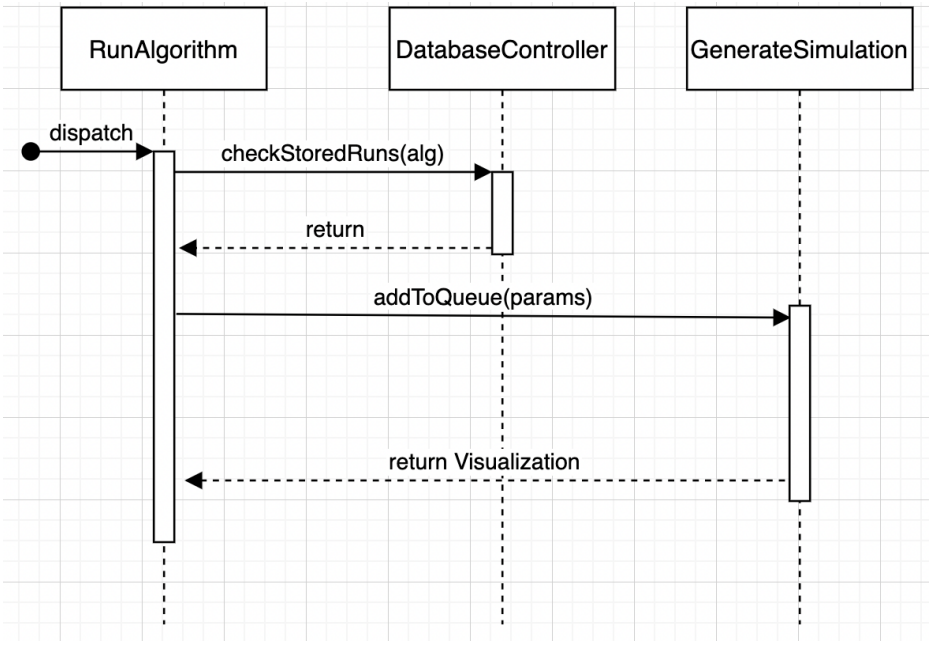

Figure 3: Sequence Diagram

In the case of long run times, we could implement a notification system, featured in our design overview (*Figure 1)*. 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 4: Login / Account Creation Mockup

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

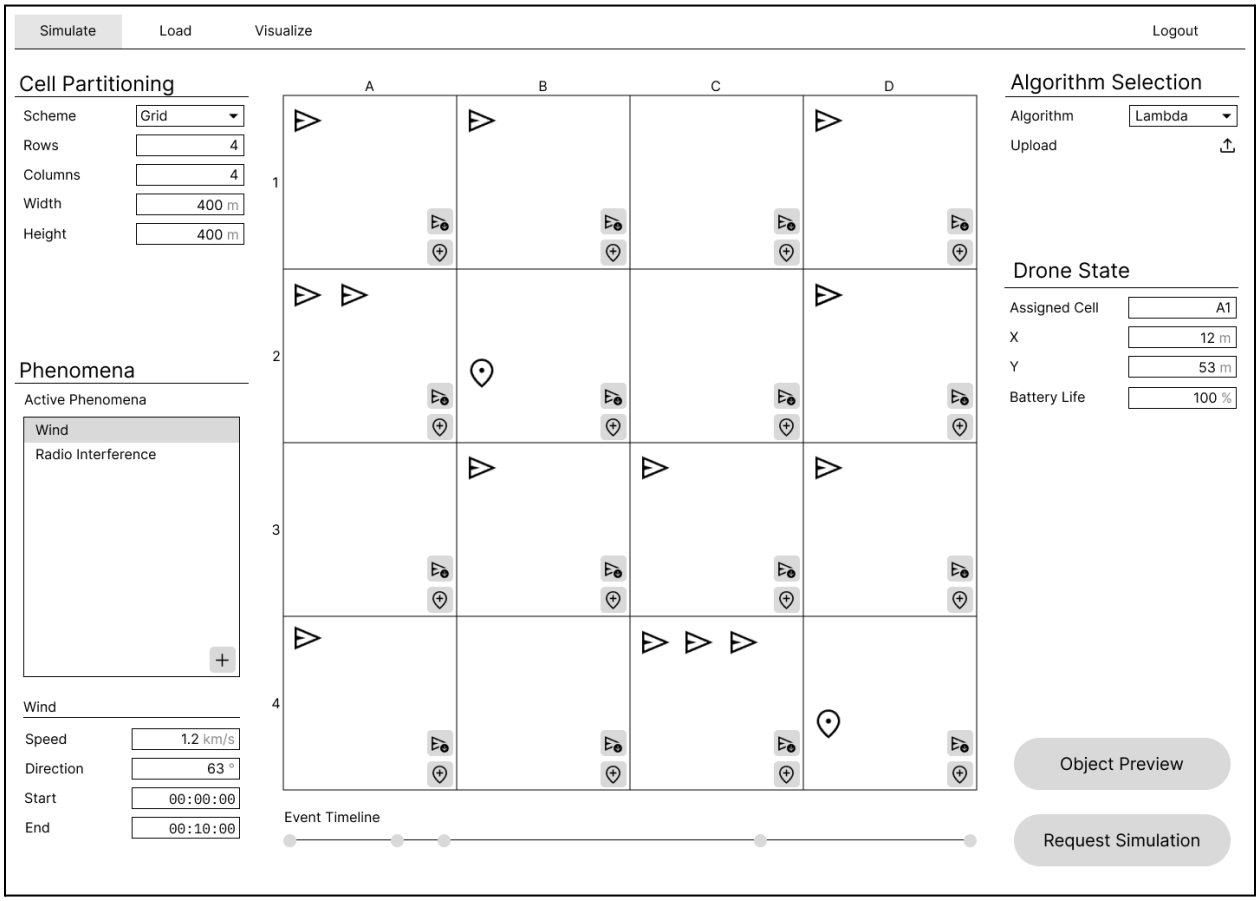

Figure 5: Simulation Setup Mockup

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)

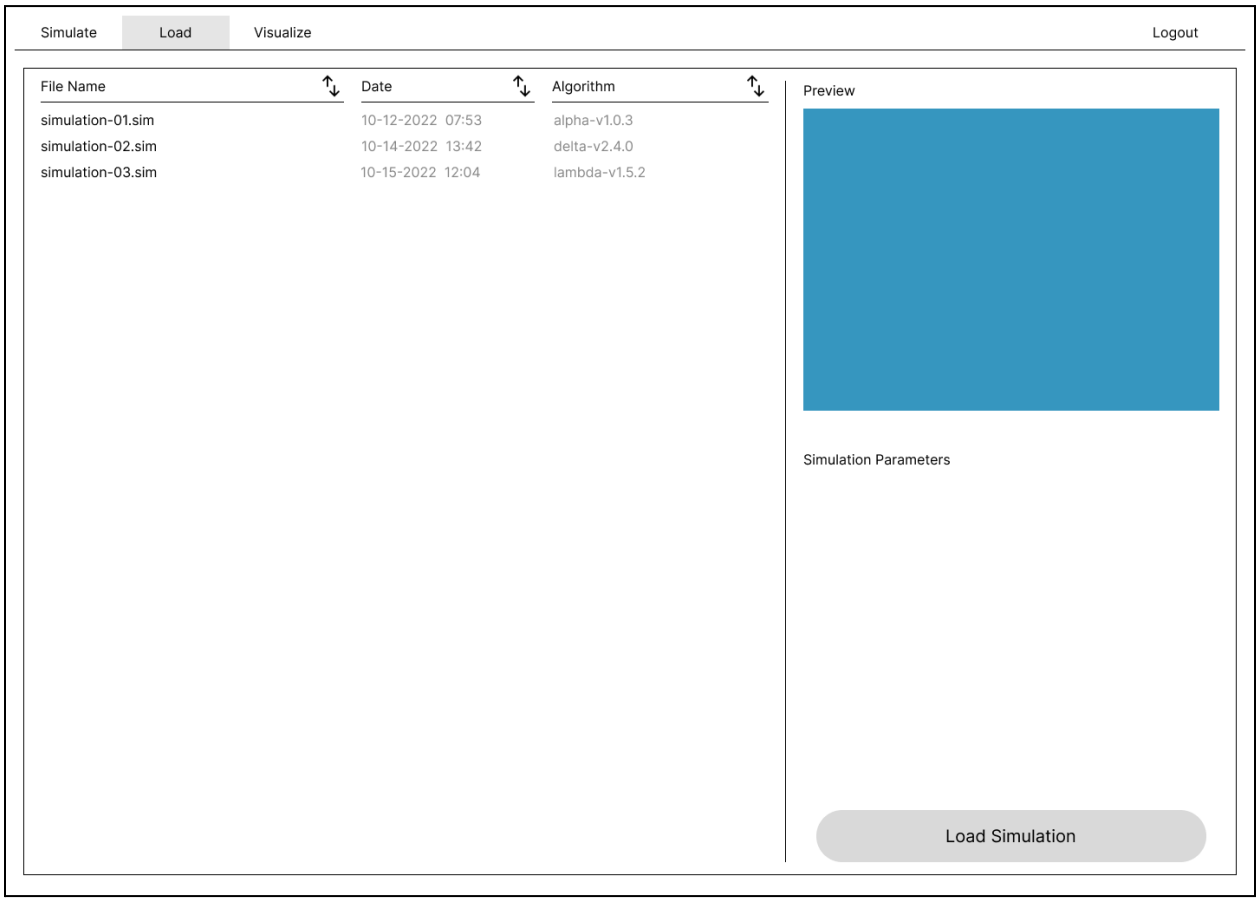

Figure 6: Load Simulation Mockup

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

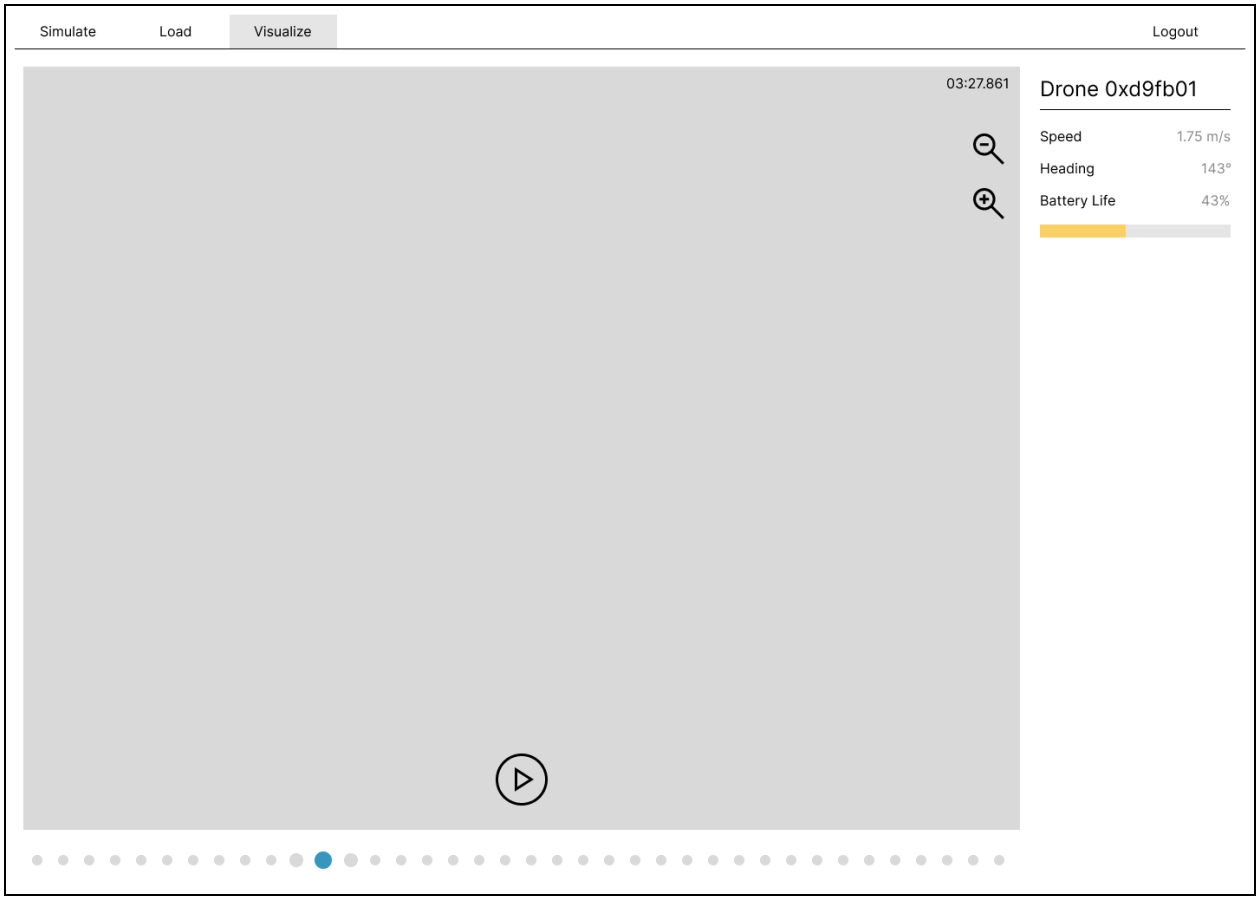

Figure 7: Visualization Mockup

4.3.4 Areas of Concern and Development

How well does/will the current design satisfy requirements and meet user needs?

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.

Based on your current design, what are your primary concerns for delivering a product/system that addresses requirements and meets user and client needs?

What are your immediate plans for developing the solution to address those concerns? What questions do you have for clients, TAs, and faculty advisers?

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

Describe the distinct technologies you are using in your design. Highlight the strengths, weakness, and trade-offs made in technology available. Discuss possible solutions and design alternatives.

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

Figure 8: Technology Considerations

4.5 Design Analysis

Discuss what you have done so far, i.e., what have you built, implemented, or tested? Did your proposed design from 4.3 work? Why or why not? Based on what has worked or not worked (e.g., what you have or haven't been able to build, what functioned as expected or not), what plans do you have for future design and implementation work? For example, are there implications for the overall feasibility of your design or have you just experienced build issues?

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.