Convoys in Atomic and Molecular Trajectories

DESIGN DOCUMENT

Team 19

Advisors/Client: Goce Trajcevski

Team Members: Ayden Albertsen, Benjamin Hall, TJ Thielen, Ben McClannahan

Team Email: sdmay24-19@iastate.edu

Team Website: https://sdmay24-19.sd.ece.iastate.edu/

Executive Summary

Development Standards & Practices Used

IEEE/ISO/IEC 12207-2017 [1]

- Covers the common framework for the software development life cycle.

IEEE/ISO/IEC 90003-2018 [2]

- supply, development, operation and maintenance of computer software and related support services.

ISO/IEC 27001 [3]

- Standard dealing with information security and practices that should be followed.

V. Phoha, "A standard for software documentation," in Computer, vol. 30, no. 10, pp. 97-98, Oct. 1997, doi: 10.1109/2.625327. [4]

Summary of Requirements

- front-end UI that will enable:
  - (i) users to select dataset;
  - (ii) users to enter parameters;
  - (iii) selection of algorithms;
  - (iv) presentation of the results to the end-user;
- back-end that will store the molecular simulation datasets;
- "middleware" that will connect the front-end and back-end;

Applicable Courses from Iowa State University Curriculum

S E 309 - **Software Development Practices**

COM S 363 - **Introduction to Database Management Systems**

COM S 327 - **Advanced Programming Techniques**

COM S 311 - **Introduction to the Design and Analysis of Algorithms**

COS S 319 - **Construction of User Interfaces**

New Skills/Knowledge acquired that was not taught in courses

- Background knowledge of flocks and convoys
- Innerworkings of a convoy detection and related algorithms
- 3D Graphing
- Data Visualization

Table of Contents

List of figures/tables/symbols/definitions

Flock (m, k, r)

- Given a set of n trajectories of entities in the plane, where each trajectory consists of τ line segments, a flock in a time interval I, where the duration of I is at least k, consists of at least m entities such that for every point in time within I there is a disk of radius r that contains all the m entities (note that $m \in N$, $k \in R$ and $r>0$ are given constants).

Convoy

- Given a set of trajectories O, an integer m, a distance value e, and a lifetime k, a convoy query retrieves all groups of objects, i.e., convoys, each of which has at least m objects so that these objects are so-called density–connected with respect to distance e during k consecutive time points.

1 Team, Problem Statement, Requirements, and Engineering Standards

1.1 Team Members

- Ayden Albertsen
- Benjamin Hall
- Ben McClannahan
- TJ Thielen

1.2 Required Skill Sets for Your Project

- Frontend development – Design and implement the UI
- Backend development – Design and implement the API
- Database design and management – Design and implement a database to store user information
- 3D Data Visualization – Create an interactive way to view the data

1.3 Skill Sets covered by the Team

- Frontend development – Ayden Albertsen, TJ Thielen, Benjamin Hall
- Backend development – Ayden Albertsen, TJ Thielen, Benjamin Hall
- Database management – Ayden Albertsen
- Visualization – Benjamin Hall

1.4 Project Management Style Adopted by the team

Given the team’s experience working on Agile teams, it makes the most sense to implement this management style on our project. Agile also allows for iterative development which is great for developing a unique interface for customers that don’t have strict requirements.

1.5 Initial Project Management Roles

Client/Advisor Contact and Communication – Ben McClannahan

Git Management Leader – TJ Thielen

Frontend Leader – Ayden Albertsen

Backend Leader – Benjamin Hall

1.6 Problem Statement

To avoid high cost and to provide safety during exploratory stages, most manufacturers of drugs run simulations of molecular interactions and analyze the movements of atoms (in the context of multiple molecules that they belong to). The main purpose is to detect whether certain events of interest occur – which, in turn, would mean that certain properties of the drug under development are satisfied (or not). For this project's purposes, our event of interest is the formation of a Hydrogen Bond (HB) during the evolution of the chemical compound, and its persistence for a set amount of time. Our project aims to develop a system that will allow users to analyze the simulation datasets and: (1) detect the occurrence of such long-lasting HBs; (2) provide a detailed report to the user; (3) provide a visual representation of the persistent HBs, if they occur within the data set.

1.7 Requirements & Constraints

Functional Requirements:

1. Take input from user regarding data set, algorithm, parameters. Parameters will change depending on the algorithm selected. The system needs to ensure that the correct parameters are provided.
2. Output the results of the algorithm so that the user can clearly see the clusters that persist over time.
3. System needs to scale well with multiple users as well as larger datasets.
4. Computations need to be done on a server rather than the client’s machine due to high intensity computations.
5. Validation needs to be done on user input parameters to ensure that the system does not crash.
6. Runtime of the algorithms needs to be reasonably efficient (Algorithms provided)

7. Access to data should be fast and reliable.

Environmental:

- System should not make excessive/unnecessary computations and should be power efficient.

Economic:

- System should not make excessive/unnecessary computations and should be power efficient and should be extensible so that other datasets can be easily incorporated, and other criteria be added.
- Guarantees of quality of output so that drug companies can clearly see the results as real-world testing involves high costs and risks.

User Interface:

- UI needs to provide ease of navigation.
- It should be intuitive to understand the (purpose and use of) different components.
- It should provide input validation.

1.8 Engineering Standards

IEEE/ISO/IEC 12207-2017 [1]

- Covers the common framework for the software development life cycle.

IEEE/ISO/IEC 90003-2018 [2]

- supply, development, operation and maintenance of computer software and related support services.

ISO/IEC 27001 [3]

- Standard dealing with information security and practices that should be followed.

V. Phoha, "A standard for software documentation," in Computer, vol. 30, no. 10, pp. 97-98, Oct. 1997, doi: 10.1109/2.625327. [4]

1.9 Intended Users and Uses

There are several classes of stakeholders that could potentially benefit from the results of this project.

Chemical scientists and engineers:

- How will they use our project?
  - Easily filter received data and visualize results in a way that is useful to chemists.

    - Display easy to understand information about properties regarding reactions of chemical compounds to peers and others.
- What do they gain from our project?
    - Speeds up the process of drug development.
    - Reduces time and effort on evaluating the data our application does for them.

Pharmaceutical Investors and Businesses:

- How will they use our project?
    - Easy, high-level tool to understand drug interactions without needing to understand the details.
- What do they gain from our program?
    - Reduce the risks when investing in development for potential drugs.

Educators:

- How will they use our project?
    - Safe environment for demonstrating properties of chemical reactions to students.

2 Project Plan

2.1 Task Decomposition

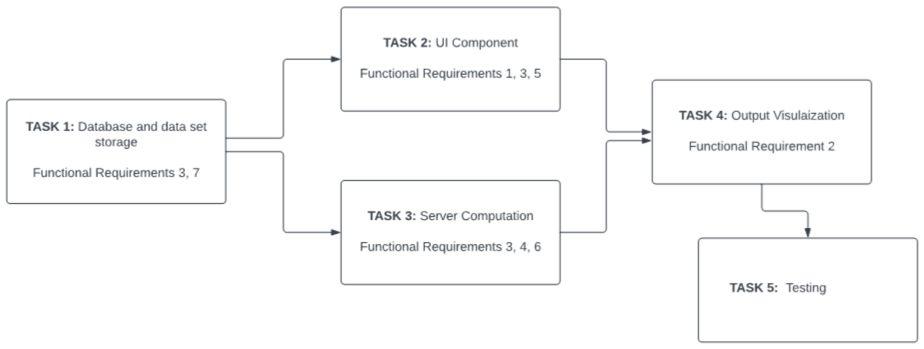

TASK 1: Database and data set storage
Functional Requirements 3, 7
TASK 2: UI Component
Functional Requirements 1, 3, 5
TASK 3: Server Computation
Functional Requirements 3, 4, 6
TASK 4: Output Visulaization
Functional Requirement 2
TASK 5: Testing

Figure 1. Task Decomposition

For our project we derived 5 major tasks that need to be completed in order to fulfill the functional requirements that we defined in the previous section. From the above tasks we have defined more specific subtasks that will need to be completed in order to fully complete the tasks.

Task 1: Database

1a. Pick a database to store npz datasets (Could potentially use file system)

1b. Pick relational database to store user information and results

1c. Determine tables needed for relational database

1d. Create ER diagram for relational database

1e. Translate ER diagram to SQL statements

1f. Set up/deploy database server

Task 2: UI

2a. Design Prototype for UI (Figma)

2b. Choose UI framework (React, Angular)

2c. Determine list of detection algorithms and required parameters

2d. Project Setup

2e. Select Database/Algorithm/Parameters Page (Functional Requirement 1)

2f. Import Dataset Page

2g. Login/Create user Functionality

2h. Job Status Page

2i. View Results/Select Convoy to Visualize Page

2j. Connect UI to backend (Axios)

Task 3: Server/API

3a. Choose Backend Framework (Flask, Spring)

3b. Project Setup

3c. Obtain all algorithms being used for the project

3d. Endpoint to take in job parameters and start a convoy detection job

3e. Import Dataset Endpoint

3f. User Login/Creation Endpoints

3g. Retrieve Job Status Endpoint

3h. Retrieve results of convoy detection algorithm endpoint

Task 4: Convoy Visualization

4a. Discuss specific visualization needs with client

4b. Choose visualization framework (Plotly, VMD as failsafe)

4c. Create endpoint that produces a 3D visualization of a convoy

4d. Send convoy visualization to frontend

4f. Display an interactive visualization to the user in the UI

Task 5: Testing

5a. Ensure that all components of the project can communicate with each other and produce the desired output

5b. Ensure that all component unit tests are correct.

5b. Ensure that all functional requirements are met.

2.2 Project Management/Tracking Procedures

Our group will be using an agile project management style. Our tasks/subtasks will be broken down into 2 weeklong sprints (Roughly 8 sprints). The goal of our project is to produce a user friendly and easy to use product which requires constant communication, flexibility, and adaptability. By choosing the agile methodology we can iteratively develop each of our subtasks and get meaningful feedback after each sprint which is crucial when developing a customer focused product.

Our progress and tasks will mainly be tracked using GitLab issues and milestones. GitLab allows issues to be assigned to specific group members and have wights assigned to them ensuring a even workload distribution for our group members. Daily standups and communication will mainly be done through Discord.

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

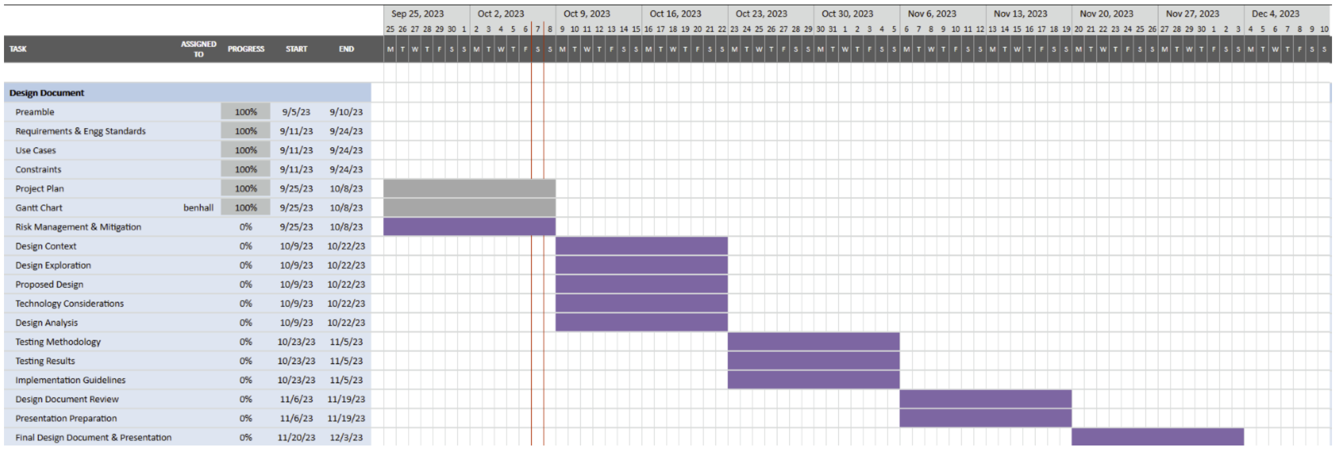
Milestone	Metrics

Relational Database is designed, implemented, and deployed (Task 1)	Database captures/maintains the required data for the project. Database response time: Less than 10 milliseconds for 95% of queries. Transaction throughput: Handle up to 10 concurrent users without performance degradation.
Method to store and retrieve element datasets is implemented (Task 1)	Efficiency: Large datasets are stored efficiently with a compression ratio of at least 30% Data retrieval should be reasonably fast, and dataset should take no longer than 1 minute to be loaded into the computation system
Login/Register functions are implemented (Task 2)	Prospective users are able to register new accounts and login with them. Security: all users should have their passwords and data encrypted.
Dataset/Algorithm/Parameter selection is implemented (Task 2)	Ensures that all of functional requirement 1 is met. Users can easily select the algorithm and parameters for convoy detection in an intuitive and easy to use way
Upload Dataset Functionality (Task 2/Task 3)	System should be able datasets as large as 5GB. Users should be able to drag and drop datasets to upload them to the server Upload speed should only be affected by client's internet speed.
UI component is able to communicate with backend (Task 2)	Only authenticated users should be able to interact with backend Response time should be less than 100ms
Convoy Detection Algorithms are implemented (Task 3)	Convoy algorithm runtimes should be the same as described in journal articles. Users should have a way of seeing the status of the algorithms

All systems are able to work and communicate with each other (Task 5)	Latency between communicating from one system to another should be less than 100ms Proper https encryption standards are used when communicating over the internet

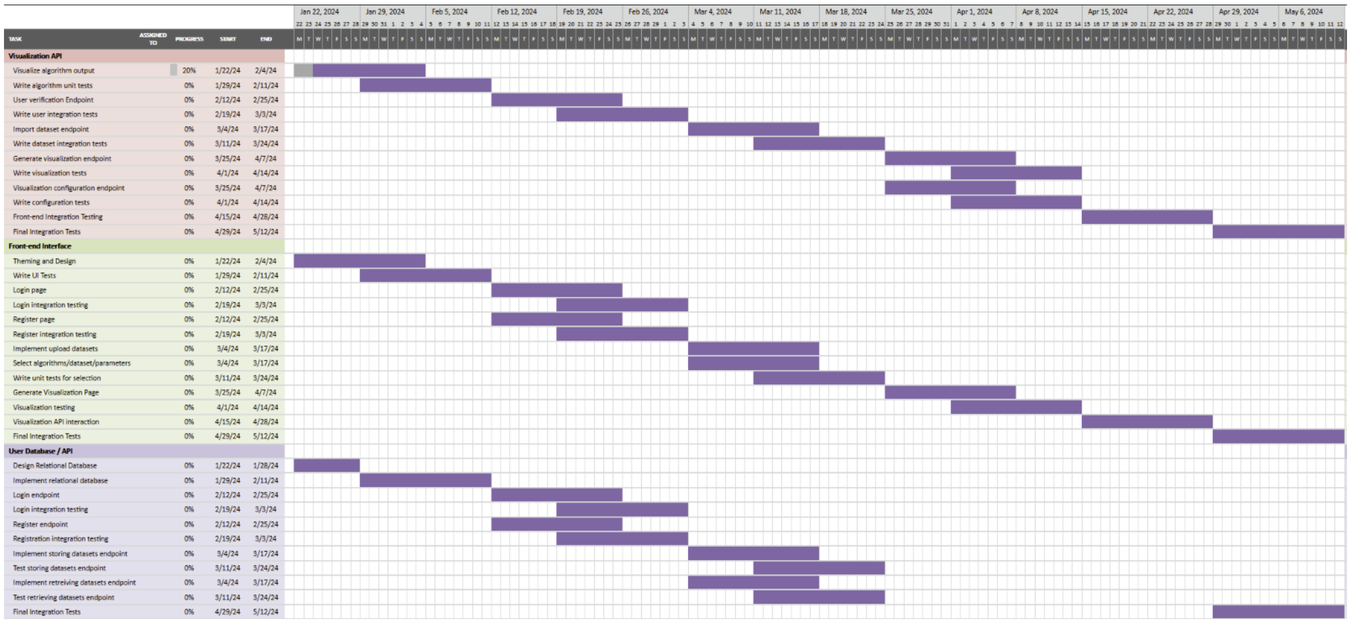
Figure 2. Milestones and Metrics Table

2.4 Project Timeline/Schedule



TASK	ASSIGNED TO	PROGRESS	START	END
Design Document				
Preamble		100%	9/5/23	9/10/23
Requirements & Engg Standards		100%	9/11/23	9/24/23
Use Cases		100%	9/11/23	9/24/23
Constraints		100%	9/11/23	9/24/23
Project Plan		100%	9/25/23	10/8/23
Gantt Chart	benhall	100%	9/25/23	10/8/23
Risk Management & Mitigation		0%	9/25/23	10/8/23
Design Context		0%	10/9/23	10/22/23
Design Exploration		0%	10/9/23	10/22/23
Proposed Design		0%	10/9/23	10/22/23
Technology Considerations		0%	10/9/23	10/22/23
Design Analysis		0%	10/9/23	10/22/23
Testing Methodology		0%	10/23/23	11/5/23
Testing Results		0%	10/23/23	11/5/23
Implementation Guidelines		0%	10/23/23	11/5/23
Design Document Review		0%	11/6/23	11/19/23
Presentation Preparation		0%	11/6/23	11/19/23
Final Design Document & Presentation		0%	11/20/23	12/3/23

Figure 3a. Gantt chart for the Fall 2023 Semester sdmay-24-19 Gantt Chart.xlsx



TASK	ASSIGNED TO	PROGRESS	START	END
Visualization API				
Visualize algorithm output		20%	1/22/24	2/4/24
Write algorithm unit tests		0%	1/29/24	2/11/24
User verification Endpoint		0%	2/12/24	2/25/24
Write user integration tests		0%	2/19/24	3/3/24
Import dataset endpoint		0%	3/4/24	3/17/24
Write dataset integration tests		0%	3/11/24	3/24/24
Generate visualization endpoint		0%	3/25/24	4/7/24
Write visualization tests		0%	4/1/24	4/14/24
Visualization configuration endpoint		0%	3/25/24	4/7/24
Write configuration tests		0%	4/1/24	4/14/24
Front-end Integration Testing		0%	4/15/24	4/28/24
Final Integration Tests		0%	4/29/24	5/12/24
Front-end Interface				
Theming and Design		0%	1/22/24	2/4/24
Write UI Tests		0%	1/29/24	2/11/24
Login page		0%	2/12/24	2/25/24
Login integration testing		0%	2/19/24	3/3/24
Register page		0%	2/12/24	2/25/24
Register integration testing		0%	2/19/24	3/3/24
Implement upload datasets		0%	3/4/24	3/17/24
Select algorithms/dataset/parameters		0%	3/4/24	3/17/24
Write unit tests for selection		0%	3/11/24	3/24/24
Generate Visualization Page		0%	3/25/24	4/7/24
Visualization testing		0%	4/1/24	4/14/24
Visualization API interaction		0%	4/15/24	4/28/24
Final Integration Tests		0%	4/29/24	5/12/24
User Database / API				
Design Relational Database		0%	1/22/24	1/28/24
Implement relational database		0%	1/29/24	2/11/24
Login endpoint		0%	2/12/24	2/25/24
Login integration testing		0%	2/19/24	3/3/24
Register endpoint		0%	2/12/24	2/25/24
Registration integration testing		0%	2/19/24	3/3/24
Implement storing datasets endpoint		0%	3/4/24	3/17/24
Test storing datasets endpoint		0%	3/11/24	3/24/24
Implement retreiving datasets endpoint		0%	3/4/24	3/17/24
Test retrieving datasets endpoint		0%	3/11/24	3/24/24
Final Integration Tests		0%	4/29/24	5/12/24

Figure 3b. Gantt chart for the Spring 2024 Semester sdmay-24-19 Gantt Chart.xlsx

2.5 Risks And Risk Management/Mitigation

Task	**Risk**	**Mitigation**	**Probability**
Task 1: Database	Data becomes compromised.	Follow ISO/IEC 27001 standards	0.1
Task 1: Database	Data is deleted.	Keep a backup of the data so that it can't be deleted or become unavailable	0.1
Task 2: UI	Framework does not meet our requirements	Define a backup frontend framework in case our first choice does not work	0.2
Task 2: UI	Unable to communicate with backend	Fallback to a different http request library	0.1
Task 4: Visualization	Visualization Framework unable to handle the amount of data	Reduce resolution of data being displayed, only show some time slices, render elements as simple dots.	0.8
Task 4: Visualization	Visualization API cannot handle several connections and requests	Implement queue system for visualization requests	0.3

Figure 4. Risk Decomposition Table

2.6 Personnel Effort Requirements

Task	Setup/Research Hours	Implementation Hours	Explanation
1a. Pick a database to store .npz datasets (Could potentially use file system)	3	1	There may only be a few options available
1b. Pick relational database to store user information and results	2	0	Will mostly likely use MySQL but need to ensure that it will meet our requirements.
1c. Determine tables needed for relational database	5	0	Will take some time to figure out what tables are needed for the project
1d. Create ER diagram for relational database	2	4	Some setup time to learn Lucid Charts or some other diagram tool
1e. Translate ER diagram to SQL statements	2	4	Just some research and testing for simple SQL statements.

1f. Set up/deploy database server	1	4	Getting access to Iowa State servers and setting up the database on it.
2a. Design Prototype for UI (Figma)	2	10	Learn Figma if the team has not learned it yet and create program flow and design themes.
2b. Choose UI framework (React, Javascript)	1	0	Decide on which framework would be best suited for our team.
2c. Determine list of detection algorithms and required parameters	2	0	Research the algorithm to determine all the parameters needed from the user.
2d. Project Setup	0	1	
2e. Select Dataset/Algorithm/Parameters Page (Functional Requirement 1)	6	12	Might have to pull available datasets and algorithms from the backend for this.
2f. Import Dataset Page	6	12	Developing an upload function with security concerns and writing tests in conjunction with the backend to test functionality.
2g. Login/Create user Functionality	6	12	Developing a user account system with security concerns in mind. Writing tests to confirm security and functionality.
2h. Job Status Page	8	10	
2i. View Results/Select Convoy to Visualize Page	12	16	Might take some research with the visualization aspect as the output is an HTML file.
2j. Connect UI to backend (Axios)	12	24	Integration with backend for user and dataset storage. Will need to test login, register, upload, and retrieval functions.
3a. Choose Backend Framework (Flask, Spring)	2	0	Research what framework meets our requirements.
3b. Project Setup	1	0	Setting up the project in GitLab.
3c. Obtain all algorithms being used for the project	1	0	Receive all algorithms from client.

3d. Endpoint to take in job parameters and start a convoy detection job	9	19	Will also have to develop or research queue framework for jobs.
3e. Import Dataset Endpoint	9	18	Requires integration with the user database and API.
3f. User Login/Creation Endpoints	9	18	Integration of User API and database.
3g. Retrieve Job Status Endpoint	6	12	Research the best way to implement and retrieve job status.
3h. Retrieve results of convoy detection algorithm endpoint	4	8	Requires integration testing from visualization API and frontend.
4a. Discuss specific visualization needs with client	4	0	Meeting with client.
4b. Choose visualization framework (Plotly, VMD as failsafe)	4	0	Research and decide which framework fits our client's needs.
4c. Create endpoint that produces a 3D visualization of a convoy	9	18	Need to optimize and refine usage of visualization library.
4d. Send convoy visualization to Frontend	2	4	Need to research ways to optimally send data securely and efficiently.
4f. Display an interactive visualization to the user in the UI	9	18	Researching and designing an elegant way to display the results of the algorithm.
5a. Ensure that all components of the project are able to communicate with each other and produce the desired output	9	18	Writing and running integration tests.
5b. Ensure that all functional requirements are met	9	18	Comprehensive testing and client feedback.

Figure 5. Personnel Effort Breakdown Table

2.7 Other Resource Requirements

Algorithms and datasets will need to be obtained from the clients. We will also need to obtain a server to host our various project systems.

4 Design

4.1 Design Content

Our design attempts to address the needs of the client by creating two fundamental systems; (a) a frontend that provides the UI/UX to the user as well as display the rendering of the convoy displayed to the user, and (b) a backend that handles data access, computation, and visualization rendering.

4.2 Design Complexity

The design consists of multiple components/subsystems that each utilize distinct scientific, mathematical, or engineering principles.

Frontend

1. Authentication/Authorization
    a. Users need to view only data assigned to them.
    b. Users need to log in to access their data
2. UI/UX
    a. Database, algorithm, parameter selection
    b. View Job Status
    c. View Results
3. Visualization of convoys
    a. Interactive
    b. 3D

Backend

4. Data Access and Storage
    a. Upload and retrieve stored datasets in the database.
    b. Load datasets into algorithm parameters.
5. Job Broker
    a. Start job on API request and return Job Identifier.
    b. Portion out computer resources for algorithms.
    c. Retrieve and return status of Job from an identifier.
6. Convoy Computation/Processing
    a. Load large datasets into algorithm.
    b. Run a computationally difficult algorithm while several requests are going on.
    c. Return processed data to visualization.
7. Visualization Rendering
    a. Take in the processed data and render the dataset to an HTML format for easy and dynamic viewing by the user.
    b. Reduce size for output.
    c. Reduce resolution of data.

4.3 Modern Engineering Tools

1. Backend

Flask [6] provides a stable but updated library for developing and hosting an API through Python.

Celery [10] is an asynchronous job queue library that will be used for starting and retrieving job status. This tool will provide the most flexible and fast response to users. It provides interactivity with Flask which makes it perfect for this use case.

Redis [11] is an in-memory database which can be used as a broker for job statuses for Celery so clients can dynamically request the status of the job queued through the API.

Docker [12] allows for quick deployment of the multiple backend scripts like Flask and Redis, making it a more deployable package.

Plotly [7] provides an expansive visualization library which can output a HTML rendering of the data after processing, which can be returned by the API.

MySQL [13] provides a traditional database structure perfect for storing user data and the stored data sets uploaded by the user.

2. Frontend

React [5] is the largest, most popular frontend library used to develop fast and dynamic web pages. It will be used to dynamically update web pages with information retrieved from our backend.

4.4 Design Context

Our project is designed to be used as an aid for drug manufacturers to detect how certain substrates and drug molecules may interact over time as well as detect events of interest such as convoys. Because of this, both chemists and non-technical industry professionals need to be able to use and understand our software.

Area	**Description**	**Examples**
Public health, safety, and welfare	How does your project affect the general well-being of various stakeholder groups? These groups may be direct users or may be indirectly affected (e.g., solution is implemented in their communities)	Aid in providing a comprehensive tool to automate convoy detection and allow chemists to develop potentially lifesaving drugs faster.
Global, cultural, and social	How well does your project reflect the values, practices, and aims of the cultural	Chemists' workload will be lightened by removing the tedious

	groups it affects? Groups may include but are not limited to specific communities, nations, professions, workplaces, and ethnic cultures.	repetitive task of detecting convoys by hand enabling a more effective exchange of findings and processes.
Environmental	What environmental impact might your project have? This can include indirect effects, such as deforestation or unsustainable practices related to materials manufacture or procurement.	The servers that run the convoy detection algorithms will use energy, however, our system will enable more efficient separation of potential reactions among atoms without sacrificing the effectiveness.
Economic	What economic impact might your project have? This can include the financial viability of your product within your team or company, cost to consumers, or broader economic effects on communities, markets, nations, and other groups.	Drug research can be extremely costly and have risks involved. Our system will mitigate some of these costs by using simulated data and detecting convoys with efficient algorithms. This can make for faster drug research as well as quicker time to market.

Figure 6. Design Considerations Table

4.5 Prior Work/Solutions

There are no comprehensive systems that provide convoy detection and visualization using a cohesive user interface. There has been prior work done with studying molecular flocks [9l] and convoys [8]. The downside of these two studies is that they neglect to consider tolerance gaps in convoy detection which is still interesting to drug manufactures. There also exists software that can visualize molecular trajectories but none that work in conjunction with a convoy detection system.

4.6 Design Decisions

Backend Framework

Defined Criteria

- Needs to be able to work with the data and algorithms provided by the client.
- Needs to be able to scale with multiple users.
- Needs authentication support.
- Needs to be able to communicate with frontend using HTTP
- Support for rendering visualizations

Given the criteria we narrowed down our options to two different frameworks:

- Java Spring
    - Group Familiarity/Experience

        - Fast
        - Authentication Support
        - Testing Support / Dependency Injection
        - Good Documentation
        - Bloated Codebase
        - Not compatible with given datasets
    - Python Flask
        - Older framework
        - Little group experience
        - Compatible with given datasets and algorithms
        - Performant
        - Lightweight
        - Can work with many visualization and job queue frameworks.

After analysis of our two options, we have decided to go with Flask to create our backend. Its compatibility with the given datasets being its biggest draw. It will also give the group an opportunity to learn a new framework and for some of us a new language. Though this will come at the cost of added time as members may have to do more initial research before completing their tasks. Python also has a larger number of graphing/visualization framework available which also lead us to choose Flask.

Frontend Framework

Defined Criteria

- Needs to be able to give a good user experience.
- Interface with our visualization framework to display information

Given the criteria we narrowed down our options to two different frameworks:

- React
    - Most popular frontend framework
    - Group Familiarity
    - Flexible
    - Larger community
    - Larger file size
- Vue
    - Second most popular frontend framework
    - Smaller file sizes
    - Better for simplicity

After analysis of our two options, we have decided to go with React to create our frontend. Of all our members, we have the most experience with React. Additionally, it has great community support as it is the most popular frontend framework that developers use.

- Visualization Framework
    - Display convoys through various time periods
    - Output multiple convoys
    - Highlight hydrogen bonds within convoys.
    - Output results to our frontend
    - Fast and efficient

Given the criteria we narrowed down our options to two different frameworks:

- VMD
    - External application
    - Unknown API
    - Suggested by client
    - Built for modeling molecules
    - Built-in scripting
    - Lacks interactivity
- Plotly
    - Python library
    - Interactive
    - Flexible
    - May lack complexity and tooling for molecule modeling.

After analysis of our two options, we have decided to go with Plotly to create our visualization framework. We concluded that VMD would require too much work to interact with our frontend and it would be better to use a more flexible tool like Plotly over one tool designed for molecule modeling. Plotly being a Python library also allows for simplicity without backend framework being developed in Python as well.

4.7 Proposed Design

4.7.1 Design 0 (Initial Design)

Design Visual and Description

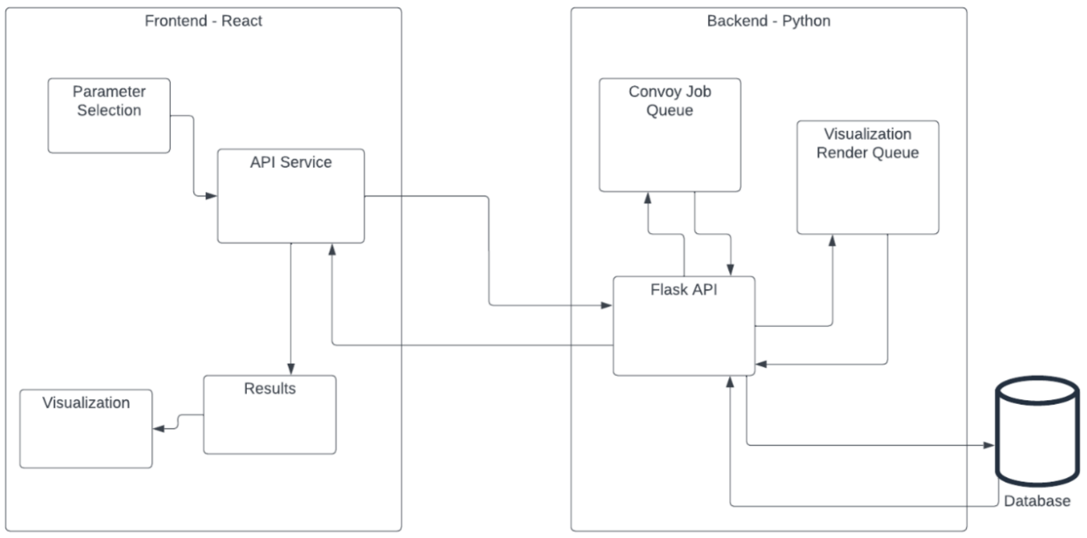


Figure 7. Design 0 Diagram

This project can be broken down into two main parts: the frontend, and the backend. The frontend will encompass parameter selection, the API service, the Results, and the Visualization. The backend will contain the Flask API, the Convoy Job Queue, the Visualization Render Queue, and communicate with the database.

The process will start on the frontend, where a user is able to upload a dataset for the convoy detection algorithms to run on. This will go through the API Service to communicate with the Flasks API, which will then upload that data to the database. After this is done, the user can use the Parameter Selection component to determine the specifics of how the convoy detection algorithm will work. It will pass that information to the API service, which will communicate with the Flask API. The Flask API will send this information to the Convoy Job Queue. This will execute the algorithm needed to detect convoys and return the results to the API. Next, it will send these results to the Visualization Render Queue. This will create a visualization using Plotly to display information about the convoys. Finally, the Flask API will send back this visualization to the API Service, which will send this information to the frontend for the user to view.

Components in relation to requirements:

Frontend

- Parameter Selection
    - This helps to satisfy the requirement of taking input from the user including the dataset, parameters, and algorithm. It also helps to ensure the parameters are valid to prevent crashing.
- API Service
    - This is a necessary component because it handles the communication between the data/parameters and the execution of the algorithms.
- Results
    - This component is needed to process the results being returned from the backend, necessary for generating the visualization on the frontend.
- Visualization
    - This helps to meet the requirement of displaying a visualization for the user to view on the frontend.

Backend

- Flask API
    - This is a necessary part of the communication between the frontend and backend, allowing for the parameters and dataset to be properly used.
- Convoy Job Queue
    - This component satisfies the requirement of executing the algorithms on a remote server.
- Visualization Render Queue
    - This component meets the requirement of creating the visualization so it can eventually be outputted on the frontend for the users to view.
- Database
    - This enables the program to access the data both quickly and reliably.

Functionality

This design is intended to operate in the real world by first allowing chemists who need to view the formation of convoys and hydrogen bonds between molecules to upload their data to the database. Next, they can select the necessary parameters and algorithm to execute. Both this and the visualization of the results will be done remotely and sent back to the frontend for the users to view. This process will meet the requirements as specified above.

4.7.2 Design 1 (Design Iteration)

Design Visual and Description

In this design iteration we expanded on the frontend and backend systems. For the frontend we completed a UI prototype using Figma. This prototype includes detailed versions of the parameter selection, results, and visualization sections in our initial design. It also includes implementations of pages outlined in our task decomposition sections.

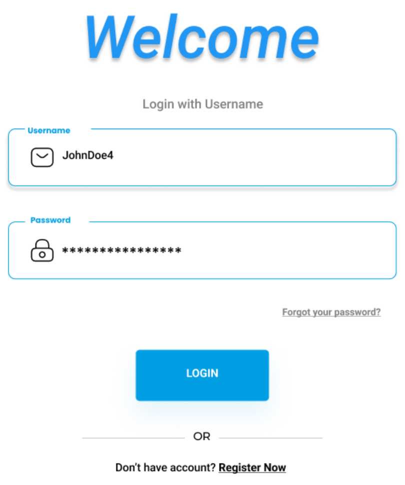
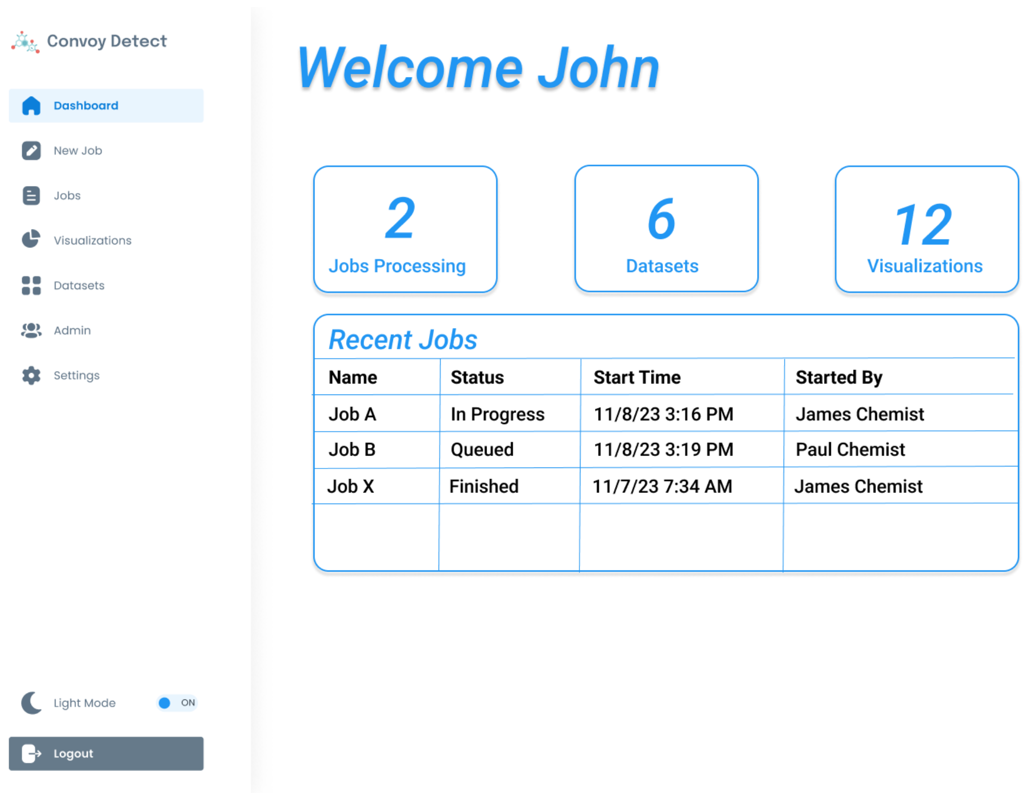


Figure 8. Frontend Login Page



Welcome John

2 Jobs Processing

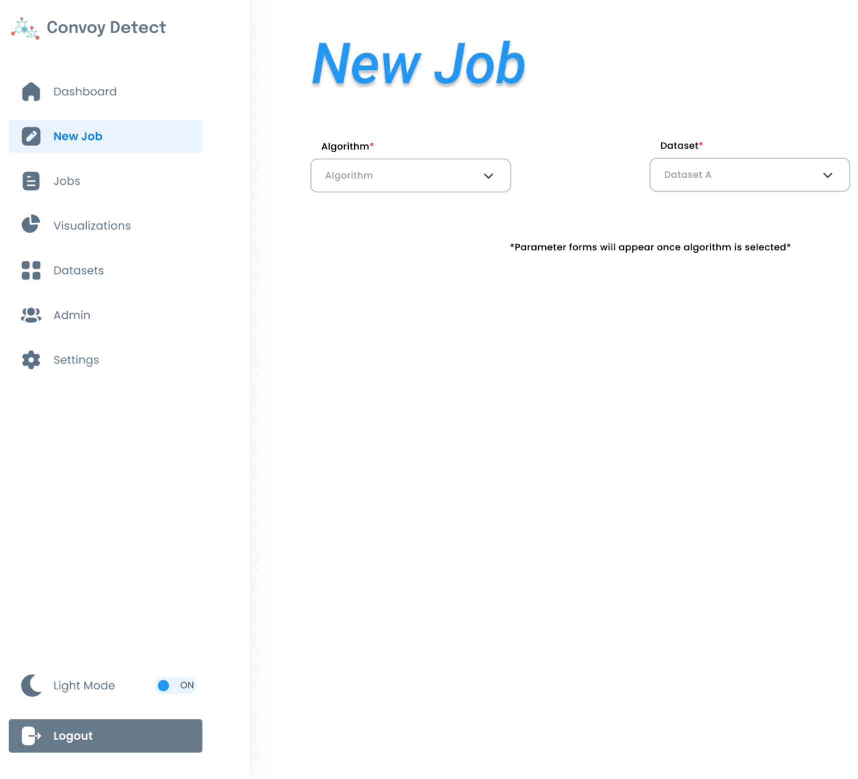
6 Datasets

12 Visualizations

Recent Jobs

Name	Status	Start Time	Started By
Job A	In Progress	11/8/23 3:16 PM	James Chemist
Job B	Queued	11/8/23 3:19 PM	Paul Chemist
Job X	Finished	11/7/23 7:34 AM	James Chemist

Figure 9. Frontend Dashboard Page



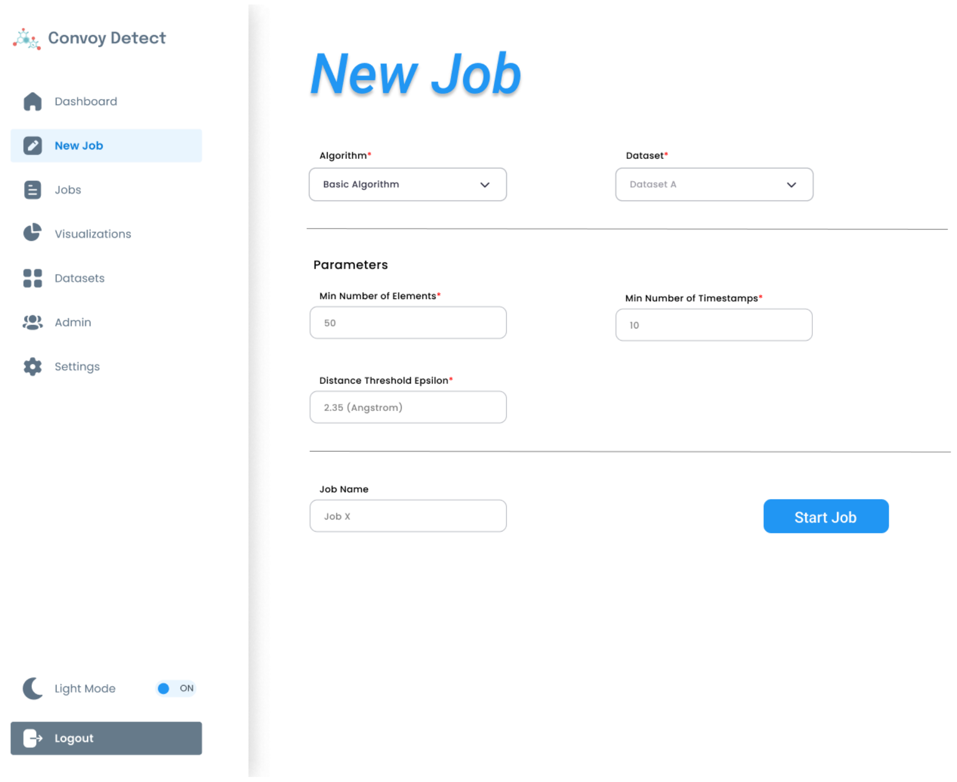
New Job

Algorithm*: Algorithm

Dataset*: Dataset A

Parameter forms will appear once algorithm is selected

Figure 10. New Job Page



New Job

Algorithm*: Basic Algorithm

Dataset*: Dataset A

Parameters

Min Number of Elements*: 50

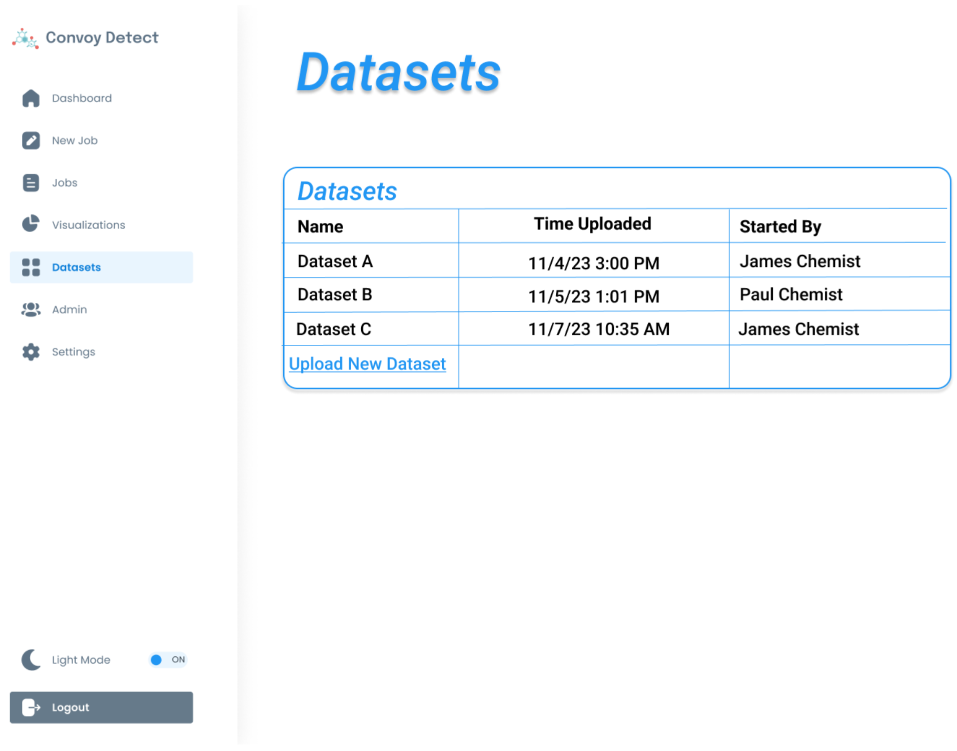
Min Number of Timestamps*: 10

Distance Threshold Epsilon*: 2.35 (Angstrom)

Job Name: Job X

Start Job

Figure 11. New Job Page Cont.



Datasets

Datasets

Name	Time Uploaded	Started By
Dataset A	11/4/23 3:00 PM	James Chemist
Dataset B	11/5/23 1:01 PM	Paul Chemist
Dataset C	11/7/23 10:35 AM	James Chemist
Upload New Dataset		

Figure 12. Datasets Page

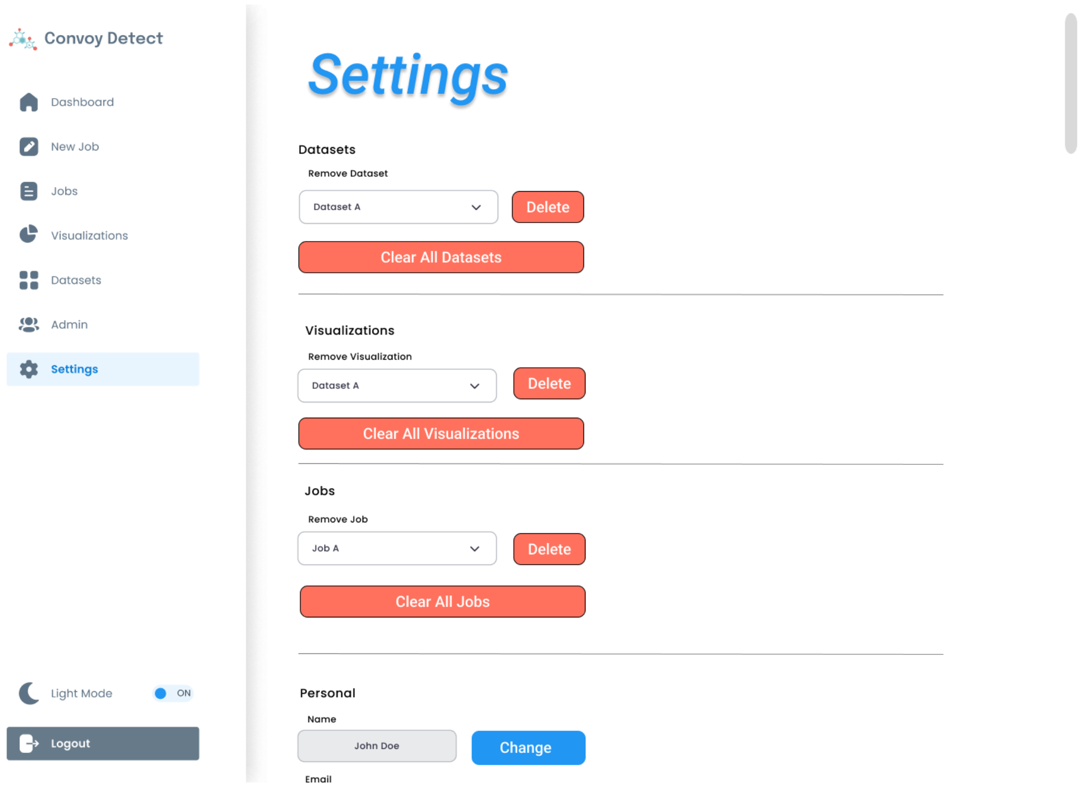


Figure 13. Settings Page

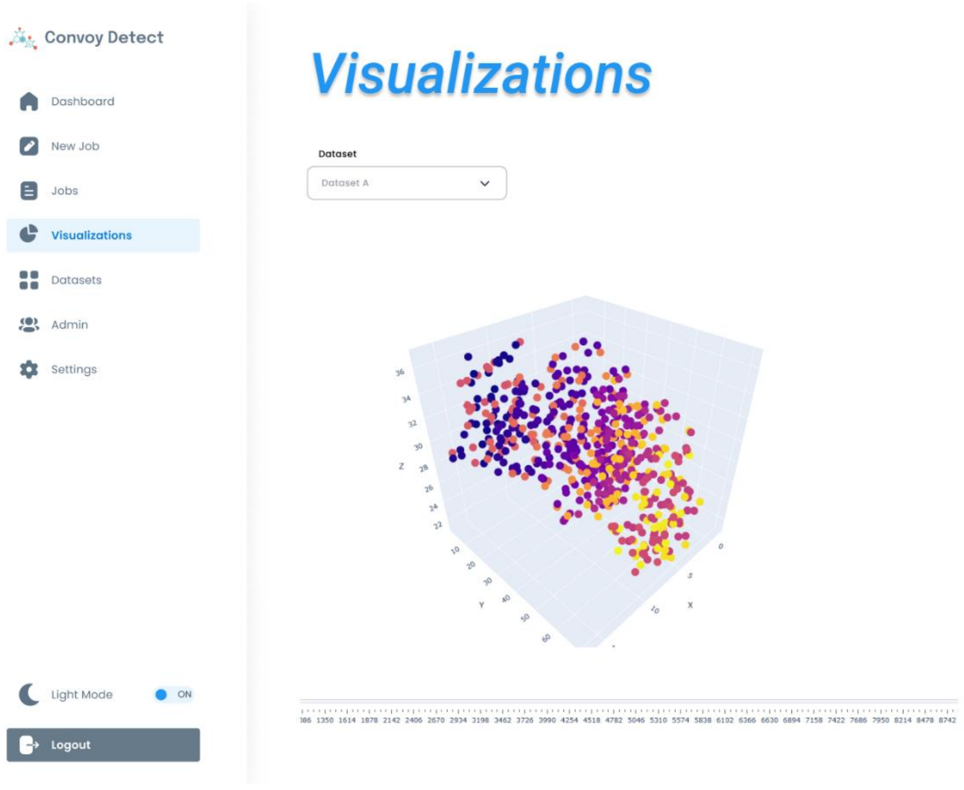


Figure 14. Data Visualization

For the backend we created a more detailed version of our design below. Here, the client/frontend interacts with Flask, which itself communicates with a MySQL Database through uploading data sets, authenticating the user, and allowing the user to log in and/or register. When Flask starts a job, it hands things over to Celery. Celery stores the job's ID and job status inside a Redis Database, and then sends the job off to the appropriate algorithm, which retrieved the specified data set from the MySQL database. Once the algorithm is finished, it uses Plotly to visualize the data and stores the html file that is created back into Celery, returning that to Flask, allowing the user to view previous and current jobs.

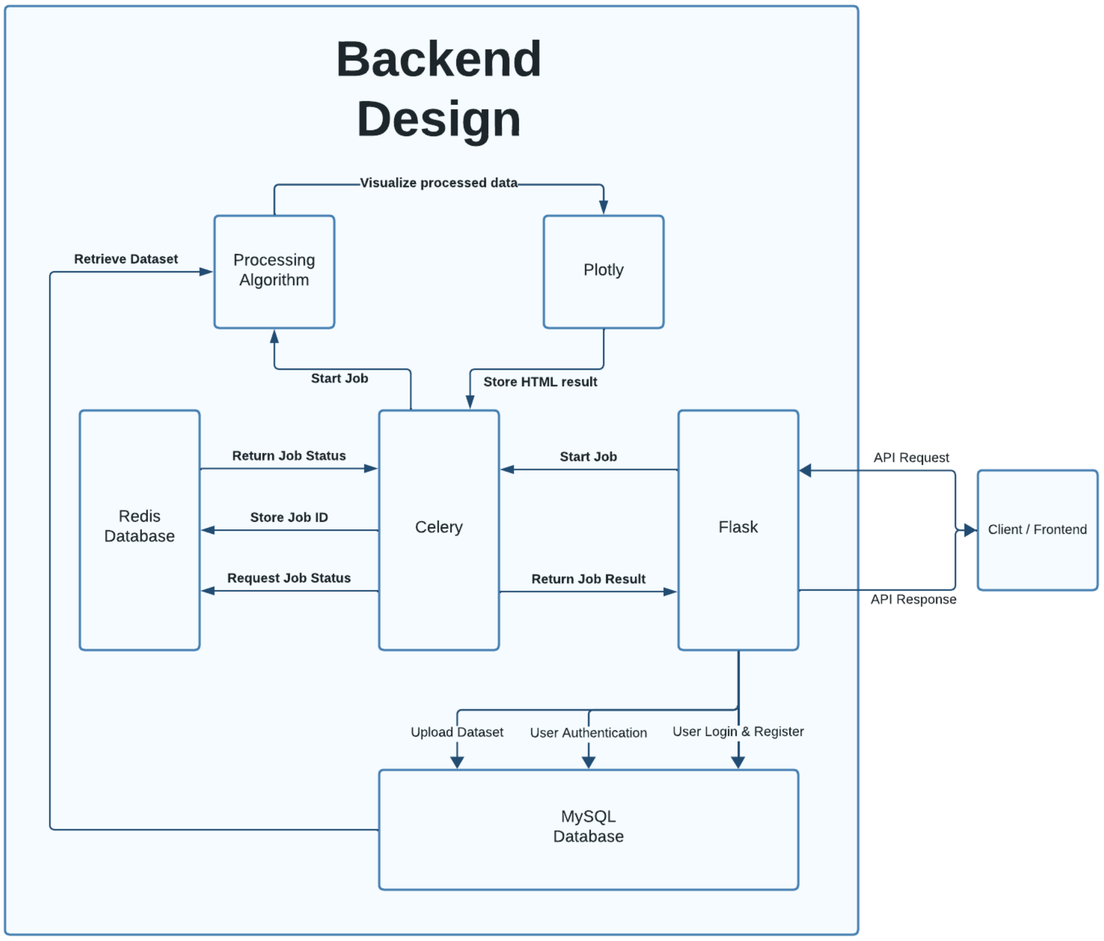


Figure 15. Data Visualization

For our visualization we put together a prototype of our visualization software that we created using the Plotly library that we discussed in previous sections. This protype was created using example data that was sent to us by our client and would be a typical use case scenario.

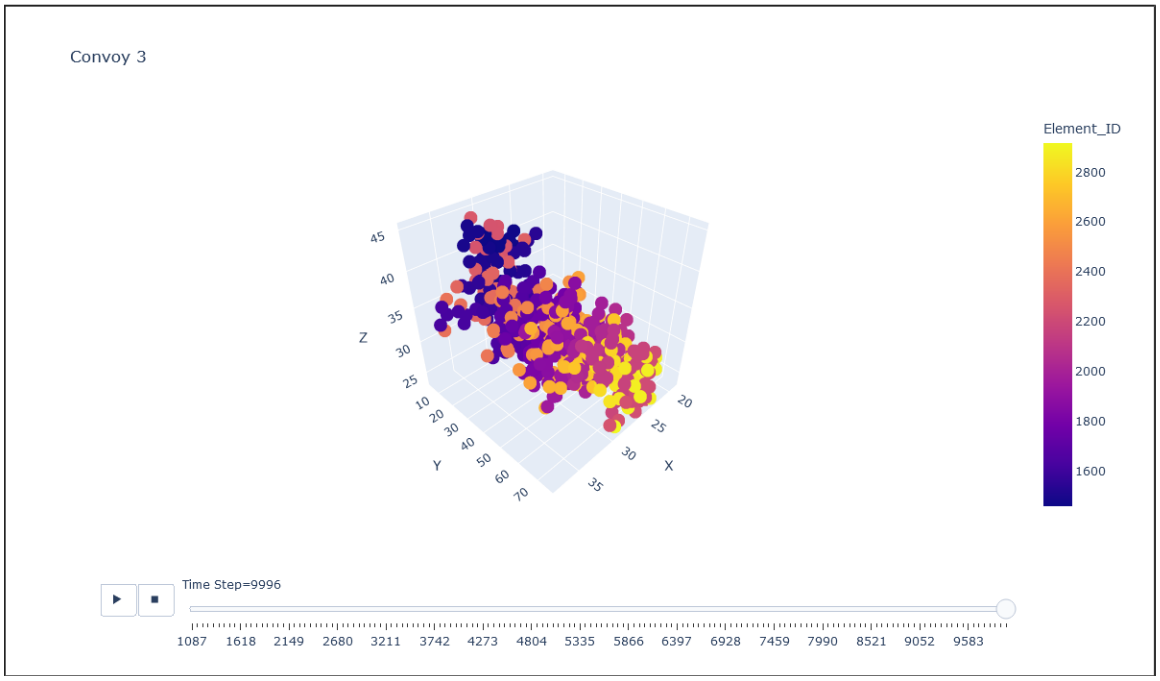


Figure 16. Data Visualization Prototype

4.8 Technology Considerations

When making the design decision on the backend language, we analyzed the strengths and weaknesses of both Java Spring and Python Flask. We ended up landing on Flask due to its good performance, light weight, and ability to work with many visualizations and asynchronous jobs. There were some disadvantages including being an older framework, and a lack of Flask experience within our group. We could have gone for Java Spring, which would have been easier to get started with due to more group experience, however we determined the advantages of Flask outweighed those of Spring.

Another decision that was made was with our visualization technique. It was initially suggested that we try to use VMD, but after further research we found a much better solution – plot.ly. Plot.ly is a python library that allows you to create a visualization similar to what is possible with VMD, however it is lighter and works really well with our current design, so we opted to choose plot.ly.

Finally, we had the opportunity to choose either React or Vue for the frontend framework. They had very similar strengths and weaknesses but given that our group has more experience in React, we decided to use React.

4.9 Design Analysis

Our proposed design from 4.7 ended up working. It succeeded due to our careful iteration over each of our design decisions. While our design works, we could potentially make it clearer for those reading over this document and trying to interpret our solutions through our design diagram. For example, we have a much more detailed design diagram of the backend in 4.7.2, and more information in that could be added into our initial design for more clarity.

5 Testing

5.1 Unit Testing

Frontend

- Components
- Forms
- Buttons
- Functions
- Methods

Backend

- API endpoints
    - Job Queue endpoints
    - User function endpoints
- Functions
    - Algorithm functions
- Methods

For the frontend we plan to use jest and React's built in testing library to test all our essential react components, primarily to ensure that form and field validation is working correctly. For the backend user and visualization APIs, we will need to use a third-party library called PyTest to unit test our backend.

5.2 Interface Testing

The main interface in our design is our React UI which consists of multiple components rendered together. React fortunately has built in test utilities that work great for interface testing. Our backend also has the API interface that allows the frontend to communicate with the backend. Again, we should be able to use PyTest and data dependency injection to ensure that data is being processed and returned in the correct format.

5.3 Integration Testing

There are many integration paths that will need to be tested in our application such as user authentication, data retrieval and data transformation, and state management. All these examples require “round trip” data paths that flow through the entire system. A great tool to test these paths is Cypress as we can ensure that all components in the system are interacting correctly with one another.

5.4 System Testing

System testing will be carried out by modeling pre-calculated models to check that the system works. The system testing will use unit tests from the algorithm and visualization API and ensure that the transfer from the backend to frontend integration is working properly. It will model basic user interactions using UI tests. The entire process will be executed to ensure that the software meets the basic functional requirements provided by the Client. This will all be executed by the CI/CD pipeline process on GitLab and will report data on failures and successes.

5.5 Regression Testing

Regression testing will be performed throughout the CI/CD pipeline where builds will be compiled, tested, and eventually published to ensure that all parts are working together. We will be able to see points of failure in the system when all the parts are integrated together as a part of the CI/CD pipeline. More specifically, we will be using the GitLab CI/CD pipeline to carry this out.

5.6 Acceptance Testing

Acceptable testing will be performed by both unit testing and client confirmation. The unit tests will test the algorithms and parameters to ensure that it produces the expected output. The clients will analyze and review the output of the program to ensure compliance with their functional requirements and accuracy.

5.7 Security Testing (if applicable)

In our design we are going to be prioritizing testing the functionality of the system as we have a limited time to implement and test. As such, if time permits, we will ensure that no major security concerns exist in the system. We could also use source code analysis tools to scan for any known vulnerabilities in our system. Mainly we want to ensure that our login system does not let unauthorized users into the system. One in the system we also want to ensure that the current user can only access data that belongs to them/their organization.

5.8 Results

As with any software product testing can help with various aspects of concern such as:

- Identifying and fixing bugs
- Verifying functionality

- Validating requirements
- Ensuring components can communicate with each other.

Testing is a vital aspect of our project plan which is why we have budgeted time in our project plan to implement the described tests. Actual results will be provided once the system is implemented, and tests are written.

5.9 Criteria of Success

The criteria for success of our testing will be determined by the functionality of the compiled project.

6 Implementation

- Ensure that all team members have appropriate software and applications required to start the implementation of the project at the start of the semester.
- Ensure that any data or servers that the team needs access to are working properly and can be accessed remotely so that the team may work with them as needed.
- Ensure that the Continuous Integration and Development pipeline is set up as needed to ensure that dynamic development and testing can be performed at the start of the semester.
- Implement features as described in figure 5

7 Professionalism

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

7.1 Areas of Responsibility

Area of responsibility	Definition	NSPE Version	Description	Performance
Work competence	Perform work of high quality, integrity, timeliness, and professional competence.	Perform services only in Areas of their competence; avoid deceptive acts.	Product - software engineers shall ensure that their products and related modifications meet the highest professional standards possible.	High
Financial responsibility	Deliver products and services of realizable value and at reasonable costs.	Act for each employer or client as faithful agents or trustees.	Management - Software engineering managers and leaders shall subscribe to and promote an ethical approach to delivering products at a reasonable cost	N/A

Communication honesty	Report work truthfully, without deception, and understandably to stakeholders.	Issue public statements only in an objective and truthful manner; Avoid deceptive acts.	Judgment - Software engineers shall maintain integrity and independence in their professional judgment.	High
Health, safety, wellbeing	Minimize risks to safety, health, and well-being of stakeholders.	Hold paramount the safety, health, and welfare of the public.	Public - Software engineers shall act consistently with the public's safety.	N/A
Property ownership	Respect property, ideas, and information of clients and others.	Act for each employer or client as faithful agents Or trustees	Client and Employer - Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest	Medium
Sustainability	Protect environment and natural resources locally and globally		Public - Software engineers shall create products that do not unnecessarily impact the environment or use unnecessary resources	Medium

Figure 17. Areas of Responsibility

7.2 Project Specific Professional Responsibility Areas

Work competence (*High*)

- This area of responsibility directly applies to our project and probably applies to every project. Our goal as a team is to create a system that meets our client's needs and addresses each requirement expressed by the client.

Financial responsibility (*N/A*)

- Because this project is for a student assignment, and we are not commercializing it this area of responsibility does not fall within our project's context.

Communication honesty (*High*)

- As the implementation of our project requires constant communication with our clients/advisors we need to ensure that we are reporting work accurately and consistently

Health, safety, wellbeing (*N/A*)

- This project doesn't necessarily have direct impact on the safety and wellbeing of stakeholders.

Property ownership (*Medium*)

- Algorithms are provided and created by the client, so acknowledging this may be important to the overall project.

Sustainability (*Medium*)

- Our project has the potential to use a lot of energy as the algorithms that we are implementing can be very computationally intensive.

7.2 Most Applicable Professional Responsibility Area

Most of the responsibility areas partially relate to our project, however work competence is one that holds the most importance in our project. We desire to deliver a high-quality project that not only meets the client's expectations, but also exceeds it. This includes finishing tasks in a timely and professional manner to uphold the integrity of the team and the project.

8 Closing Material

8.1 Discussion

The main result of the product is the design specifications and content. Our group has communicated and worked alongside the client to create a design that fits their requirements and vision for a tool to make modeling convoys easier. While implementation has yet to be made, small experiments in implementation of core features have been tested to be successful.

8.2 Conclusion

So far, we have completed all the necessary prerequisites including determining which teammate will take on which roles, how long each task is expected to take, the technique that will be used in implementing each feature, how each feature will be tested, and what the ideal frontend/backend will look like. Our main goal going into this was to establish an outline for anything that needs to be done so all that remains is writing the code. All design decisions should be made beforehand. We were temporarily constrained from achieving this goal due to our inexperience with detecting convoys and lack of clarity on the algorithms, but through communication and hard work, we were able to complete everything we needed.

8.3 References

[1] IEEE SA, "IEEE Standards Association," *IEEE Standards Association*. https://standards.ieee.org/ieee/12207/5672/

[2] IEEE SA, "IEEE Standards Association," *IEEE Standards Association*. https://standards.ieee.org/ieee/90003/7197/

[3] "ISO/IEC 27001:2022," *ISO*, Feb. 02, 2023. https://www.iso.org/standard/27001

[4] "A standard for software documentation," *IEEE Journals & Magazine | IEEE Xplore*, Oct. 01, 1997. https://ieeexplore.ieee.org/document/625327

[5] "React." https://react.dev/

[6] "Welcome to Flask — Flask Documentation (3.0.X)." https://flask.palletsprojects.com/en/3.0.x/

[7] “Plotly.” https://plotly.com/python/

[8] H. Jeung, M. L. Yiu, X. Zhou, C. S. Jensen, and H. T. Shen, “Discovery of convoys in trajectory databases,” *Proceedings of the VLDB Endowment*, vol. 1, no. 1, pp. 1068–1080, Aug. 2008, doi: 10.14778/1453856.1453971.

[9] J. Gudmundsson and M. Van Kreveld, “Computing longest duration flocks in trajectory data,” Nov. 2006, doi: 10.1145/1183471.1183479.

[10] “Introduction to Celery — Celery 5.3.6 documentation.” https://docs.celeryq.dev/en/stable/getting-started/introduction.html

[11] “Redis,” *Redis*. https://redis.io/

[12] “Docker: Accelerated Container Application Development,” *Docker*, Oct. 18, 2023. https://www.docker.com/

[13] “MySQL.” https://www.mysql.com/

8.4 Team Contract

Team Name sdmay24-19

Team Members:

1) Ayden Albertsen 2) Benjamin Hall

3) Ben McClannahan 4) TJ Thielen

Team Procedures

1. Day, time, and location (face-to-face or virtual) for regular team meetings:
    a. Virtual - Thursday @ 3:30
2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):
    a. Email/Discord
3. Decision-making policy (e.g., consensus, majority vote):
    a. Majority Vote
4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):
    a. Git Issues
    b. Google Docs

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:
    a. Each member is expected to attend all meetings unless extenuating circumstances are applicable.
2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:

   a. Each team member will commit to a similar amount of development effort of git issues as voted on by the team.
   b. Each member of the team is expected to contribute to written assignments on an even basis.
   c. Once a deadline has been decided to complete a task, the team member must complete it by the decided deadline.
3. Expected level of communication with other team members:
   a. Each team member of the team is expected to communicate regularly through discord and email about progress, blockers, or any other relevant information about completion of their assigned work
4. Expected level of commitment to team decisions and tasks:
   a. Each team member is expected to commit to a similar effort of work

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):
   a. Team Leader?
   b. Frontend Leader?
   c. Backend Leader?
      i. Benjamin Hall
   d. Client/Advisor Communication
      i. Ayden Albertsen
2. Strategies for supporting and guiding the work of all team members:
   a. Team members are encouraged to discuss tasks with other team members as well as reach out to the team TA if they need assistance with their work.
3. Strategies for recognizing the contributions of all team members:
   a. Each task should be logged and effort rated in GitLab, even non-development work.

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.
   a. Ayden Albertsen - 2+ Years of internship experience working as a full stack developer for an online banking system using React frontend and Spring backend. Experience working on an AGILE development team, attending AGILE ceremonies, completing and writing user stories.
   b. Benjamin Hall - No formal experience directly in software development, but Cybersecurity Internships that required knowledge of Python, Javascript, and AGILE development. I've worked on software projects for about 7 years now, starting with year-long club projects in high school.
   c. Ben McClannahan - Several years experience working with Python, Java, and C++. Have been working as an intern at Collins Aerospace.
   d. TJ Thielen - Internship at Source Allies doing software development for the past 2 summers that required using typescript and python, with a React frontend. Practiced AGILE development including writing stories, doing TDD, and attending and leading AGILE ceremonies.
2. Strategies for encouraging and support contributions and ideas from all team members:

    a. Team members are expected to take on tasks that they may not be initially comfortable with, in these cases it's encouraged to reach out to other team members, TAs, or advisors to help them understand and complete the task.

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. In the case of a disagreement, collaboration issues, or inclusion issues, the team will first try and resolve the problem internally via the team leader. If the team leader is unable to resolve the conflict, issues will then be brought up to the TA.

Goal-Setting, Planning, and Execution

1. Team goals for this semester:
    a. Complete the design document and get an A grade
    b. Learn new technologies
    c. Exposure to working on a team
2. Strategies for planning and assigning individual and team work:
    a. In team meetings, work will be divided into git issues, each issue will be assigned an effort value, team members will be assigned a similar total effort of stories.
3. Strategies for keeping on task:
    a. In team meetings, each member will give a progress update on the status of their issues and when they expect to complete them by. Progress of each issue will be tracked in GitLab.

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?
    a. Any infractions to this contract will be immediately brought to the attention of TA or instructor if applicable.
2. What will your team do if the infractions continue?
    a. If infractions continue, the team member will receive poor performance reviews and the TA/instructor will be notified of the team member's continued infractions.

****

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) Ayden Albertsen DATE 9/5/23

2) Benjamin Hall DATE 9/7/23

3) Ben McClannahan DATE 9/8/23

4) TJ Thielen DATE 9/10/23