3 Project Plan

3.1 Project Management/Tracking Procedures

Our project will implement the Agile methodology for project management and task completion. We chose this method because it is centered around frequent scrum meetings which allow for group discussion. Since we have such a small team and our project is mainly software development related, these scrum meetings will allow our team to gather insight from other team members as software-related obstacles arise.

Since many of the tasks we have designated will have tasks that build on each other, the Agile methodology will allow us to work from the beginning and power through each task in the scrum without strict deadlines. Although we have attempted to make a rough estimate of time that will need to be allocated for each task, it is difficult to predict where specific obstacles with integration and programming will fall. Because of these reasons, the Waterfall method was less appealing to our team since the time estimates are not as reliable with a mainly software development project where we are utilizing new tools we are not familiar with.

Our team will track progress through Github with individual folders for front-end and backend work. Since our project will utilize a VM and MySQL database, we will continue documenting progress in Google Docs and communicating via Text for information that is not on Github. We will follow the task decomposition outlined in section 3.2 for knowledge of which task should be started after completing a previous one.

3.2 Task Decomposition

Our whereabouts web application can be broken down into two major subsections - frontend and backend. Where the frontend, client side, will be responsible for managing user input, while the backend, server side, will run whereabout algorithm computation and handle database interaction. Since our team is adopting an Agile approach, team members will begin by working on the basic user authentication tasks and continue down the tasks listed in the decomposition legend - Figure 1. The major task groupings and the corresponding sub-tasks are highlighted below. Figure 1 illustrates the major tasks and how they interact from a systems-level perspective.

- Frontend
    - User authentication
        - Create login display, and form for accepting user input
        - Submit credentials to backend for authentication
    - Dataset uploads and format specification process
        - Create dataset upload and configuration display
        - Complete local file browse and select functionality
        - Check uploaded files formatting for compatibility
    - Query creation, configuration, and submission requests
        - Create query creation and configuration display
        - Query validator, prevent “long” query runtimes as specified in constraints section.

- Visualization window and underlying engine
  - Setup inclusion of necessary plotting libraries
  - Add visualization window the the interface display
  - Create plotting handlers for datasets and whereabout data

- Backend
  - Setup Database
  - Authenticate login/signup
    - Query database for submitted credentials or add new user
    - Send error message or authentication token
  - Commit datasets to database along with the format specification
    - Query database and add new entries
  - Setup query endpoints for supported query types (i.e. range, contact)
  - Implement whereabout algorithms and package results to be sent to the frontend for visualization.

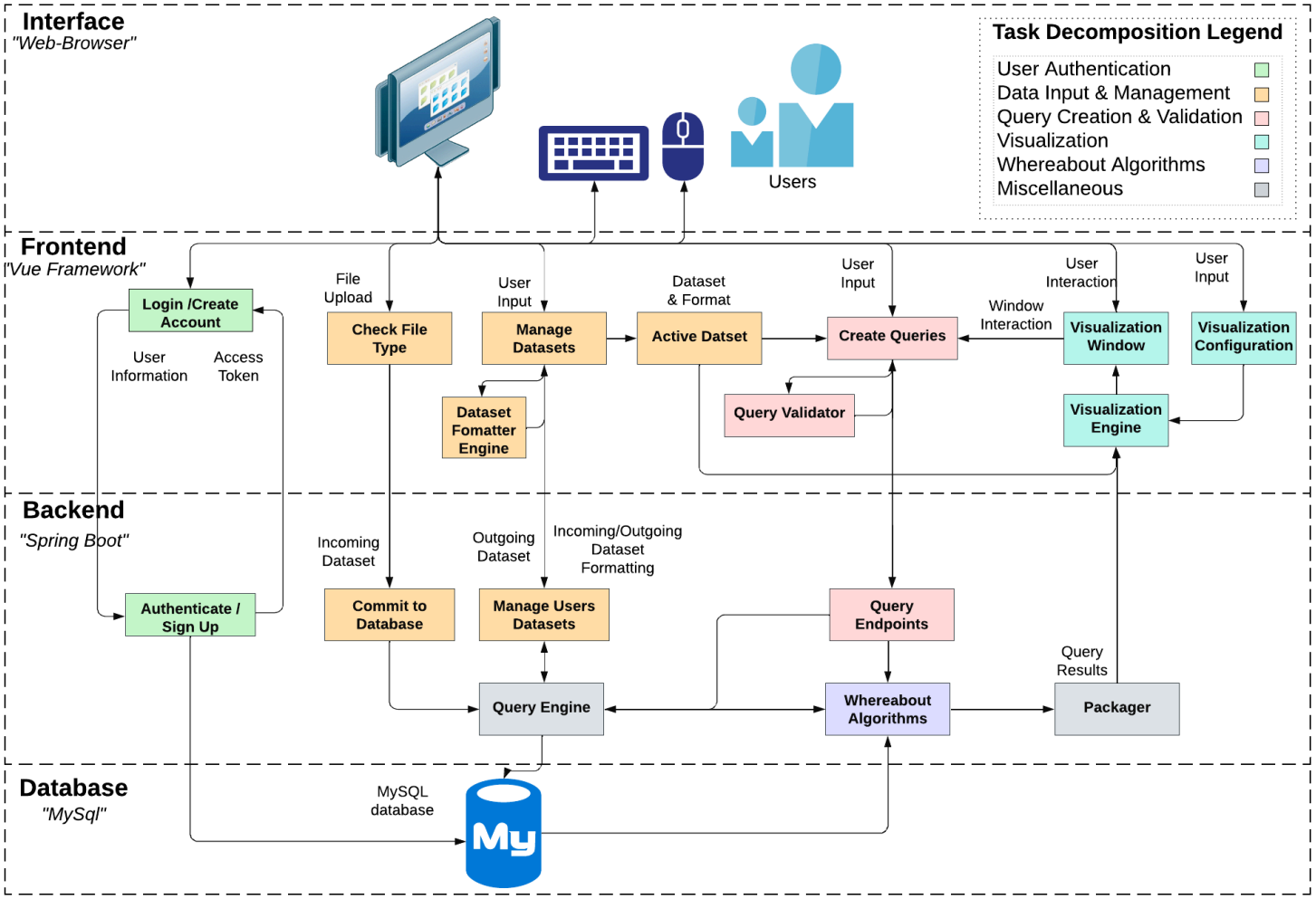


Figure 1 - *Systems diagram perspective of task decomposition.*

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

This section seeks to set progres goals and determine what successful accomplishment should look like.

Milestones

Shown below are the planned milestones we expect to accomplish. The milestones generated are based on the tasks and subtasks listed in section 3.2. For estimated completion of each task, reference the Gantt charts included in section 3.4.

Spring Semester

- Complete project frameworks and libraries
- Define and identify backend API, the contract between function calls from frontend and backend.
- Setup code base (Git, Vue Project)
- Develop preliminary versions of:
    - UI components
    - Database setup
- Finalize design documentation

Fall Semester

- Frontend
    - Render HTML passed from backend to frontend interface
    - Update the HTML from Vue application instance
    - Successfully make a request from frontend to backend
    - User can load file through interface
    - Can distinguish a loaded files format compatibility (i.e. able to be parsed with single delimiter specified)
    - Users can view a dataset through the interface and specify columns/rows (i.e. what data is - positional, timestamp, label, etc.)
    - Displaying a visualization window using plot libraries provided examples
    - Producing an aesthetic visualization window with data points from user dataset
    - Working whereabout query interface that supports a single query algorithm and type where data is manual entered
    - Visualization engine can take whereabout algorithm output and render meaningful interpretation to the visualization window
    - Whereabout query interface supporting multiple query types
    - Visualization engine supports visualization of multiple query types
- Backend
    - Store and retrieve test credentials in JSON format to database
    - Successfully register a new user to the database
    - Correctly authenticate an existing user by serving authentication token
    - Store three datasets with unique formatting into database
    - Successfully upload a new dataset and store to database
    - Whereabout query endpoint can successfully take an input query from frontend and fetch necessary resources needed by the corresponding whereabout algorithm

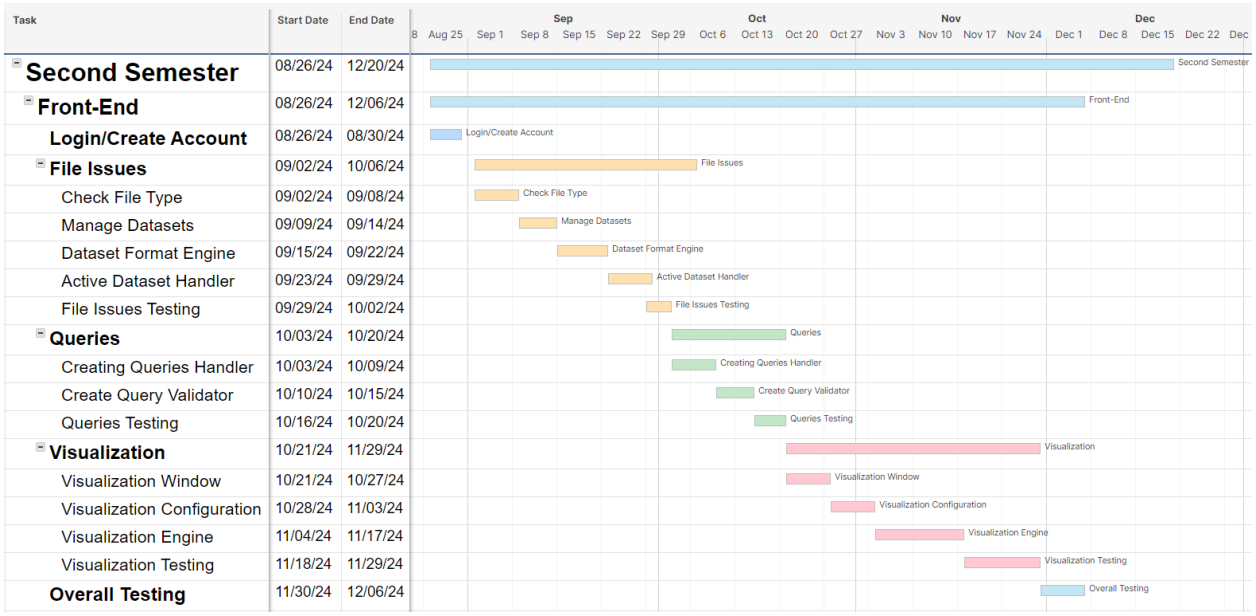
- Single whereabout algorithm implemented and producing correct output
- Whereabout algorithm output interpreted and formatted in accordance to frontend needs
- Backend supports multiple query types and produces correct output

Metrics and Evaluation Criteria

To measure the progress of our efforts and milestones completion we will be using the following metrics:

- User interface components will complete the given action within a timeframe of a few seconds as the upper-bound. Action will be verified through use of print statements or an equivalent.
- Executing algorithms on the backend should complete their action within a timeframe given by our constraints section - no longer than 30 seconds. A timer will monitor the duration of execution.
- Visualizations of whereabout algorithm output should be correct. This will be evaluated by using known scenarios (i.e. test data provided by the client). This methodology will also be used to verify algorithm implementation correctness.

3.4 Project Timeline/Schedule



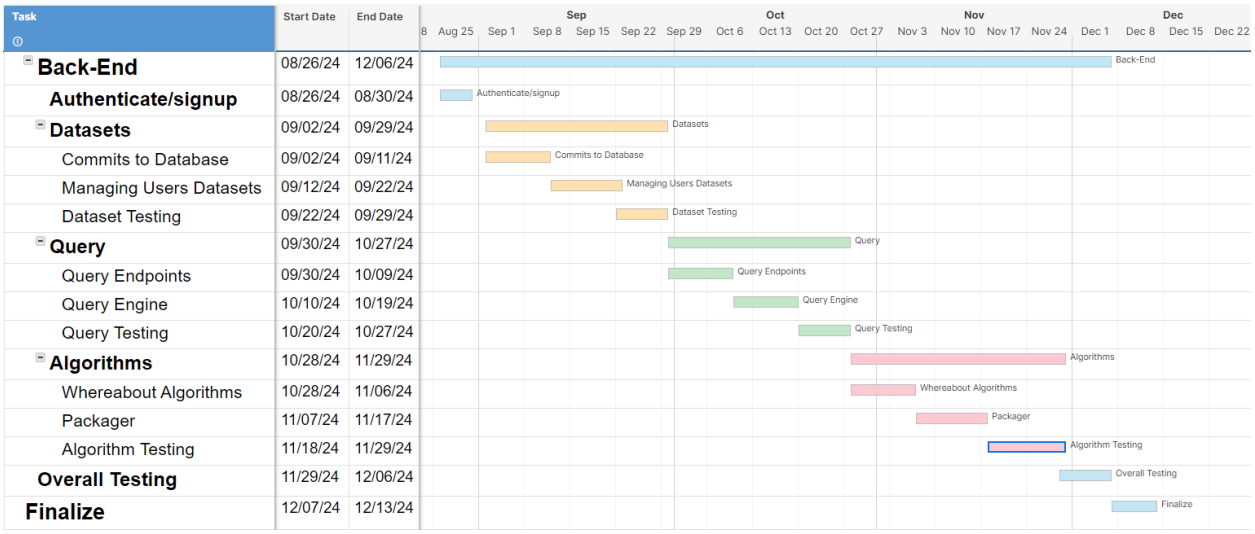


Figure 2 - *Gantt Chart*

3.5 Risks And Risk Management/Mitigation

The following section contains a breakdown of the possible risk, description of risk, probability of occurrence, and correlated mitigation.

- Frontend
  - User authentication
    - Security vulnerabilities/issues, compatibility issues with certain devices or browsers.
    - Probability: .25
    - Testing across multiple platforms during development to ensure compatibility.
  - Dataset uploads and format specifications process
    - uploading large datasets could lead to eventual performance issues, also file formatting might become problematic.
    - Probability: .3
    - Limit to only accept one type of file for uploading data, optimize server side processing to help handle larger data sets.
  - Query creation, configuration, and submission requests
    - More complex/extensive queries may lead to performance issues, failure to validate and process certain types of query configurations for different desired outcomes.
    - Probability: .35
    - Optimizing query execution and extensive testing of all intended end user combinations/types of queries from data sets.

    - Visualization window and underlying engine
        - Performance degradation with larger data sets and local device constraints, integration issues with visualization tools such as pan, zoom, and move in several axes
        - Probability: .45
        - Testing tool compatibility and implementing strategies to manage larger datasets more efficiently without sacrificing performance.

- Backend
    - Setup database
        - Issues with scaling in terms of volume of data, limitations in speed might create bottlenecks.
        - Probability: .5
        - Performance testing and optimization of both database queries and indexes.
    - Authenticate login/signup
        - Security vulnerabilities in terms of authentication
        - Probability: .1
        - Ensuring correct security measures in user login/signup.
    - Commit datasets to database (and format specification)
        - Data integrity issues during insertion into database, compatibility issues with database configurations
        - Probability: .15
        - Testing compatibility across different database environments during development, implement transaction integrity checks for data entering the database.
    - Setup query endpoints for supported query types
        - Inefficiency processing of different query types might lead to performance bottleneck issues, full support missing from all different required query types.
        - Probability: .4
        - Continuous refinement and adjustment of query endpoint functionalities based on user feedback and issues/concerns discovered during overall development.
    - Implement whereabouts algorithms to send to frontend for visualization
        - Ensuring correct visualization for easy user interpretation of data sets, greater complexity of algorithm can result in performance degradation, integration issues with backend architecture.
        - Probability: .6
        - Performance optimization in optimizing algorithm(s) to ensure most efficiency when it comes to computation, involving continuous monitoring and logging methods to detect anomalies in behavior of algorithm(s), implement extensive documentation to aid in understanding and troubleshooting eventual issues.

3.6 Personnel Effort Requirements

The table below contains a list of tasks, estimated hours necessary to complete, and a brief explanation of each task. Each task's hours may be divided between multiple team members as will be determined by the completion of blocking tasks.

Task	**Estimated Hours (total not individual)**	**Explanation**
Login/Create Account	10	Setting up GUI and Backend Calls to the Spring Boot server.
Check File Type	15	Make sure the file type is correct and formatting in the new file is correct. Send dataset to Spring boot server.
Manage Datasets	20	Receive datasets from the database that the user selects. Update selected dataset if necessary.
Dataset Format Engine	25	Checks that formatting in the updated dataset is correct.
Active Dataset Handler	20	Handles Current dataset being used for visualization.
File Issues Testing	15	Test all File Issues from Check File to Dataset Handlers works individually and when combined together for the front-end.
Creating Queries Handler	25	Selection GUI and background processes for creating Queries related to current dataset.
Create Query Validator	20	Checks that Query from user is valid to be sent to backend and processed in Query endpoints.
Queries Testing	15	Testing For Creating and validating to see that they work individually together and with the file modules.
Visualization Window	20	Create the window for the visualization of the datasets with interaction for the user and with the created query for it.

Visualization Configuration	15	GUI and Configuration methods for visualization of Current dataset with options for how the user wants to customize the visualization
Visualization Engine	40	Setting up Mapbox tool to use the modified query from the backend, the active dataset, and the visualization configuration from the user to visualize in the way the user wants
Visualization Testing	30	Testing visualization parts individually and checking that it integrates with the other front end modules correctly and effectively
Authenticate/signup	10	Create backend request receivers for Authenticating user signin and creating new users. Create a MySQL database.
Commits to Database	20	Receives new datasets from the front end and sends them to the query engine to be stored in the database.
Manage Users Datasets	20	Receives dataset request from user and sends info to query engine to get dataset. Sends chosen dataset from users database, and receives user updated datasets from the front end and sends them to the query engine to update dataset in database.
Dataset Testing	20	Test that all Receiving and Sending from the front end is working as required and will work with the front end.
Query Endpoints	25	Takes received query from front end and sends to query engine for saving to database and use in the algorithms.
Query Engine	25	Saves received data to MySQL. returns requested info from MySQL database.
Query Testing	20	Test that query modules work together and with all connected modules
Whereabout Algorithms	40	Uses user selected algorithm to ready query for visualization.

Packager	40	Packages algorithm output to be sent to Visualization in front end
Algorithm Testing	25	Tests that all algorithms in backend works with query options from the front end

Table 1 - *Tasks and estimated hours to complete.*

3.7 Other Resource Requirements

. Below is a preliminary list of resources that will be necessary to consider in the completion of this project:

- Github Page
- VM
- MySQL database
- Javascript/Springboot Libraries
- Website/Domain Name