Project 2

Jarrod Thomisee

Jarrod.thomisee@snhu.edu

2023-12-10

**Grazioso Salvare Dashboard Project**

**Project Overview**

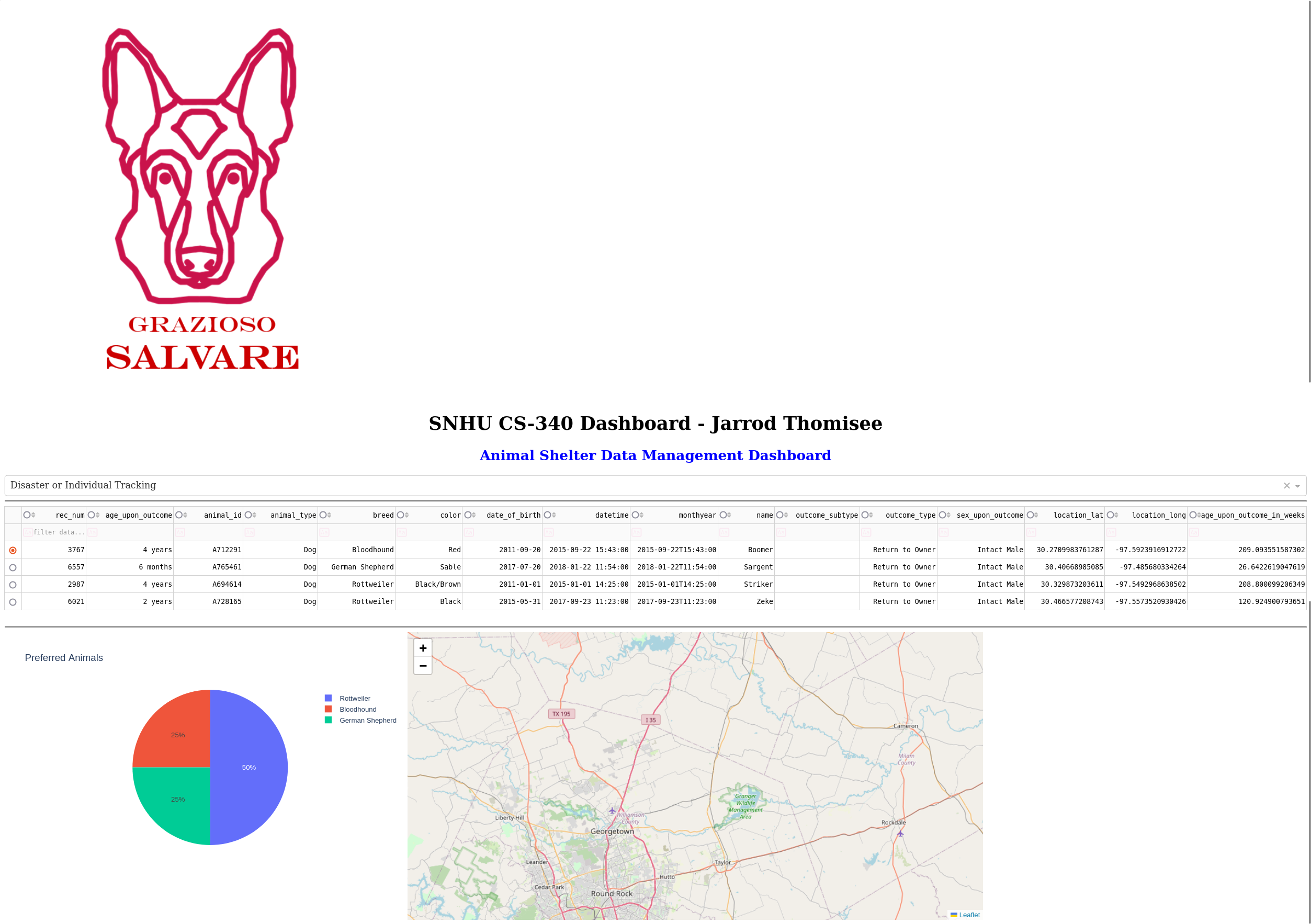
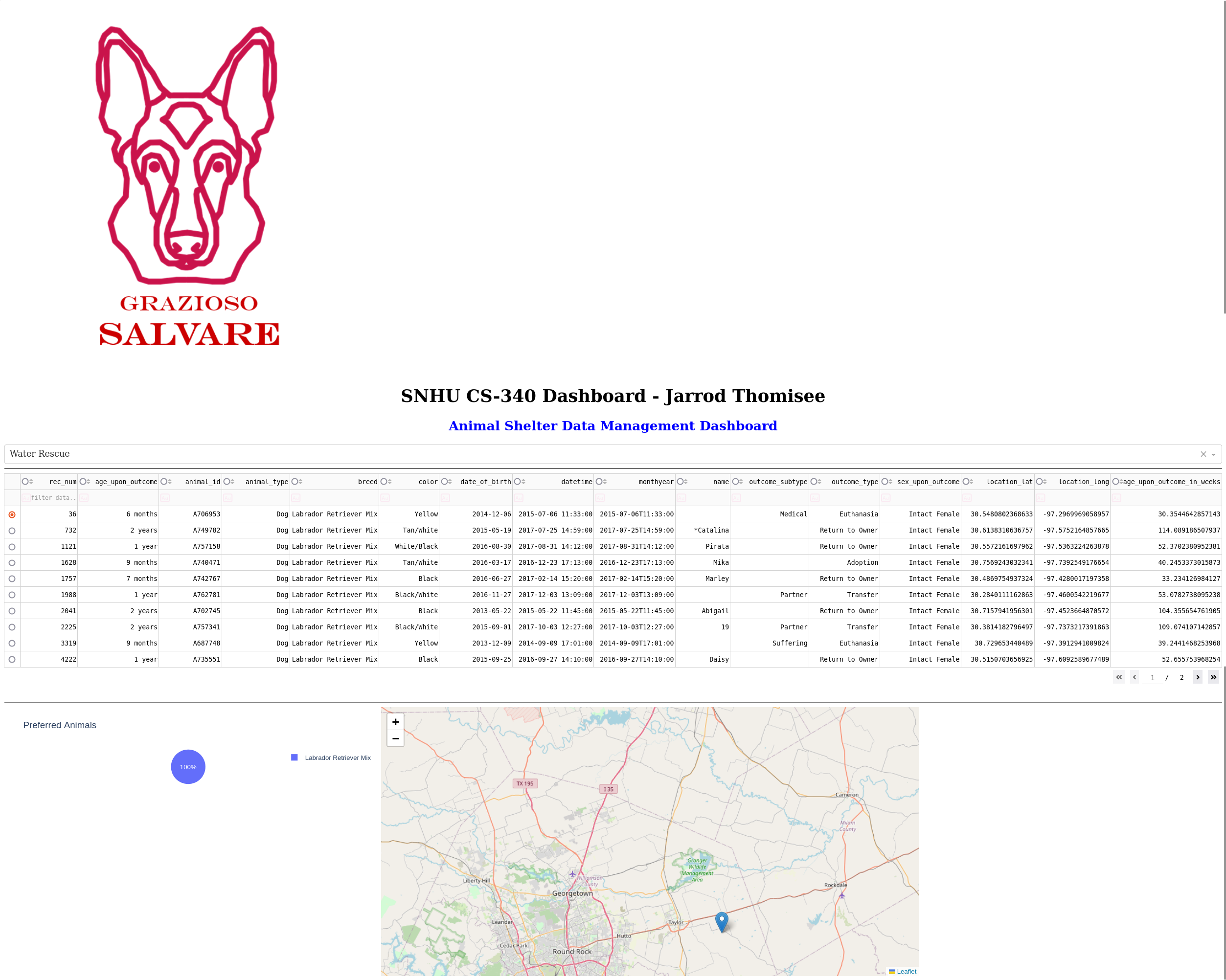
This project involves the development of an interactive web-based dashboard for Grazioso Salvare, a company specializing in training rescue dogs. The dashboard provides a user-friendly interface for accessing and visualizing data from the Austin Animal Center Outcomes dataset.

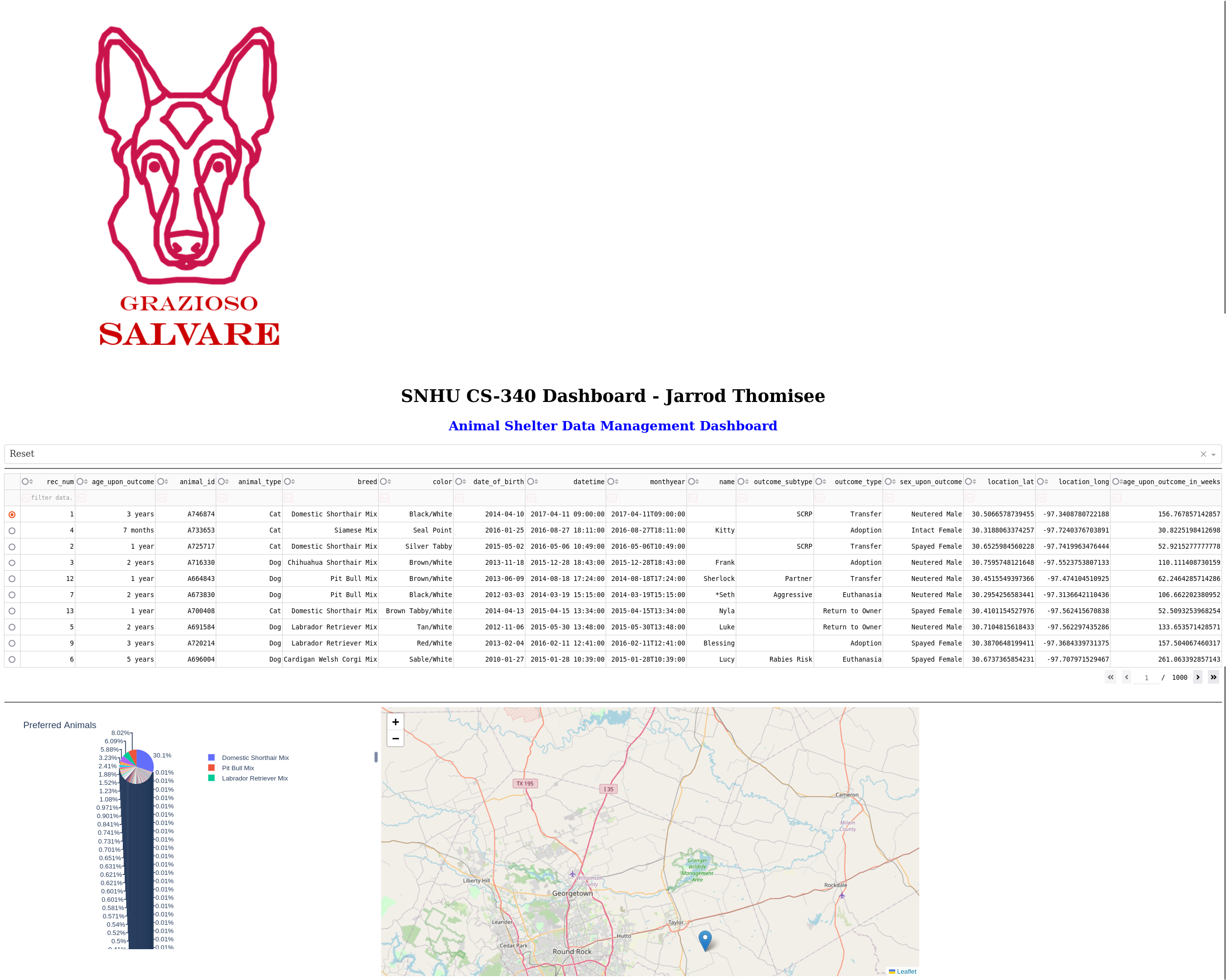
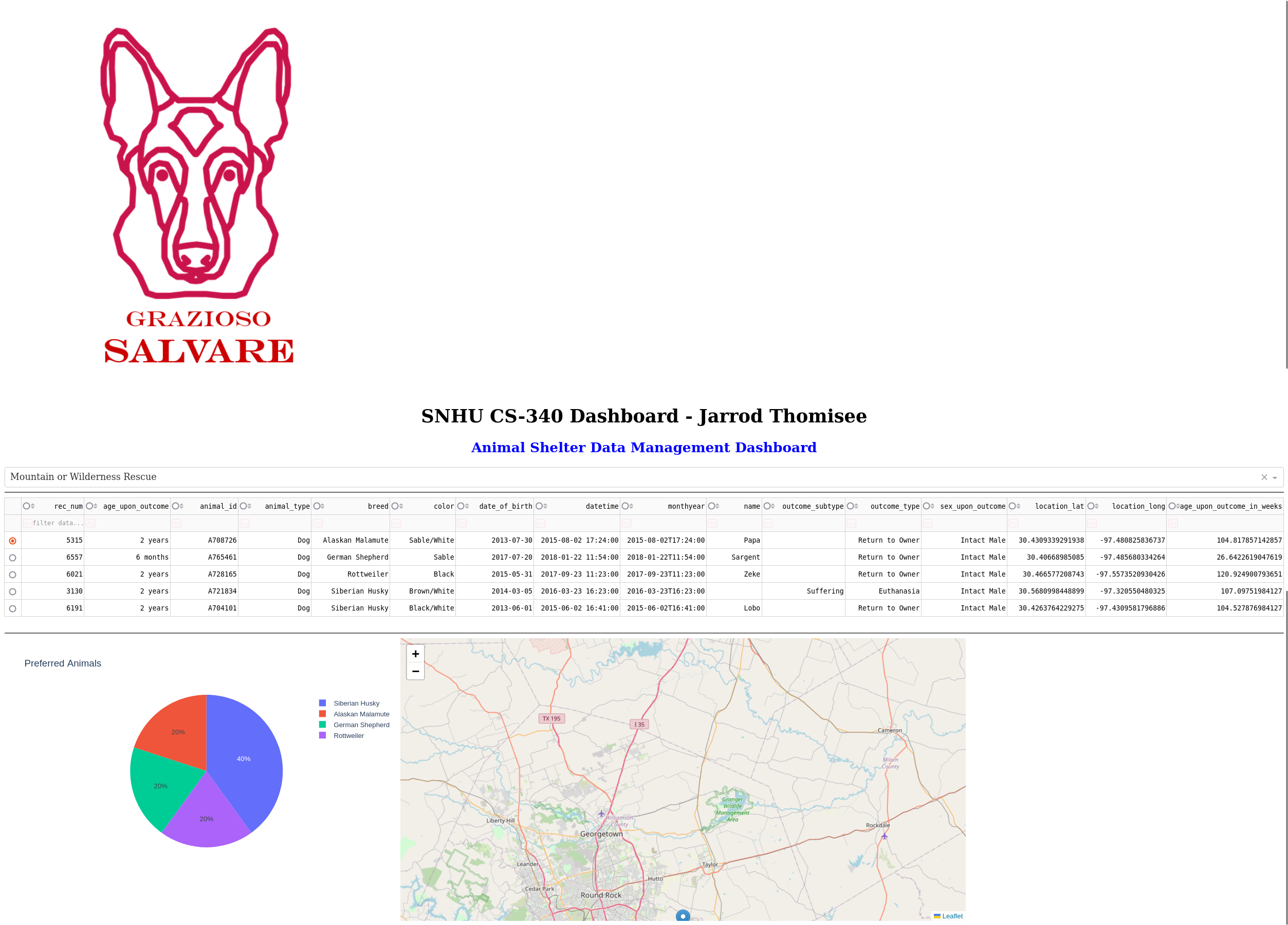
**Required Functionality**

The dashboard is designed to provide the following functionalities:

* Interactive filters for selecting specific types of rescue dogs (e.g., Water Rescue, Mountain or Wilderness Rescue, Disaster or Individual Tracking).
* A dynamic data table that updates based on the selected filters.
* Interactive charts that respond to the filter selection, providing insights into the data.
* Geolocation mapping for visualizing the location data of the animals.

Proof of Functionality



****

**Tools Used**

**MongoDB**

**Rationale**: MongoDB, a NoSQL database, was chosen for its flexibility in handling diverse data types and its scalability. Its dynamic schema is ideal for accommodating the various attributes of the animal data.

**Integration with Python**: MongoDB interfaces seamlessly with Python, allowing for efficient data retrieval and manipulation using PyMongo. This compatibility is crucial for the backend operations of the dashboard.

**Dash Framework**

**Overview**: Dash, a Python web application framework, is utilized to create the frontend of the dashboard.

**View and Controller Structure**: Dash enables the development of a responsive UI (View) and the handling of user interactions (Controller) through callbacks, making it an excellent choice for interactive web applications.

**Additional Tools**

**Plotly**: For creating interactive charts.

**Dash Leaflet**: For geolocation mapping.

**Pandas**: For data manipulation and analysis in Python.

Resource Links

* [MongoDB Documentation](https://docs.mongodb.com/)
* [Dash User Guide](https://dash.plotly.com/introduction)
* [Plotly Python Graphing Library](https://plotly.com/python/)

**Project Development Steps**

1. **Setting up MongoDB**: Configuring the MongoDB database and defining the schema based on the Austin Animal Center Outcomes dataset.
2. **Developing the Backend**: Writing Python code to interact with the MongoDB database, including CRUD operations.
3. **Designing the Dashboard Layout**: Using Dash to create the layout of the web application, including filters, data table, charts, and map.
4. **Implementing Interactivity**: Coding the callback functions in Dash to enable dynamic updating of the dashboard components based on user input.
5. **Testing and Deployment**: Rigorously testing the dashboard to ensure all functionalities work as expected and deploying the final version.

**Challenges**

**Automatically Updating Charts with Filtered Data**

A key challenge in the development of the Grazioso Salvare Dashboard was ensuring that charts updated dynamically and accurately in response to filter selections. The specific goal was to have the charts immediately reflect the first set of data from the filtered results, enhancing the dashboard's interactivity and user engagement.

**Solution**

The solution to this challenge was focused on automating the selection of the first element in the filtered data set:

**Auto-Selecting the First Data Element**: The callback function linked to the filter dropdown was enhanced to automatically select the first entry in the filtered data set. This meant adjusting the logic to not only update the data based on the selected filter but also identify and select the first row of the new data.

**Synchronizing Chart Updates**: By auto-selecting the first data point in the filtered results, the charts were then able to immediately reflect this data. This ensured that upon selecting a filter, the charts would dynamically display information pertinent to the first entry in the filtered data, providing immediate visual feedback to the user.