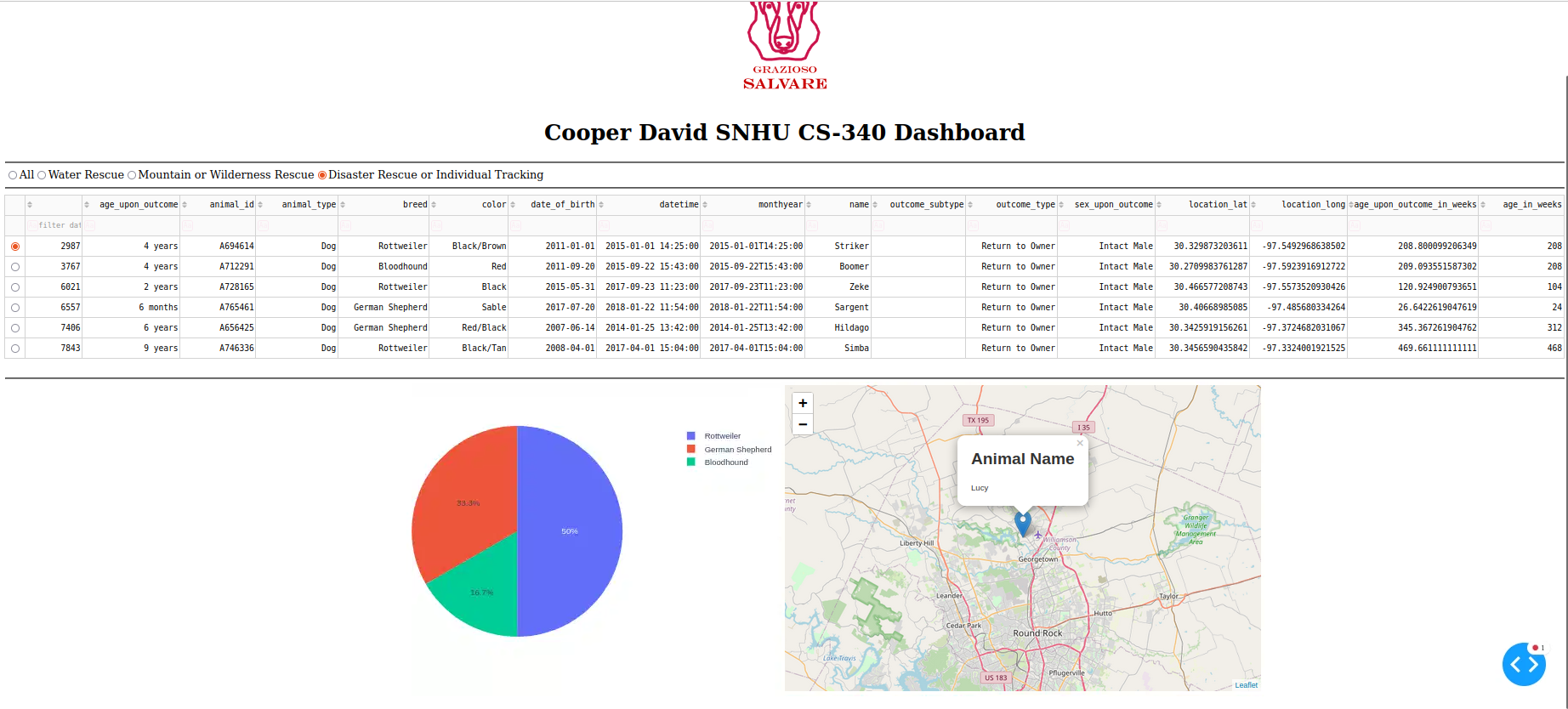
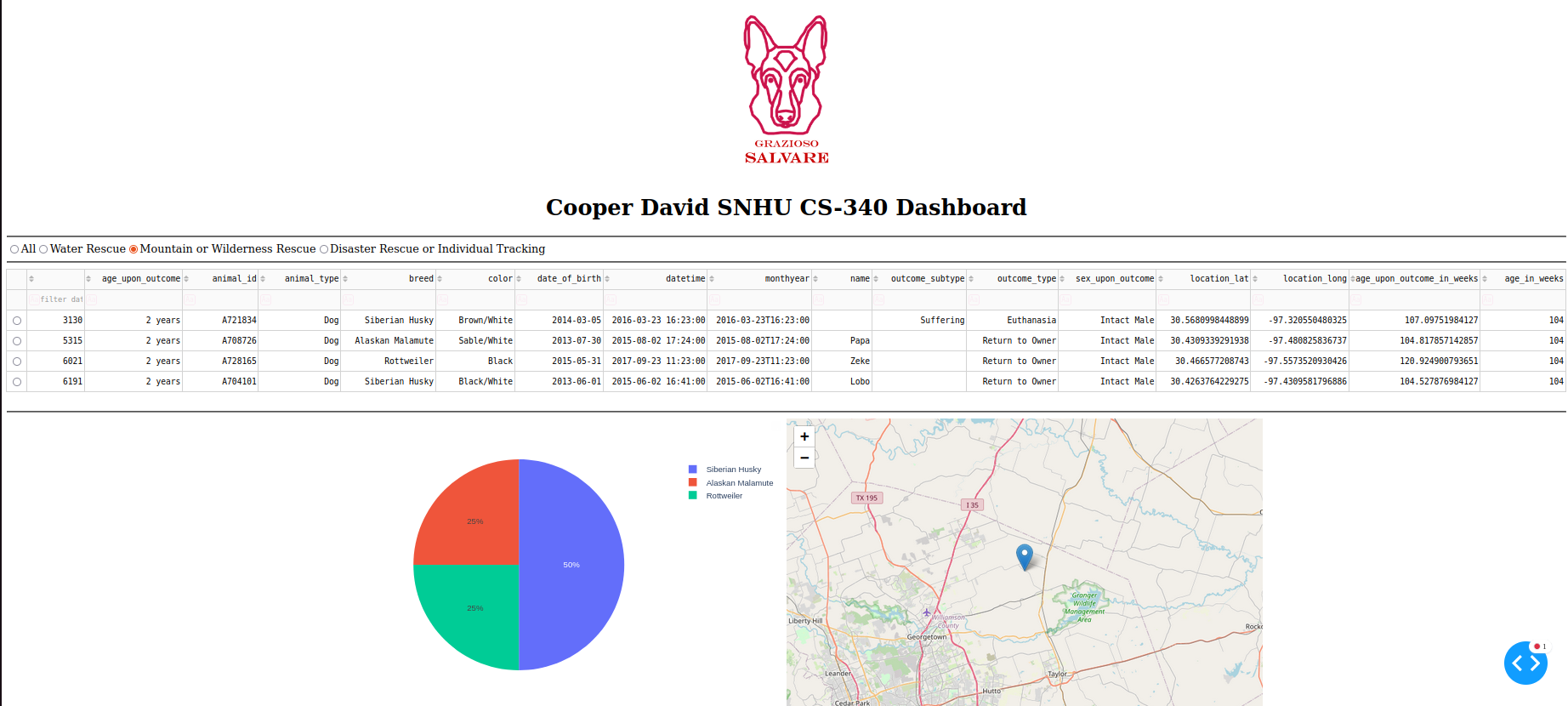
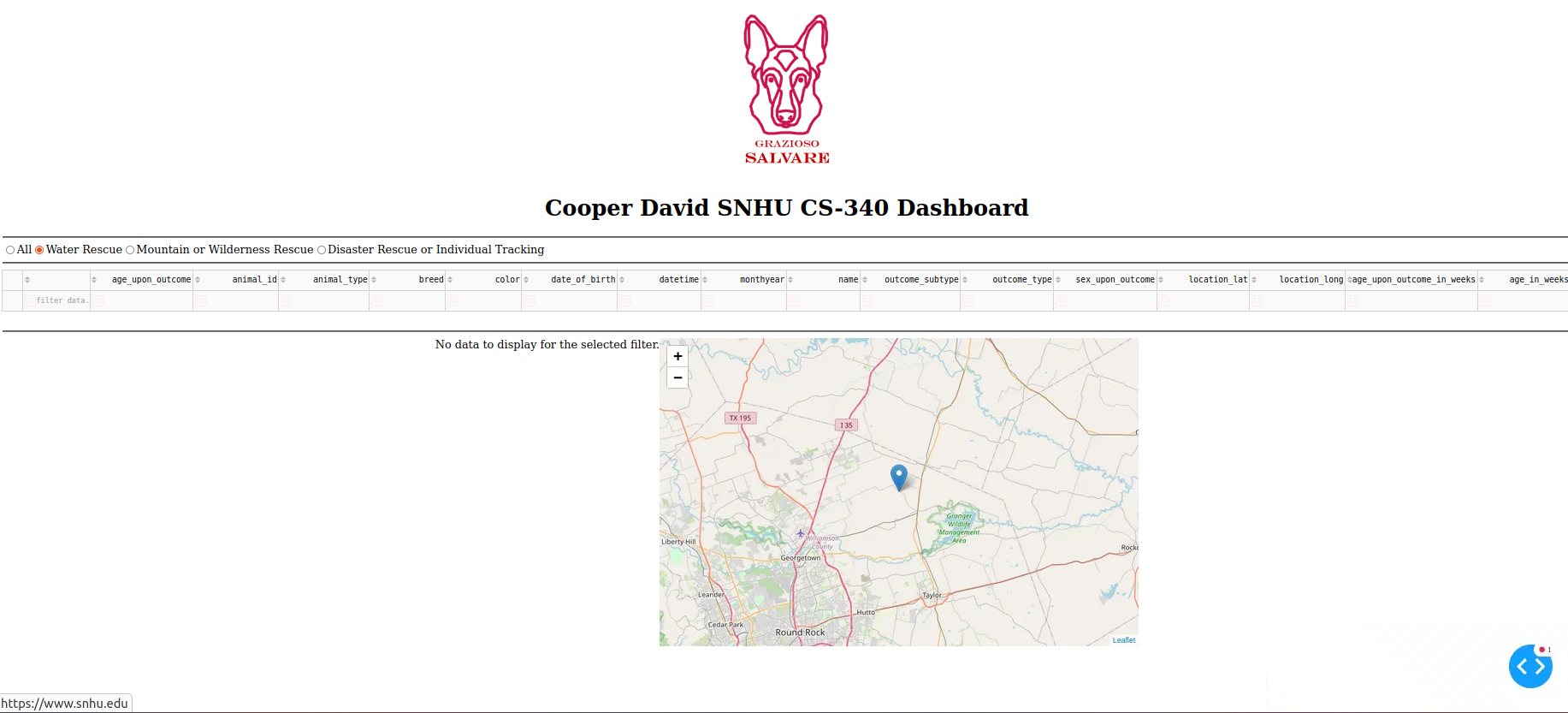
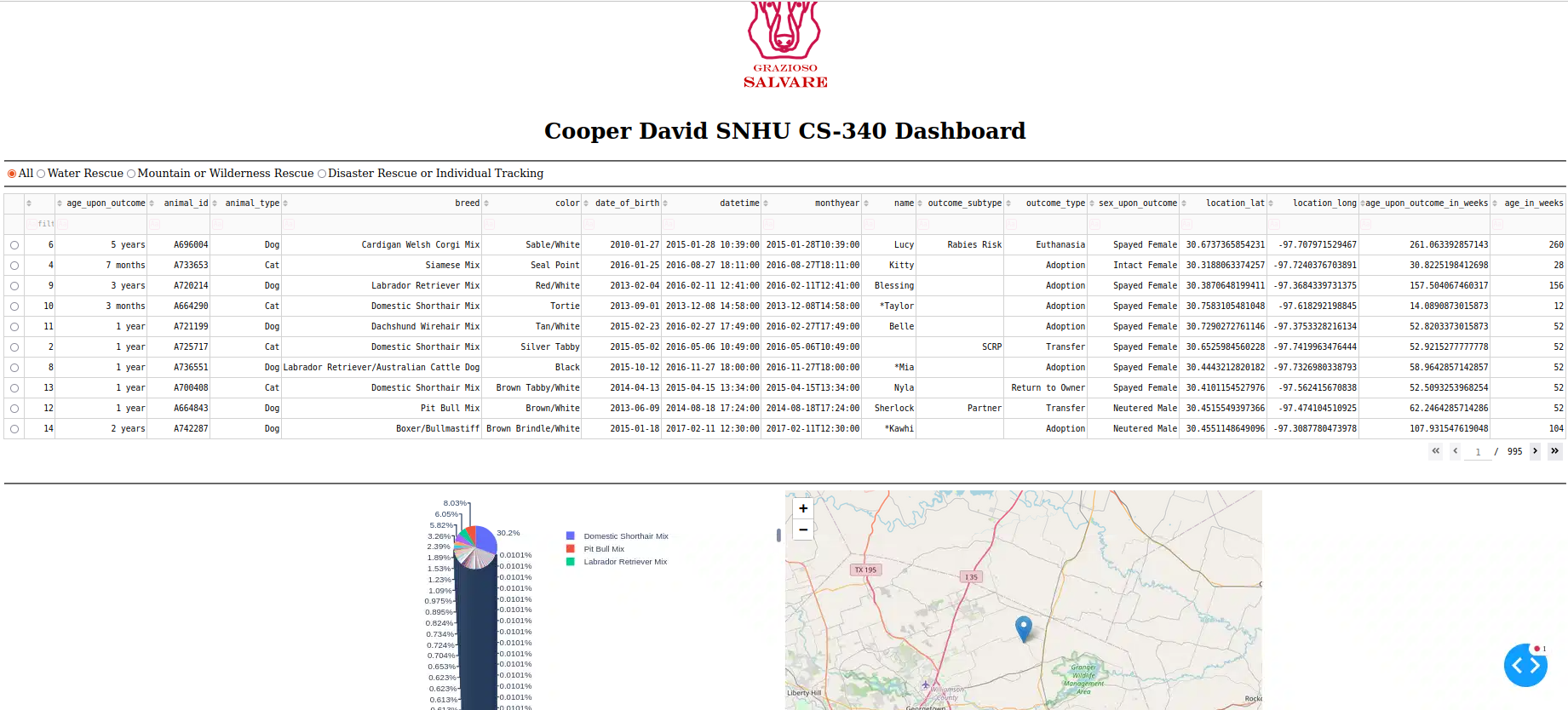
This project is a web dashboard developed using the Dash framework, which visualizes animal shelter data. The dashboard allows users to filter animal data based on various criteria, allows them to view detailed information in a table, and visualize data distributions through interactive pie charts and maps.

# Screenshots



# Tools Used

1. **Dash**: The main framework I used to create the web application. Dash allows for the development of interactive web applications using Python, making it easy to create data-driven visualizations.
2. **Plotly Express**: Used for generating visualizations (like pie charts). It's built on top of Plotly and provides a simplified interface for creating various types of plots.
3. **Pandas**: A data manipulation library that provides data structures for storing and manipulating large datasets.
4. **Dash Leaflet**: Used for displaying maps and markers. This library integrates well with Dash, allowing for more interactive geographical visualizations.
5. **MongoDB**: I used this as the database to store animal shelter records. It provides flexible data storage with a document-oriented structure that is well-suited for handling varying data fields.

# Reasoning for Tools Used

* **Dash** was chosen for its capability to create interactive web applications with a simple Python interface.
* **Pandas** is essential for data manipulation, allowing us to easily filter and transform our dataset.
* **Plotly Express** simplifies the process of creating visualizations, making it easier to implement complex charts without deep knowledge of JavaScript or D3.js.
* **MongoDB** provides scalability and flexibility in handling data, allowing for quick queries and easy updates.

# Why MongoDB?

I selected MongoDB for its document-oriented data model, which provides the following benefits:

* **Schema Flexibility**: It allows for different structures in the same collection, which is beneficial for storing varied animal records.
* **Scalability**: Easily accommodates large volumes of data, making it suitable for expanding datasets.
* **Rich Query Language**: It supports complex queries that can be easily executed from Python using libraries like PyMongo.

# The Dash Framework

Dash is a Python framework designed for building analytical web applications. It combines Flask (for backend capabilities) with React (for frontend interactivity). This framework allows developers to focus on building features without needing to manage the underlying web server or client-side code directly.

# Project Steps

1. **Project Initialization**: Set up a new Dash project and install required packages.
2. **Database Connection**: Implement a connection to MongoDB to retrieve animal shelter records.
3. **Data Manipulation**: Use Pandas to clean and prepare the data, including creating derived columns (e.g., converting age to weeks).
4. **Building the Dashboard Layout**: Design the layout using Dash components such as dcc.RadioItems, dash\_table.DataTable, and dcc.Graph.
5. **Implement Callbacks**: Create interactive components using Dash callbacks to update the data table, pie chart, and map based on user input.
6. **Testing and Refinement**: Test the dashboard functionality, making adjustments as necessary to ensure a smooth user experience.

# Challenges and Solutions

* **Handling Empty Data**: One significant challenge was ensuring the dashboard correctly handled cases where no data met the filter criteria. This was resolved by implementing checks in the callback functions to display appropriate messages when no records were available.
* **Handling and Filtering Different Types of Data:** Another challenge I had was making sure the data was filtered correctly when applying a filter to it. It was difficult to get the filters not only correct, but also functional as well when interacting with the dash framework. I converted the age strings into weeks, then put that into a new column and sorted from there. (Side note, I didn’t see the column where it said “age\_upon\_outcome\_in\_weeks” until AFTER I had completed everything, but I wanted to include that extra bit of code anyway in case age wasn’t provided in weeks)