Dante Lee

19 June 2023

CS-340-T5535

**Project Two README**

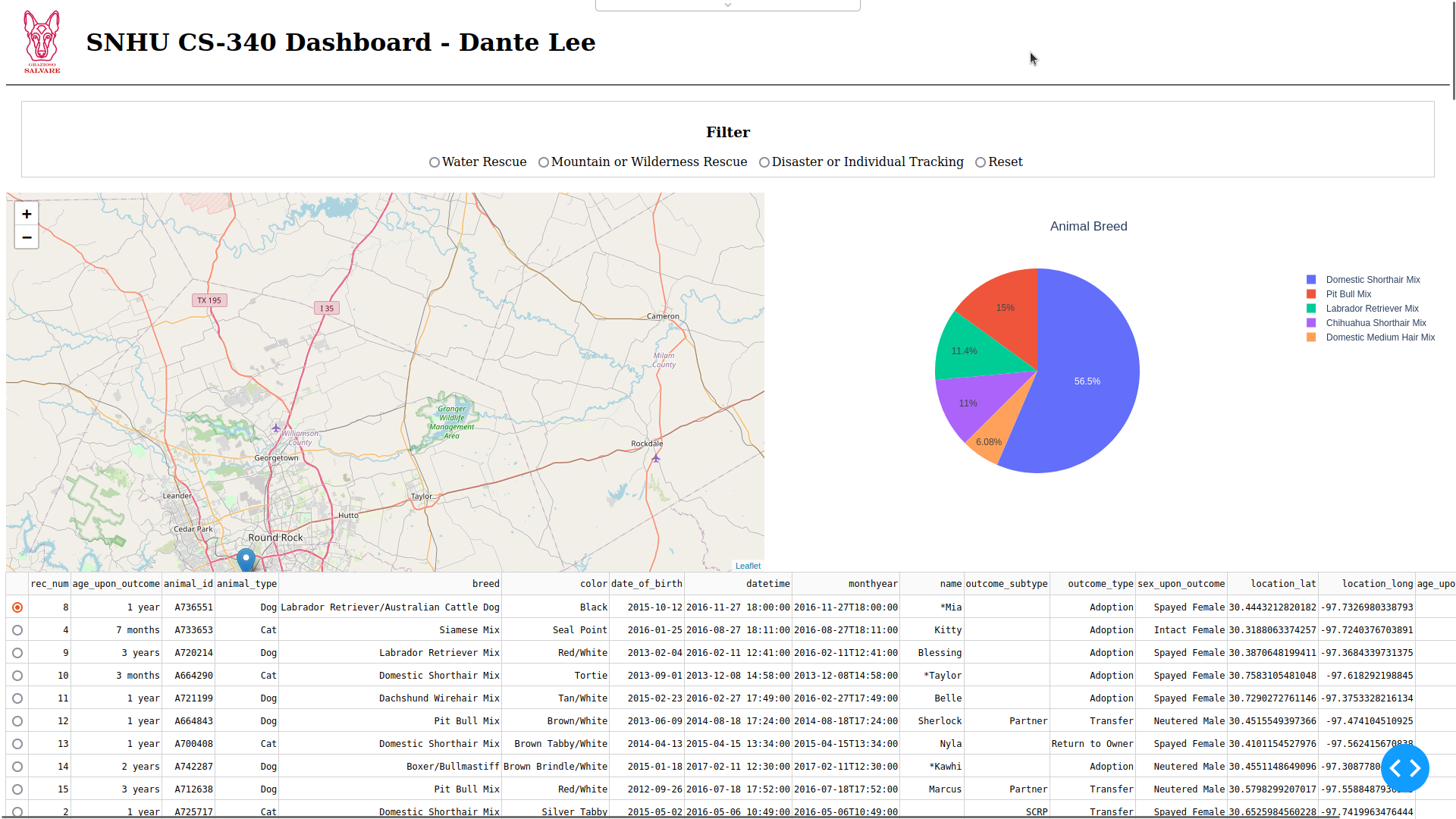
**Project Description and Functionality**

This project aims to develop a fully functional MongoDB dashboard for Grazioso Salvare, an international rescue-animal training company. The dashboard allows users to interact with and visualize data from the animal shelters in the Austin, Texas region. The key functionality of the project includes:

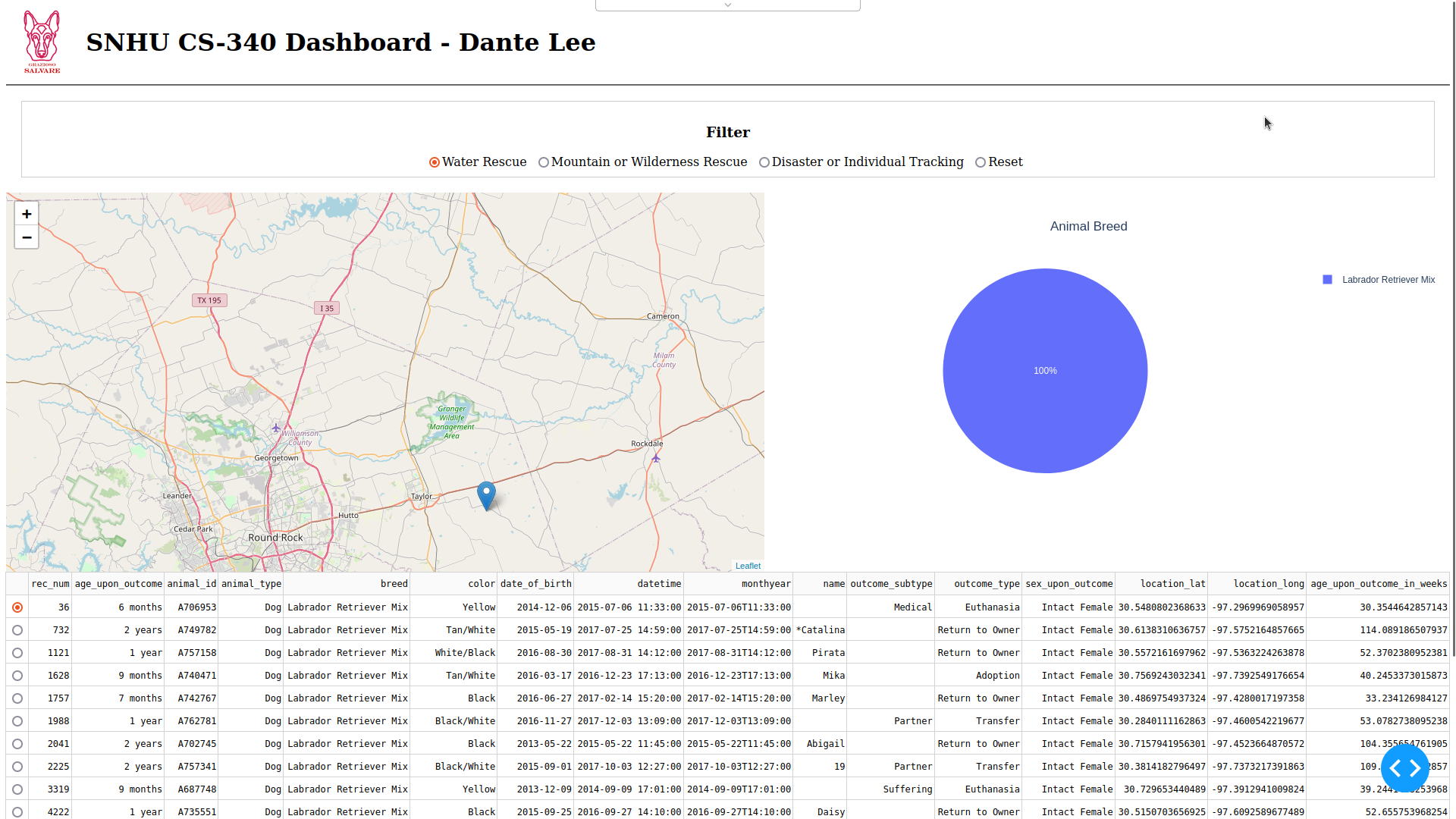
1. Interactive options to filter the Austin Animal Center Outcomes dataset.
2. An interactive data table that dynamically responds to the filtering options.
3. A geolocation chart and an additional chart that dynamically respond to the filtering options.

To demonstrate the achieved functionality, the project includes screenshots/screencasts of the dashboard at different stages of data filtering: Water Rescue, Mountain or Wilderness Rescue, Disaster or Individual Tracking, and Reset (unfiltered state).

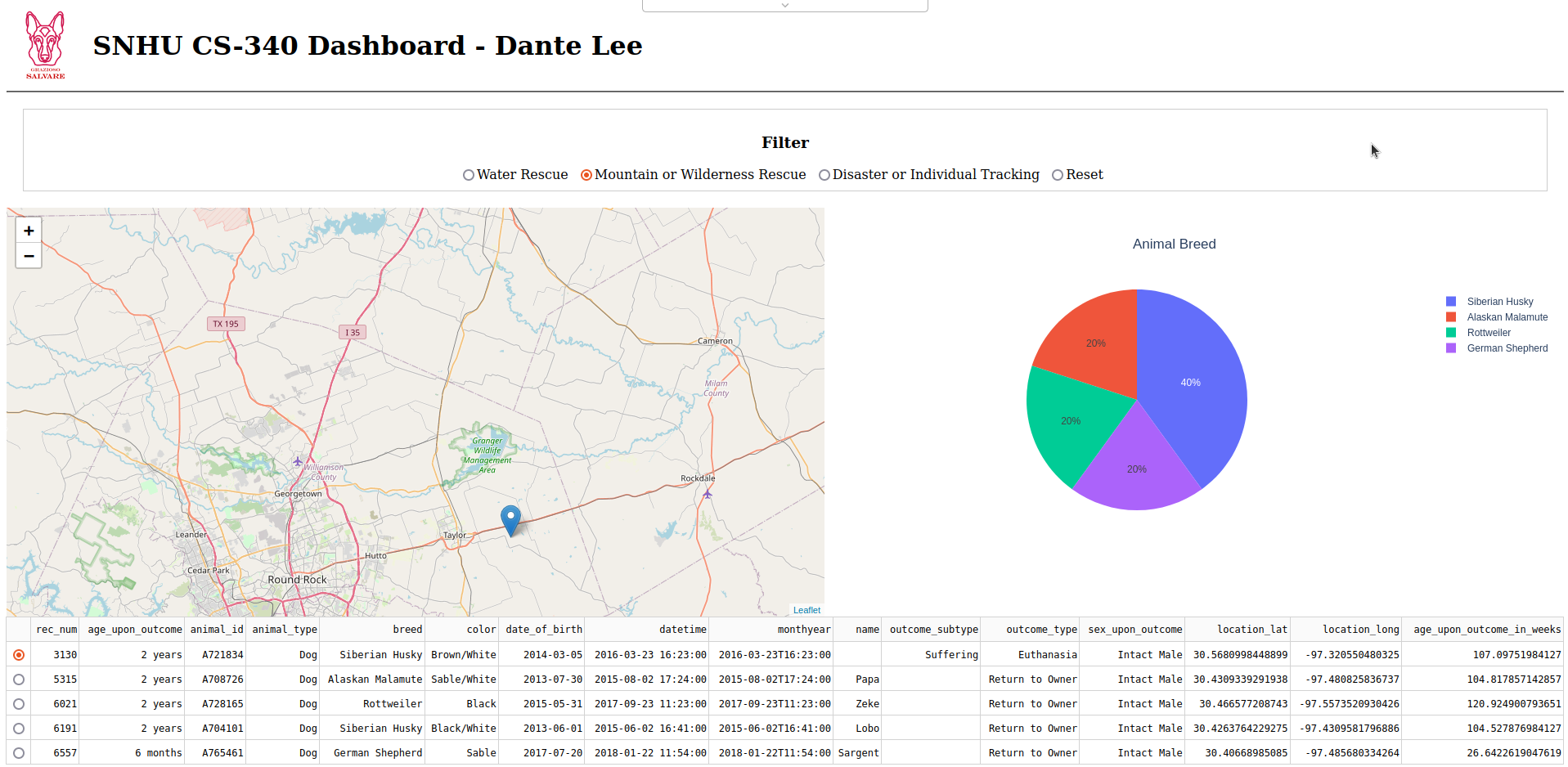
**Starting State of Dashboard**

****

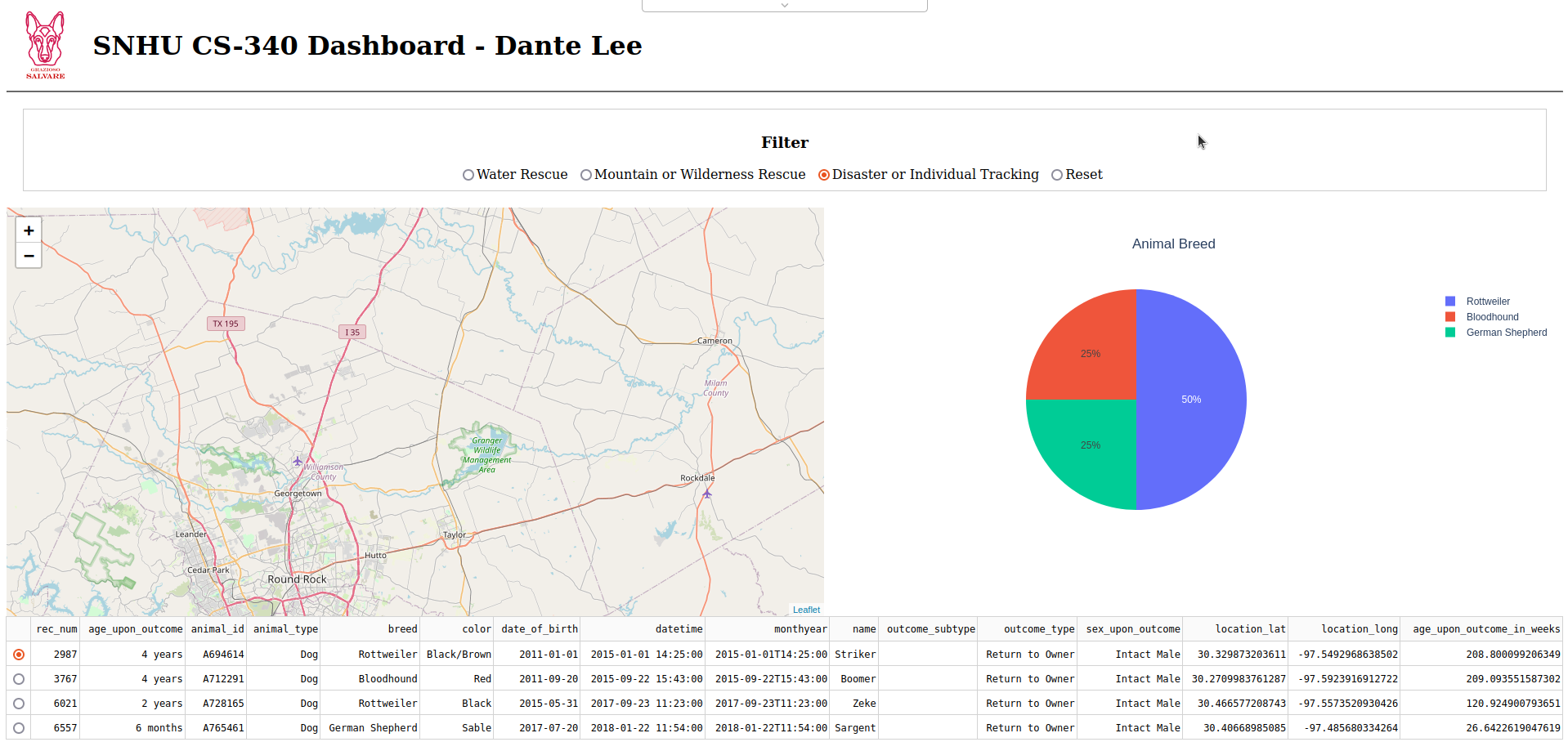
**Water Rescue Filter Applied**

****

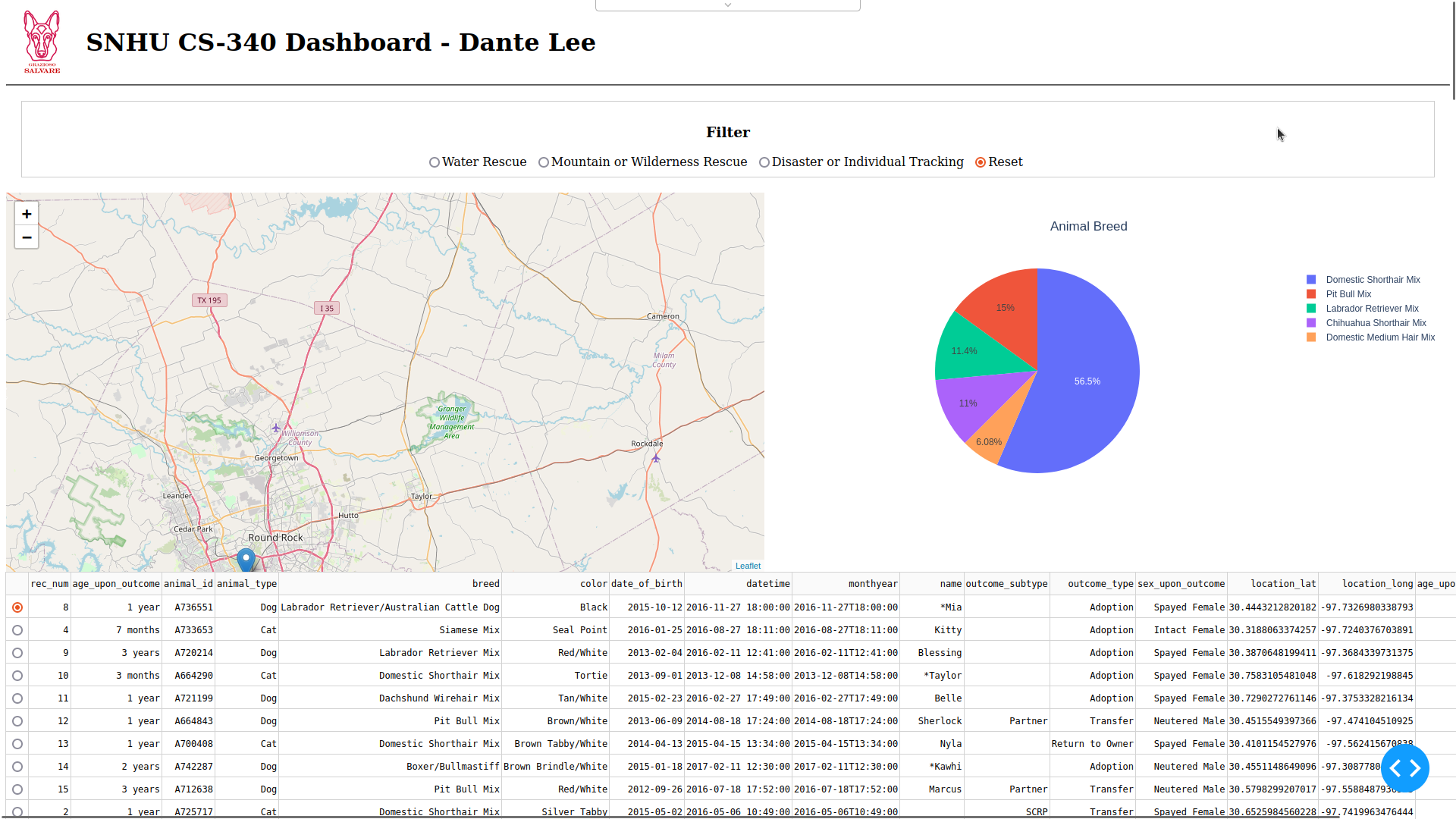
**Mountain or Wilderness Rescue Filter Applied**

****

**Disaster or Individual Tracking Filter Applied**

****

**Reset Filter Option Selected**

****

**Tools Used**

The following tools were used to achieve the desired functionality:

1. Python: The programming language used for developing the dashboard and database interface logic.
2. MongoDB: Used as the model component for the development due to its flexibility, scalability, and compatibility with Python. It allows for efficient storage and retrieval of structured data.
3. Dash Framework: Provides the view and controller structure for the web application dashboard. It enables the creation of interactive web applications with Python.

**MongoDB Usage and Benefits**

MongoDB was chosen as the model component for several reasons:

1. Flexibility: MongoDB's flexible document-based data model allows for easy representation of complex data structures, such as the animal shelter data. It accommodates changing requirements and allows for easy schema evolution.
2. Scalability: MongoDB is designed to scale horizontally, making it suitable for handling large datasets. It supports sharding and replication, ensuring high availability and performance.
3. Compatibility with Python: MongoDB has a mature and well-documented Python driver (PyMongo), which simplifies the integration with Python code.
4. Querying Capabilities: MongoDB's rich query language and indexing support enable efficient retrieval of data based on various criteria, which is crucial for the dashboard's filtering functionality.

**Dash Framework**

The Dash framework was utilized to provide the view and controller structure for the web application dashboard. It offers several benefits:

1. Interactive Web Applications: Dash enables the creation of interactive web applications using Python, allowing for rapid development and easy customization.
2. Data Visualization: Dash provides a wide range of interactive visualization components, making it well-suited for creating charts and graphs that respond to user inputs.
3. Integration with Plotly: Dash is built on top of Plotly, a popular open-source data visualization library. This integration enhances the dashboard's data visualization capabilities.
4. Easy Deployment: Dash applications can be easily deployed on various platforms, such as local servers or cloud services.

**Resources and Software Used**

During the development process, the following resources and software applications were accessed:

* MongoDB: A NoSQL document database used for storing and retrieving data.
* Dash and Plotly: Libraries used for building the web application dashboard.

**Project Completion Steps**

The project was completed in the following steps:

1. Database and CRUD Development: In the initial phases, a database was created, and a Python module enabling CRUD functionality for MongoDB was developed. This module allowed for the retrieval of data from MongoDB using the "retrieve all" query.
2. Data Table Creation: A data table was implemented on the dashboard to display an unfiltered view of the Austin Animal Center Outcomes dataset. The existing CRUD Python module was utilized to retrieve the data from MongoDB.
3. Database Query Development: Database queries were developed to match the required filter functionality. The Rescue Type and Preferred Dog Breeds Table provided in the Dashboard Specifications Document helped construct these queries. The CRUD Python module from Project One was used to develop these queries, with the "aacuser" account credentials hardcoded.
4. Interactive Options Development: Controller components were developed to create interactive options for data selection based on filtering functions. Different interactive options such as radio items or drop-down menus were implemented using Dash Core Components.
5. Widget Development: The data table created in Step 2 was modified to become an interactive data table that responded to input from the interactive options. Additionally, charts, including a geolocation chart and a second chart of choice, were created. These charts dynamically update based on the data table and the interactive options.
6. Testing, Deployment, and Documentation: The dashboard was thoroughly tested to ensure the functionality of all components. It was then deployed, and screenshots/screencasts were taken to capture the different stages of data filtering. A README file was created to document the project, including the achieved functionality, tools used, steps taken, and any challenges encountered.

**Challenges and Solutions**

Throughout the project, the following challenges were encountered:

* Complex Data Filtering: Implementing the filtering functionality based on different criteria, such as rescue type and preferred dog breeds, required careful query construction. However, by referring to the Rescue Type and Preferred Dog Breeds Table and leveraging the MongoDB querying capabilities, the challenges were overcome.
* Dashboard Widget Integration: Ensuring that the interactive options, data table, and charts interacted seamlessly required coordination between the controller and view components. By following the MVC pattern and utilizing the capabilities of the Dash framework, these integration challenges were successfully resolved.
* Deployment and Documentation: Deploying the dashboard and capturing screenshots/screencasts required careful consideration of the environment and proper documentation. Through the utilization of appropriate deployment methods and comprehensive documentation, these challenges were effectively addressed.

By applying problem-solving techniques, utilizing available resources, and leveraging the capabilities of the chosen tools, these challenges were overcome, resulting in a successful completion of the project.