CS 340 README Template

About the Project/Project Title

The Austin Animal Center project tracks records of animals at the Austin Animal Center.

Motivation

This project exists as a convenient way for users to keep track of animals that are at the Austin Animal Center and the actions that are taken regarding the animals (e.g. transfers from another location to the shelter, returning the animal to an owner, euthanization, etc.) by way of an intuitive dashboard.

Getting Started

To get a local copy up and running, follow these simple example steps:

1. Cone the repository: git clone https://github.com/your-username/austin-animal-center.git

2. Install dependencies (see next section of this README 'Installation')

3. Configure the MongoDB connection using the following connection variables:

USER = 'aacuser'

PASS = 'yourpassword'

HOST = 'your-mongodb-host'

PORT = 31860

DB = 'AAC'

COL = 'animals'

4. Test that the Python CRUD Module (project\_1.py) functions properly by running the associated test file: aac\_crud\_tests.ipynb

5. Start the project by running the first cell in the file: ProjectTwoDashboard.ipynb

6. Navigate to the application in the browser indicated by the url noted below the cell (i.e. Dash app running on <url>)

Installation

Install the following tools:

1. Python: download and install the latest version from the official website https://www.python.org/downloads/ and run the command 'python --version' to verify that the installation was successful.

2. MongoDB: download and install the latest version of MongoDB Community Edition for your operating system from the official website https://www.mongodb.com/try/download/community. Start the MongoDB server to verify that the installation was successful by running the command 'mongod'.

3. Git: download and install the latest version from the official website: https://git-scm.com/downloads and run the command 'git --version' to verify that the installation was successful.

4. Jupyter Notebook: Install Jupyter Notebook with pip by running the command 'pip install notebook' and verify that the installation was successful by starting the notebook using the command 'jupyter notebook'.

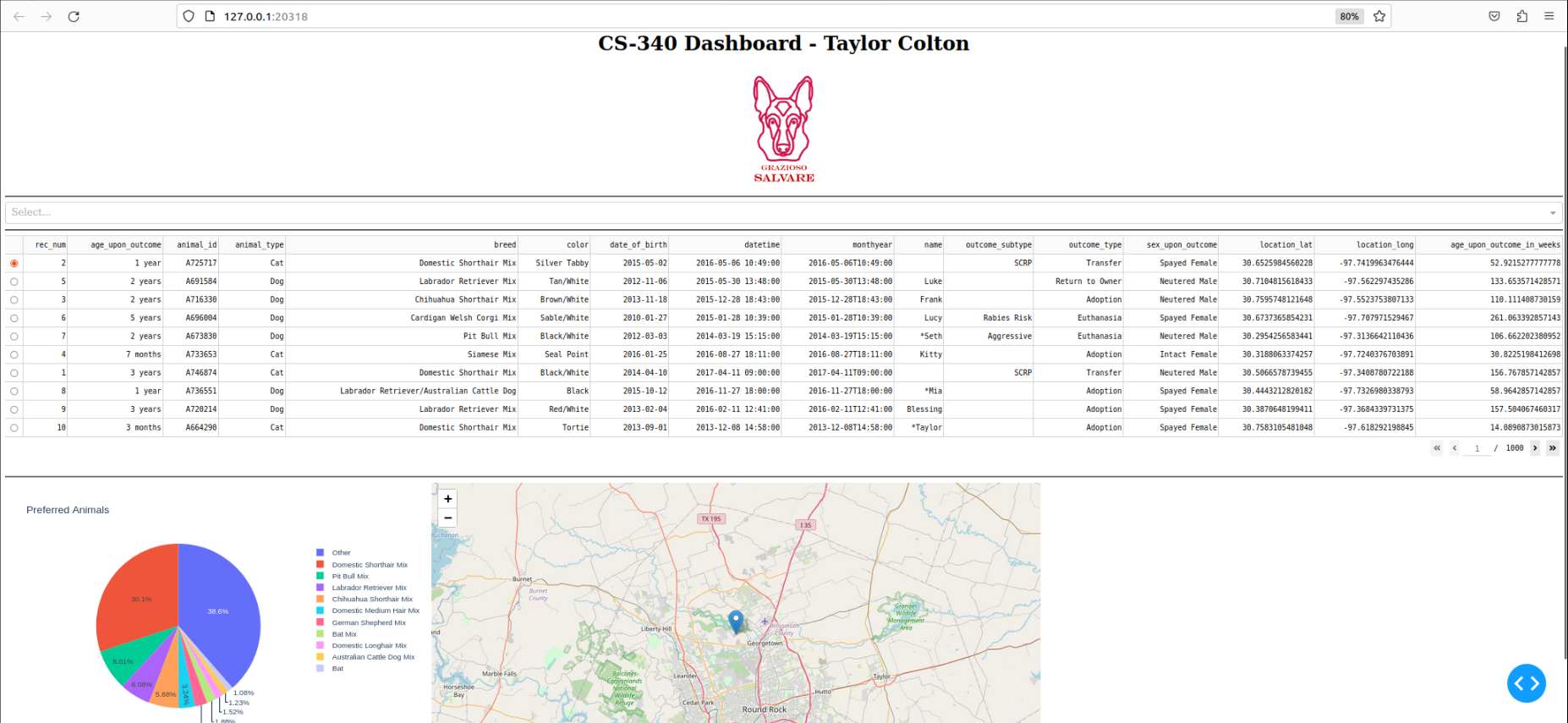
Steps Taken to Implement Required Functionality

This application needed to perform several actions in order to be as useful and easy to understand as is necessary for its users, including:

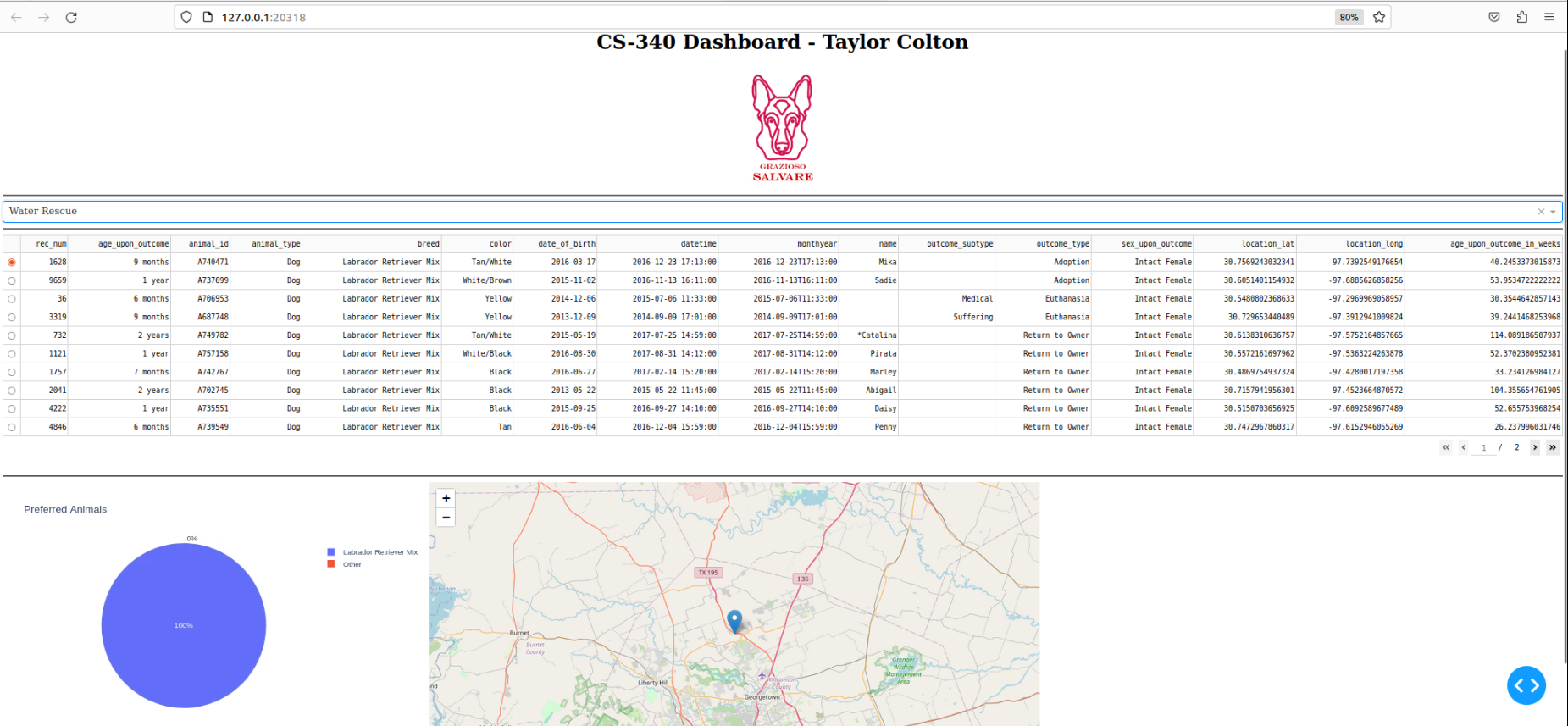
1. The dashboard filter options needed to properly retrieve data from the database (i.e. database queries had to be created that matched the required filter functionality). To do this I had to create a drop down and connect it to the data table to allow for accurate filtering based on types of rescues performed.
2. The controller pieces had to be developed to create interactive options that allowed for the selection of data based on the filtering functions. These interactive options enabled the control of other dashboard widgets, such as the data table and charts.
3. The dashboard widgets needed to be created/modified to receive input from the interactive options and subsequently present those dynamic updates to the client. Some of which included:
   1. The interactive data table that responds to input based on row selection and filtering from the drop down.
   2. The pie chart and map that displays data in response to updates from the data table.
   3. The pie chart that displays data in response to user selection of which breeds they want to be displayed in the chart.

Screenshots of Required Functionality

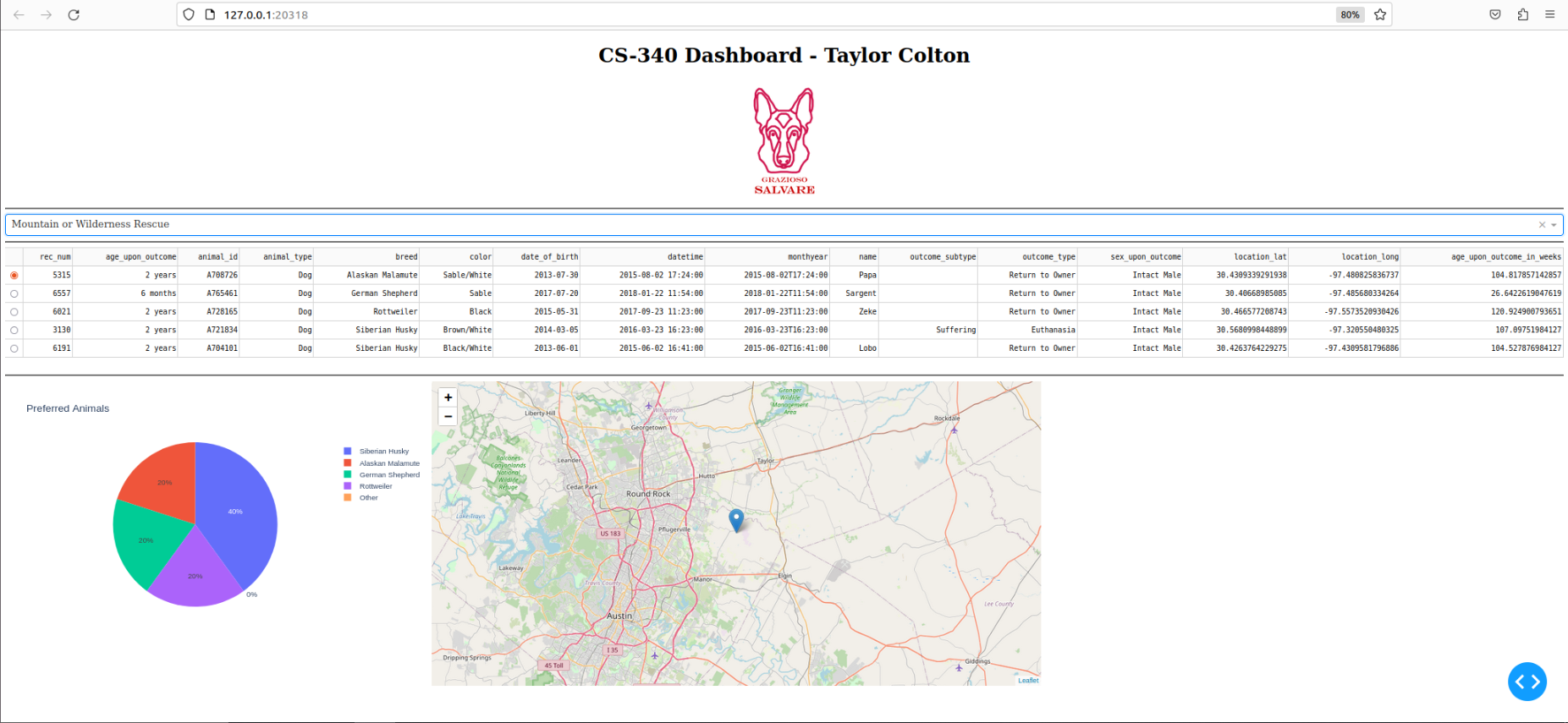
ProjectTwoDashboard in initial state:



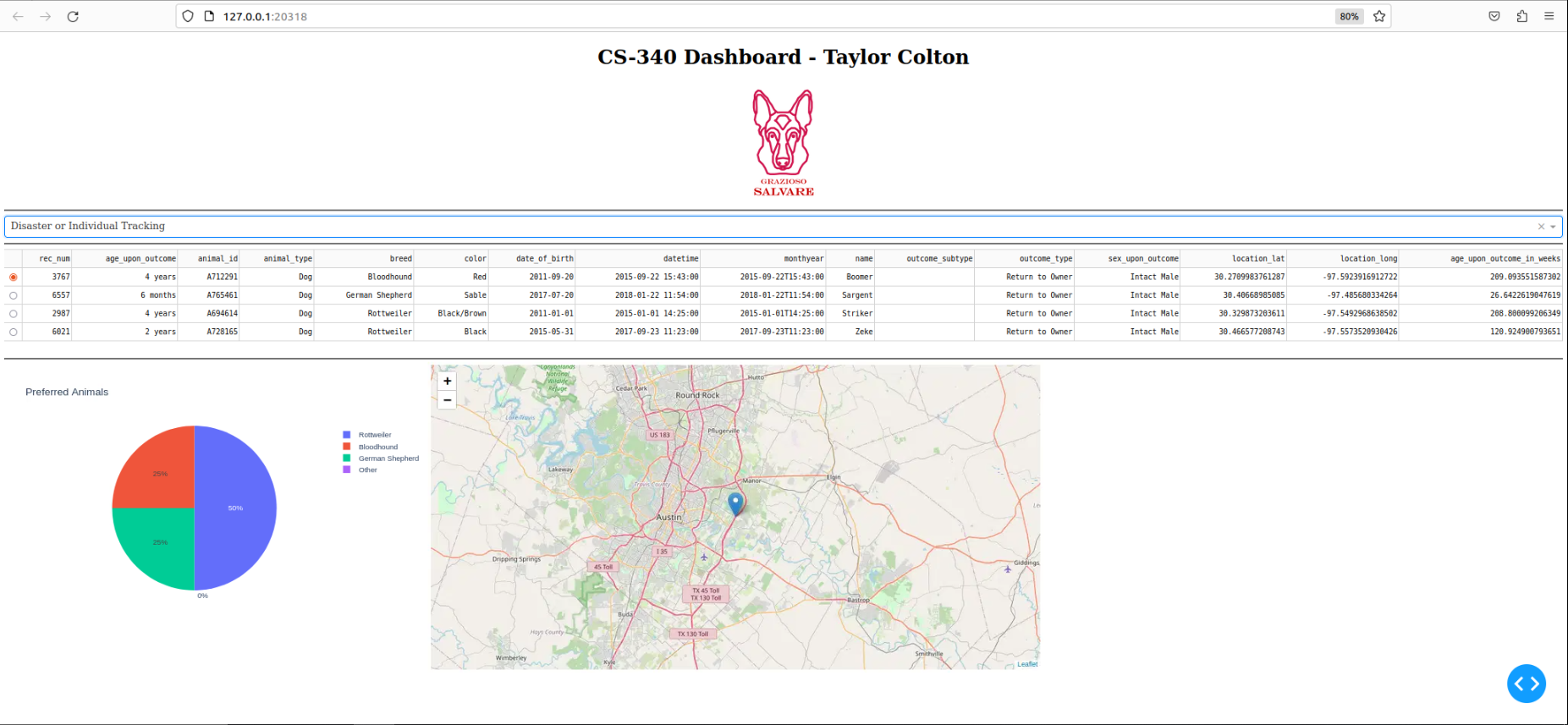
ProjectTwoDashboard after filtering by Water Rescue via drop down:



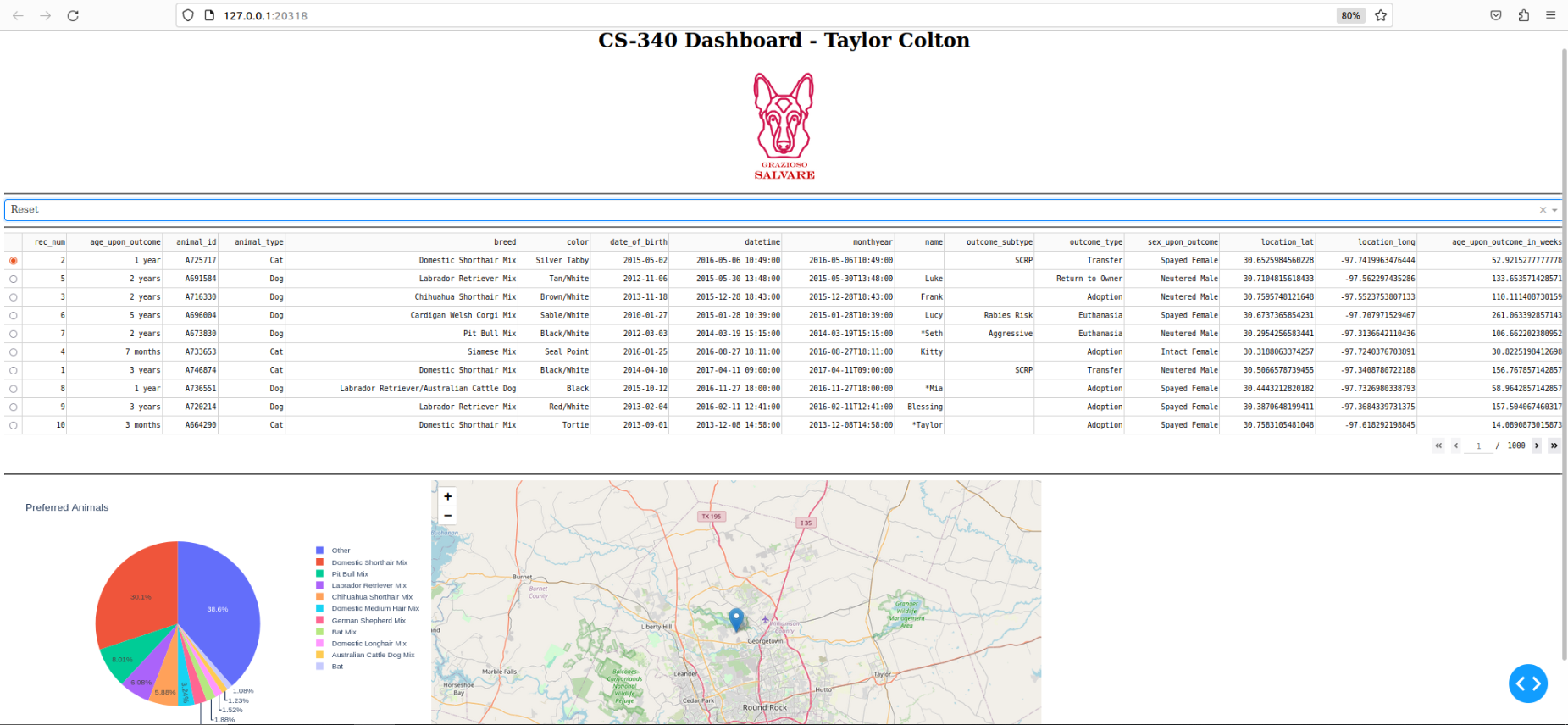
ProjectTwoDashboard after filtering by Mountain or Wilderness Rescue via drop down:



ProjectTwoDashboard after filtering by Disaster or Individual Tracking via drop down:



ProjectTwoDashboard after filtering by Reset via drop down:



Tools & Resources

MongoDB:

1. MongoDB enables flexible, schema-less data storage by storing data in the binary JSON (BSON) format. When storing different animal attributes without strict schema requirements, this is particularly helpful.
2. Through the PyMongo package, MongoDB and Python interact seamlessly, making it simple to use Python code to perform CRUD (Create, Read, Update, Delete) operations.
3. MongoDB makes it simple to effectively obtain particular data by supporting sophisticated queries that include filtering, aggregation, and sorting. Sorting animals by breed, age, or outcome is an example of this.

Dash:

1. To create user interfaces declaratively in Python, Dash offers components such as dash\_table, html, and dcc (Dash Core Components). One of the resources I used several times throughout this project is the Dash Components documentation.
2. The has the ability to react to user inputs dynamically without reloading the page thanks to Dash's callback functions. This offers a seamless, dynamic experience, including real-time updates to data tables, graphs, and maps.

Challenges Encountered

Most of the challenges I encountered were related to proper use of callbacks. Two of the errors that I received when debugging were “TypeError: 'NoneType' object is not subscriptable” and “IndexError: list index out of range”. These errors typically occurred due to missing data and/or improper handling of inputs from Dash components like derived\_virtual\_data and selected\_rows. I was able to overcome these challenges by checking the outputs I was receiving via print statements and updated input data using custom queries.

Contact

Your name: Taylor Colton