**Grazioso Salvare Dashboard Project README**

**Project Overview**

This project is a dashboard web application developed for Grazioso Salvare, an international rescue-animal training company. The application interfaces with a MongoDB database to identify and categorize dogs suitable for search-and-rescue training. The dashboard allows users to filter, visualize, and interact with data from multiple animal shelters in the Austin, Texas, region.

The application was developed using Python, with the Dash framework used for the front-end interface and MongoDB as the back-end database. The code for this project is open source and available on GitHub [here](https://github.com/jjs003/CS-340).

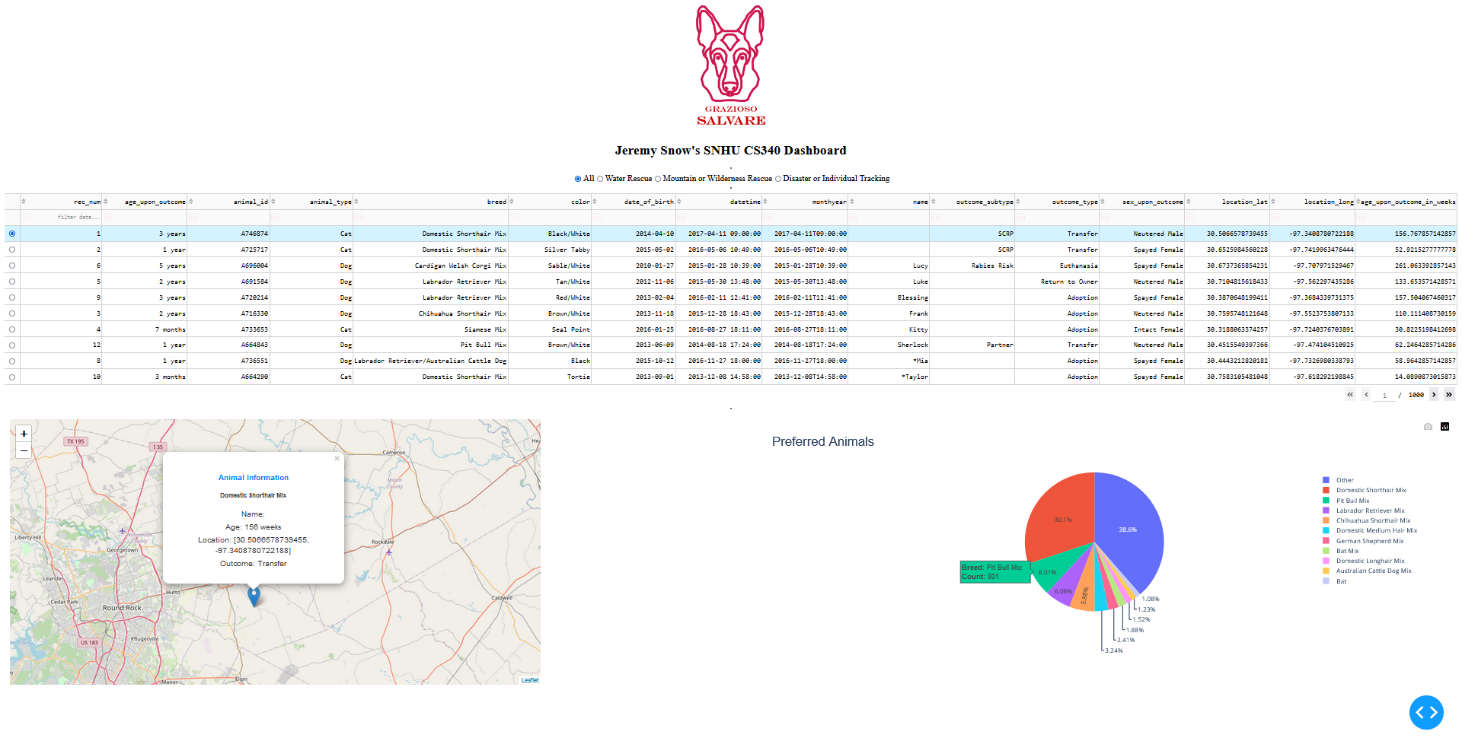
**Functionality**

The dashboard includes the following key features:

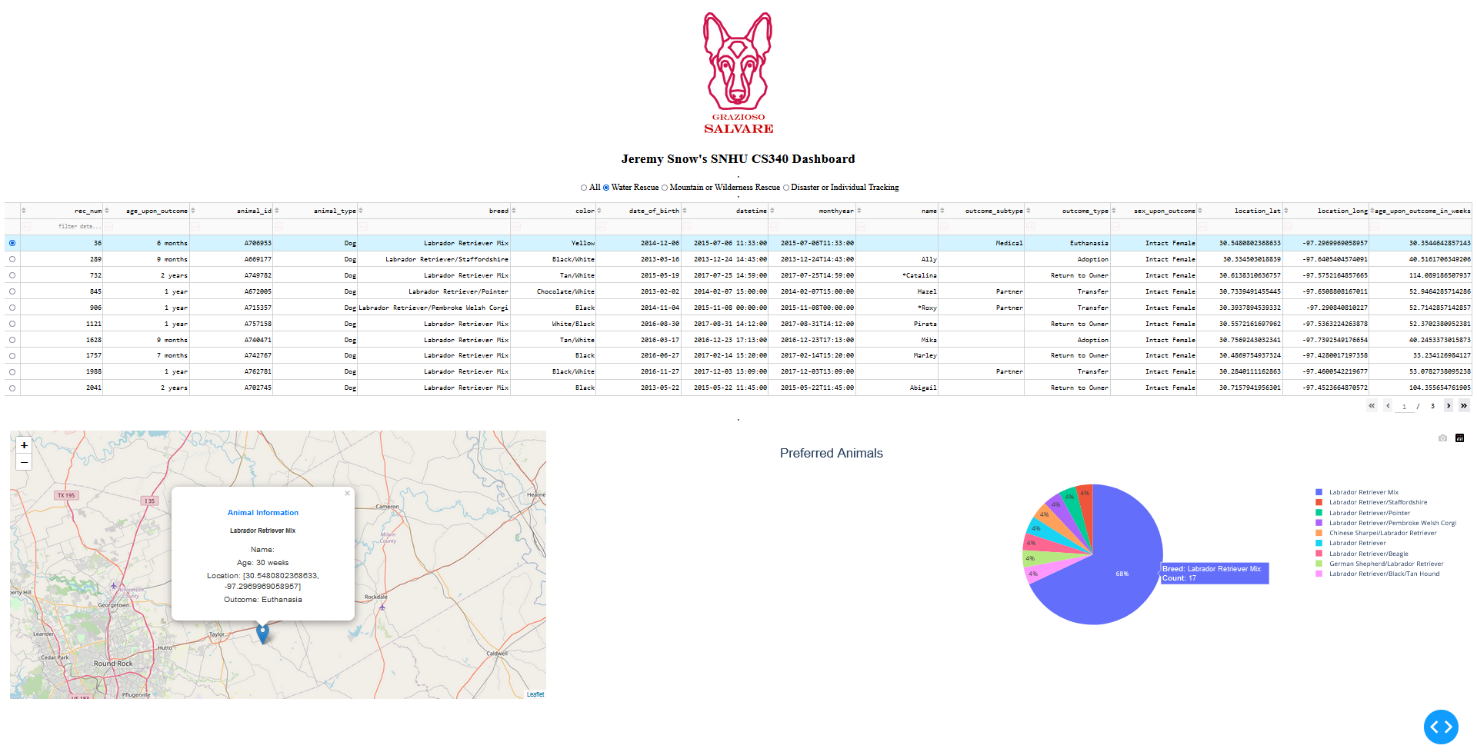
* **Unfiltered Data Table (Initial State)**: Displays all records from the Austin Animal Center Outcomes dataset, retrieved using MongoDB queries. Radio button selection is ‘All.’
* **Filter Options**: Users can filter the dataset by rescue type (Water Rescue, Mountain or Wilderness Rescue, Disaster or Individual Tracking) using radio buttons for each type. Clicking all resets the data table to its initial state.
* **Interactive Data Table**: The data table dynamically updates based on the selected radio button filters and also allows users to use native sorting and filtering built into the table by entering search parameters.
* **Geolocation Chart**: Visualizes the geographical location of the animal selected from the data table.
* **Pie Chart**: Shows the distribution of animal breed types for the selected filter.

**Screenshots of Functionality**

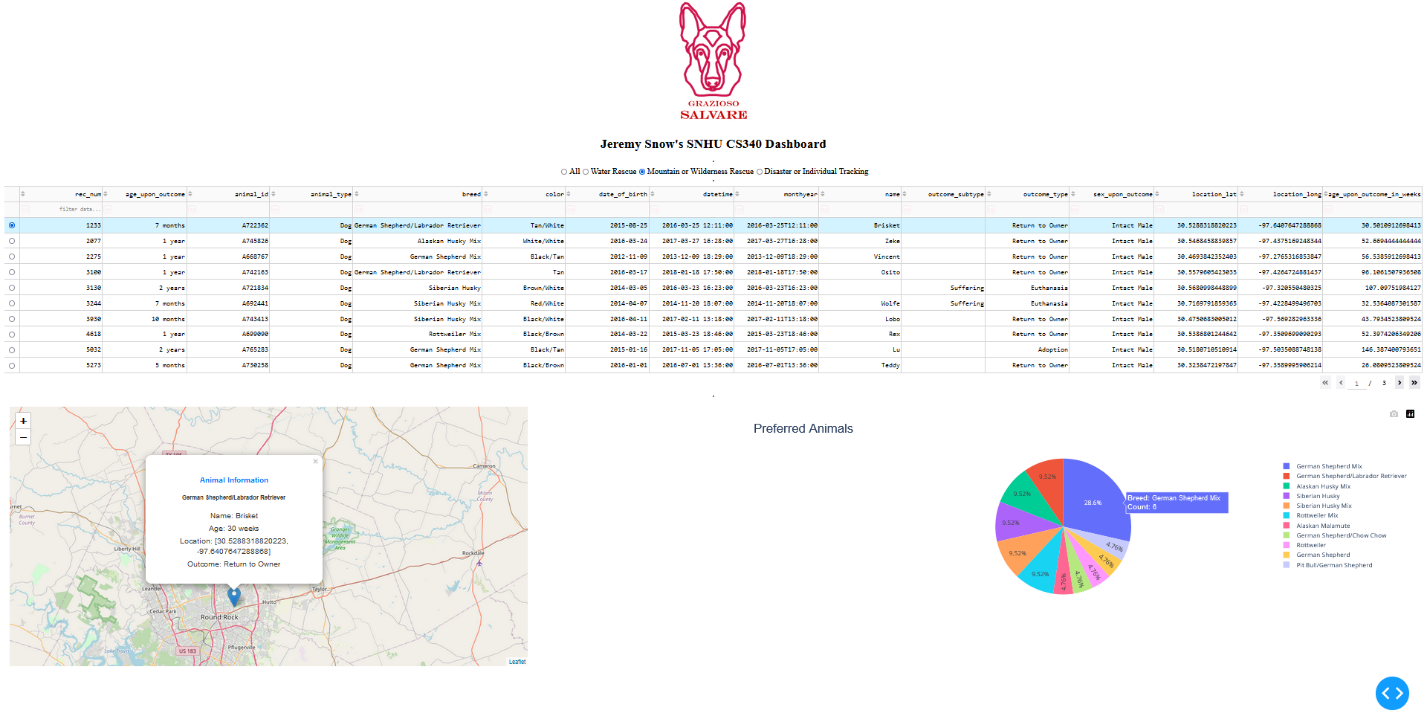
**‘All’ filter selected (unfiltered/reset)**

****

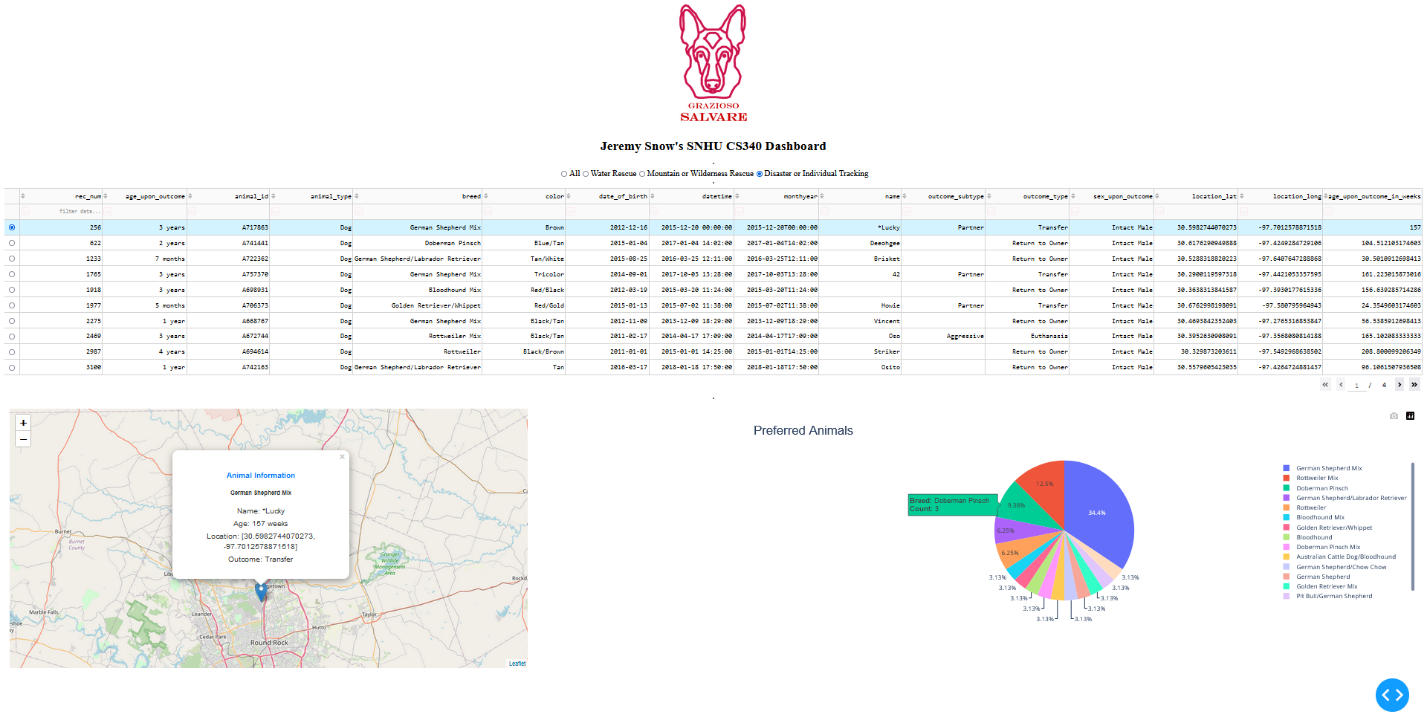
**‘Water Rescue’ filter selected**

****

**‘Mountain or Wilderness Rescue' filter selected**



**‘Disaster or Individual Tracking’ filter selected**

**Screen Capture of Functionality**



**Tools and Technologies**

**MongoDB v7.0.2 / PyMongo v4.8.0**

MongoDB was chosen as the database technology for this project due to its flexibility and scalability. Its document-oriented data model aligns well with the dynamic and hierarchical nature of the dataset. Additionally, MongoDB is easily integrated with Python (the programming language for this project) through the PyMongo library which enables efficient CRUD operations, making it an ideal choice for this project. ([mongodb.com](https://www.mongodb.com/)) ([pymongo](https://pymongo.readthedocs.io/en/stable/))

**Dash Framework**

The Dash framework was used to build the dashboard because of its simplicity and powerful features for creating interactive web applications with Python. Dash abstracts much of the complex web development process, allowing for the rapid development of a user-friendly interface. It also provides a seamless way to integrate data visualization and interaction, essential for this project. ([dash.plotly.com](https://dash.plotly.com/))

**Additional Tools**

* **Python v3.8.2**: The programming language used for developing the application. ([python.org](https://www.python.org/))
* **Jupyter Notebook**: Used for iterative development and testing of the dashboard. ([jupyter.org](https://jupyter.org/))
* **GitHub**: Used for version control and hosting the open-source project. ([github.com](https://github.com/))

**Steps Taken to Complete the Project**

1. Install the tools and technologies listed in the previous section.
2. Place aac\_shelter\_outcomes.csv file and AnimalShelter.py in a common folder to launch Jupyter Notebook from.
3. Set up a user account in MongoDB to load data into a database.

A computer screen shot of a blue screen

Description automatically generated

1. Import csv file into MongoDB database.

A screenshot of a computer program

Description automatically generated

1. **Install PyMongo:** Run the following command to install PyMongo from the shell:

pip install pymongo

1. **Install Jupyter Notebook:** Run the following command to install Jupyter Notebook from the shell:

pip install jupyter

1. Launch Windows PowerShell and navigate to the directory where aac\_shelter\_outcomes.csv file and AnimalShelter.py are located. Run the following command to start Jupyter Notebook:

jupyter notebook

1. Load existing Jupyter Notebook file from [GitHub](https://github.com/jjs003/CS-340) or create a similar version to build the dashboard.
2. Run Jupyter notebook and test dashboard functionality, documenting proper functioning with screenshots.

**Challenges and Solutions**

The main challenge that I ran into while finishing this project was getting the dashboard to run without errors. Each time I would figure out how to fix one error, another would pop up. One such issue was how to deal with the update\_map callback when data isn’t loaded quickly enough. This involved creating an if statement to catch instances when data isn’t loaded. I highly recommend adding functionality as incrementally as possible and testing very often to catch bugs before they stack up. Another issue that I came across was dealing with the data in the csv file that was not formatted very well. This was dealt with through utilizing regular expressions to capture breeds that matched the search constraints.

**Contact**

Jeremy Snow – Jeremy.Snow@SNHU.edu