# Rescue Dog Identification Application README

## About the Project/Project Title

The Rescue Dog Identification Application is used to access an existing database of shelter dogs to select the best candidates for rescue dog training for various types of rescues. These include specific dogs for water rescue, mountain and wilderness rescue, and disaster and individual tracking. The application utilizes a python module to connect the dashboard to the animal shelter’s database to perform filtered queries. All a user is expected to do is to select what type of rescue they want to find a dog to train for. It will filter using breed, age, and sex to select the very best candidates and map them on an interactive geolocation map and utilize a pie chart to further understand the breed distribution of those type of rescue dogs.

## Tools and Technologies Utilized

The tools required to use this project are:

* **Current Python programming language** including the following libraries:
  + **Plotly** for creating charts and graphs

<https://plotly.com/python/>

* + **Pandas** to converting data from MongoDB into readable data tables

<https://pandas.pydata.org/docs/>

* + **Dash** Leaflet to integrate geolocation map

<https://dash-leaflet-docs.onrender.com/>

* **MongoDB** for the database as it allows for more flexibility and does not require specific table structures because it uses NoSQL so data does not have to fit a perfectly rigid format. PyMongo makes MongoDB highly compatible with Python by allowing a large variety of querying options and filtering commands.

<https://www.mongodb.com/docs/>

* **Jupyter** **Notebook** as an IDE and web application development

<https://pymongo.readthedocs.io/en/stable/>

* **PyMongo** which the driver needed to connect Python to MongoDB databases

<https://pymongo.readthedocs.io/en/stable/>

* **Mongosh** as the Mongo shell

<https://www.mongodb.com/docs/mongodb-shell/install/>

* **Dash Framework** to build a web-based application that only requires python code. This makes it simpler to create an interactive dashboard without having to use HTML, CSS, and JavaScript code. Also, it allows for its callback functionality which immediately updates with user input. The buttons in the dashboard are a great example of this.

<https://dash.plotly.com/>

**Setup**

* Ensure you have a working CRUD python module to interact with the database and your dashboard. An example of the database connection is below.

A screenshot of a computer program

AI-generated content may be incorrect.

This instance connects to a collection named “animals” within the “AAC” database. It also utilizes security-minded authentication in the form of username and password arguments while restricting access to only the “animals” collection. Next the CRUD methods should resemble the screenshot below:

A screenshot of a computer

AI-generated content may be incorrect.

These methods allow this module to be used to create, read, update, and delete operations without direct access to the MongoDB database.

To begin setting up the dashboard within Jupyter Notebook, you need to import the following libraries and the CRUD Module.

A screenshot of a computer

AI-generated content may be incorrect.

Once imported, python language can be used to implement html headers and layouts, and Dash core components can be used through plotly to create interactive widgets such as the buttons used in this project or other functional items such as dropdown menus, sliders, and checkboxes. Within the dash framework, these widgets update in real time ensure interactivity.

[*https://dash.plotly.com/dash-core-components*](https://dash.plotly.com/dash-core-components)

Dash DataTable through plotly creates interactive tables that create the foundation of sorting, querying, and viewing data in easy-to-read formatted tables. It also allow the ability to select rows and update other widgets based on selection.

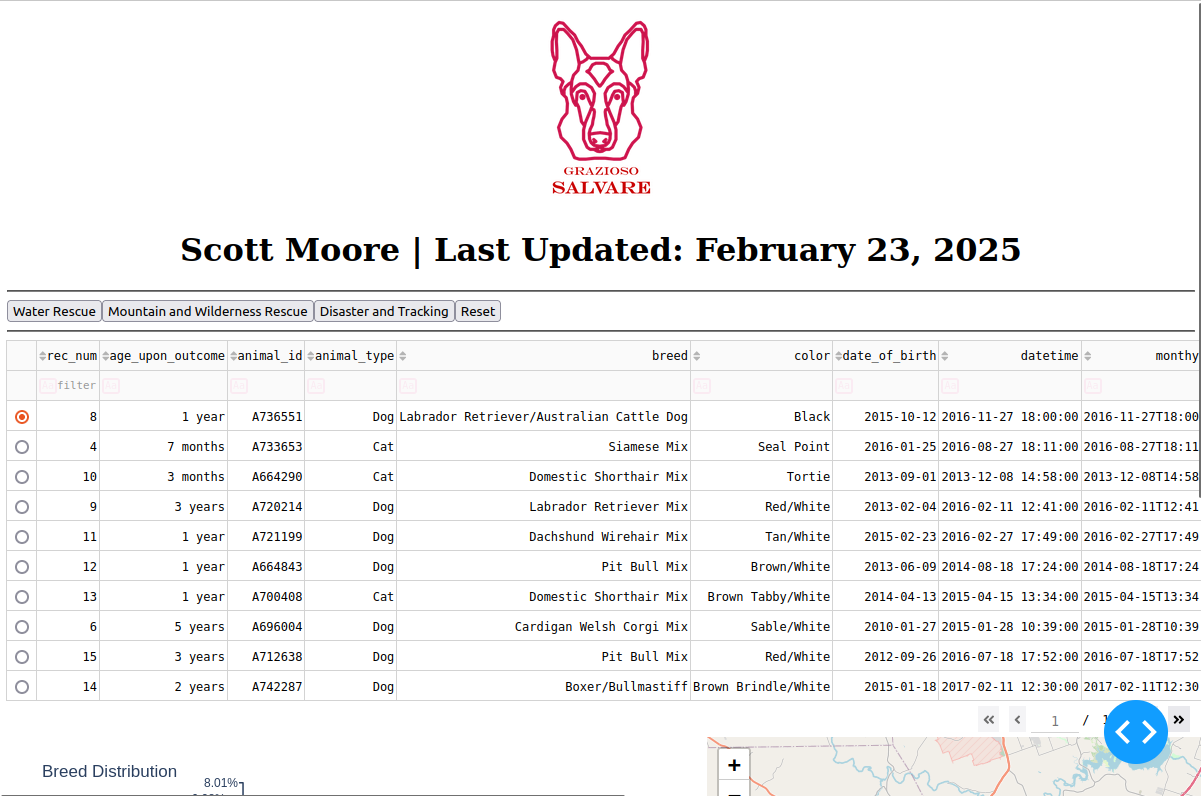
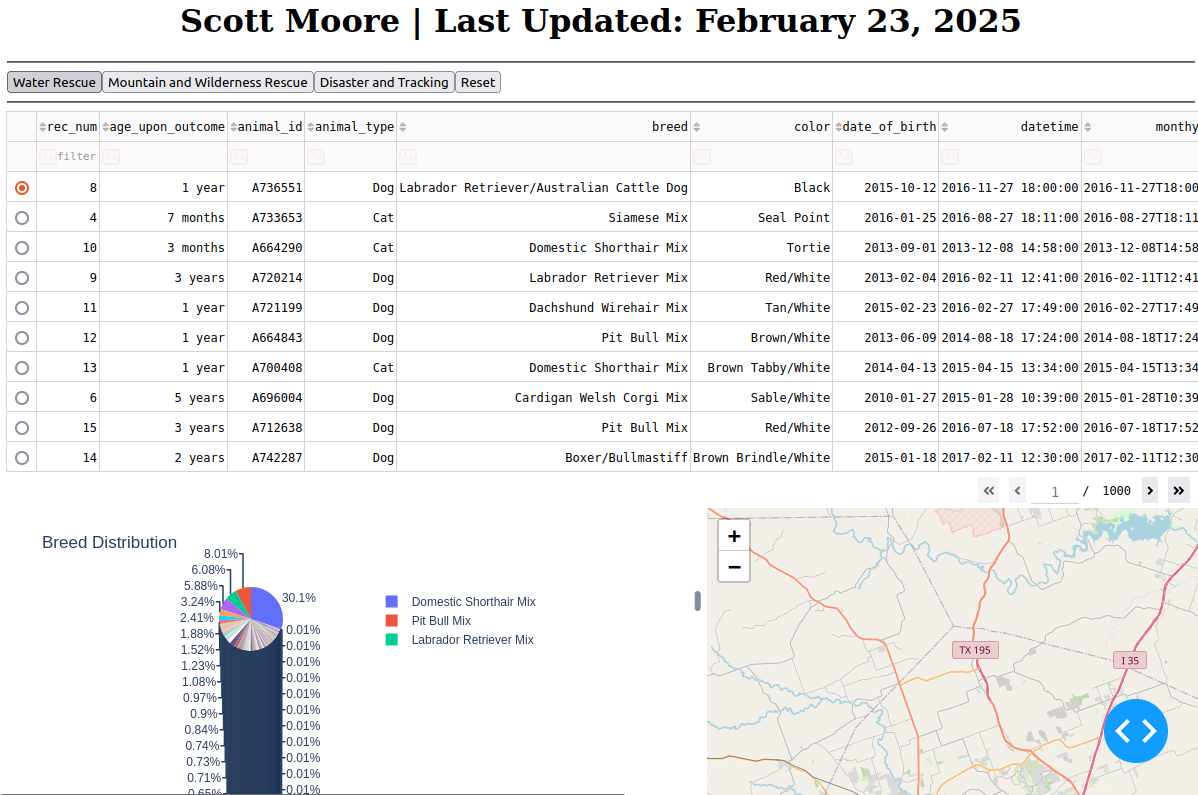
<https://dash.plotly.com/datatable>

Lastly this dashboard utilized the Dash Leaflet plugin to use a geolocational map that used longitudinal and latitudinal coordinates to mark up an interactive map. This project’s database did contain geographical coordinates for us to map for each animal.

<https://dash-leaflet-docs.onrender.com/>

## Usage

Once connected to the correct python module, and MongoDB database, the dashboard allows for very easy filtering queries for rescue dogs. A pie chart displaying breeds for the selected filters and an interactive geolocational map also make for easy analysis of what dogs are available and where. Below are some examples:



Figure

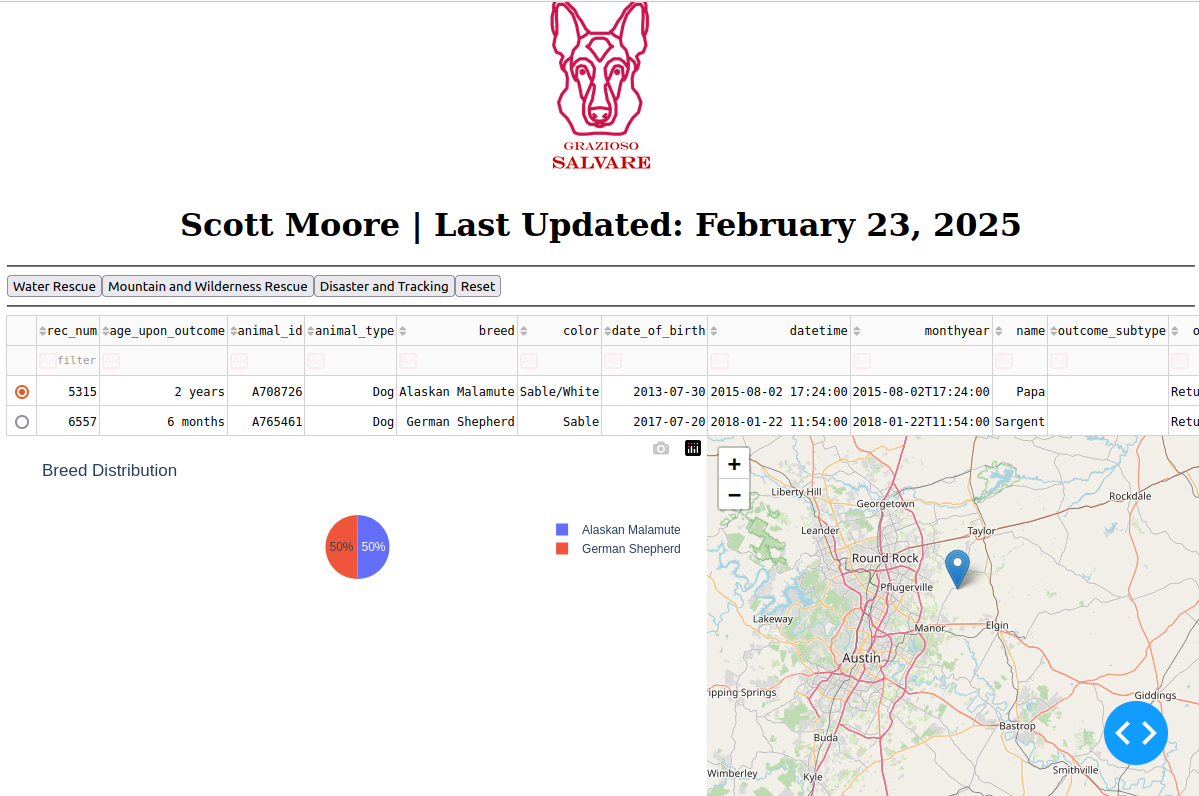
### Figure 1 shows two screenshots with the initial page when no filters are selected. The output shows 10 animals from the animal shelter, the distribution by breed in the pie chart for all dogs, and a geolocational map that defaults to the first animal selected.

A screenshot of a computer

AI-generated content may be incorrect.

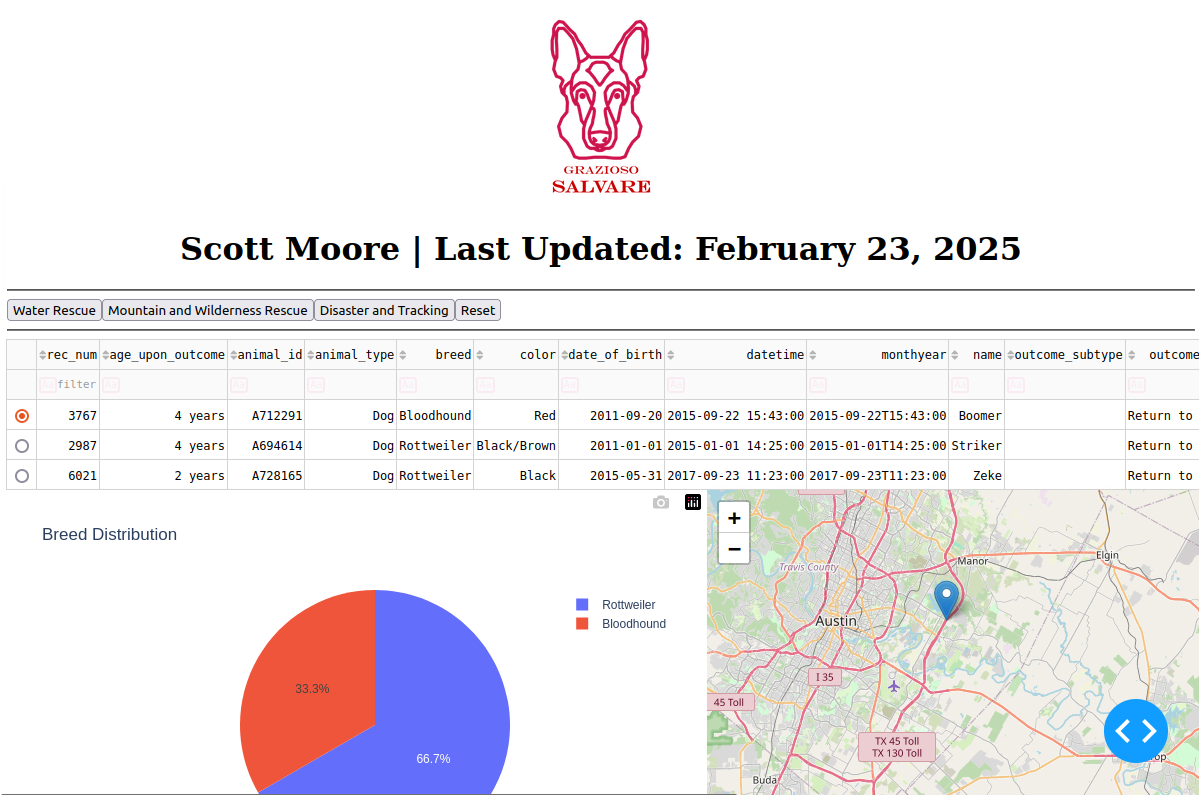
Figure

**Figure 2 shows a screenshot of when the “Water Rescue” button is pressed. The only result matched is a labrador retriever so the breed graph shows 100% Labrador Retriever and it is plotted on the geolocational map.**

****

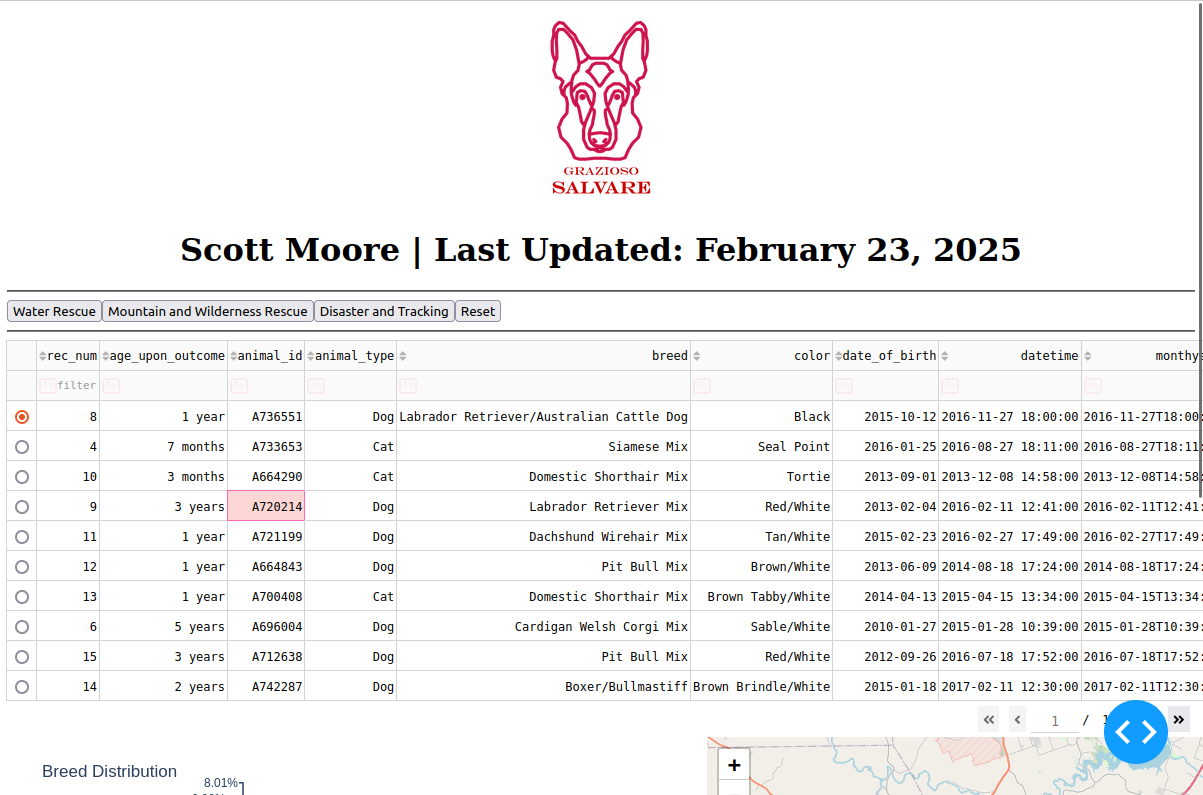
Figure

**Figure 3 is a screenshot from when the “Mountain and Wilderness Rescue” button is clicked. The results are updated to show a new breed distribution, and it maps the selected dog.**

****

Figure

### Figure 4 is a screenshot illustrating the results when the “Disaster and Tracking” button is clicked by the user. This updates the view to list the acceptable dogs for this category and maps the dog selected.



Figure

### Figure 5 shows the results when the reset button is clicked while viewing another category. It goes back to the default selection with the same pie chart and map from figure 1.

**Challenges Encountered During Implementation**

* Initial attempts to execute the program did not account for empty results.
  + Solution: Ensure the dash code has built-in responses to empty data.

**A screenshot of a computer code

AI-generated content may be incorrect.**

* Python requires strict adherence to spacing and indents.
  + Solution: If receiving indentation error, scan code especially with embedded lines for indentation errors.
* MongoDB’s \_id column can cause tables to crash
  + Implemented an “If” statement for instances when \_id column was trying to be placed in the table that would drop that column.
* Image not loading into webview with a “FileNotFoundError” message
  + Created /asset folder with the project to ensure the image was available

**Roadmap**

* Add more charting to allow more visualization such as bar and/or line graphs
* Account for dogs that are enrolled or have completed their rescue training

**Author**

Scott Moore