# CS 340 README

## About the Project/Project Title

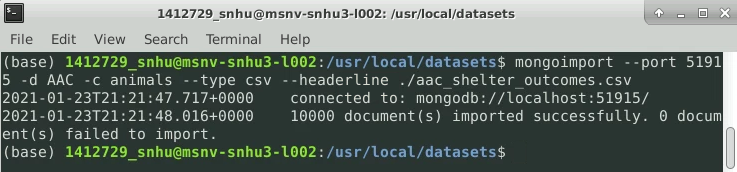
The *Grazioso Salvare Shelter Dog Finder* was created so that Grazioso Salvare can find candidates for search and rescue training. Through this interface with a previous MongoDB database, the filtering of dogs based of certain criteria can be done. The filtered data is then displayed in a list of entries, a pie chart with the number of each dog breed in the filtered data, and a map showing the location of each dog in the filtered data.

## Motivation

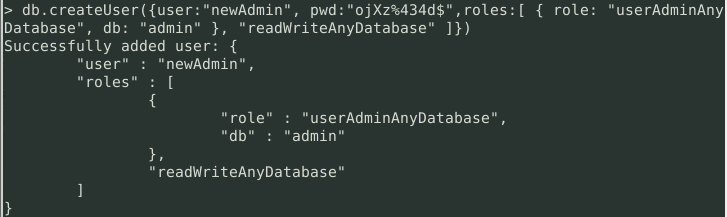
This app was created to provide an easy way for Grazioso Salvare to find shelter dogs. Without this interface, it would be much more difficult to find candidate dogs in the MongoDB database.

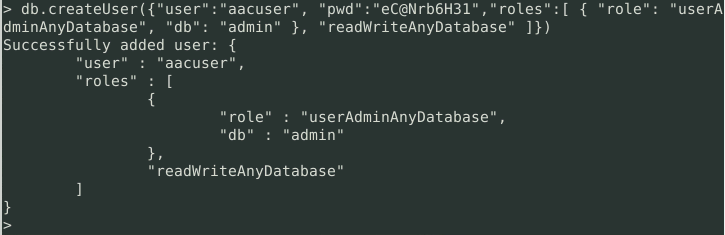
## Getting Started

To begin with, make sure you have a MongoDB server running with an imported database. In this example, I import a database named “AAC”

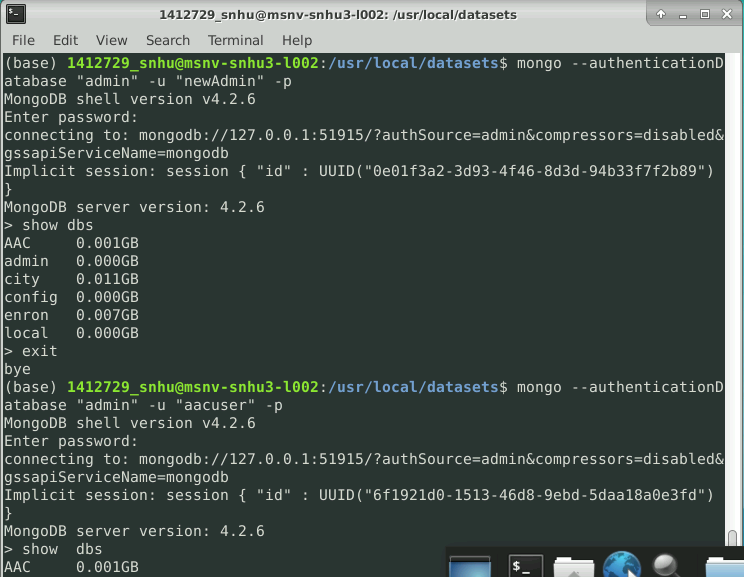


Next, create an admin account for your MongoDB server. In this example I created two accounts, NewAdmin and aacuser.





As you can see in the following image, both created users are able to access the test server.

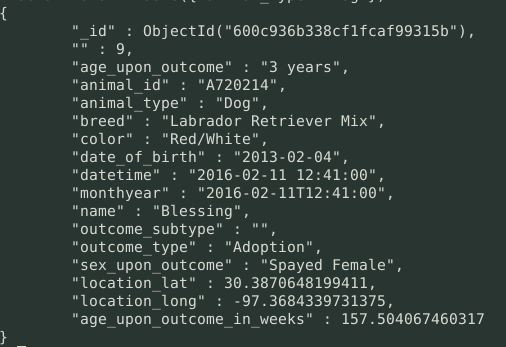


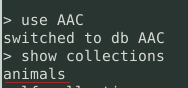
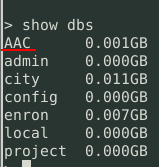
Next, start your MongoDB server and take note of the port (in this case port 51915).

Port Number

Also take note of the desired database (in this case “AAC”) and the desired collection (in this case, animals).

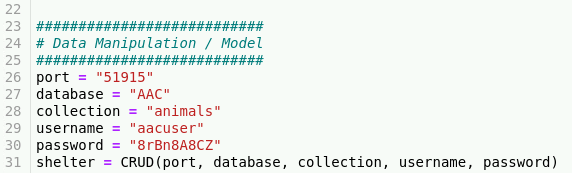
Note:(the app will only work with a database with documents formatted such as in the following screenshot)





Next download the *crud.py* file, the *GraziosoSalvareShelterDogFinder.ipynb* file, and the *Grazioso Salvare Logo.png* file and put them in the same folder.

After this, edit the GraziosoSalvareShelterDogFinder.ipynb file variables on lines 26-30 with your given port, database, collection, username, and password such as in the following screenshot.



## Installation

To use this class, there are a few tools to install.

1. Python 3

To install this, open the terminal and input “sudo apt-get install python3-pip"

1. MongoDB
2. PyMongo

To install this, open the terminal and type “python -m pip install PyMongo”

1. A text editor or python IDE
2. Jupyter Notebook

To install this, open the terminal and type “pip3 install Jupyter”

## Usage

This app consists of 4 elements in the dashboard, the radio buttons, the data table, the pie chart, and the map. These elements were implemented using the “dash” framework which allows for the use of HTML in python code.

The dash framework was used for it’s vast capabilities regarding the link between HTML and python, and for it’s extensive collection of useful elements such as the map and pie charts.

The database used was a NOSQL database called MongoDB. This database type was used because of it’s flexible model and for it’s useful interaction with python through the PyMongo external tool.

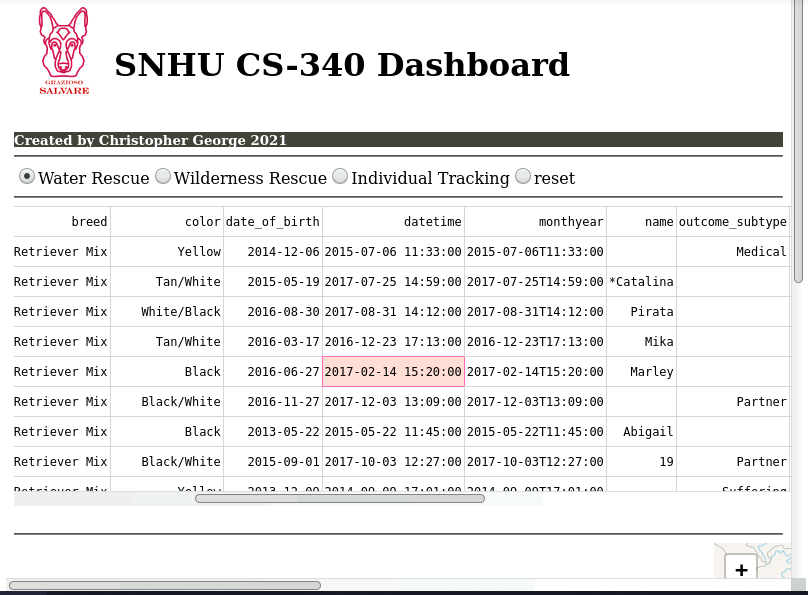
To use this app, simple select the desired filer by selecting a radio button (Water Rescue, Wilderness Rescue, Individual Tracking, or Reset), and then looking through the displayed data in each element.

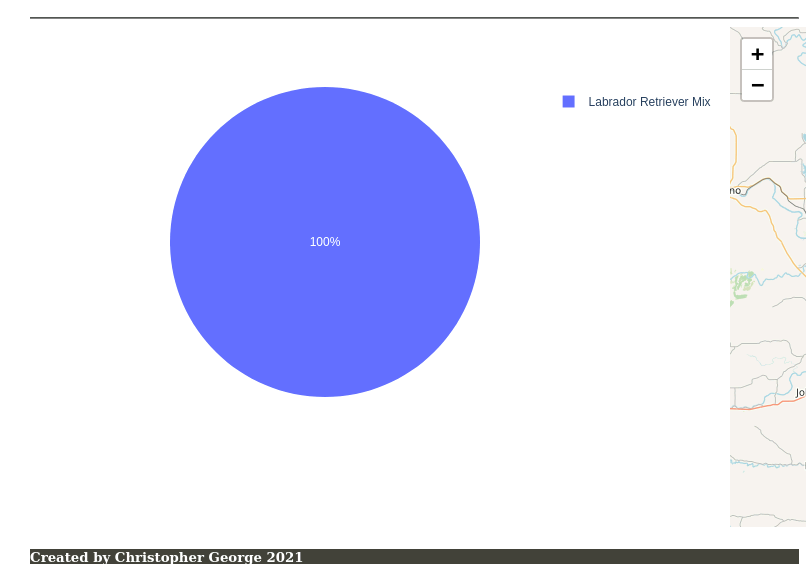
The data table displays all documents which match the filter, the pie chart displays the number of each breed which match the filter, and the map displays all locations of the dogs that match the filter along with a hover tag which displays the dog breed and a clickable tag which displays the dog name.

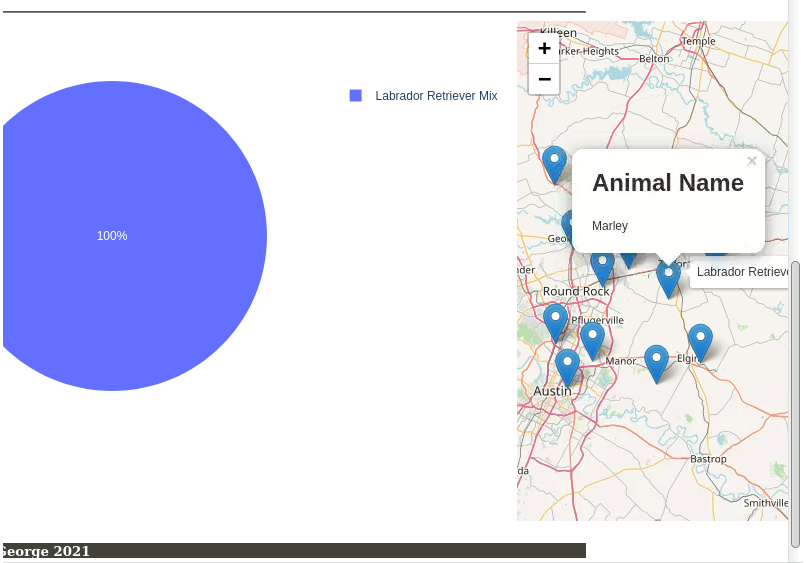
The usage and outputs of the radio buttons are shown below with a test database being used.

***Inputs and outputs***

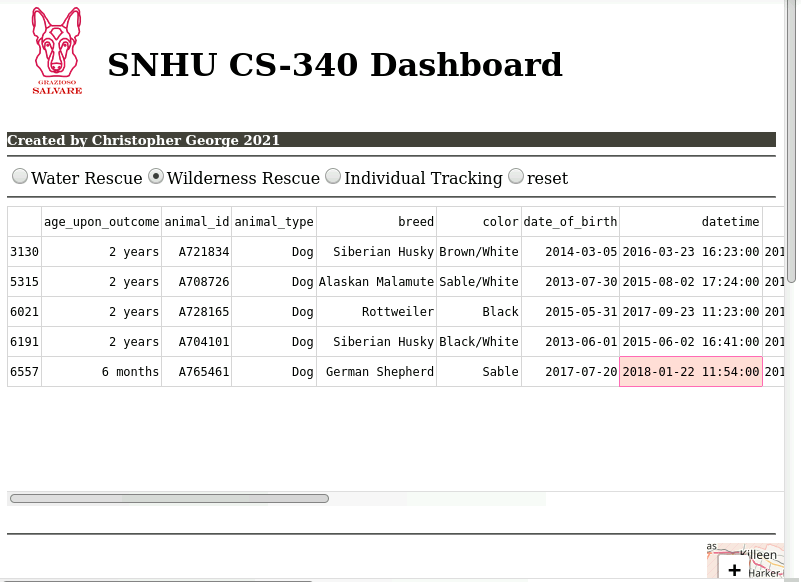
*Water Rescue Button*

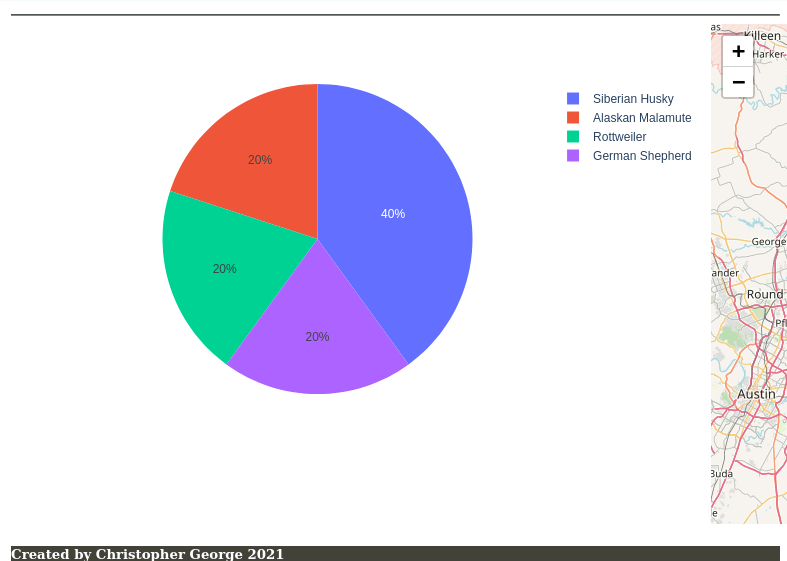
**

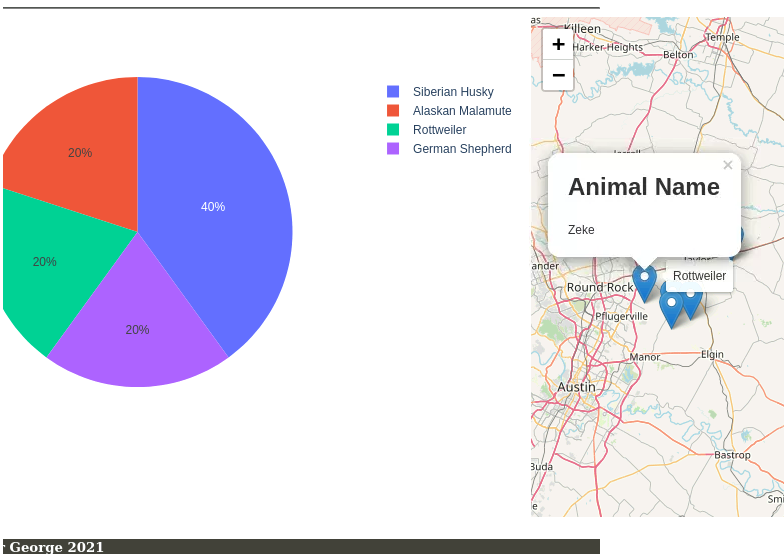
**

**

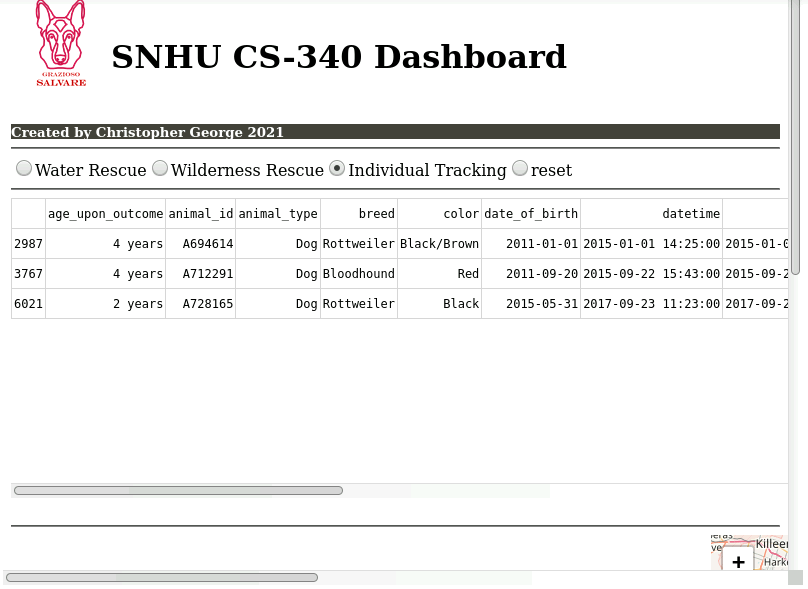
*Wilderness Rescue Button*

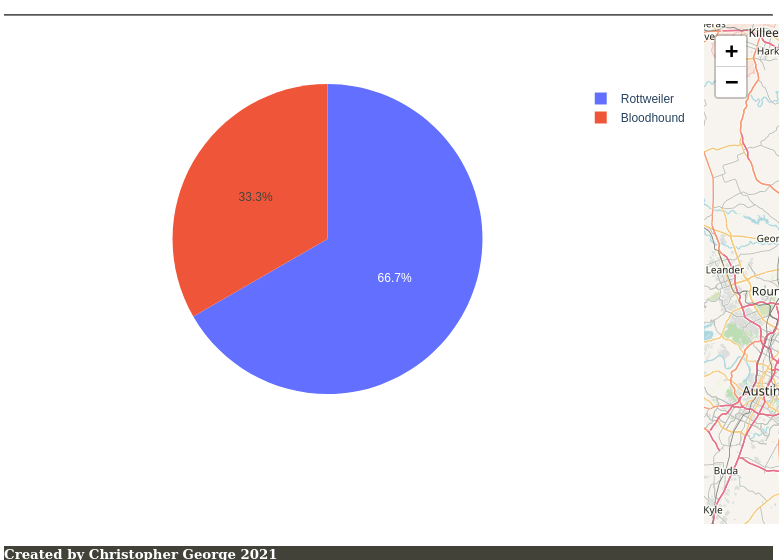
**

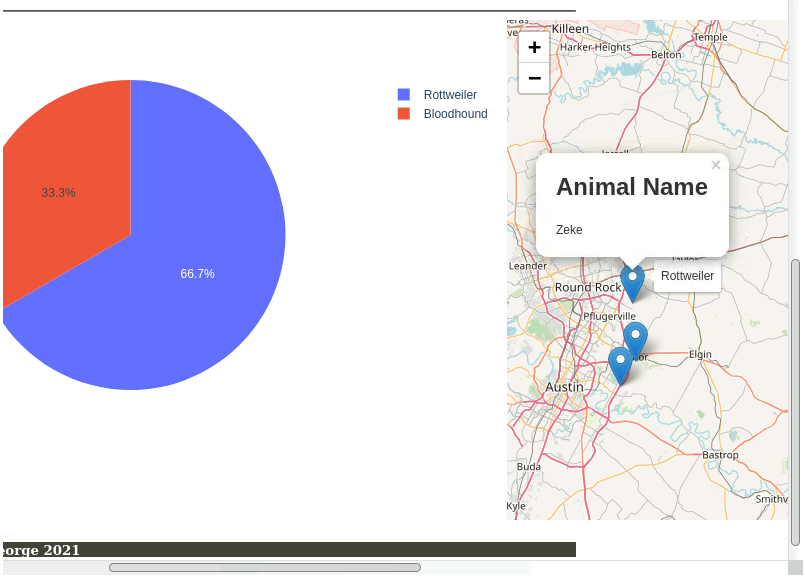
**

**

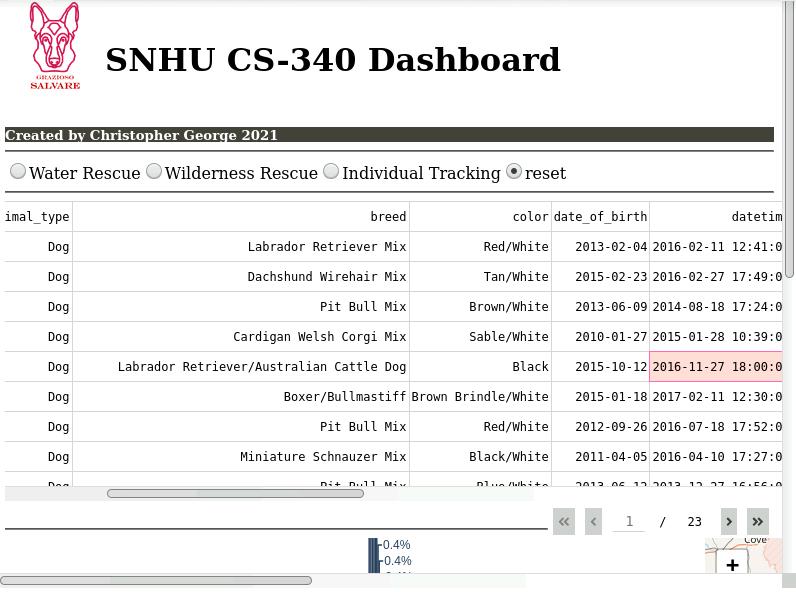
*Individual Tracking Button*

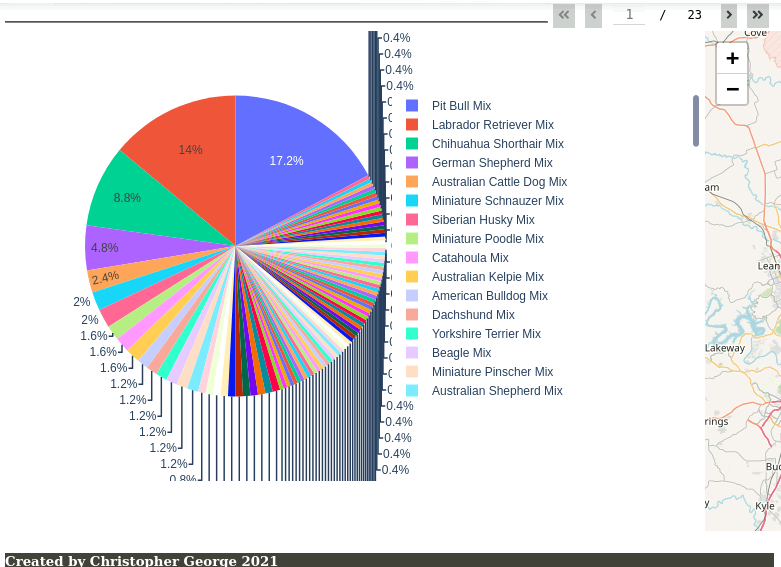
**

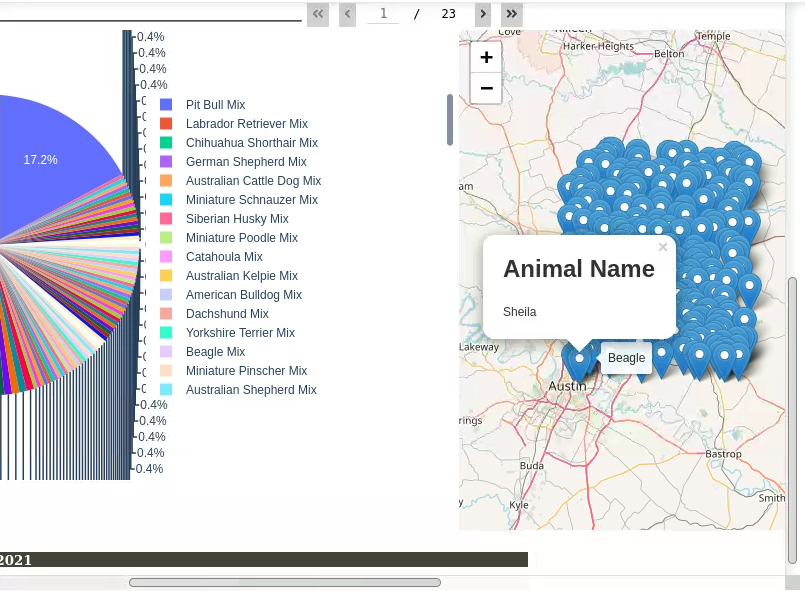
**

**

*Reset Button*

**

**

**

***Radio Buttons***

There are 4 radio buttons that each filter the raw data in specific ways. Depending which button is pressed, a different database read query will be used to retrieve the data from the database and display it in the subsequent elements.

. The “Water Rescue” button displays all dogs matching the attributes defined in the specification document for Water Rescue dogs. The “Wilderness Rescue” button displays all dogs matching the attributes defined in the specification document for Mountain or Wilderness dogs. The “Individual Tracking” button displays all dogs matching the attributes defined in the specification document for Disaster or Individual Tracking dogs. And finally, the reset button displays all dogs in the database without any filtering. This created directly in an HTML element.

These buttons were fairly easy to implament and simply consisted of creating radio buttons elements with the dash “RadioItems” element. A value was then set to each radio option and each value set to the id “filter-type”.

***Data Table***

This data table displays all documents which match the filter set by the radio button element.

It is created with the dash “dataTable” element. There is a function called “update\_Dashboard’ which handles the filtering and takes the filter from the radio buttons. With this filter, it creates a new dataframe replacing the old dataframe which only reads filtered documents from the database.

The dataframe is then filled with this data and the data is also passed to the other elements through the variables “data” and “columns”. This is then displayed in an HTML element with the id of “datatable-id”.

This was pretty easy to implament and after some minor query formatting issues, it was up and running.

***Pie Chart***

The pie chart is created in a function called “update-graph” which takes the data variable from the “update\_graphs” function. At this point, the data variable is basically a list of documents which are in the form of python dictionaries. In order for the pie graph to behave like I wanted it to (to list the number of each breed) I appended a new entry to each dictionary in the list called “count” each with a value of 1. With this, I was able to return a pie chart dash element with the new data table, the values of count, and the names of breed. From this point the graphs automatically adds up the value of count for each dictionary and successfully displays the proper breed numbers. This is displayed in an HTML element with the id of “graph-id”.

This feature took me a really long time to implament because I was misunderstanding the datatype passed to this function. As such , I was trying to make changes to the data variable in ways that don’t make sense for it’s datatype.

***Map***

The map is created in the “update\_map” function which takes the data variable and then uses that information for return a map element with the proper number of children and proper data. To do this, an array is created with information from the data variable which is appended to the end of the created array using a for loop. This array is then used in the return of the dash map element. This is then displayed in the HTML element with the id of “map-id”.

This element wasn’t too hard to create and just required a little knowledge about dictionaries to complete.

## Contact

Christopher George