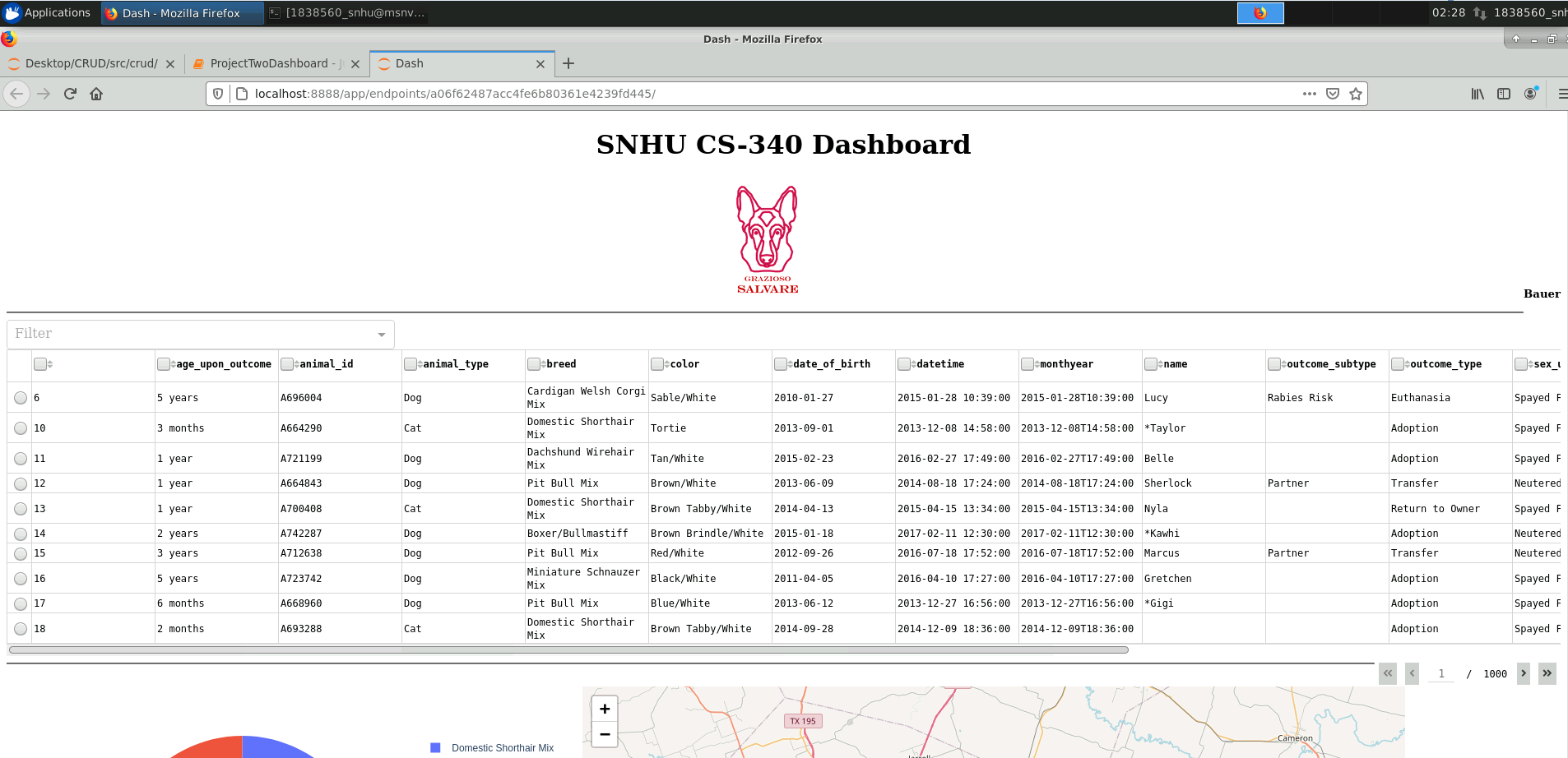
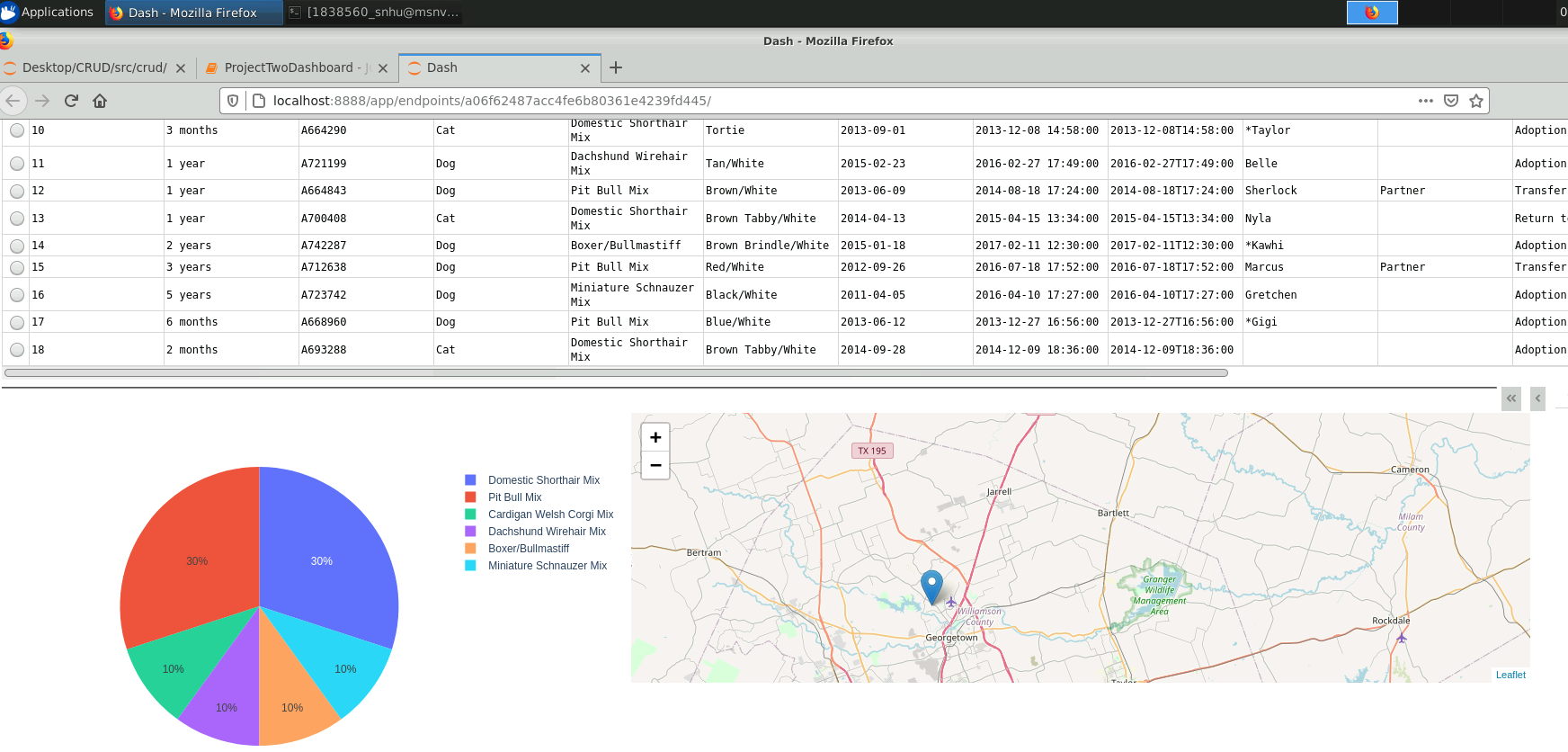
# CS 340 README Template

## Functionality

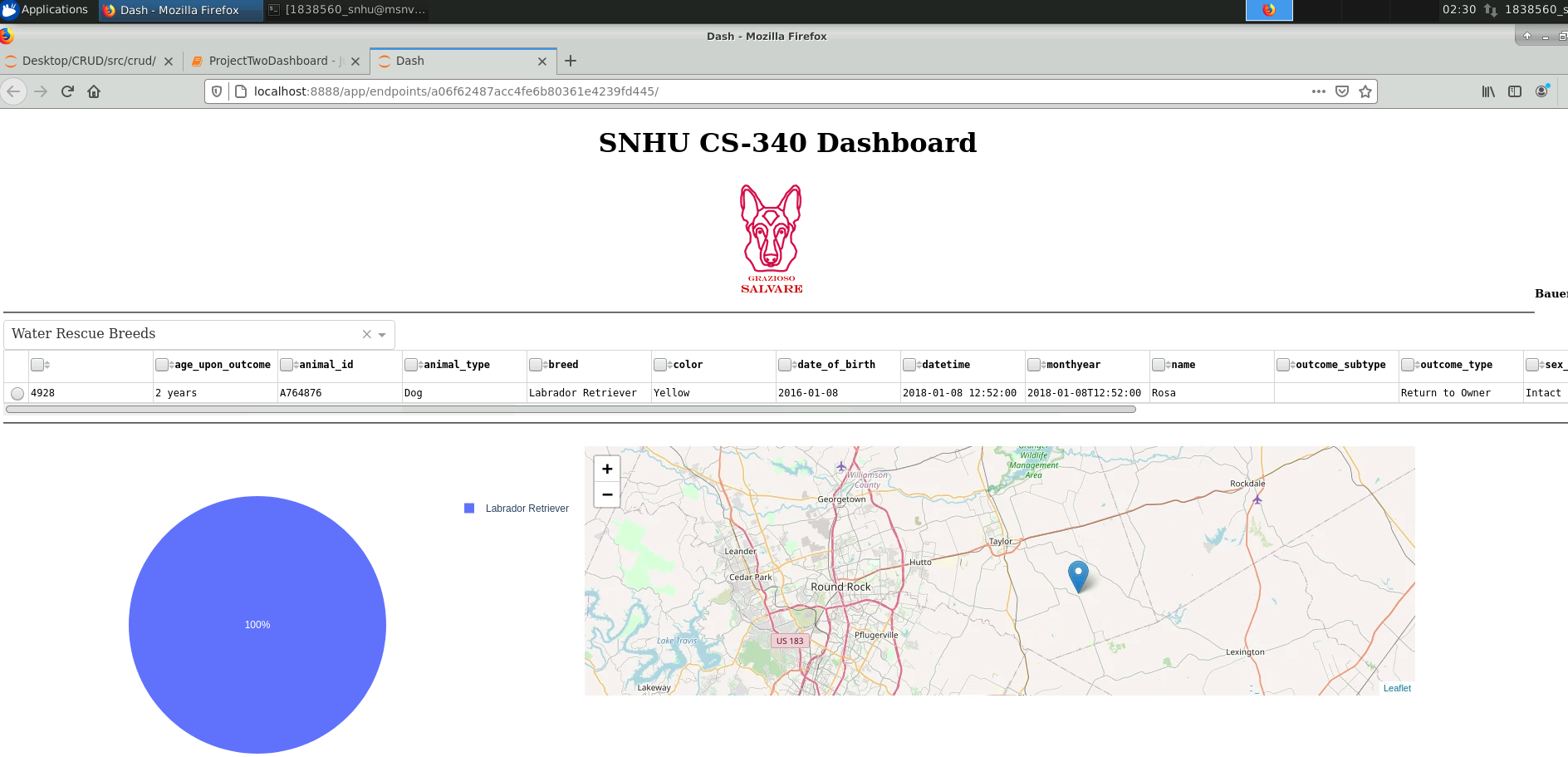
This reads the database into a table to be easily read. This table allows filtering for the types of breeds for water rescue, wilderness rescue, or disaster rescue. The data on the page is shown as a pie graph that breaks down the percentage of that breed that is on the page. There is a map that shows the location of the first dog by default but allows the user to select the radio button next to each animal to show their location on the map.

The pictures below show the default data that is loaded into the table.

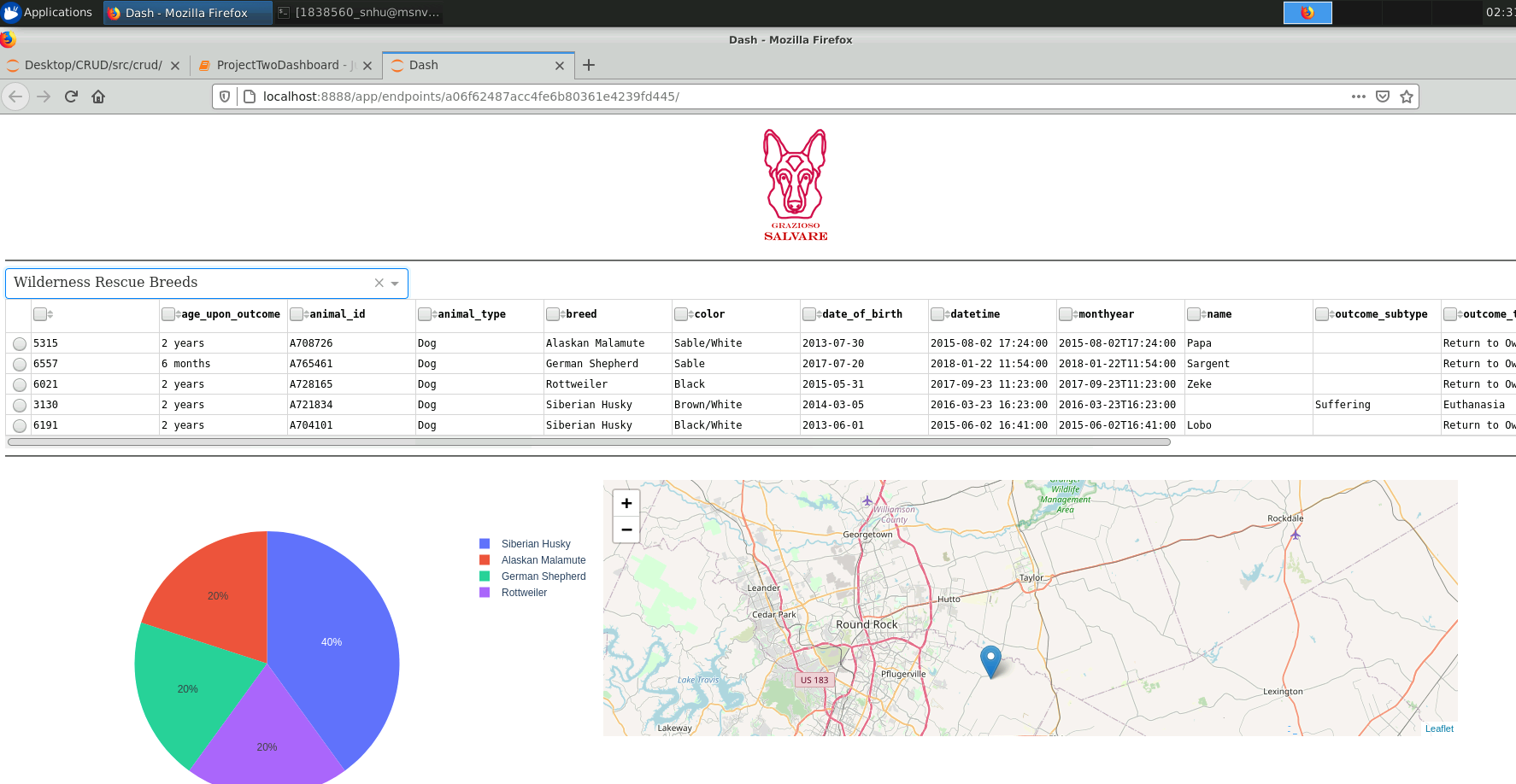




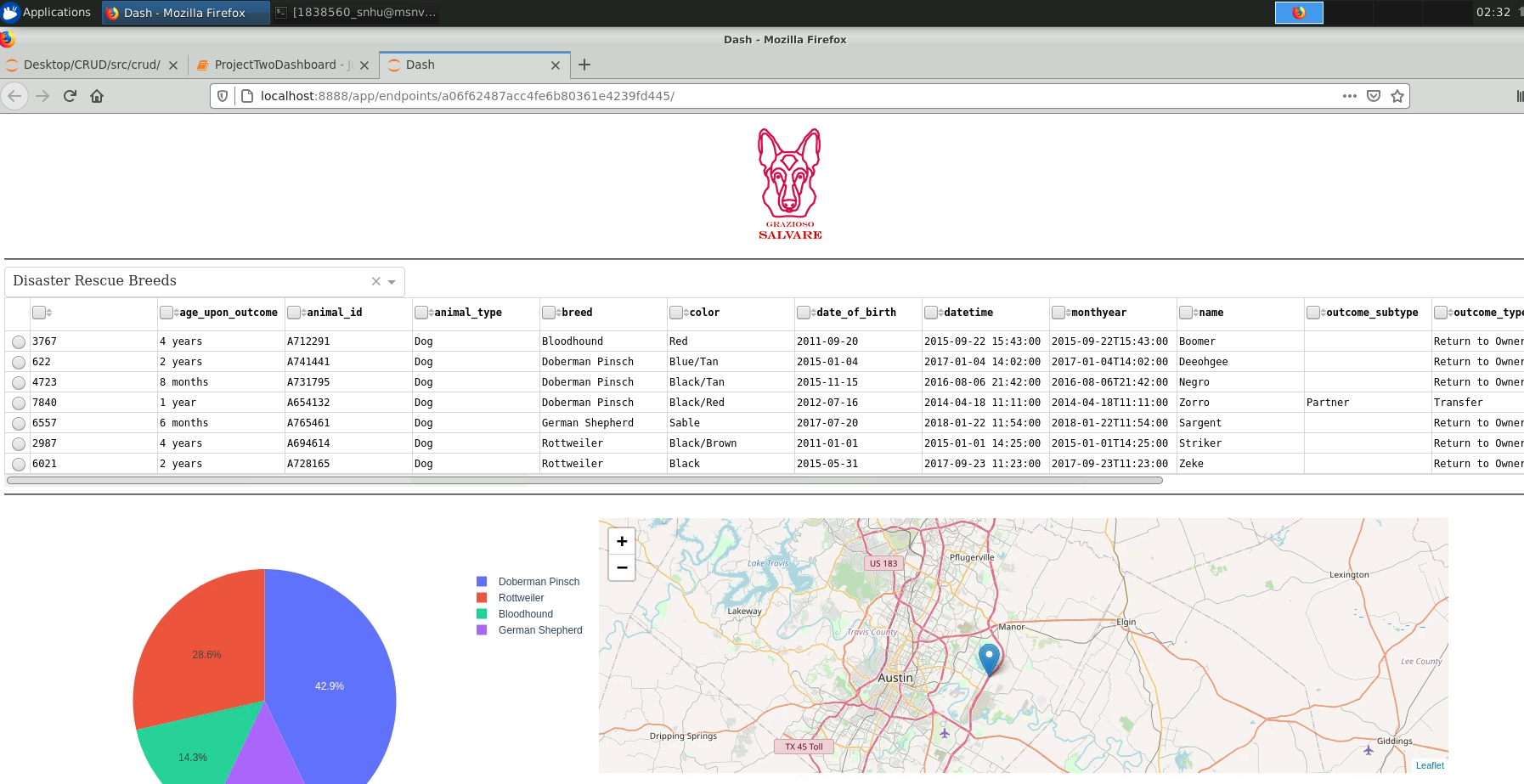
The picture below shows the water rescue breeds filter for the data in the table.



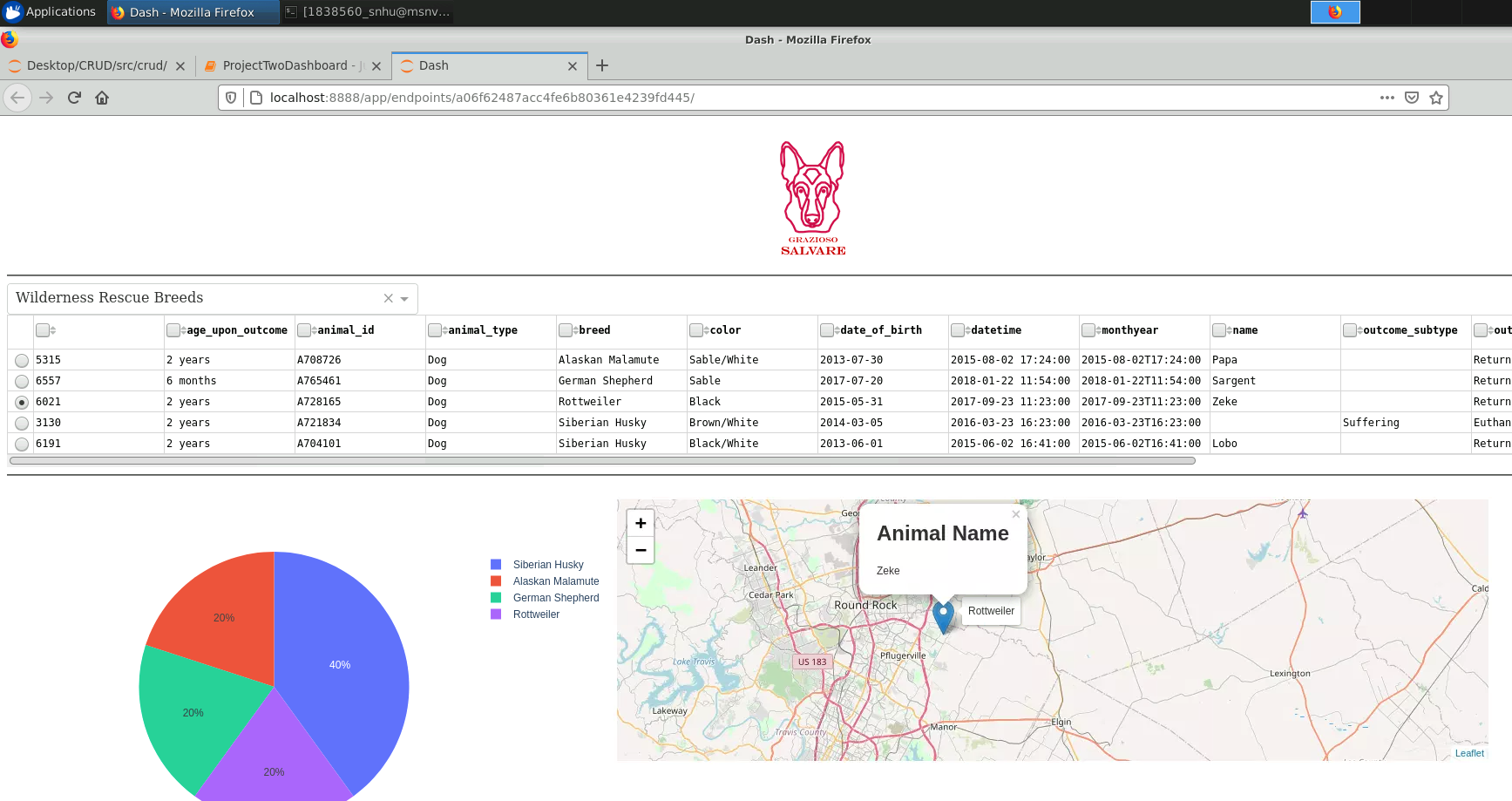
The picture below shows the wilderness rescue breeds filter for the data in the table.



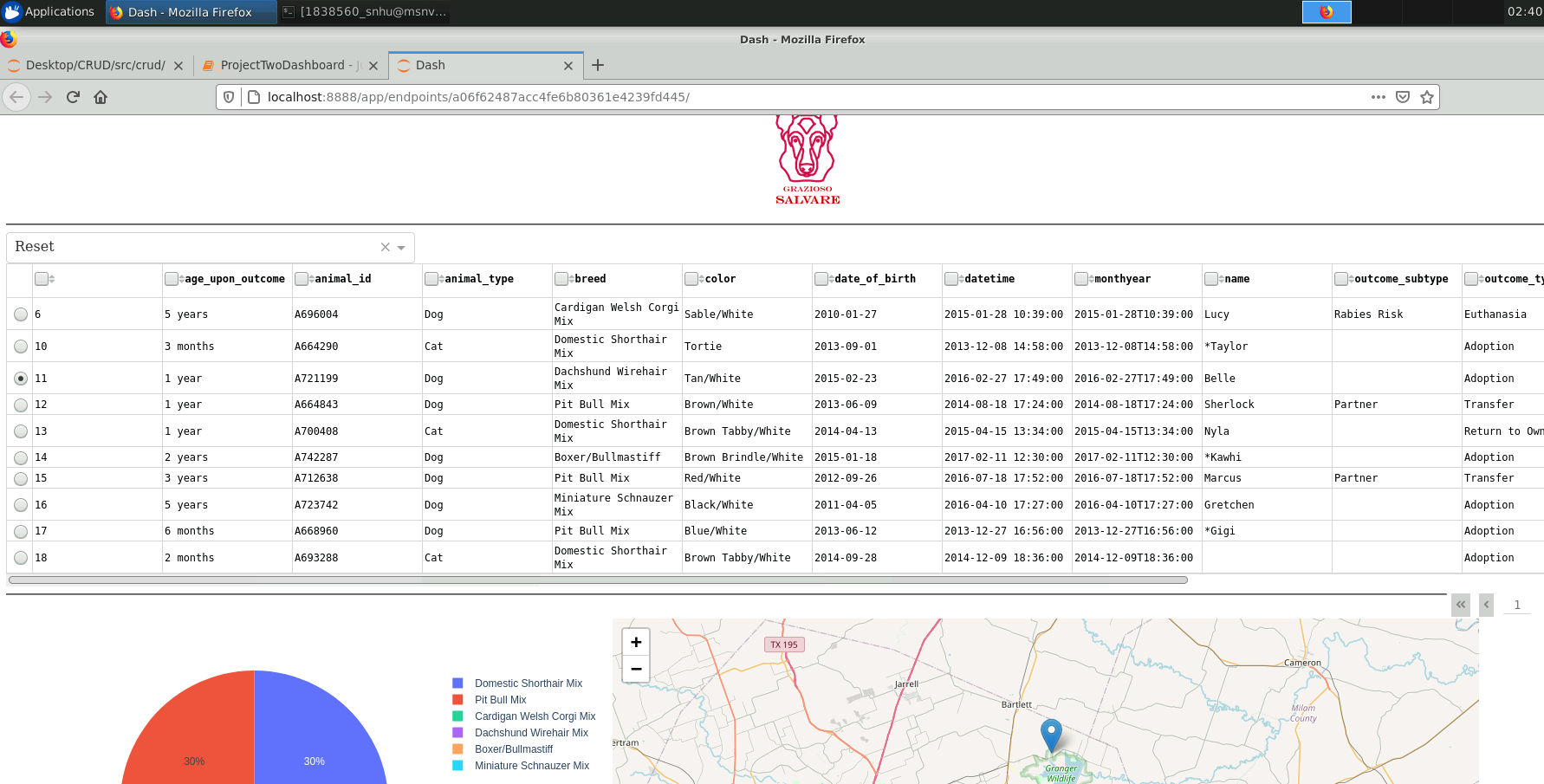
The picture below shows the disaster rescue breeds filter for the data in the table.



The picture below shows the disaster rescue breeds filter and selecting the third dog in the table which updates the map to that dog’s location.



The picture below shows the reset filter to set the table back to default.



## Tools

**Mongodb**

Mongodb is the database used to hold the data for all the animals. This database is very straight forward and easy to use. Mongodb has great functionality and flexibility to achieve varied results. Mongodb goes hand in hand with python using pymongo. Being able to use python allows for further flexibility. Since python uses dictionaries and mongodb does as well, this allows the data to be easily read in and manipulated without a bunch of conversions.

**Dash**

The dash framework was used to provide the structure the data was put into. The table, map, and graph used the dash framework. Dash is a relatively easy framework to start to use for basic structures but has advanced parts to allow for further customization. Dash is python based which makes it work with mongodb without much issue. The table used is a datatable in dash that displays the data brought in from mongodb. There are filters in a dropdown that dash provides in the framework that allows the data to be filtered and returned to the table with the new filtered data. The data that is being viewed is sent to the graph and broken down by percentage of the breed listed. The first animal’s location is sent to the map or the selected animal’s location is sent to the map. These are all features within dash’s framework.

**Links**

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

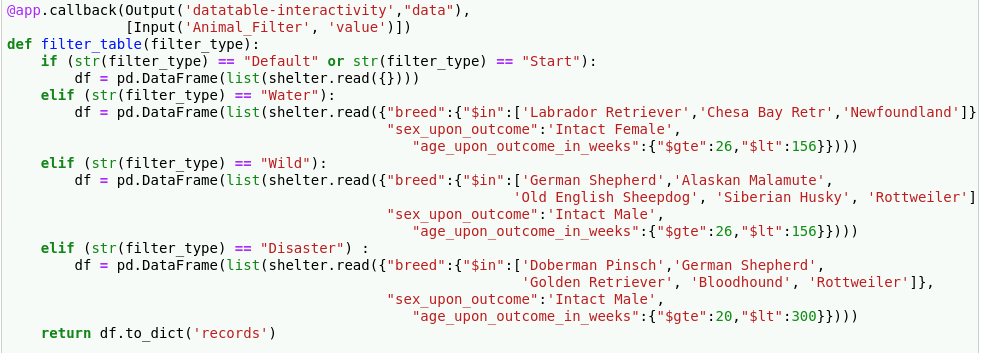
<https://docs.mongodb.com/manual/>

**Steps (Included is the code to reproduce this)**

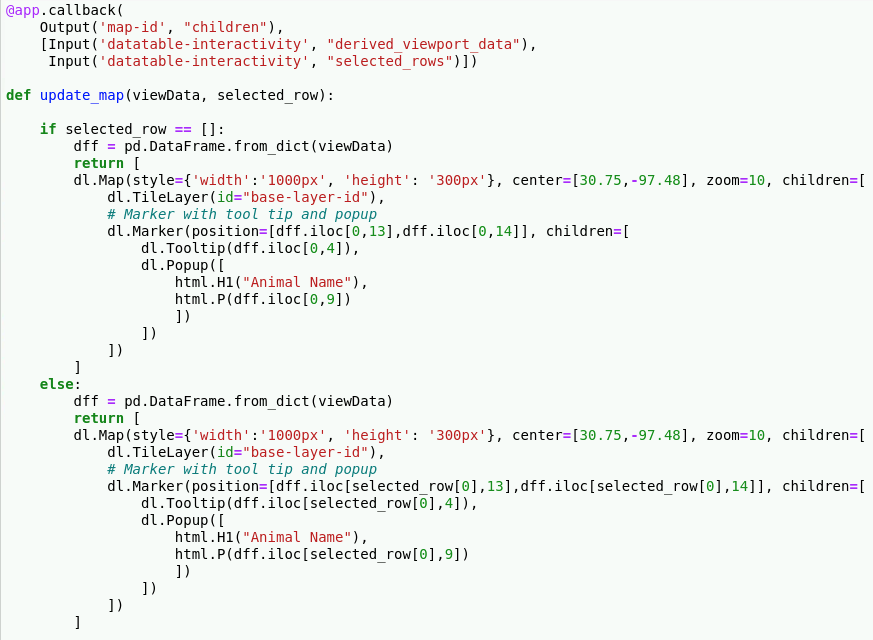
The steps taken were:

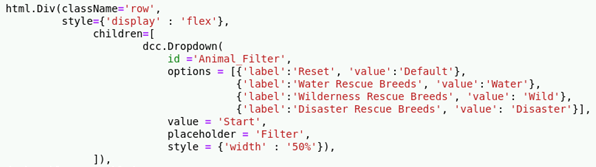
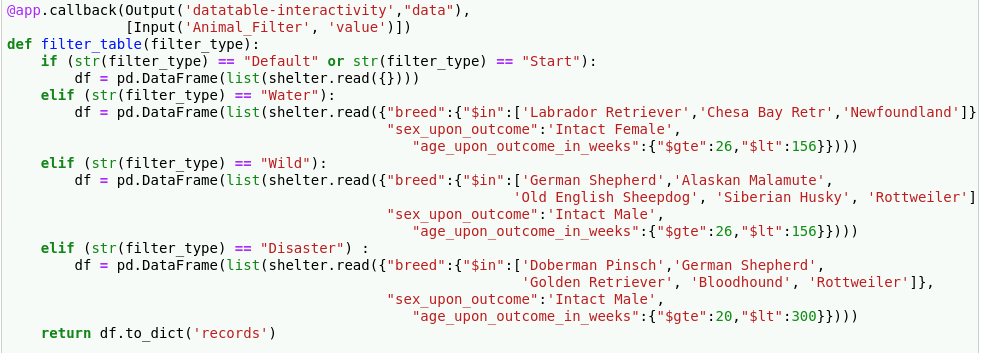
1. First to connect to the server and be able to load the data.

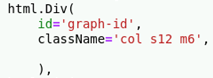


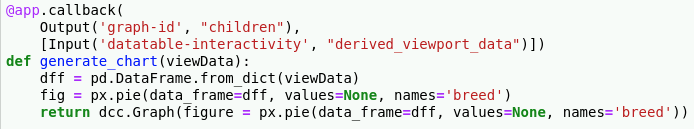
1. Take the data loaded and send it into a table in dash
   1. This creates the table
   2. The first if statement loads the data into the table 
2. Then create a map from dash and load the location from the first animal into the map to show the location
   1. This creates a spot for the map

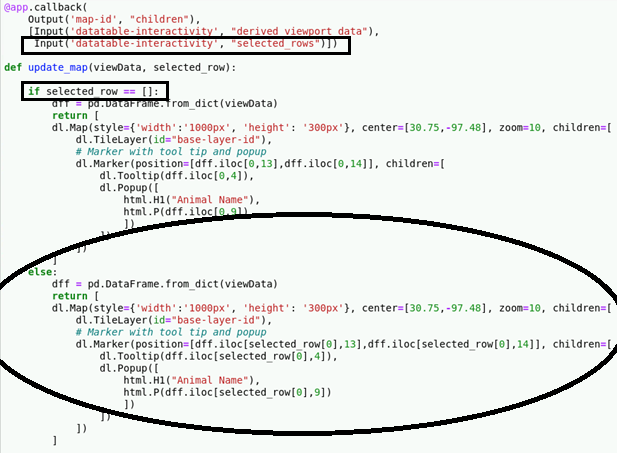
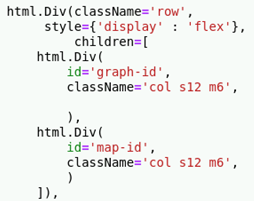


* 1. This sends the map code and location of the first animal or selected animal to the map

1. Then adding filters to filter the table for the desired breeds, sex, and age based on the type of rescue.
   1. This adds the dropdown to the table and sends the value to the callback when the user selects a specific dropdown. 
   2. This reads the value sent from the dropdown and filters based on what dropdown was selected
2. After that a graph was added and the data from the current page of the table was sent to the graph to display the breeds by percentage.
   1. This creates a spot for the graph



* 1. This sends the graph code and percentage of breeds of the first page to the graph 

1. Next was to add a radio button to select a specific animal to show on the map and have that specific animal’s location sent to the map.
   1. This adds the radio buttons next to each animal on the table 
   2. This sends the location of the selected animal to the map. The input “selected\_row” is checked if it is an empty list in the if statement. The else statement sends the selected row to the map if the user makes a selection 
2. Finally adding the logo, title, customizing and cleaning the look of the page was done.
   1. This adds the logo, title, and author 
   2. This makes the graph and map appear next to each other on the page

## Challenges

The biggest challenge was being unfamiliar with html type of code in general which is similar to dash. Not being fully comfortable with dash or front end development was a hurdle I overcame by doing research on the code and how it works. Another challenge was being a bit rusty with python and having to remember or look up how things worked. All of the challenges made completing take extra time but I kept at it trying different things and researching until I was able to accomplish the task.

## Contact

Michael Bauer