# Grasioso Salvare Rescue Dashboard

## About the Project/Project Title

The Grasioso Salvare Dashboard allows rescue teams to work with shelters in their areas nationwide to quickly locate rescue animals suited to various rescue types. Based on need, rescuers can quickly locate the animals needed for emergent situations.

## Motivation

By providing an at-a-glance view into the status of the animals, Grazioso Salvare can quickly react to the needs of search and rescue missions in need of search animals.

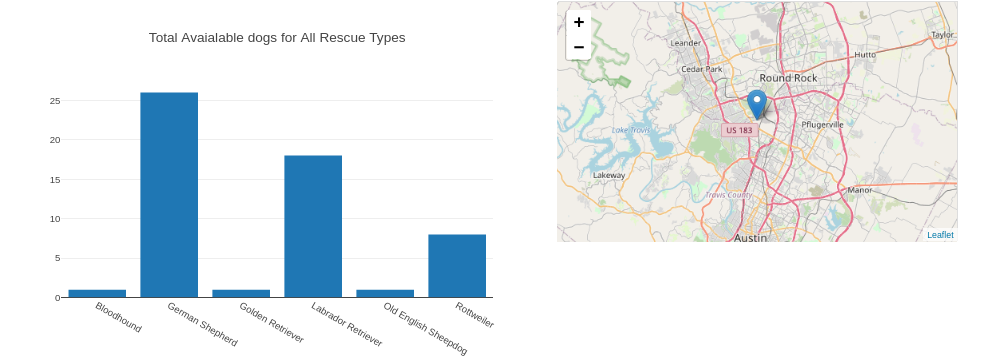
## Getting Started

The initial state of the dashboard is to load all animals into a table. The drop down filter will change the table to only those animals suited to the chosen rescue type filter. The graph and map will both update to show the total number of dogs and which breeds are available and the locator pin will also update based on which row is selected in the table.

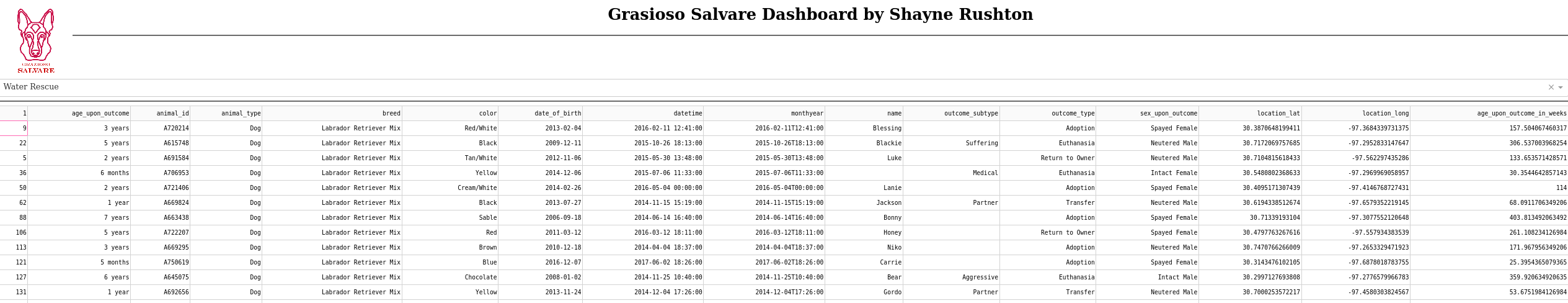
Initial State



Graph and Map:



## Filtered State:

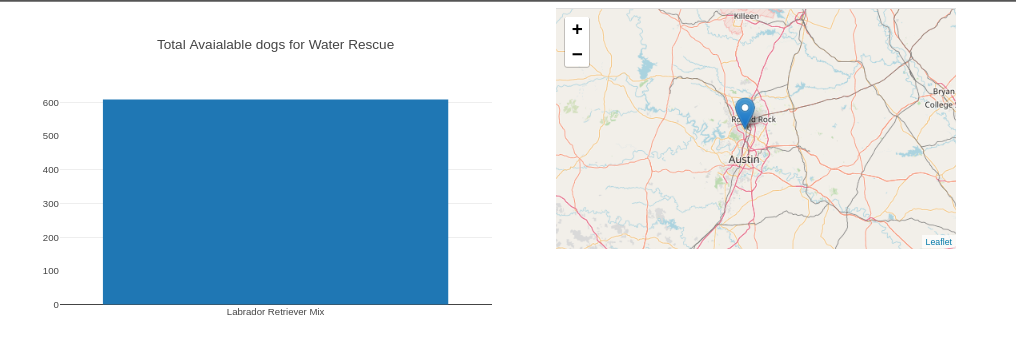


## Usage

These screenshots show the class and methods, then the test script used to test the functionality of the methods within the class. The outcome is shown in the screenshots section below. Beginning with the Animal Shelter Class:

**Test Script**

Map and Graph



### Code Example

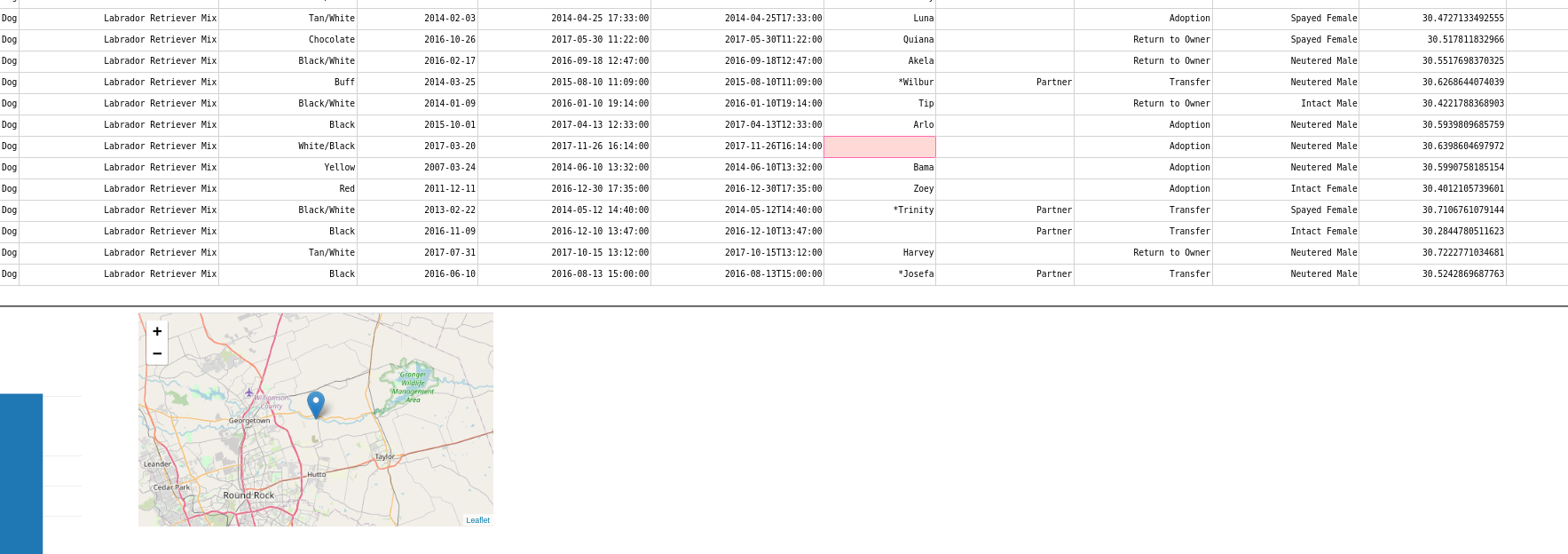
Here is code to add a document to the mongoDB collection. In our case we use a database AAC and a collection called animals

Set your database

self.database = self.client[‘yourDatabase’]

replace yourDatabase with the name of your database

Any selected cell in the table will change the map location based on the location coordinates for that row:





**Project Build**

The project was built in Jupyter Notebook using a MongoDB database. First, the queries were created to allow connection to the database and to allow insertion into the database for initial installation using the create function provided. After initial upload, the dashboard will call the get all function and display all database items.

After the information was displayed correctly a drop down option was added as a filter. Using the Dash framework and python, the callback functions will filter the data based on rescue type and update the map and graph while also allowing a location update in the map.

The rescue filter is simple. It just queries the database using the get function to map the rescue type chosen to the list of dogs selected for that rescue type. The filter then looks in the database to see how may of that breed is available and updates the table based on that.

The map filter was a little tougher. It required the use of the dcc.Store feature in Dash. First the selected cell analyzed the row and returned the fields associated with the location, which were then passed to the Store for use in the update map location call back to provide the x and y coordinates to the map API and the update the map. Using a dictionary proved problematic since the values were difficult to access and parse as actual float data types. Eventually I used a simple list and accessed the location variables via indexing and that was the final touch to the project.