# CS 340 README Project Two Brennan Reed

## About the Project/Project Title

*This project is a Web Dashboard for Grazioso Salvare (GS), an animal rescue organization based in Austin, TX. The project includes an interactive table to display data, a geolocation map and a histogram with breed breakdowns (also both interactive and responsive to filters). This is all built on top of a previously created CRUD module that provides the data/functionality ‘under the hood’ for performing queries via a MongoDB database.*

*MongoDB integrates exceptionally well with Python for a few reasons – making it a great ‘model’ for the ‘controller’ (callbacks through Dash) to integrate with for our MVC model. In addition to Python having strong data manipulation and processing abilities, it also features the NoSQL structure of JSON-like objects instead of the more rigid structure of tables, rows, etc found in a SQL environment – which makes it more flexible and adaptable. The PyMongo driver also contains an easy-to-use API which makes working with the above objects comfortable. Finally, objects retrieved from MongoDB through PyMongo are compatible with dictionaries and lists, which enables the easy manipulation and iteration of these objects as referenced earlier – a great tool when dealing with queries returning dozens, hundreds, or thousands of results.*

*Dash provides practically limitless options for the ‘view’ component of our dashboard by providing great support via callbacks to create the ‘controller’ element with various other elements for the view. For instance, importing Pandas allowed us easier/clean manipulation of data to create a table displaying our results. We couple that with other charts powered through Leaflet, Plotly, etc and use of Dash components to create a clean, intuitive interface for users.*

*Our largest challenge was implementing callbacks that allowed interactivity of the tables and charts, but we were able to persevere by examining callbacks of other working projects. Once we were able to successfully implement our first element, the others were easier to add callbacks to with our required functionality. Our design philosophy was that WE should struggle through these hurdles so that YOU don’t have to!*

## Motivation

*The project exists to allow GS employees to access and interact with their data in a much easier and more user-friendly fashion. The ability to use the dashboard to filter results to match their exact needs drastically improves the ability to help place animals by eliminating excess time spent working through records manually, and greatly improves the speed/user experience even compared to just searching through a database that does not feature an interface/dashboard like this.*

## Getting Started

*The Dashboard can be accessed by IP/hostname, which can be found at* [*http://127.0.0.1:8050/*](http://127.0.0.1:8050/) *- the only other element needed is a working Internet connection. Tips on getting started on performing queries and operations are included below in the ‘Usage’ section.*

## Installation

*Since this is a Web Dashboard, no installation is needed - nothing more is needed than to navigate to* [*http://127.0.0.1:8050/*](http://127.0.0.1:8050/) *at this time. For background, Python is the language used for the code behind this project and MongoDB is the database behind it all. Various other libraries and frameworks are used (such as PyMongo driver, Dash, Plotly, Pandas, and Leaflet), but all of that is handled behind the scenes.*

*For information about what the installation “behind the scenes” looked like, we first created a Python module containing our CRUD class (AnimalShelter) – which also included authentication components to ensure that the database and collection were accessed after authentication of user via username and password.*

*Next, the interface had to be built. We first imported the aforementioned module/class into our code, along with other necessary libraries and components (Dash + other dash components, Plotly, Pandas, etc). We then used those libraries to create a DataFrame in Pandas, a geolocation widget displaying animal location/GPS, and a histogram chart displaying a breakdown of the breeds of the current animals displayed on screen in the table. We implemented four preset filters (one of which is a reset to all) by designing specific pre-built queries for popular/frequently needed options, added native filtering to the table to allow for further customization, and made all the previously mentioned elements of the page respond to said filters. We added in branding (also required yet another import of an image encoder) and voila!*

*The geolocation chart will display basic information (name/breed) about the animal currently shown on screen upon hover and selection of the marker.*

*The histogram displays the count of each breed,* ***only*** *out of those results on screen at that time. It also displays a second element by using color to sort the columns into the different sex\_upon\_outcome options as well – this will not be noticed with any preset filter other than reset/default because those filters require one specific sex each, as a note.*

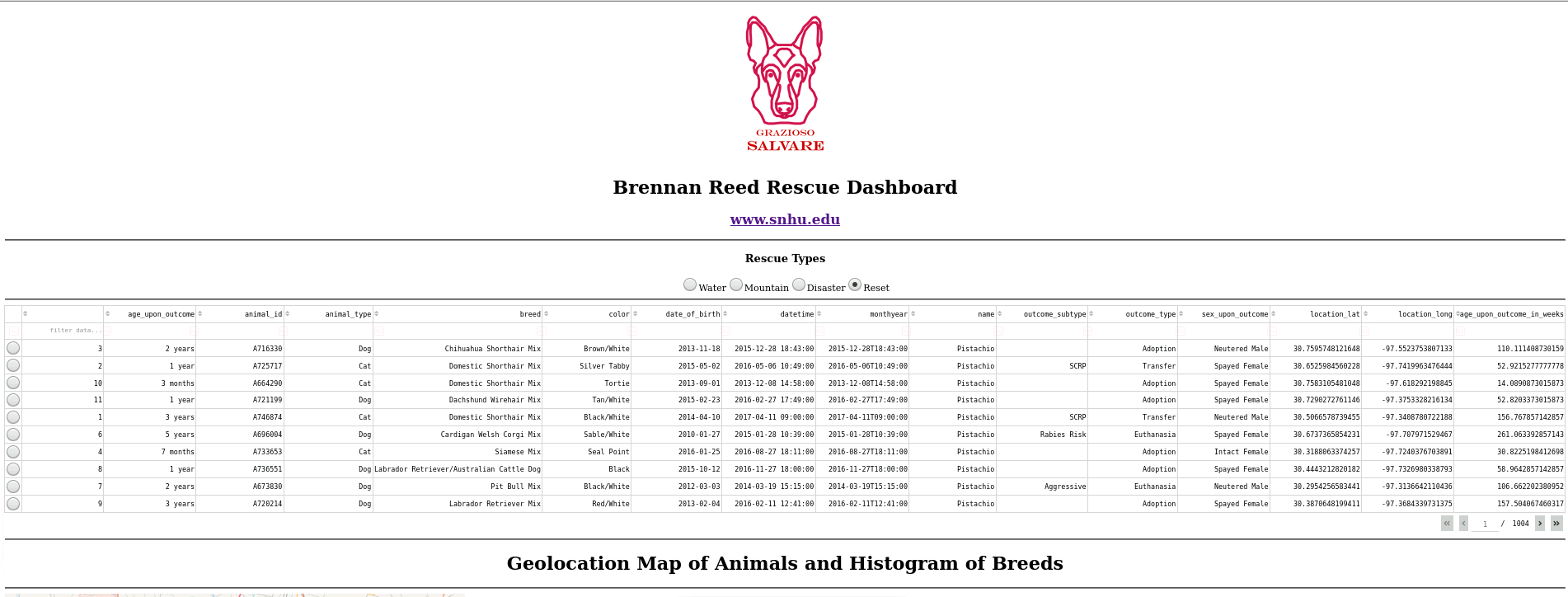
*Usage tips, further guidance, and screenshots are featured starting on the next page and contact information is listed on the last page if additional assistance is needed.*

## Usage

**Examples:**

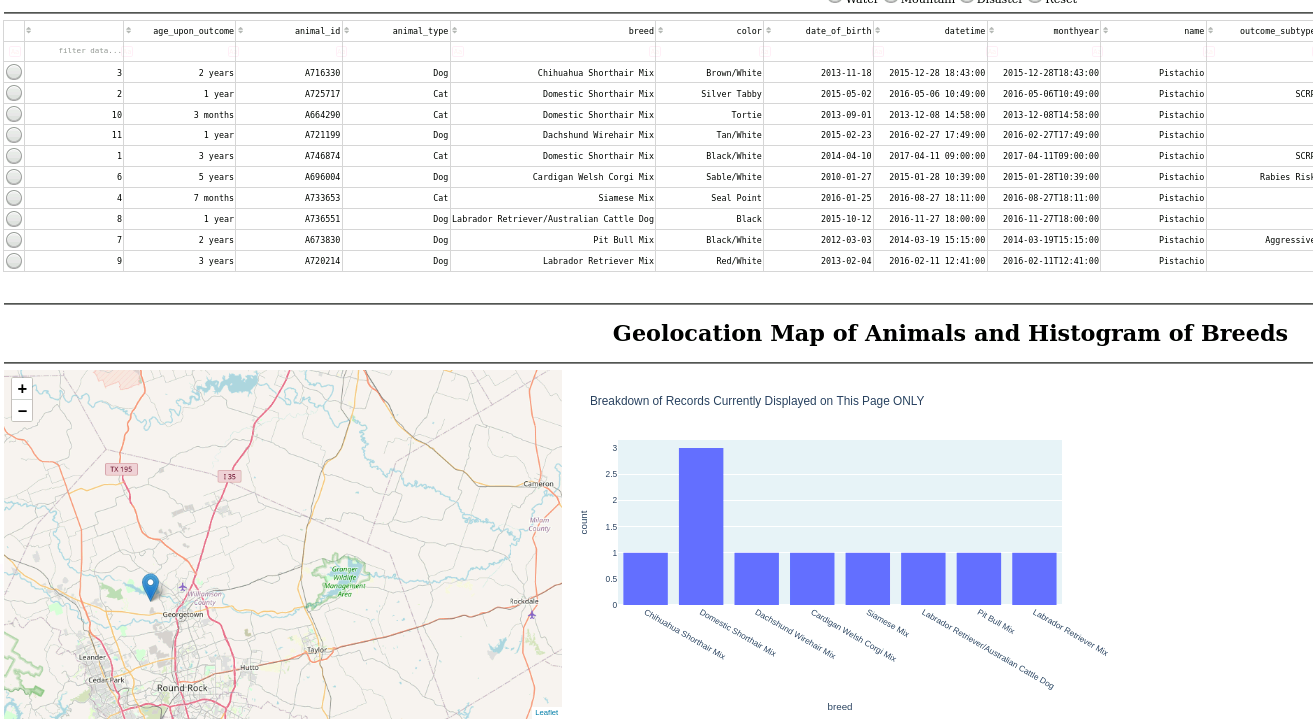
**Default dashboard** displays all results in database and shows the **‘Reset’** radio button pre-selected. If other filters are applied via the radio buttons this can be selected again to display the full database once again.

Image 1 of 2 – second image on next page is more zoomed in and shows table, graph, and map in greater detail (all with default/no filter applied)

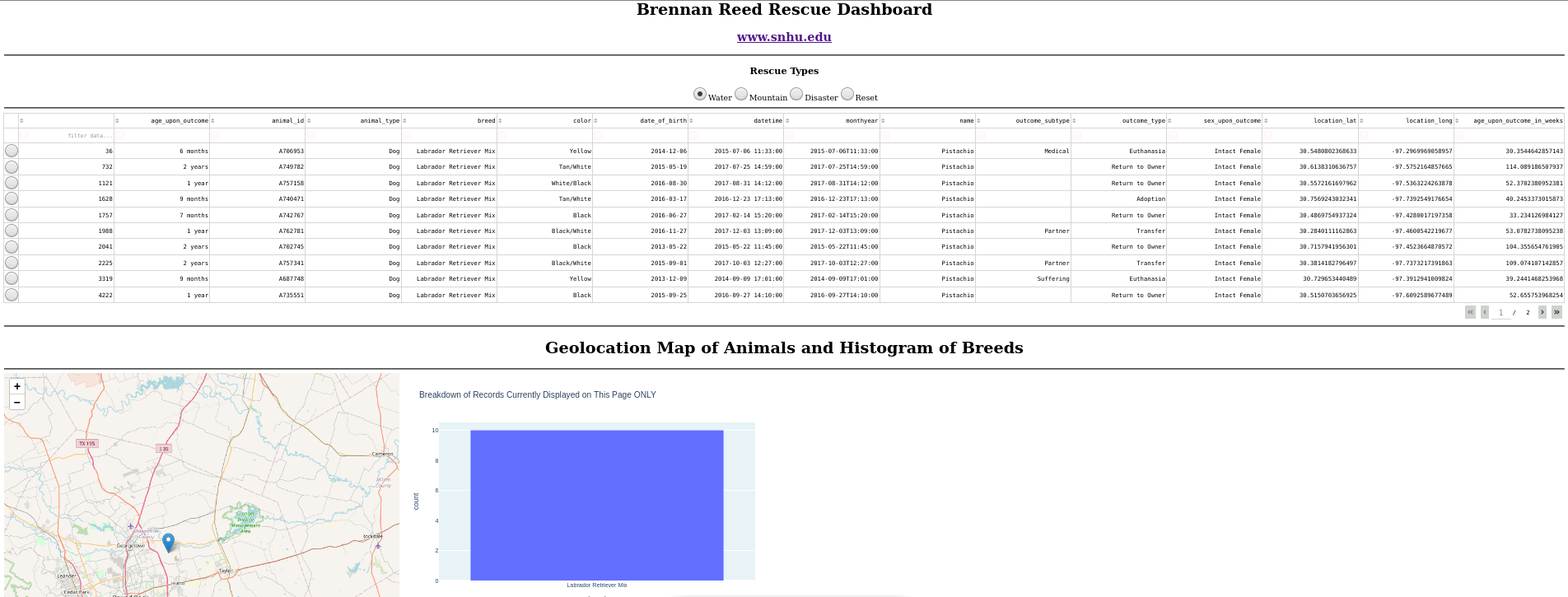


**Default dashboard** displays all results in database and shows the **‘Reset’** radio button pre-selected. If other filters are applied via the radio buttons this can be selected again to display the full database once again.

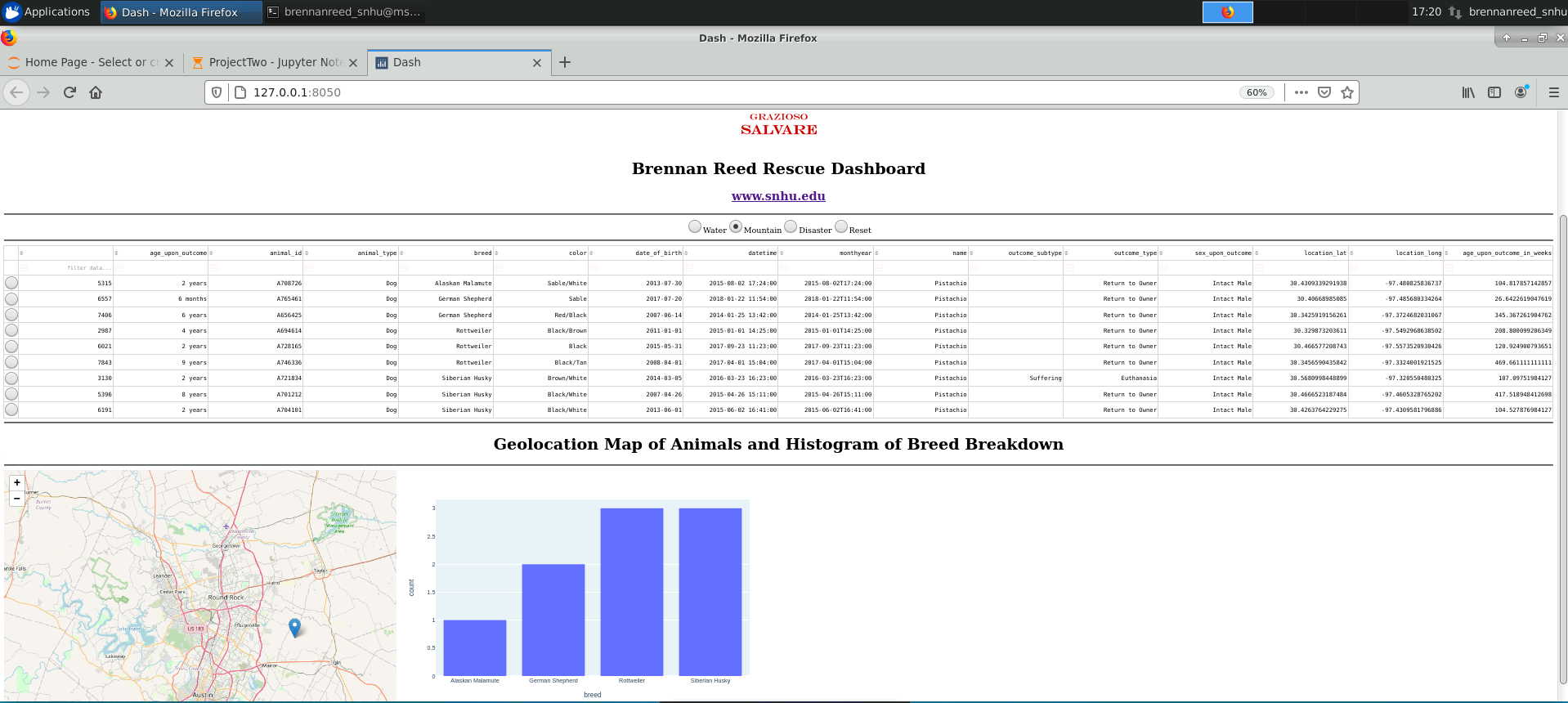
Image 2 of 2



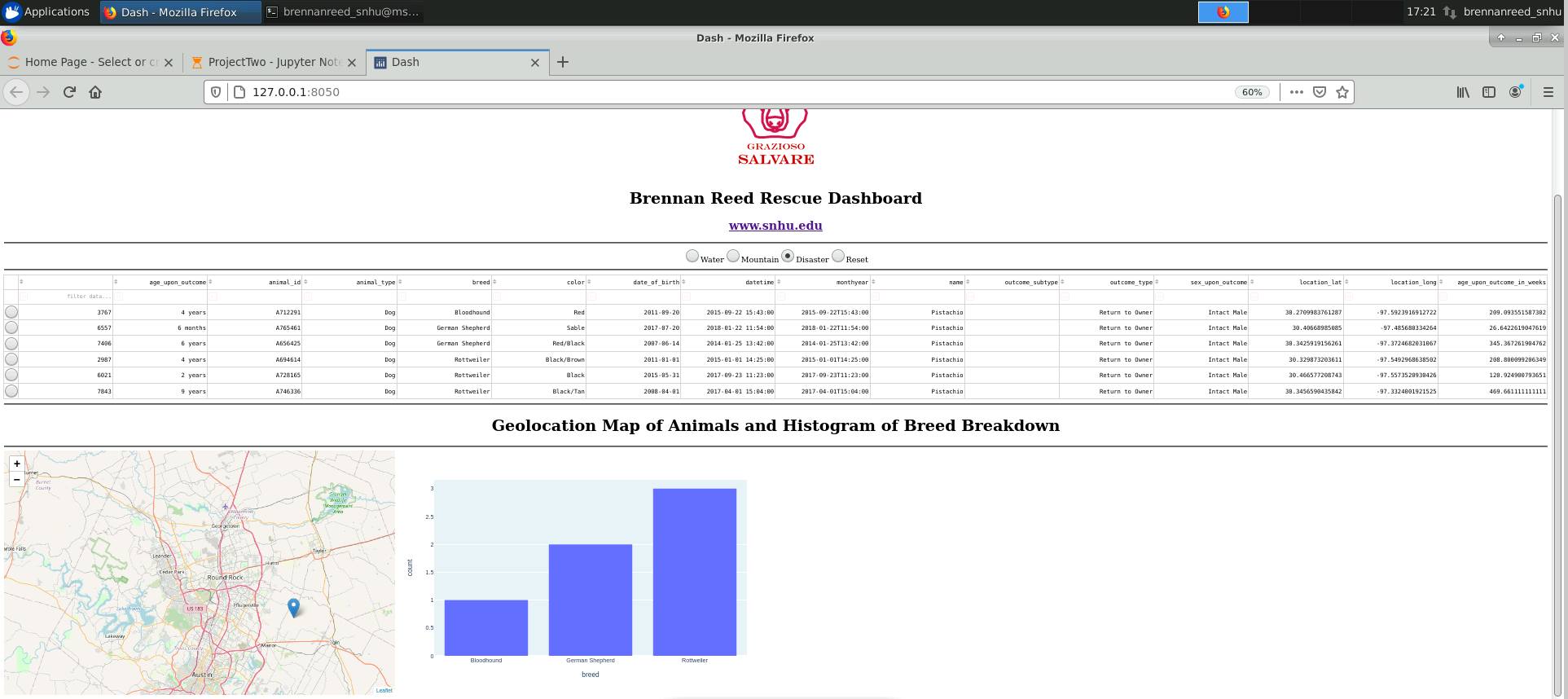
**Water** preset displays Water Rescue animals only.



**Mountain** preset displays Mountain or Wilderness Rescue animals only.

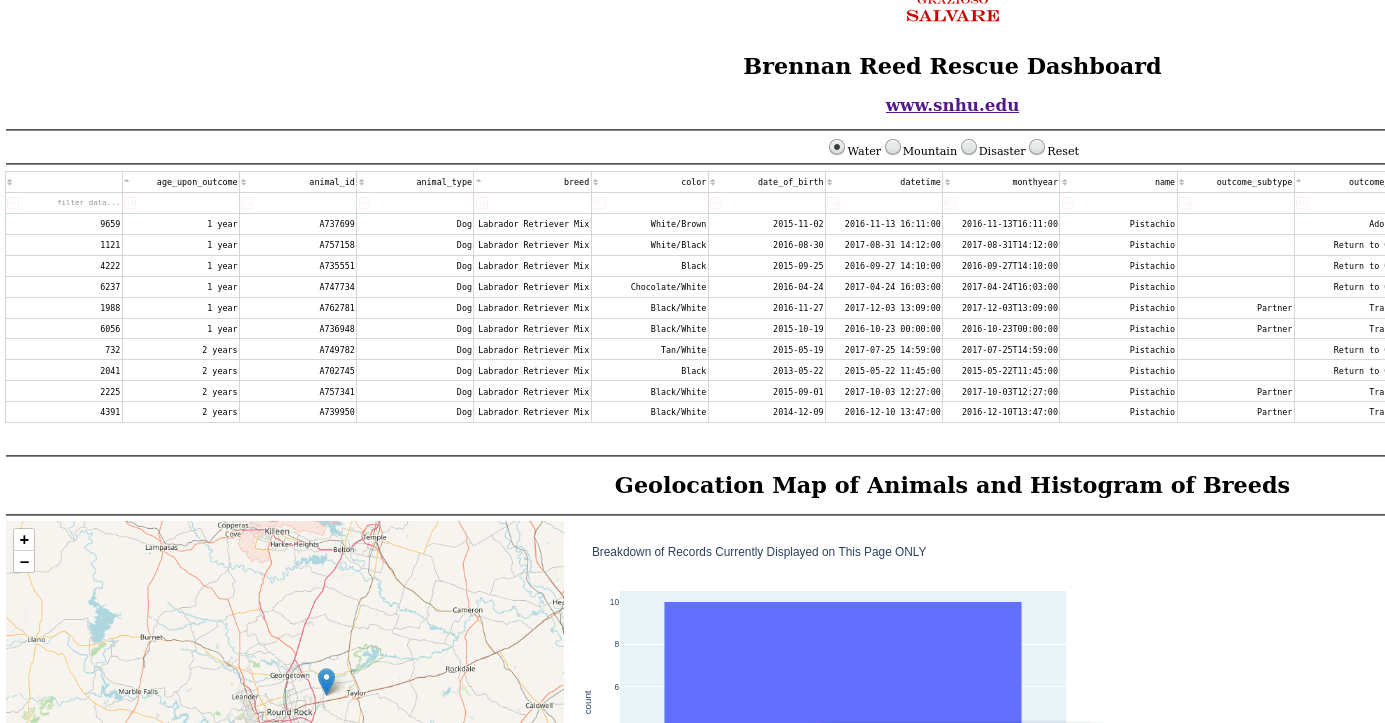
**

**Disaster** preset displays animals from the Disaster or Individual Tracking category.

**

### Tests:

Example of multiple filters applied natively (**age\_upon\_outcome** and **outcome\_type**) *and* a preset filter (water) applied at same time for an example of the customizability available

**

## Roadmap/Features (Optional)

*URL coming soon*

*Mobile app coming soon*

*Access Control coming soon*

*Ability to select multiple lines/animals and have the geolocation map and chart update to reflect selection coming soon*

## Contact

Brennan Reed

Brennan.reed@snhu.edu