# CS 340 README

## About the Project: Search & Rescue Dog Dashboard

*The “Search & Rescue Dog Dashboard” is a web application designed for Grazioso Salvare to identify good dog candidates for search and rescue training. The data is compiled from local animal shelters in the Austin, TX area.*

*The web application is built using Python’s Dash framework. This application interacts with the CRUD Python Module, which is the Middle-ware layer for the stacked development. It is the “glue” between the base level and the client level. The base level is the raw data, which is maintained using the MongoDB database management tool.*

## Motivation

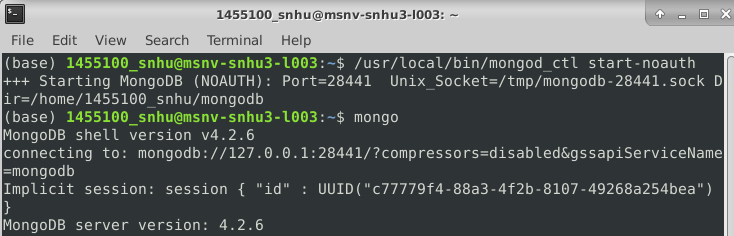
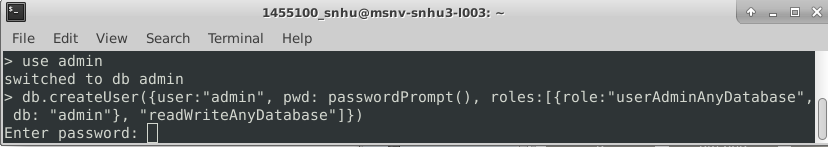
*The motivation for this project comes from our client’s desire to easily identify specific traits among a large set of canine data. These specific traits are categorized into three types of search and rescue candidates: Water rescue; Mountain & Wilderness rescue; Disaster & Individual rescue.*

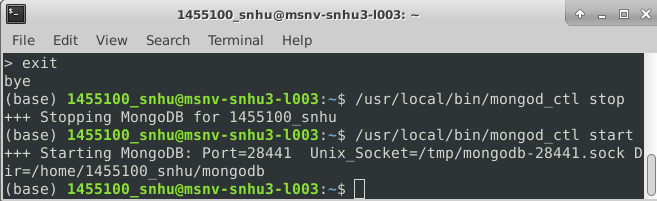
* Why MongoDB? *The desire to proceed with MongoDB, Python, and the Dash framework stems from the natural interconnectivity across these platforms. MongoDB is a free, open-source, NoSQL database tool which supports large scale and rapid development projects. The driver required for MongoDB to communicate with Python is called PyMongo. Additionally, Mongo has a great set of tools that allows for incredibly fast querying through large data sets, such as indexing. It also has great customization for querying by utilizing the aggregation pipeline tool.* 
  + *The MongoDB manual can be found here:*[*https://docs.mongodb.com/manual/introduction/*](https://docs.mongodb.com/manual/introduction/)
* Why Python?   
  *Python is a powerful programming language with a flexible set of tools for various applications. It is particularly well suited for data analysis and communicates fluently with MongoDB. The PyMongo driver allows for a quick and reliable connection to the MongoDB server. This connection is embedded in the CRUD.py module, which also stores the base code for inserting, querying, modifying, and removing documents in the* ***AAC*** *database. The code is reusable and designed to be utilized by client-facing applications for access to the AAC database.*
  + *The Python manual can be found here:*[*https://pymongo.readthedocs.io/en/stable/*](https://pymongo.readthedocs.io/en/stable/)
  + *The PyMongo manual can be found here:*[*https://pymongo.readthedocs.io/en/stable/*](https://pymongo.readthedocs.io/en/stable/)
* Why Dash?  *Dash is the Python framework for designing web applications. Like all the other software mentioned, it is free and open source. Instead of having to implement additional scripting and markup languages such as JavaScript or HTML, the Python Dash framework allows us to continue developing the full-stack using just Python. This allows for seamless communication between all three levels of the stack development: the server-side base level (MongoDB / PyMongo), the middle-ware (Python / PyMongo), and the client-side web application (Python / Dash). Dash utilizes several powerful analysis tools within Python such as the Pandas library and Plotly, but also implements HTML and core components for handling the web design and application layout. The Dash application communications directly with the CRUD.py module for importing the data table and running queries to filter results based on the type of rescue dog. Dash also implements callback functions which accepts the queried data results and sends output data into the graph and map, so as the filters are changed on the data table the visualizations reflect the queried data.*
  + *The Dash manual can be found here:*

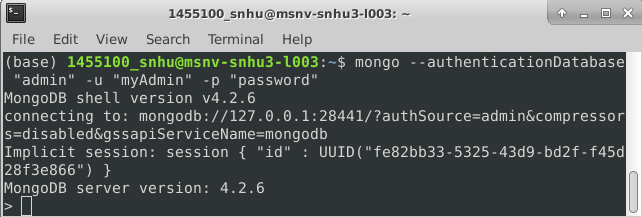
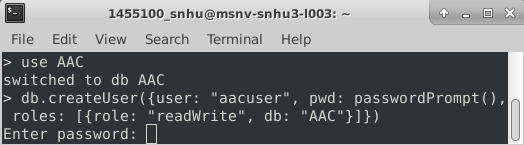
[*https://dash.plotly.com/*](https://dash.plotly.com/)

## Getting Started

*To get a local copy up and running, follow these simple example steps.*

1. *Download / Install the following tools & files (see “Installation” section on page 4):*
   1. *MongoDB*
   2. *Python (with the Pandas library & the PyMongo driver)*
   3. *Python’s Dash framework (with Dash core components, Dash HTML components, & Plotly)*
   4. *CRUD.py & AAC database (available on this project’s Github repository)*
2. *Initiate the MongoDB server in your terminal with* ***no authorization*** *by entering the following:****/usr/local/bin/mongod\_ctl start-noauth****Then start mongo by entering* ***mongo****:*
3. *Enter “****use admin****” to enter the admin database, then proceed to enter the “****db.createUser()****” command with the information from the following screenshot. This will create an admin account, and you will be prompted to enter a password for the account:*
4. *Follow the steps shown in the below screenshot to exit mongo and restart it* ***with authentication***
   1. *Enter:* ***exit***
   2. *Enter:* ***/usr/local/bin/mongod\_ctl stop***
   3. *Enter:* ***/usr/local/bin/mongod\_ctl start***



1. *Start mongo again, but this time use the admin account for authorization:****mongo –authenticationDatabase “admin” -u “username” -p “password”*** *\*\*Replace “username” and “password” with the admin account’s username & password:*
2. *Enter “****use AAC****” to enter the AAC database, then proceed to enter the “****db.createUser()****” command with the information from the following screenshot. This will create a user account, and you will be prompted to enter a password for the account:*
3. *You are now ready to import the AAC Outcomes data set and run the project\_one.py program*

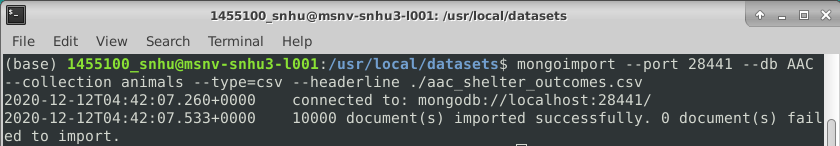
## Installation *(all applications are free)*

* *MongoDB Community Server:*[*https://www.mongodb.com/try/download/community*](https://www.mongodb.com/try/download/community)
* *Python:*[*https://www.python.org/downloads/*](https://www.python.org/downloads/)
* *PyMongo:*[*https://pypi.org/project/pymongo/*](https://pypi.org/project/pymongo/)
* *Pandas:*[*https://pandas.pydata.org/pandas-docs/stable/getting\_started/install.html*](https://pandas.pydata.org/pandas-docs/stable/getting_started/install.html)
* *Dash:*[*https://dash.plotly.com/installation*](https://dash.plotly.com/installation)
* *Plotly:*

[*https://plotly.com/python/getting-started/*](https://plotly.com/python/getting-started/)

* *CRUD.py module:*

[*https://github.com/bkoconnell/CS340-final-project*](https://github.com/bkoconnell/CS340-final-project)

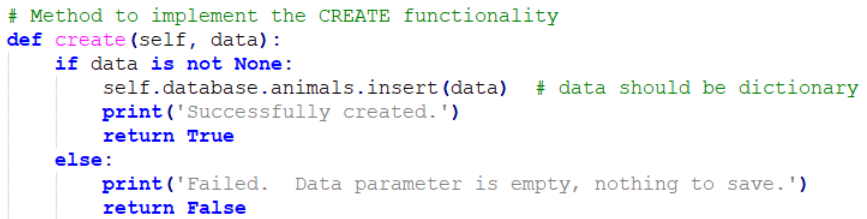
* ***AAC database:*** [*https://github.com/bkoconnell/CS340-final-project*](https://github.com/bkoconnell/CS340-final-project) *After initiating MongoDB in your terminal (see “Getting Started” section above), upload the Austin Animal Center Outcomes data set into MongoDB using the MongoDB csv import tool:*

## Usage (CRUD.py)

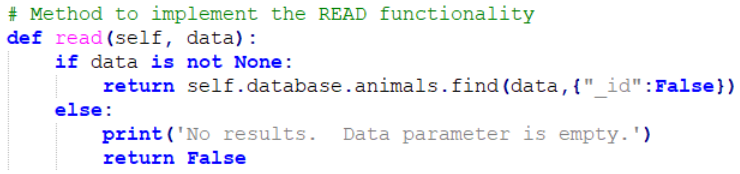
*The main purpose for using the CRUD.py module is for client/web applications to access and query the database on the server side, for creating, reading, updating, and deleting documents. Here are some examples of how the module is setup.*

### Code Example (CRUD.py)

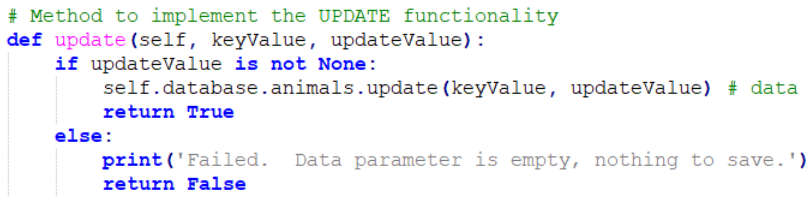
*Below is an example of the* ***create*** *function from CRUD.py. This function allows the client to create a new document in the AAC database and will confirm whether the creation is successful:*

**

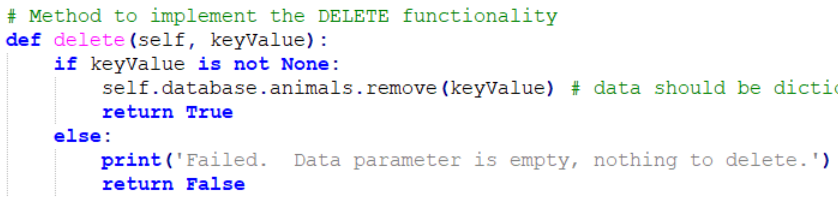
*The* ***read*** *function allows the user to query existing documents in the AAC database by comparing the user’s input data against the documents in the database, then returning the results to the client. Here is the method that implements the* ***read*** *function:*

**

*The* ***update*** *function can be utilized by the client-side application to locate an existing document and update specified data. Two arguments are required for this function. The first argument is a key/value pair that is used to locate the document in the database, and the second argument is the specific data that is to be updated. Here is an example of the code:*

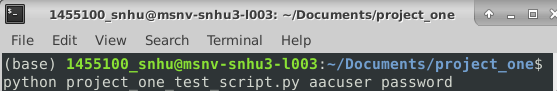
**

*The* ***delete*** *function removes documents by locating a specified document and deleting it from the database. The only argument passed to this function is a key/value pair used to identify the specified document. The code is as follows:*

**

### Tests (CRUD.py)

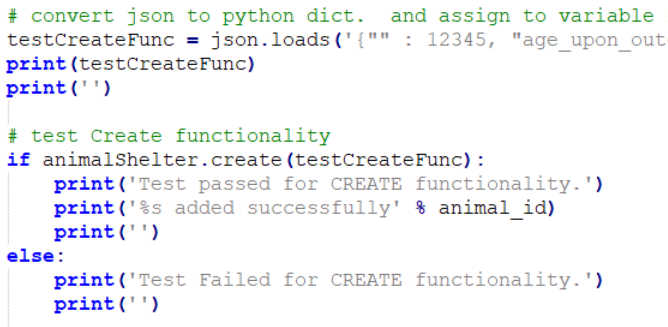
*Several sample tests have been established in the script called “project\_one\_test\_script.py”. To run these test samples, download the test script and then open your terminal and enter the following:* ***python project\_one\_test\_script.py username password****(Note: \*\*the arguments “username” and “password” should be replaced by the credentials for the AAC user account\*\*)*



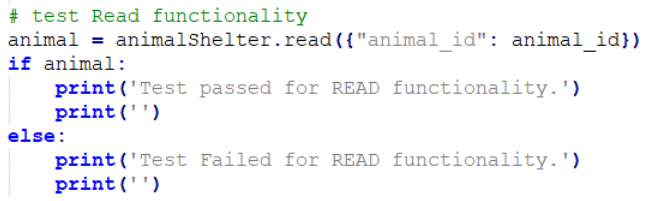
### Screenshots

*The command above will execute the test script, including a secure connection to the MongoDB database as well as unit tests for the* ***create****,* ***read****,* ***update****, and* ***delete*** *functionality. Below are several screenshots of the test script’s code for the* ***create****,* ***read****,* ***update****, and* ***delete*** *testing, as well as the results in the terminal once the script is executed.*

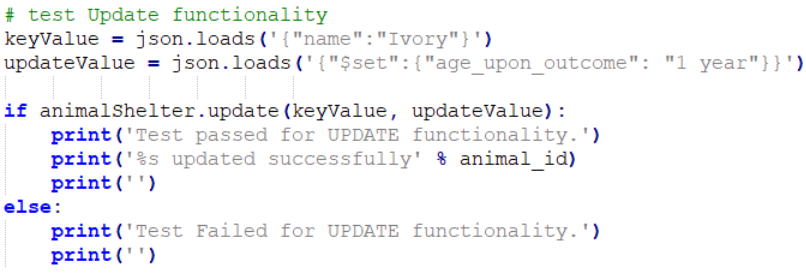
*For the* ***create*** *test, a set of document data is written as json and converted to a python dictionary then assigned to a test variable. A conditional statement then tests if the function returns True or False, determining whether the function executed properly:*

**

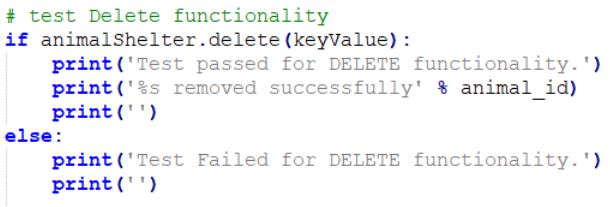
*For the* ***read*** *function test, a key/value pair is used as the argument to locate the document that was created from the previous test.*

**

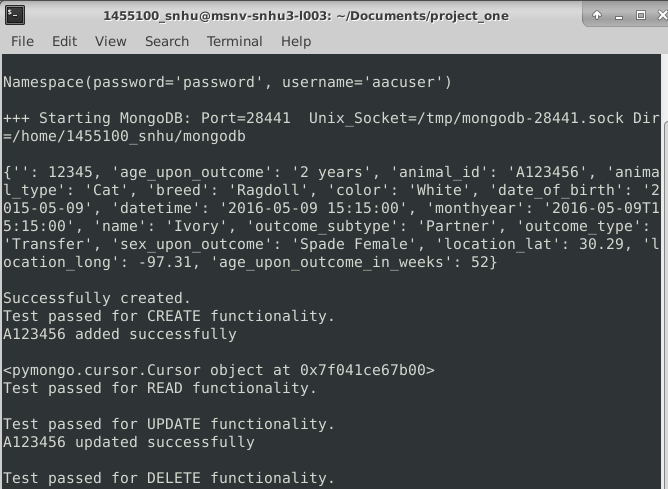
*The* ***update*** *test sends two variables as arguments to the function. The first argument is a specific key/value pair for the function to use to query the database and locate the corresponding document. The second argument is the data being updated in the selected document:*

**

*The* ***delete*** *function test uses a key/value pair as the only argument, and it is used to locate a specific document which will be delete once it is found:*

**

*Running the project\_one\_test\_script.py in your terminal will yield the following, confirming successful execution of the* ***create****,* ***read****,* ***update****, and* ***delete*** *functions:*

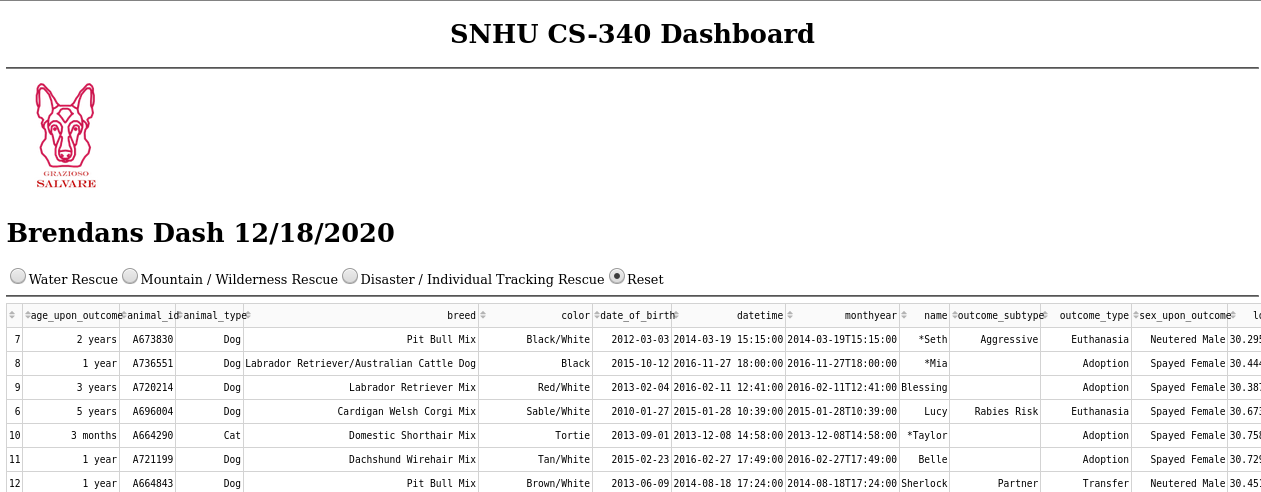
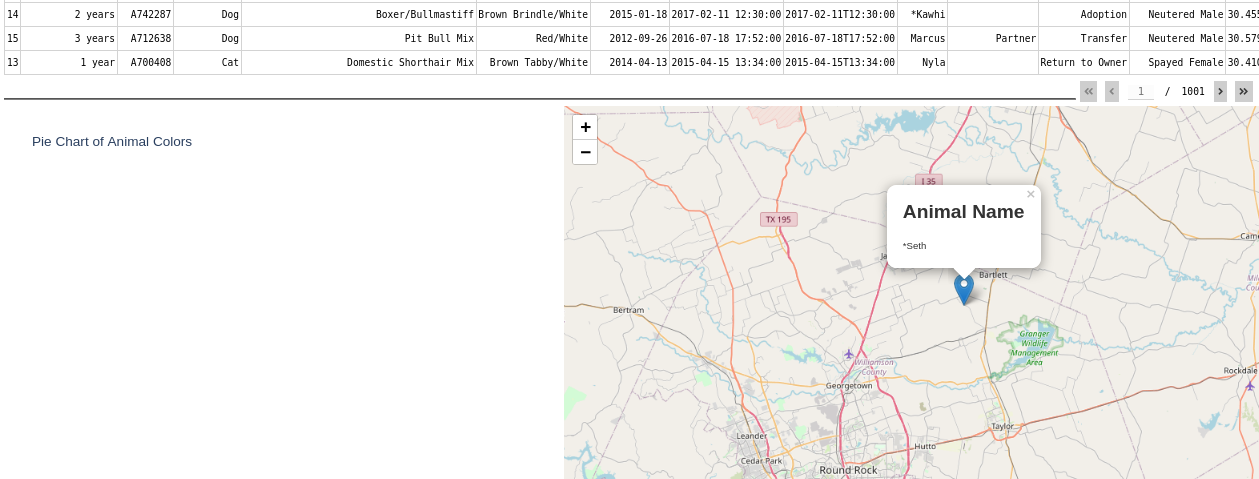


## Functionality (ProjectTwoDashboard.ipyng)

*With the database setup and the CRUD.py module fully functional, the client-side dashboard can now be utilized for purposes of filtering through the animal shelter data and identifying candidates for rescue dog training. This web application is designed with a user-friendly layout specific to the needs of Grazioso Salvare. Below are examples of the dashboard’s functionality including interactive options for filtering data, an interactive data table, and charts for visual data presentation.*

### The Dashboard

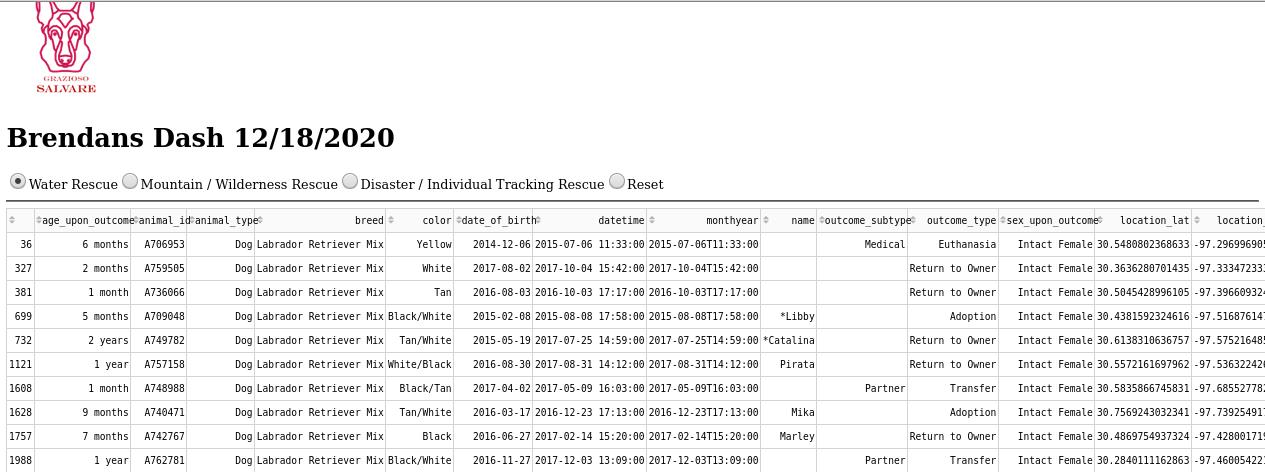
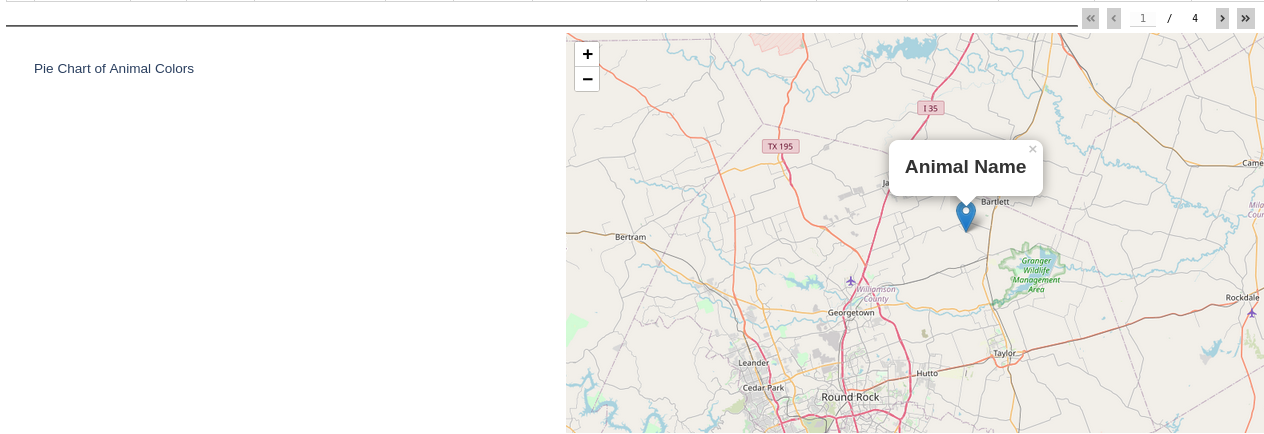
*Here is the initial state of the web application upon loading the dashboard display:*

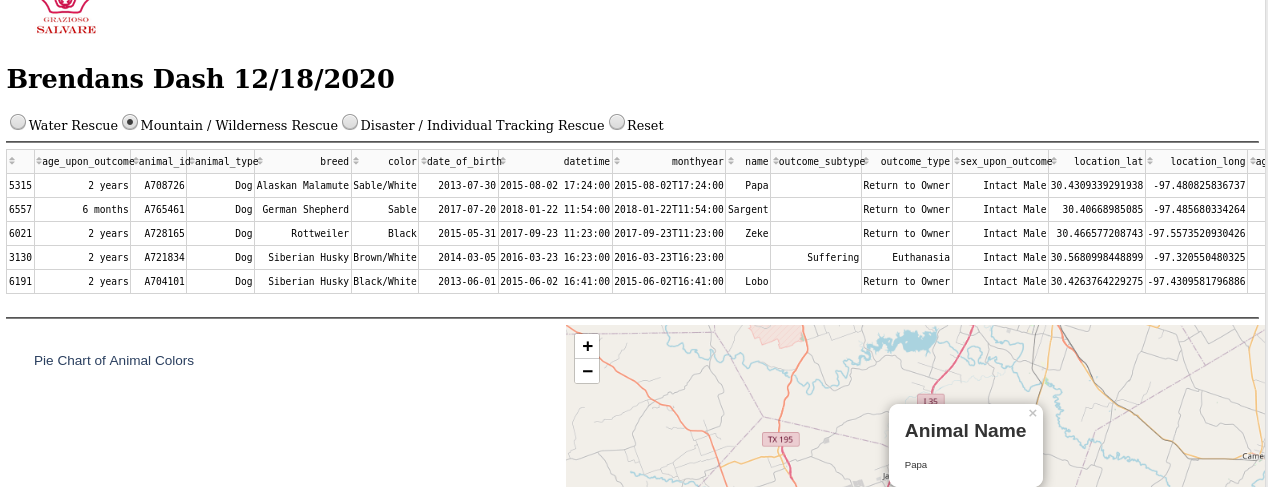
### Interactive Data Filters

*To query the database for ideal canine candidates, these radio buttons offer a quick and simple approach to filtering the data table. User’s can choose between any of the three rescue types or they can select “reset” to remove all filters. Below are several screenshots that show the usage of each radio button. Take note of how the visual display of the charts reflect the data filtering, such as the “Animal Name” tag on the geolocation map:*

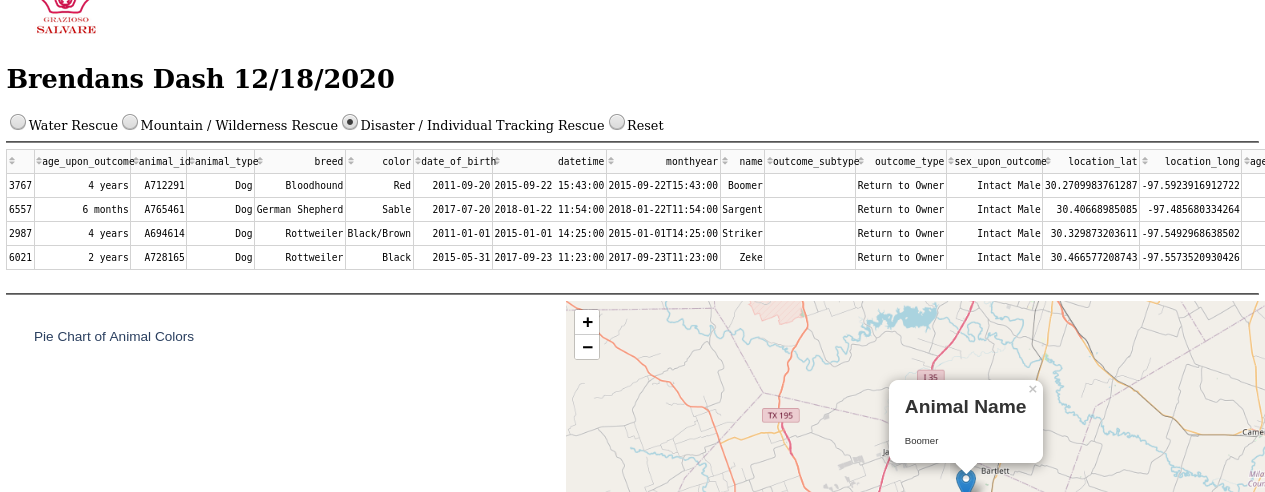
* *Water Rescue:*

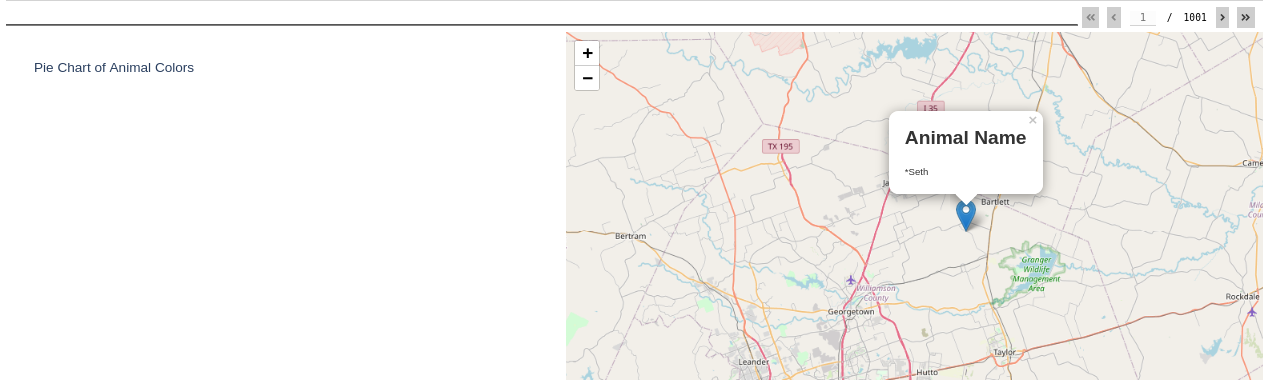
* *Mountain & Wilderness Rescue:*



* *Disaster & Individual Rescue:*



* *Reset (returns widgets to their original, unfiltered state):*

## Roadmap/Additional Features

*Although the web application is ready for use, additional revisions are required for full functionality of the pie chart below the data table. This is expected to be completed by the next released version.*

*I am always looking to improve the functionality of this application. If you have any comments or would like to contribute ideas, please feel free to contact me at the e-mail listed in the “Contact” section below.*

## Contact

Brendan O’Connell

[brendan.oconnell2@snhu.edu](mailto:brendan.oconnell2@snhu.edu)