# CS 340 Project 2 README

## About the Project

This project is a dashboard through which you can view the AAC (Austin Animal Center) database, and filter its data.

## Motivation

This project was created for Grazioso Salvare, an international animal rescue organization. They wanted an application that would interface with shelter records to help them find good candidates for rescue dogs. This application is meant to provide an easy, user friendly interface that can be used to find suitable candidates quickly.

## Getting Started

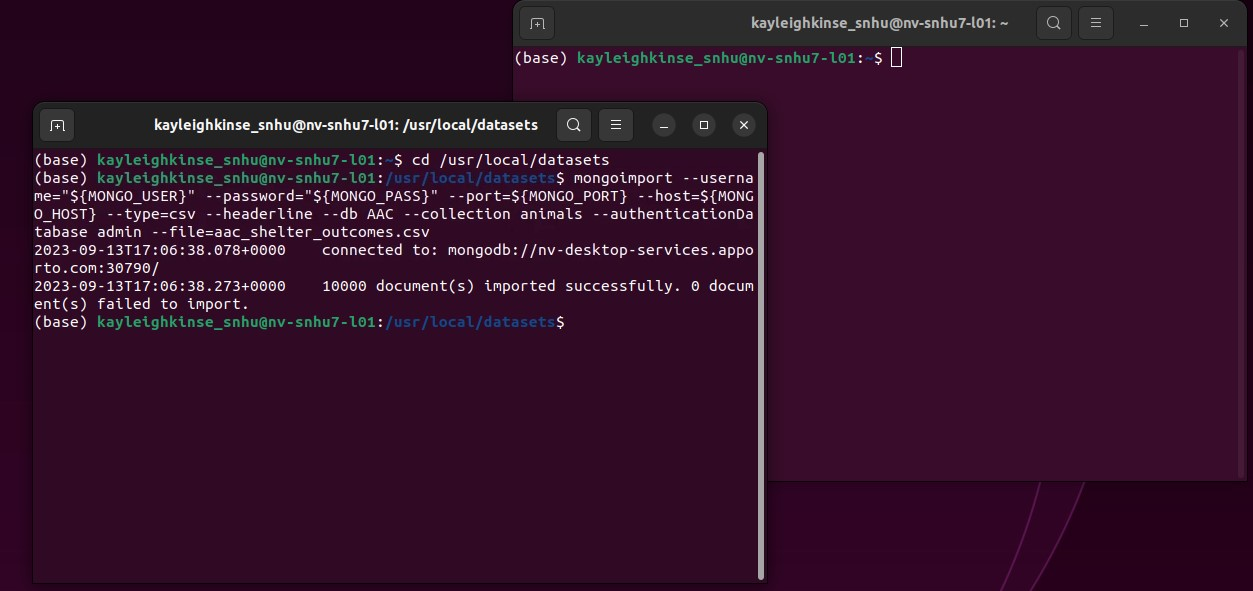
To get started, you will need the aac\_shelter\_outcomes.csv file.

To ensure your device can connect with the database, I recommend importing it to the terminal before you try it through a script.

1. Open up your terminal and navigate to aac\_shelter\_outcomes.csv’s directory.
2. Use the following mongoImport command to import the database and log in as a user:

mongoimport --username="aacuser" --password="SNHU1234" --port= --30790 host=nv-desktop-services.apporto.com --type=csv --headerline --db AAC --collection animals --authenticationDatabase admin --file=aac\_shelter\_outcomes.csv

If the documents import successfully, you’re all set to use the dashboard!

Here I’m importing the AAC Database. The database’s directory may be different on your device.

When you log in to the database, you will do so as aacuser, with the password “SNHU1234”. aacuser has read/write permissions across the database. You will need to provide the username and password whenever you access AAC DB. As of now, AAC User is the only user role available other than admin, and it’s the only one that you should need to interface with the database.

Once you’re logged in, you can interface with the database through the terminal after calling the mongosh command. Be sure to look over the Mongosh documentation if you’re interested in learning more: <https://www.mongodb.com/docs/manual/reference/method/>.

## Installation

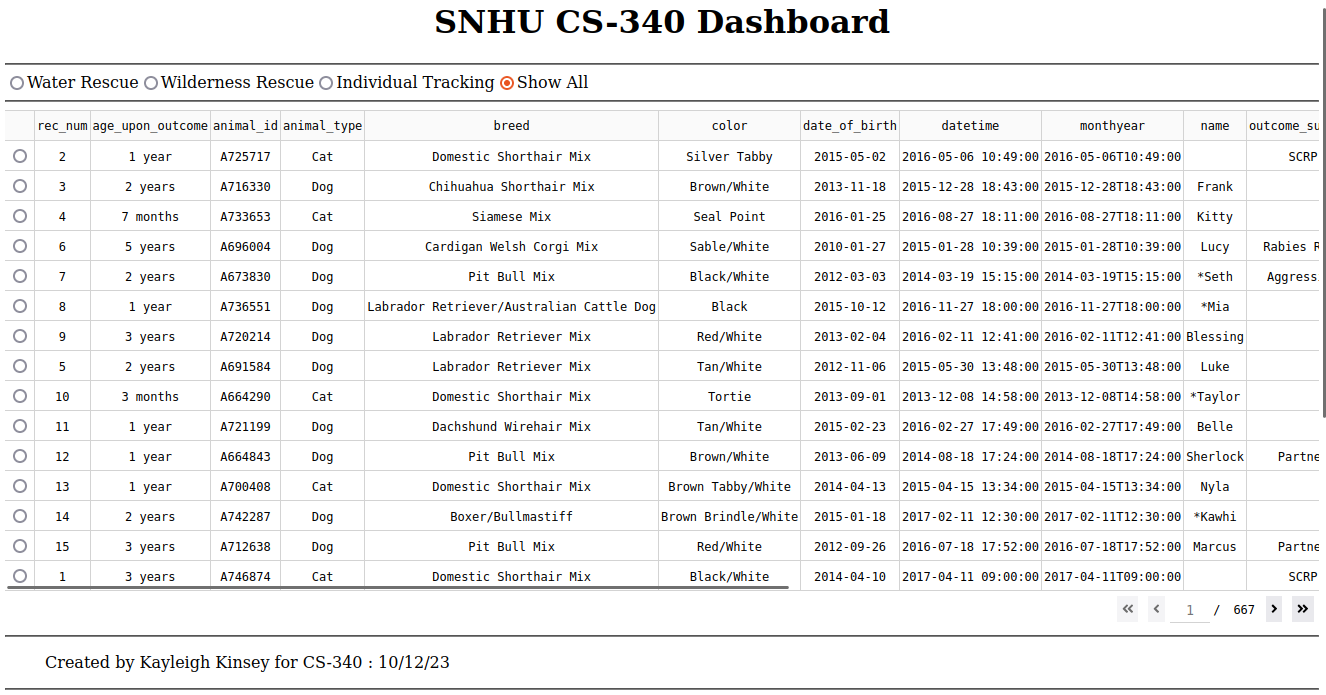
To use the dashboard, you will need an application that can read ipynb files, such as Jupyter Notebook or Google Colab. Both are free to use and easy to set up. Many IDEs such as Visual Studio can also read and edit ipynb files.

- Here is the installation guide for Jupyter Notebook: <https://jupyter.org/install>. You will need pip or another Python installation tool.

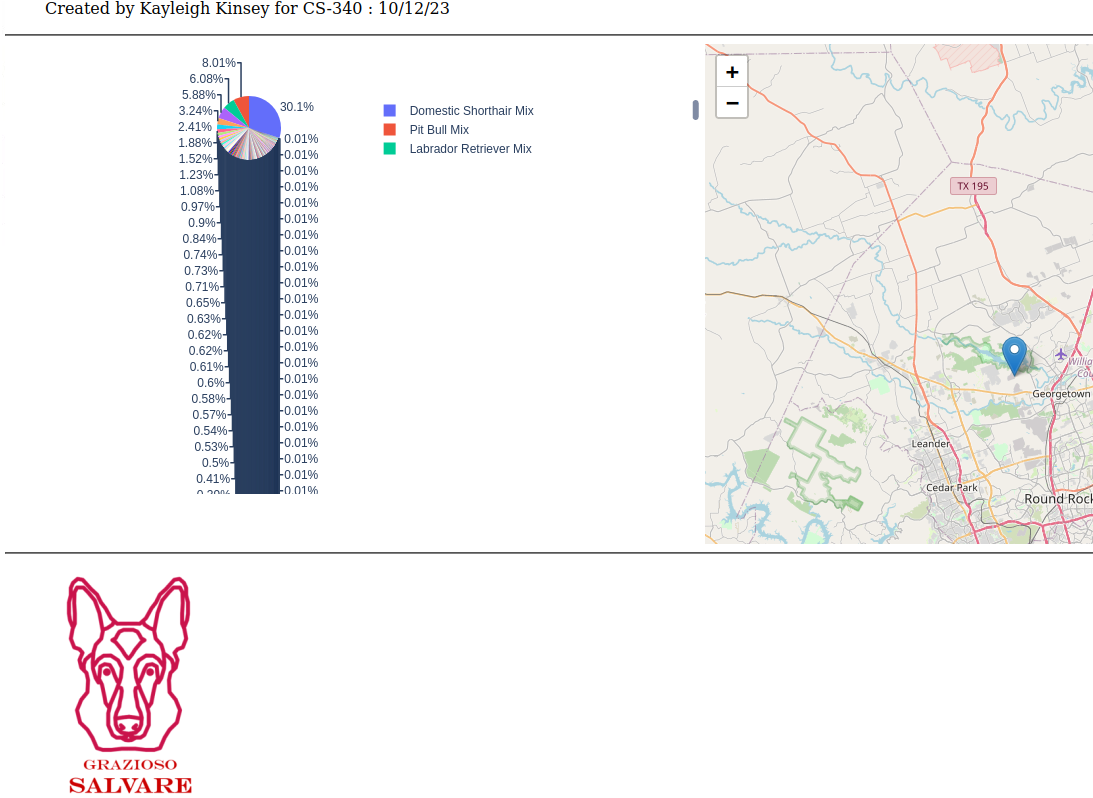
- Google Colab can be accessed here: <https://colab.google/>. You only need a Google account to get started.

## Usage

Once the application is fully loaded, you should be met with a table similar to the following:



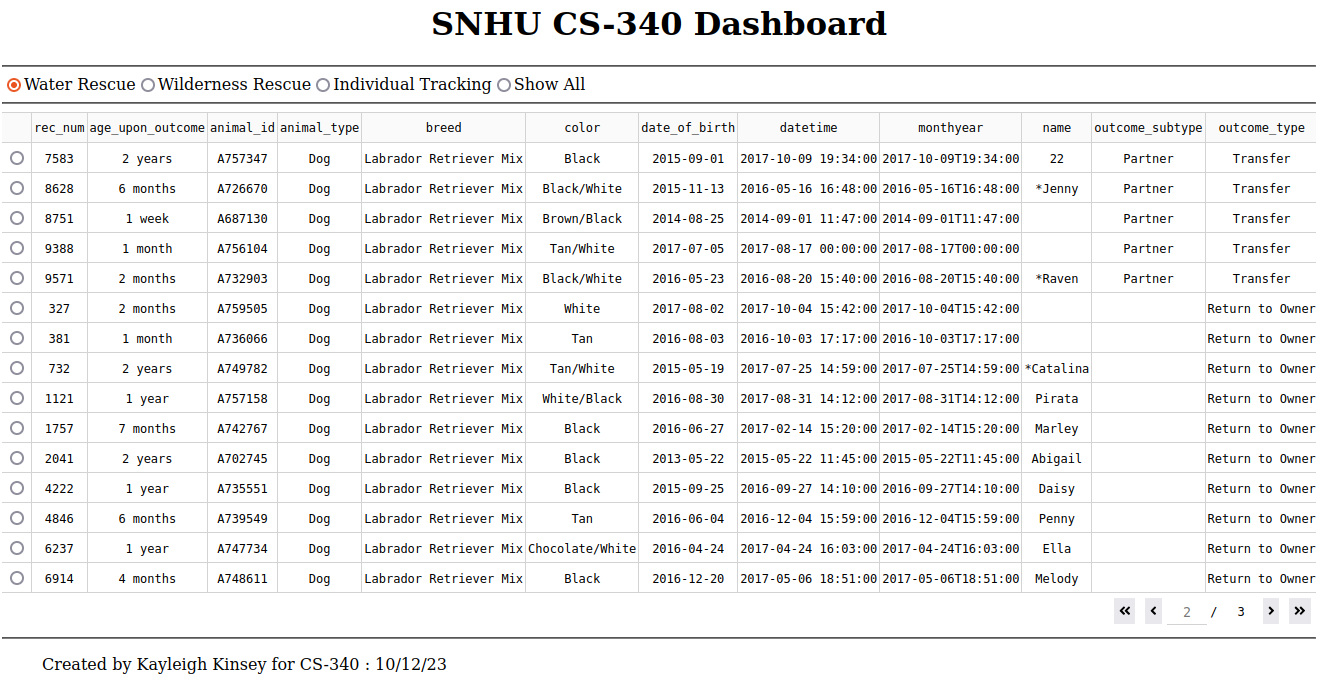
If you scroll down, you should see a pie chart, a map, and the Grazioso Salvare logo.



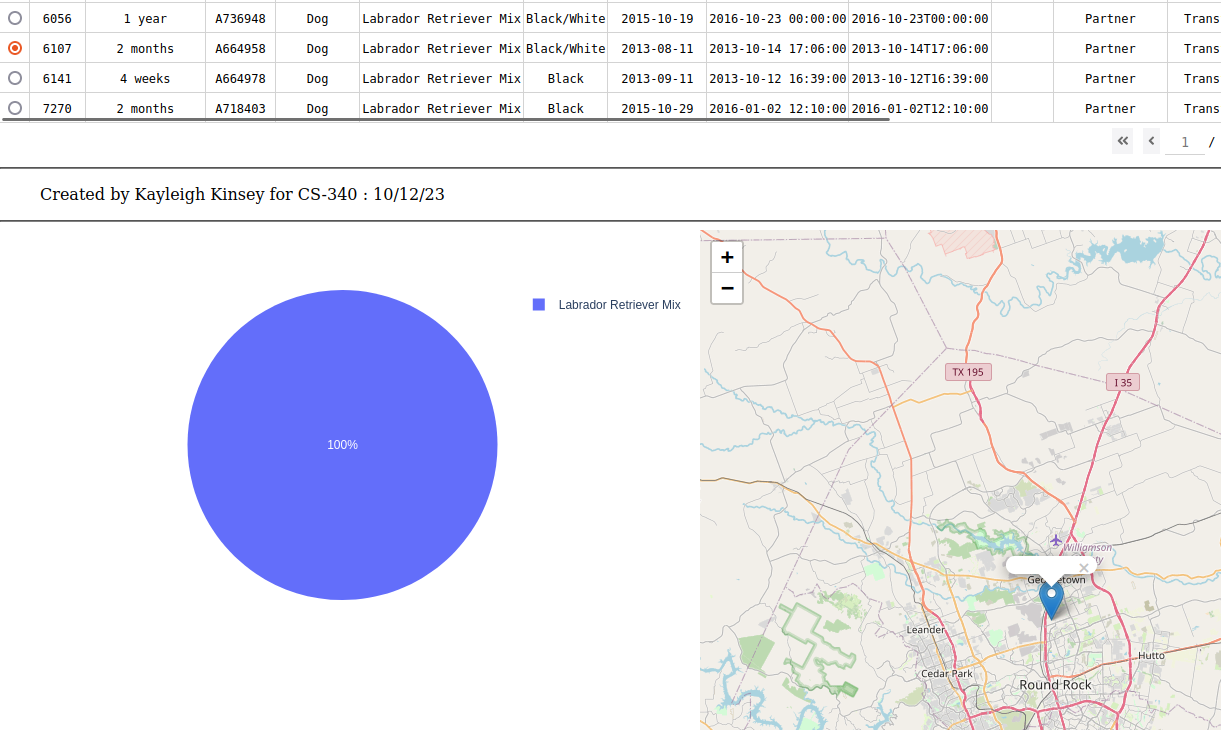
The pie chart shows the distribution of breeds in the current set of animals. Since the table is unfiltered and there is a huge variety of breeds in the database, the pie chart is difficult to read. Once you’ve applied a filter, the chart will reload and be much more comprehensible.

The map will show the location of a selected animal, which should always be around Austin, Texas (unsurprisingly).

To filter the table and get usable data, just select one of the preconfigured filter types on the top of the table.

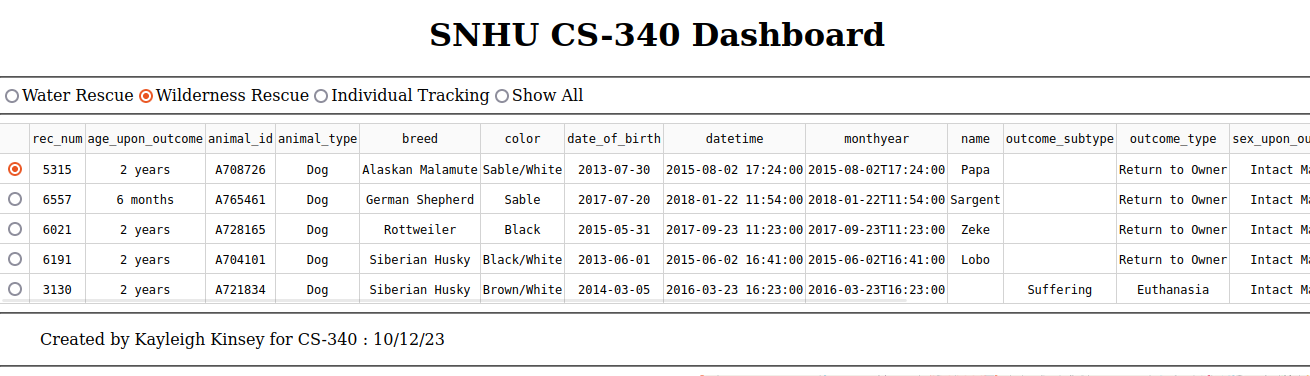


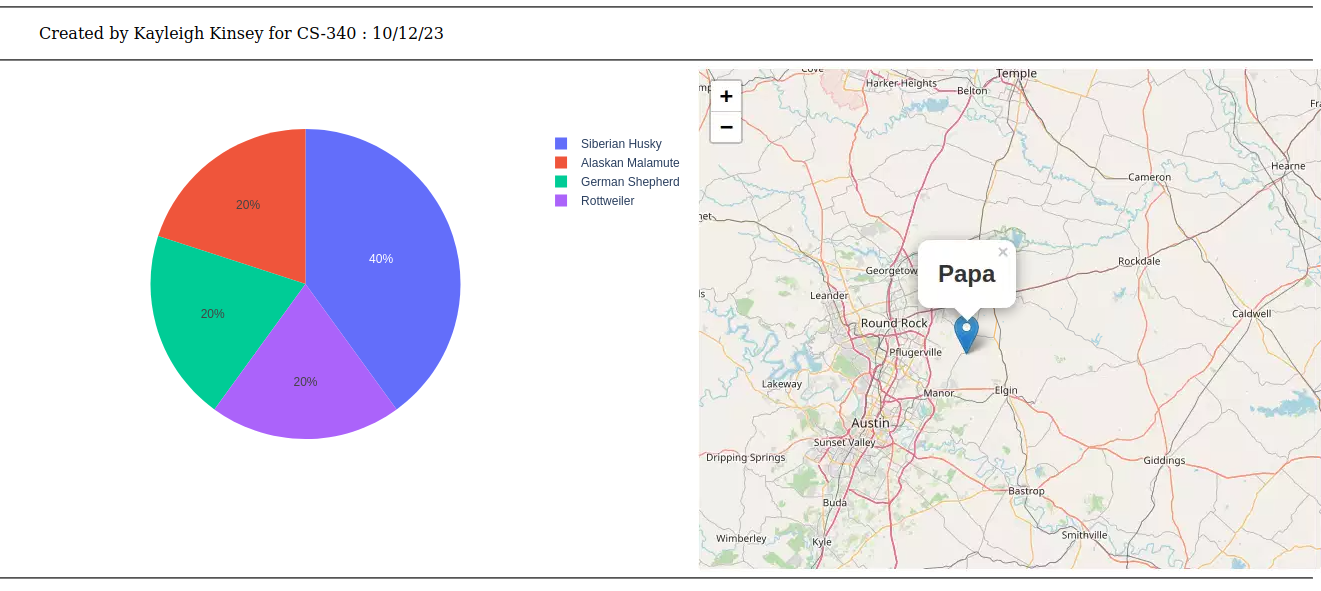
Water Rescue yields three pages of results, which, you can see below, are all Labrador Retriever Mixes.



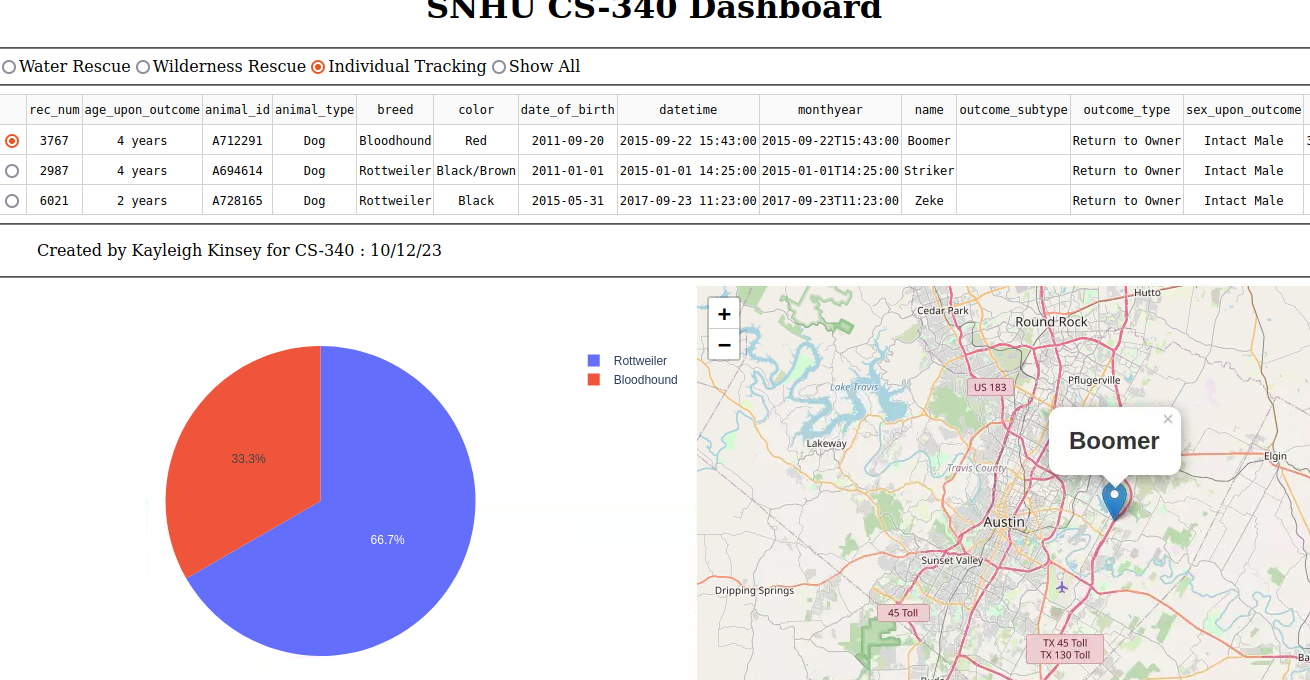
These filters were created to follow Grazioso Salvare guidelines, sifting through the database for the breeds, sex, and age range that Grazioso Salvare has identified as ideal for each type of rescue.

Below, I have filtered the data for dogs well-suited to wilderness rescue.

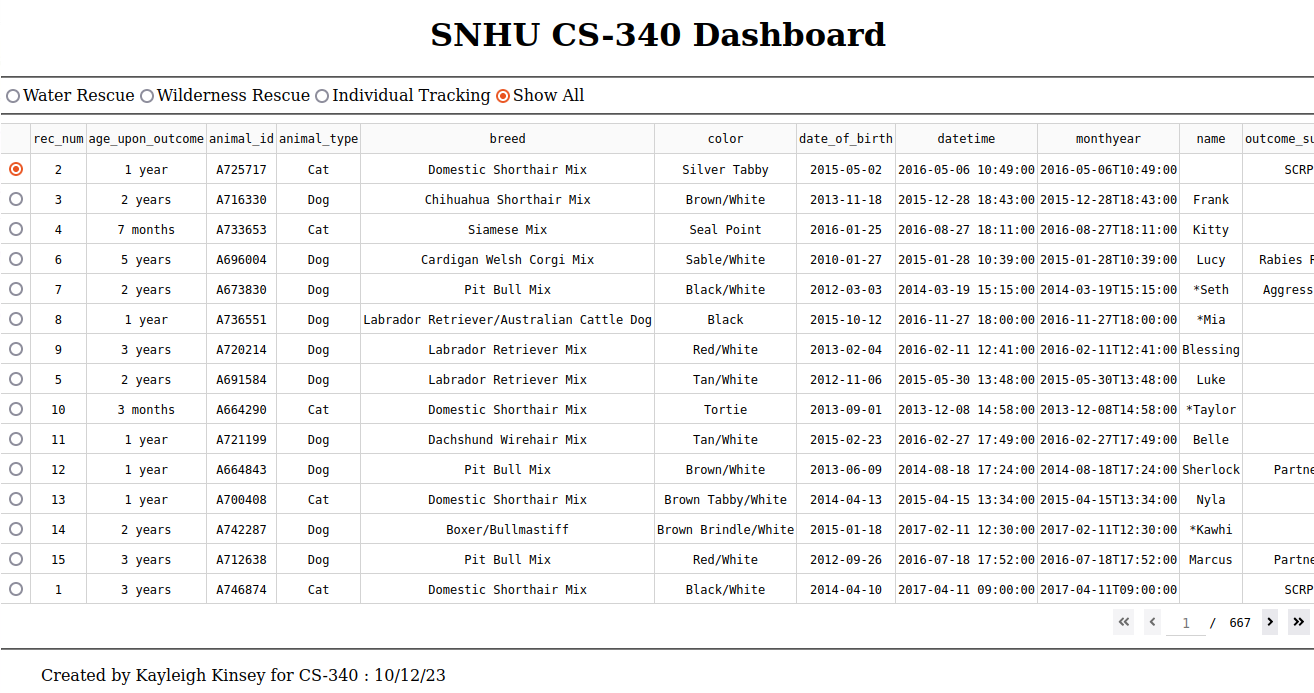




The final filter is Individial Tracking.



Should you ever want to return to the unfiltered dataset, just select the “Show All” filter and the table will be reset to its initial state. (Note that it may load for awhile when you select this option, as there is a lot of data to process.)



**Tools Used**

This application was built on MongoDB, a prominent database framework. MongoDB is extremely flexible, allowing for dynamic database creation, updating, and querying. Because this application will likely be subject to improvements and adjustments as Grazioso Salvare’s needs grow, I believe MongoDB is the best choice. This application’s queries can be refined further for more accurate results. We could even add new features to the applications, such as updating the table with new information, or saving notes on existing data. MongoDB can handle all of these changes and more without the need for me, or anyone else, to change my foundational code.

MongoDB has an official Python driver, and it takes relatively little effort to implement. I was able to create a simple Python module to interface with the AAC database, which can be used easily to conduct basic MongoDB CRUD operations. (See the CRUDModule README for more information.) This should be sufficient for most of Grazioso Salvare’s needs, but if you want to do anything more complicated, you will need to interact with MongoDB directly. Begin by following my Getting Started instructions above, and consult <https://www.mongodb.com/docs/> for more information on how to use mongo.

The application itself was created with the Dash framework. This is a Python package useful in creating charts, graphs, and other visual representations of data for html applications. It is fairly easy to use and produces visible results very quickly, with minimal code size.

For more information on any of the tools I used, consult the following resources:

[MongoDB](https://www.mongodb.com/docs/) (database framework)

[Plotly Dash](https://dash.plotly.com/) (data visualization framework)

[Pandas](https://pandas.pydata.org/docs/) (a dataframe library)

And, of course, the [Python official documentation](https://www.python.org/doc/) is always a good reference to have on-hand.

**Steps Taken**

Here are the steps that I took to complete this project, and the challenges I faced along the way:

1. First, I set up the code to import my CRUDModule and any other packages I would need.
2. Next, I copied code for my datatable and map from my last project to the new application.
3. The first thing I added to my application were the filter buttons, which I originally made html buttons. I didn’t think they looked nice, though, and felt that radio buttons would be better for filter selection. They continue to display the user’s selection until a new one is made, which is helpful for users who may forget what filter is currently in use.
4. Once the visuals were done, I created the filter queries and callback function, which was a huge challenge for me. I just didn’t understand the way that Dash (specifically its callback functions) worked, and spent hours pouring over documentation before I finally comprehended it all.
5. Next, I worked on adding a graph to my application. I started out trying to make a bar graph, and struggled with it for a long time, particularly in how to format the data I gave to the graph. I got a very clear error: “int is not iterable”, but I was iterating over a dictionary. I made test after test that gave me no insight, until at last I realized that I was using an invalid method to append an int to a list, and THAT was the source of the problem, not my loop itself.   
   When my bar graph was complete, it was unreadable because of all the data in the database, so I decided to go with a more compact pie chart instead.
6. Next, I added the Gravioso Salvare logo to my dashboard. I had difficulty accessing it at first, because in the early stages of the project, I had changed the working directory to the location of my CRUDModule, and never changed it back to the project’s local folder.  
   The logo was way too big at first, so I resized the image file.
7. Then, I added my unique identifier, with my name and the date.
8. I spent awhile shuffling the logo and identifier around, and decided that my identifier should go in the middle of the page, where it would be easy to include in every screenshot, and that the logo should go on the bottom of the screen, where it wouldn’t push around any of my widgets.
9. Finally, I made finishing touches and cleaned up some remaining errors.

This project is one of the most challenging I’ve had in awhile. I hit wall after wall, and every step was so much more difficult than I had anticipated. I like challenges though, and feel like I’ve learned a lot more about Dash than I would have if everything went the way I thought it would.