# Jarod Timms

# 6/18/2022

# Professor Jeff H. Sanford, M.S.

# CS 340

# CS 340 Week Four Assignment

## About the Dashboard

This is an interactive client designed to retrieve data from a backend database for Grazioso Salvare, which assists with the retrieval, location, and analysis of animal information. The data is stored in a background service, running MongoDB, while the dashboard handles all the user interaction.

## Motivation

The goal of the dashboard is to display information with filtering options requested by Grazioso Salvare, which includes the ability to search dog breeds relating to three rescue occupations as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Rescue Type | Preferred Breeds | Preferred Sex | Training Age |
| Water | Labrador Retriever Mix,  Chesapeake Bay Retriever,  Newfoundland | Intact Female | 26 weeks to 156  weeks |
| Mountain or Wilderness | German Shepherd, Alaskan  Malamute, Old English  Sheepdog, Siberian Husky,  Rottweiler | Intact Male | 26 weeks to 156  weeks |
| Disaster or Individual Tracking | Doberman Pinscher, German  Shepherd, Golden Retriever,  Bloodhound, Rottweiler | Intact Male | 20 weeks to 300  weeks |

To understand the options available for these fields, the dashboard will be able to provide a pie chart which displays a comparison of dog breeds which are available for each rescue occupation. Additionally, there will be a map provided that will display active location of those dogs.

## Installation and Software

Packaged tools –

For easier implementation, I suggest downloading Anaconda Navigator, used to easily download python IDEs and packages, including PyMongo listed below, from [here](https://docs.anaconda.com/anaconda/navigator/install/). This makes the process much easier as it contains a catalog of libraries you can add to your environment, such as PyMongo.

MongoDB –

This is the database that you will be interfacing with your client. MongoDB has libraries, such as PyMongo, which can be used in Python to make calls to the backend and pull data, which can be used to display elements on the dashboard. You can download the MongoDB that is appropriate for your operating system from the download page [here](https://www.mongodb.com/try/download/community).

*If you choose not to install Anaconda.*

Python –

Python is the coding language the program runs in. You will need the latest installation of Python for the client, which can be acquired from the downloads page [here](https://www.python.org/downloads/).

PyMongo –

You will need to ensure that PyMongo is installed with your version of Python. This library allows the client to connect to the MongoDB background service. This can be found [here](https://pymongo.readthedocs.io/en/stable/), or more easily downloaded in the “pip” with the command pip install pymongo.

Pandas –

Pandas is used for creating data frames from MongoDB documents returned and can be installed from their [website](https://pandas.pydata.org/).

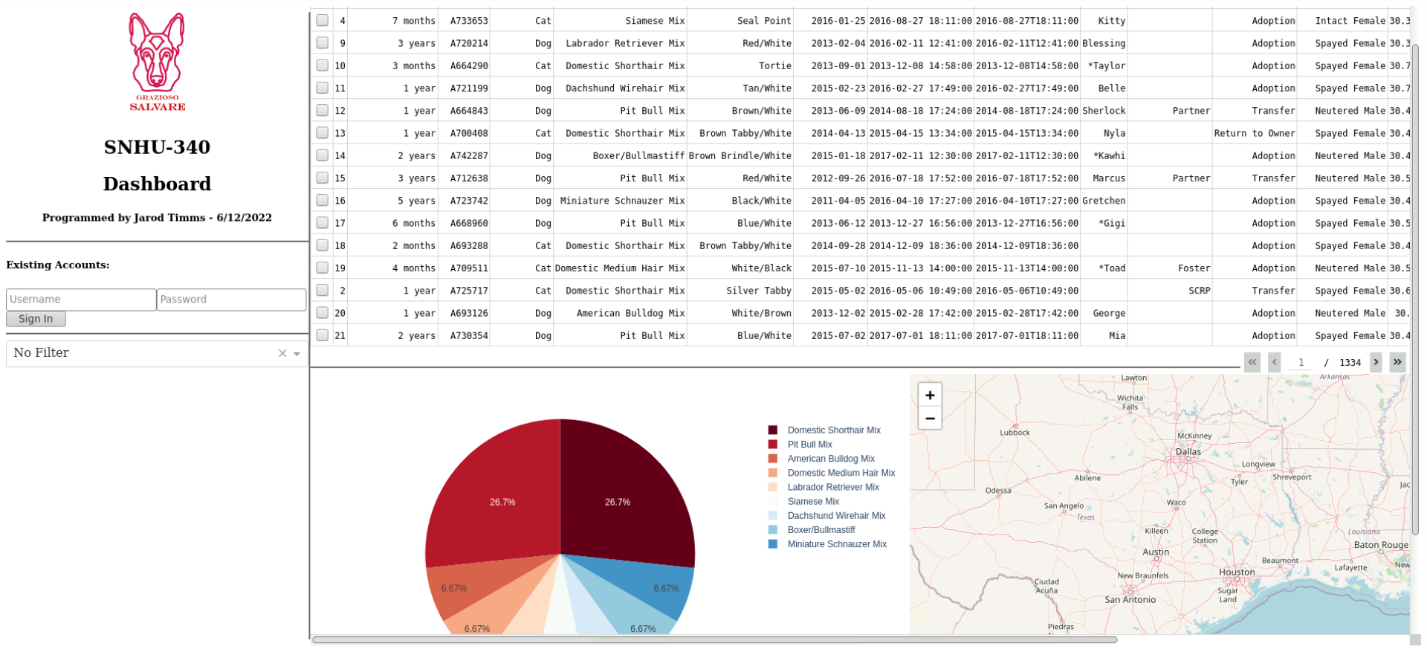
Jupyter Ploty Dash –

Dash builds the visual elements of the dashboard and can use functions with HTML components, such as the dropdown filter, map, data table, and graph in this project. This library can be downloaded from [here](https://pypi.org/project/jupyter-plotly-dash/).

Jupyter Notebooks –

This is the IDE that the main program runs in. It can be installed from the makers’ website [here](https://jupyter.org/install).

## Building the Dashboard



When building the dashboard, I decided I wanted to add a static toolbar on the left-hand side of the screen that displays the company logo and all the filtering options currently available while providing a place for more tools to be added later if desired. The viewport on the right can be navigated and contains all the visual outputs.

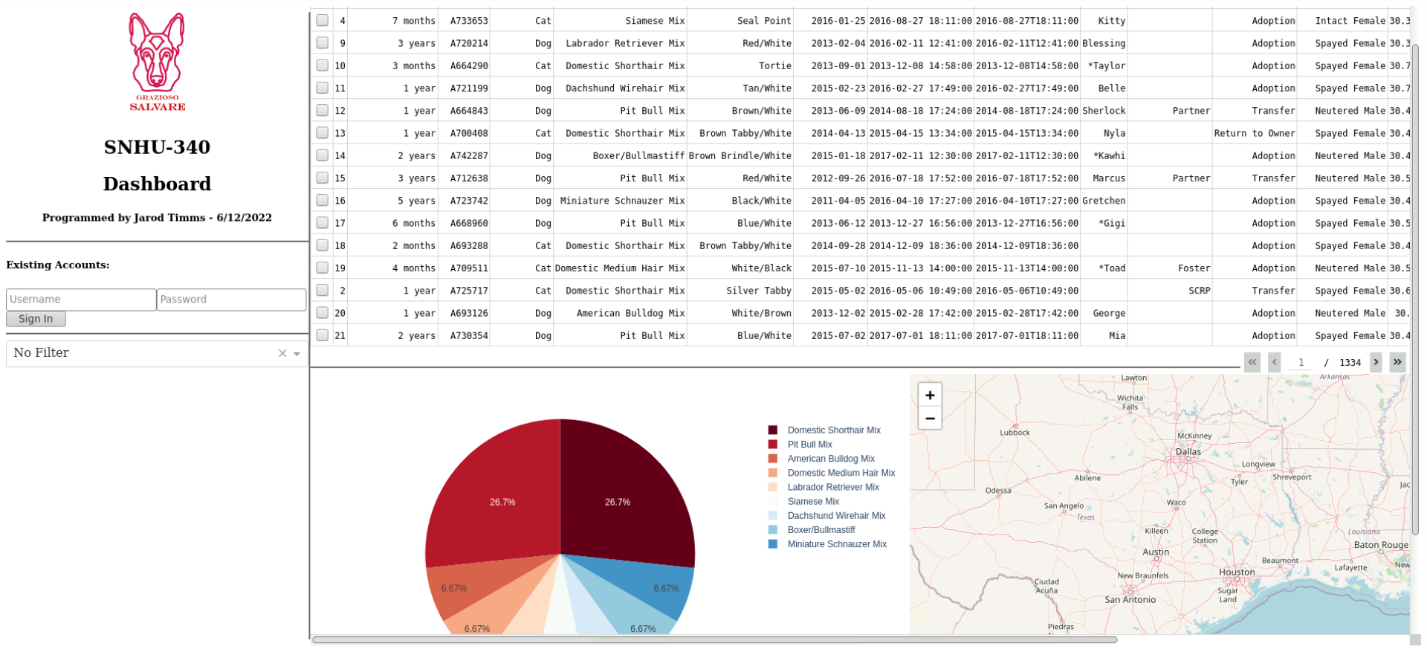
To accomplish this, I divided the layout variable in Dash to be composed of a major Div zone which orients child elements horizontally. Normally, elements are placed vertically, which is how the two child Div elements, that contain the toolbar and the viewport, are laid out. However, the root Div that contains both is limited to a screen height which contains it to the user’s browser window, forcing the toolbar to remain static. The viewport is allowed to have its own scrolling feature to handle whatever overflow occurs from the height constraint.

App callbacks are put in place to update the data table, map, and graph elements in the layout by calling functions which filter and update the data frame, which contains the query passed through MongoDB, or pull column and row data from the data table to represent it visually in the pie graph.

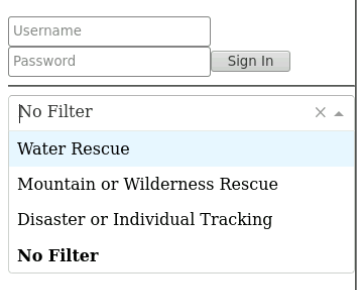
The map callback checks with the latitude and longitude columns of the data table as well as the number of rows which are selected. The map can support up to five animal locations.

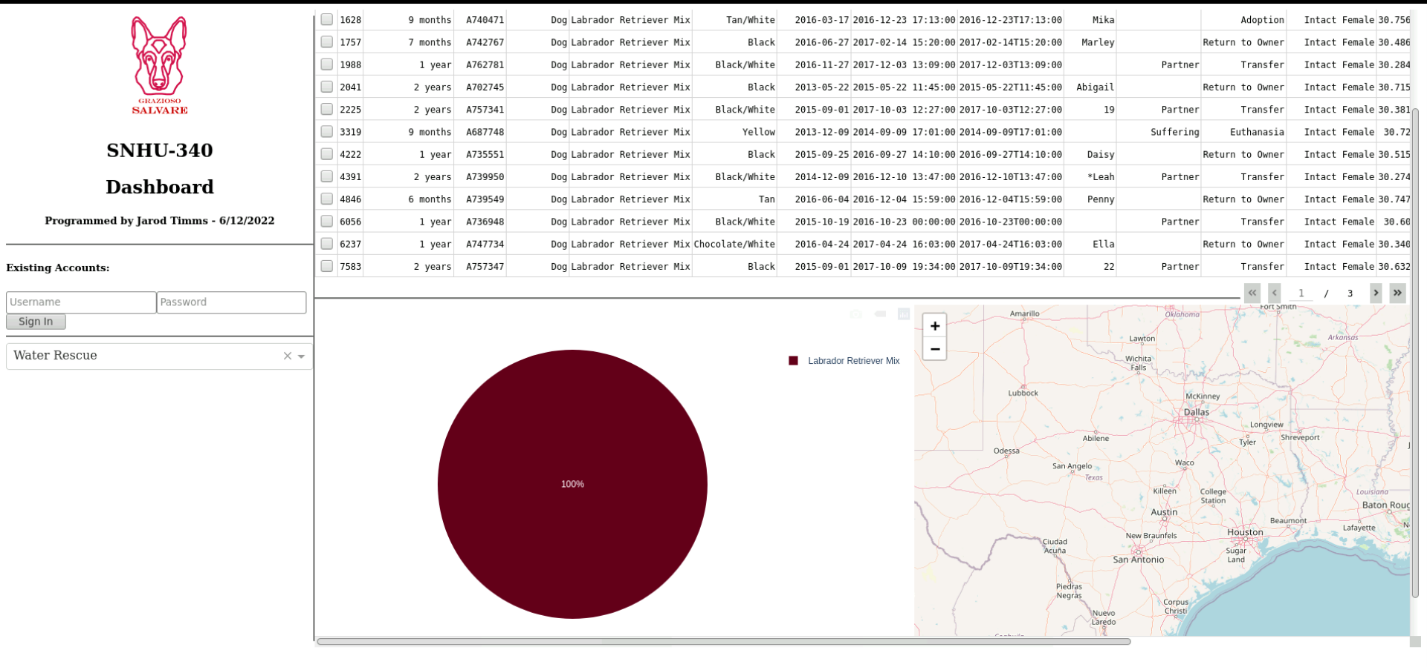
## Dashboard Usage

### Usage of the dashboard is simple in its current iteration. When you start the dashboard, the table in the viewport is unfiltered by default.



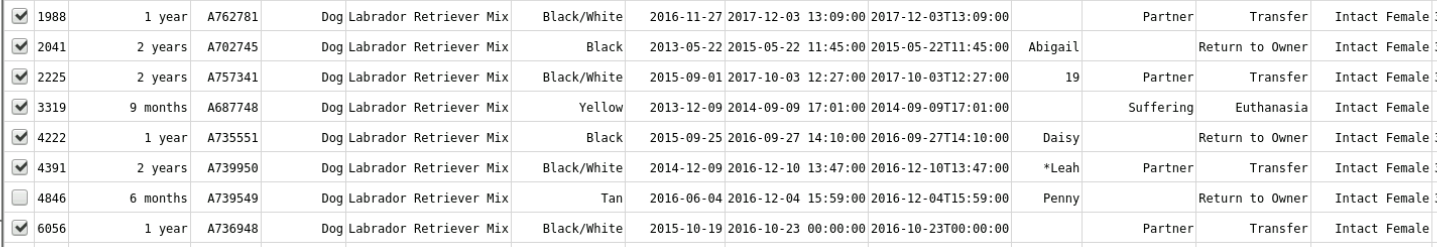
By selecting the drop down in the toolbar, you can choose from the list of available professions, or type out a profession and select the option that is returned.

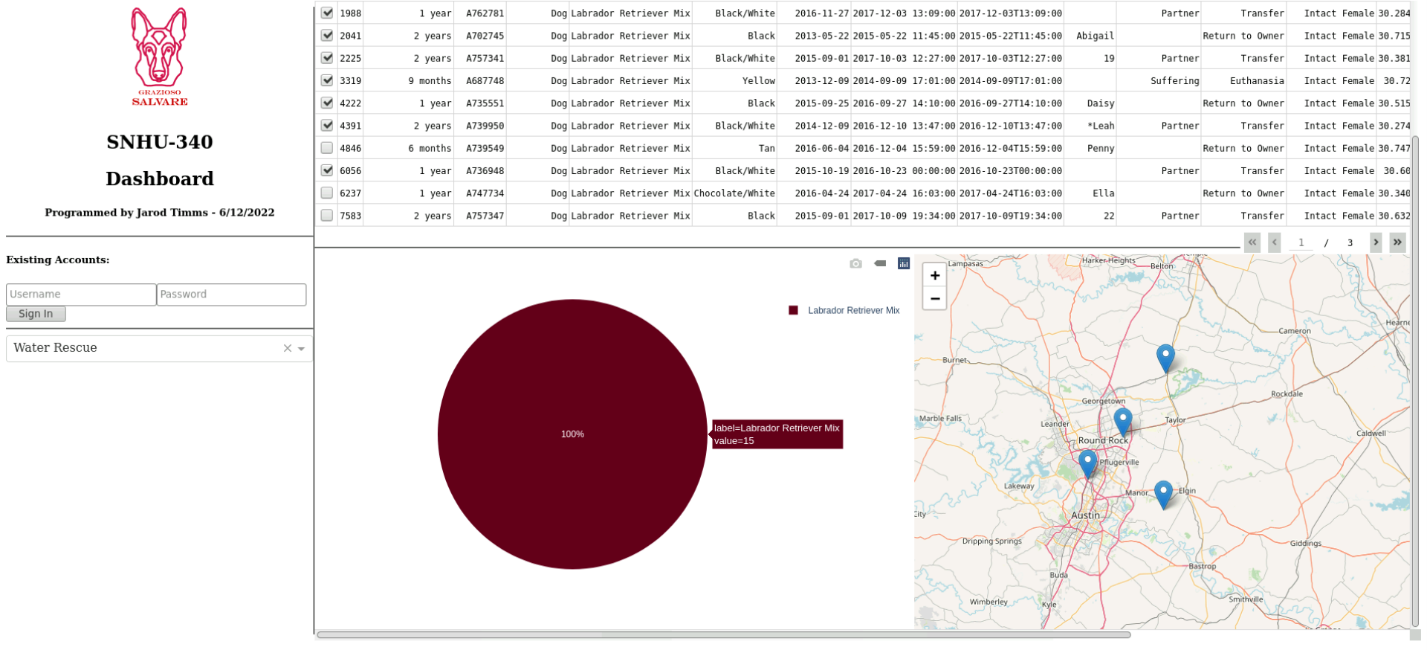




The list of available animals will adjust based on which ones best fit that profession based on sex, breed, and age, as per the requirements requested. The graph will also update based on the breed population of the table.

Selecting up to five rows on the table will begin returning locations on the map. Some locations seem to overlap if they are at the same location. After five rows are selected, every following row selection will be omitted until an older one is unselected.





## Troubles with the Dashboard

One issue I ran into was that some of the code that dealt with creating pins in the geolocation map. I would have preferred to have built pins based on how many rows were selected with a loop rather than an elif statement. Because of this, there is a lot of bloat in the map callback. Using a loop would have potentially allowed for as many pins to be on the map as allowed by the amount of rows displayed.

I also encountered an issue with the map sometimes not returning the first pin for an unknown reason. Any attempts to fix this issue resulted in the map disappearing entirely.

## Contact

Jarod Timms