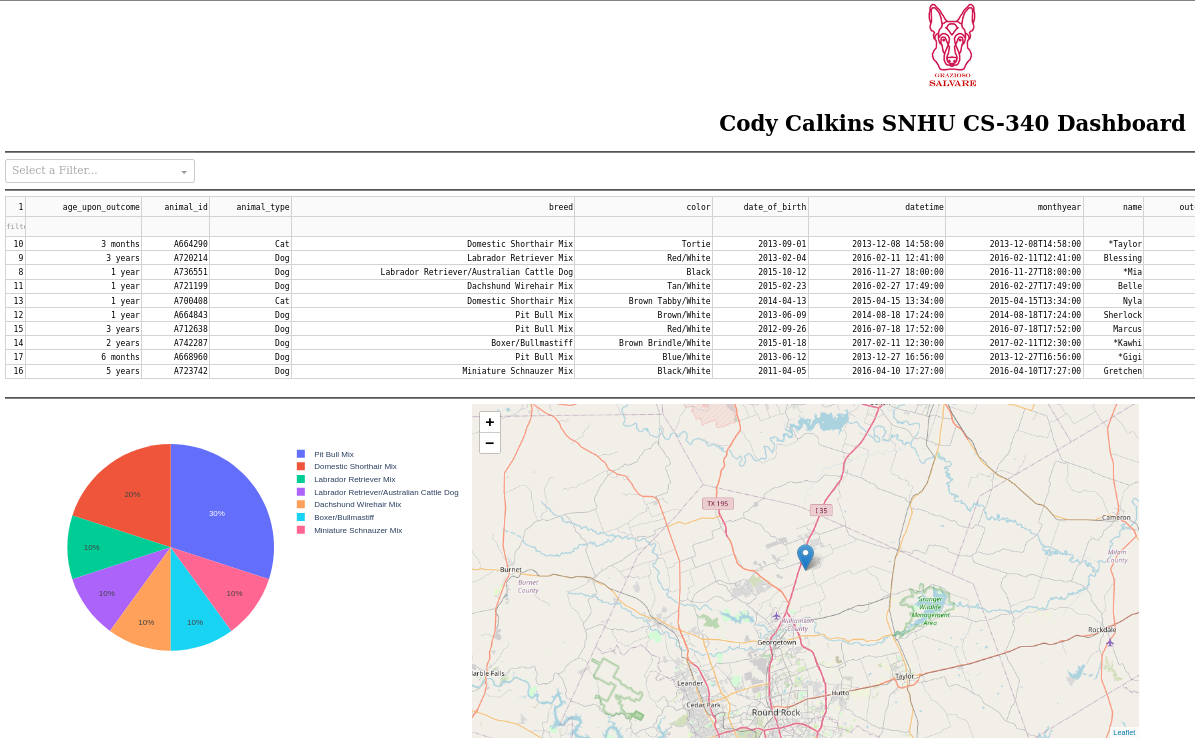
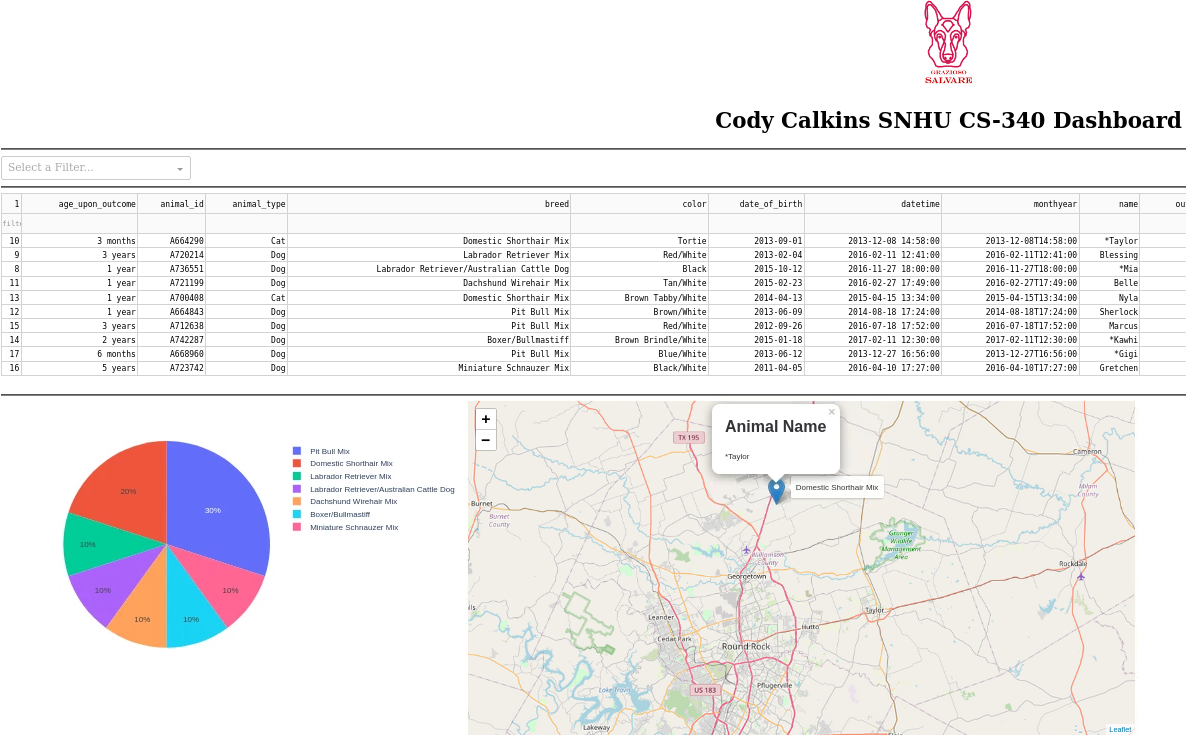
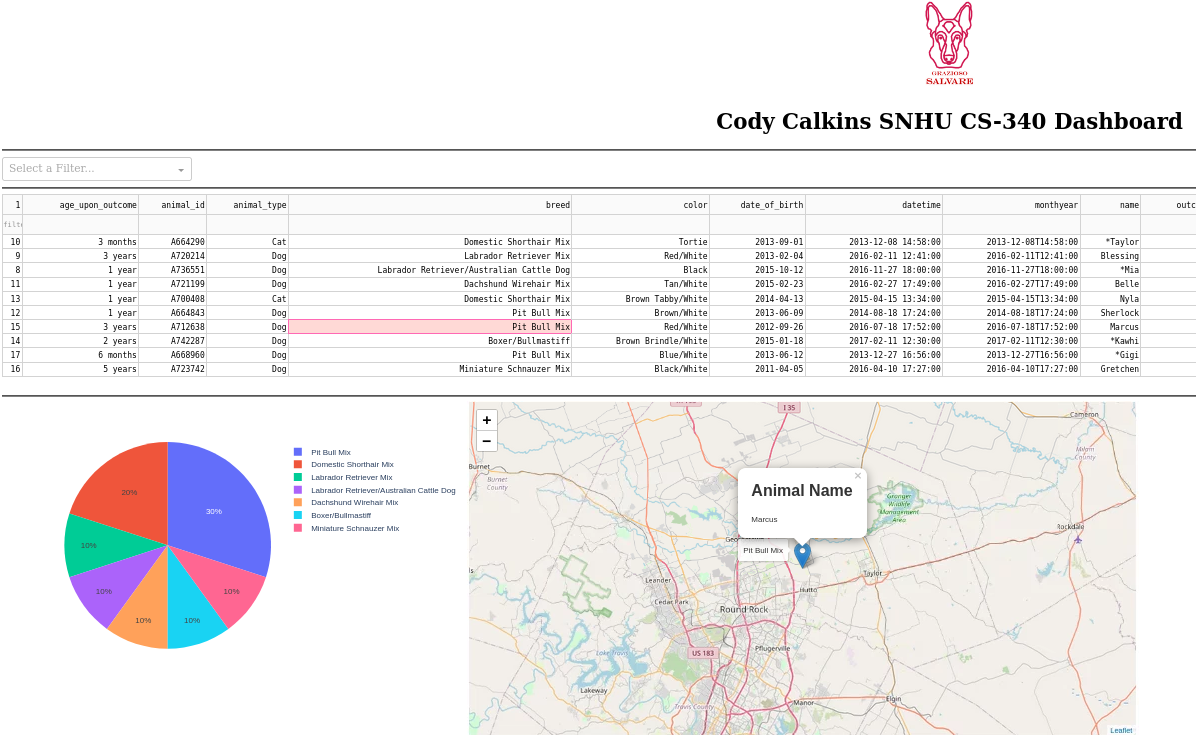
**Dashboard README**

**Functionality**

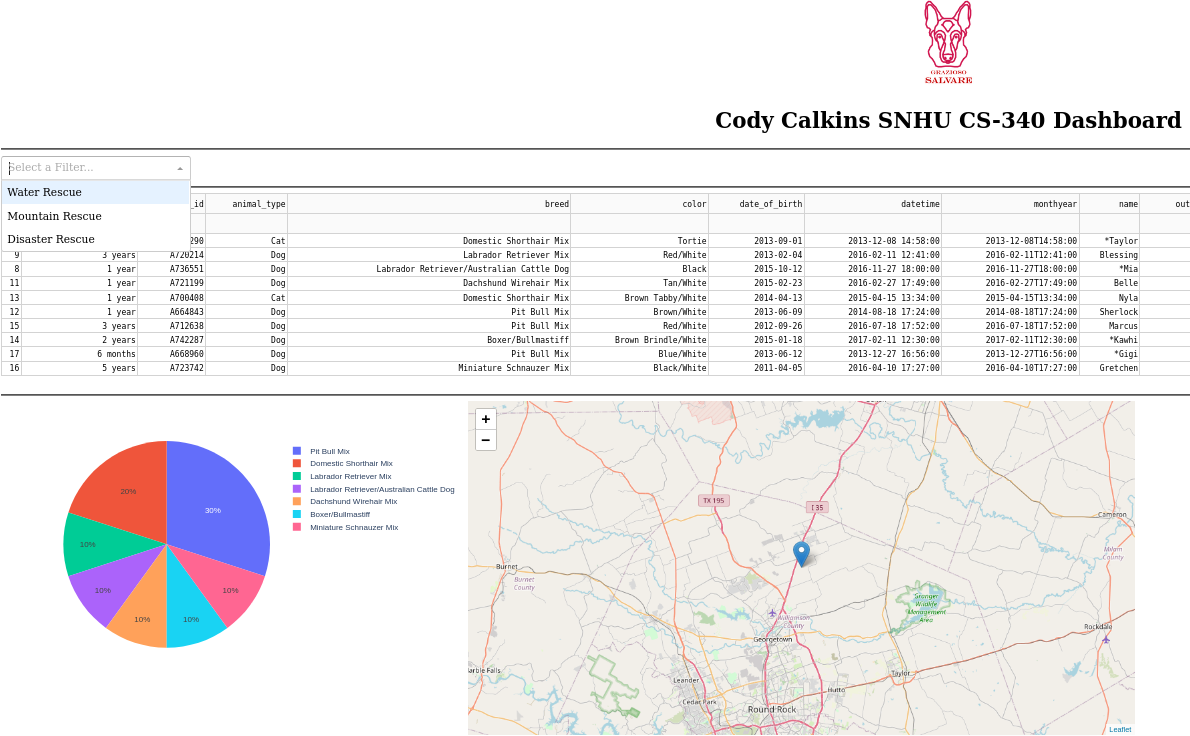
Below is the display of the dashboard when it is first opened; which includes the data table, a pie chart, and an interactive map. There is a filter dropdown that contains specified filter sets, as well as filter fields for individual filtering options. Not all columns are currently shown, so as to provide easier to read images:

If no entries in the data table are selected, the interactive map places a marker at the location of the first entry of the page. Once an entry is selected, the map will move the marker to the location of the selection. Hovering over the marker will display the breed of the entry, and clicking the marker will show the name of the entry, as seen below:

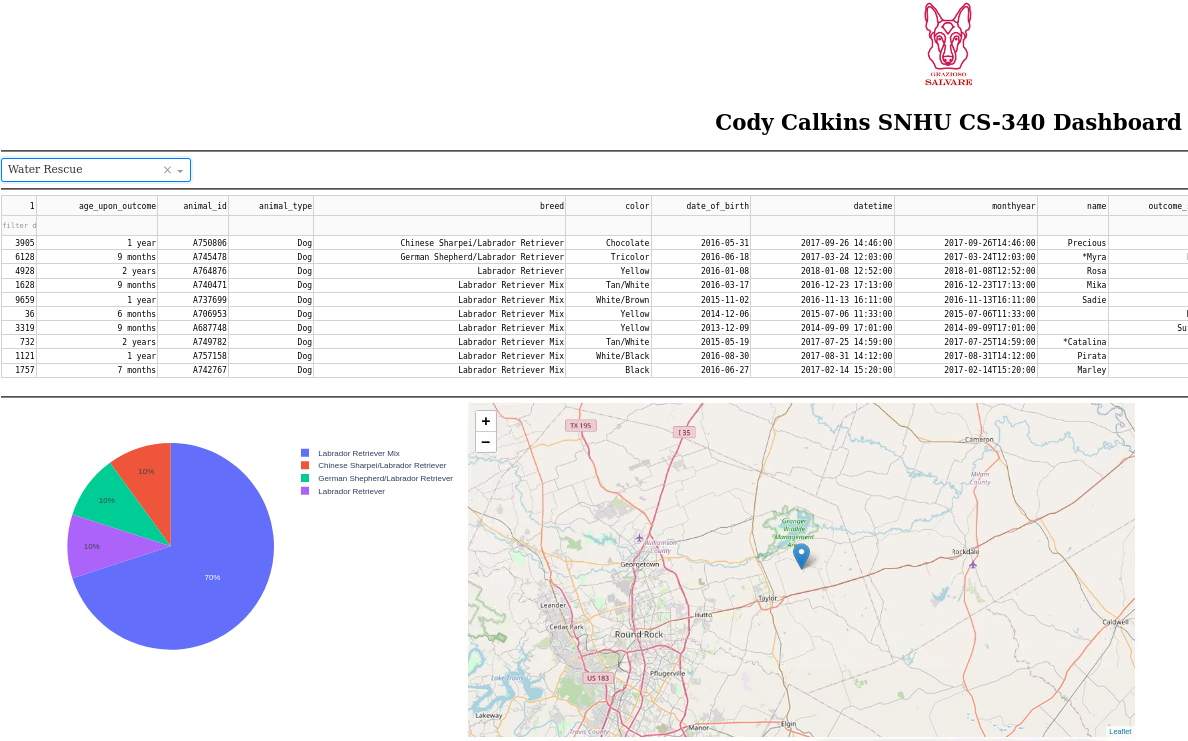


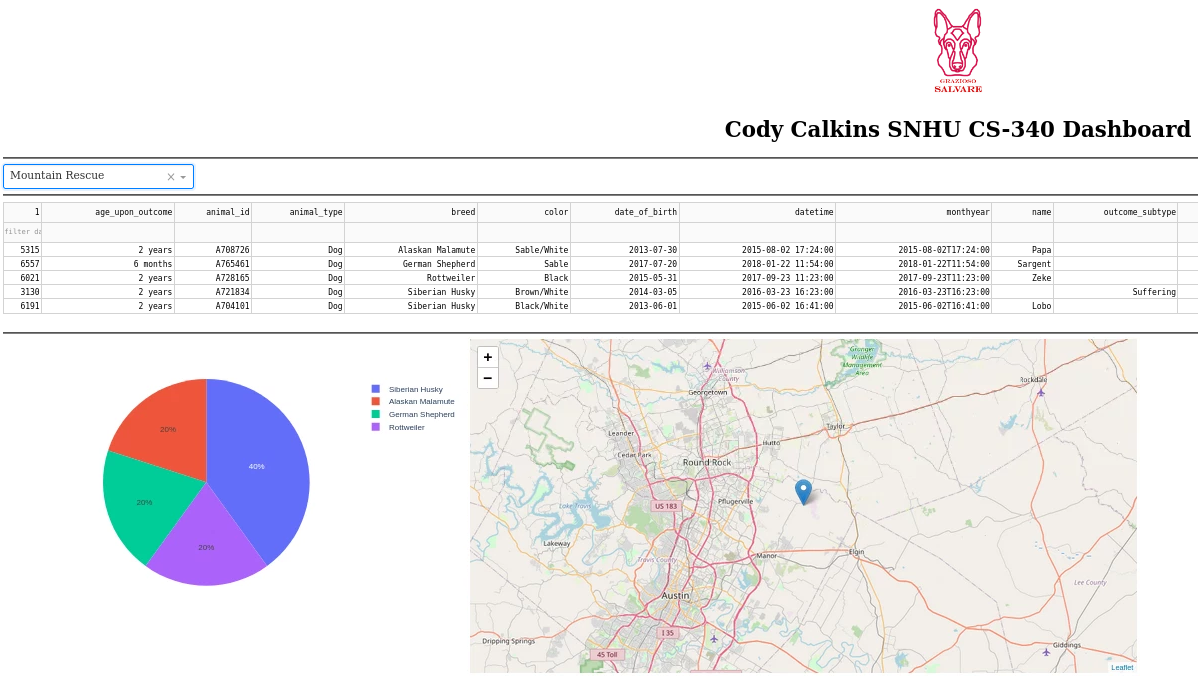


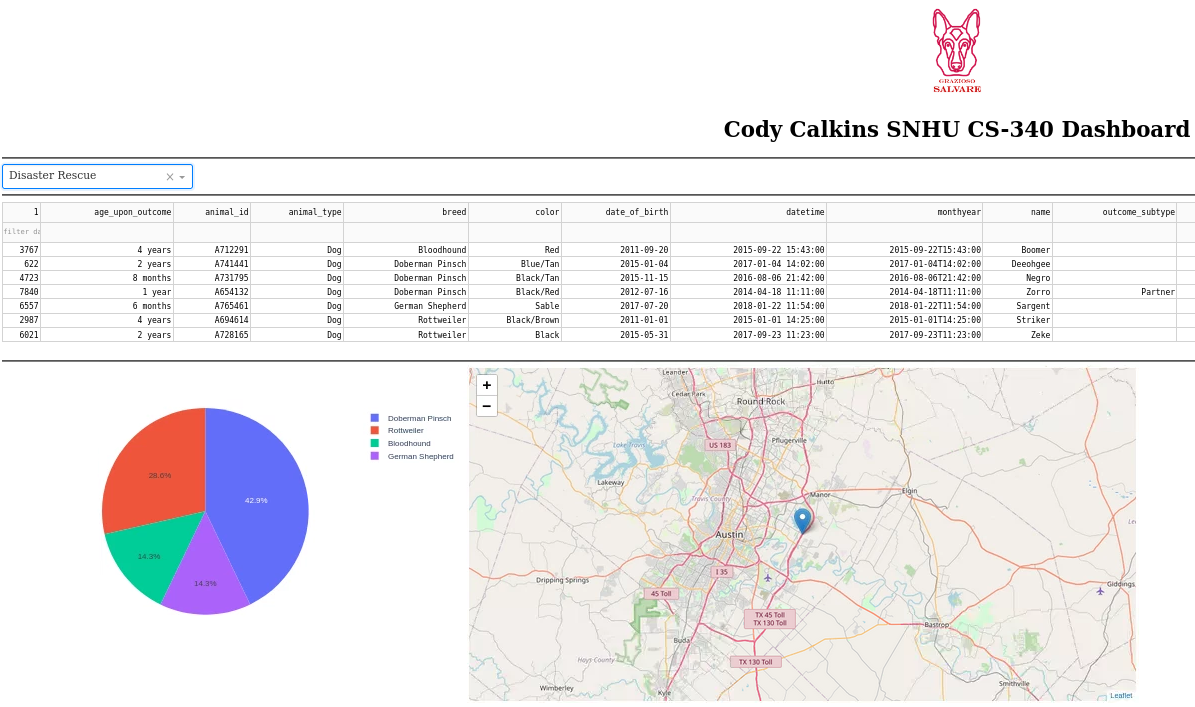
Here, we can see each of the specified filter sets programmed into the dropdown on the dashboard; Water Rescue, Mountain Rescue, and Disaster Rescue:



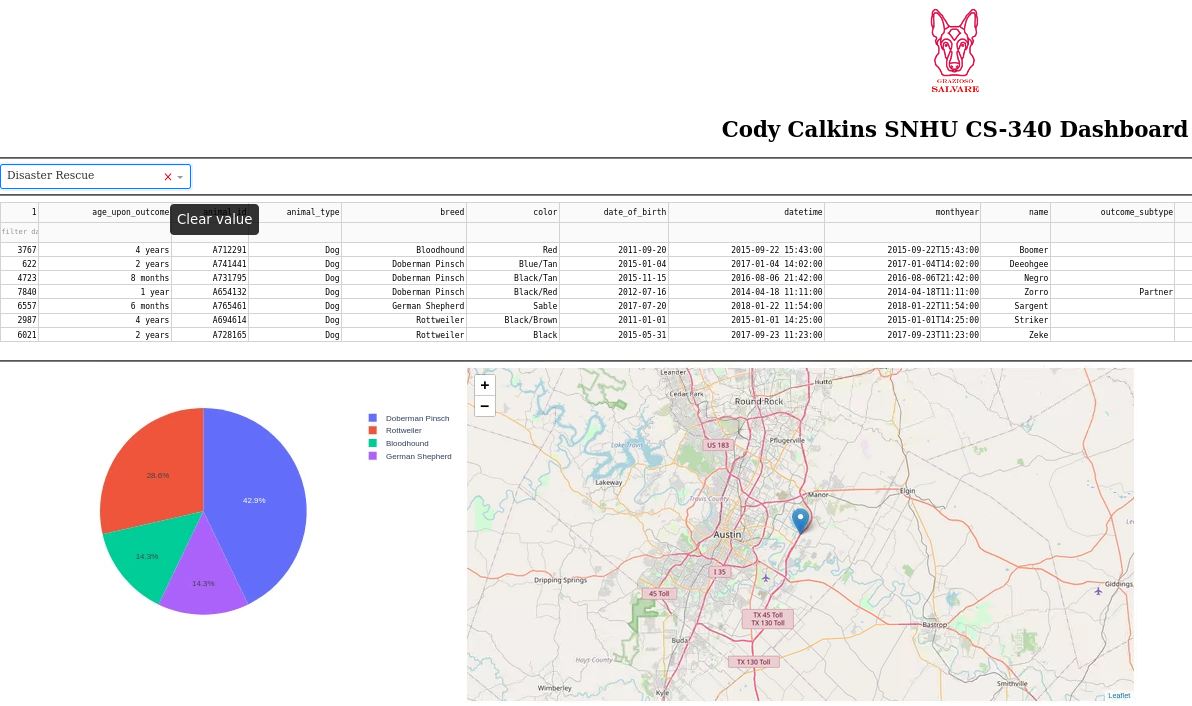
When the filter set is changed, both the interactive map and the pie chart will be updated. The map will move the marker to the first entry in the new filtered list and the pie chart will change to use the data that is currently displayed. These interactions will also happen if the next page of the data table is navigated to. Depicitons of these filters in use can be seen below:



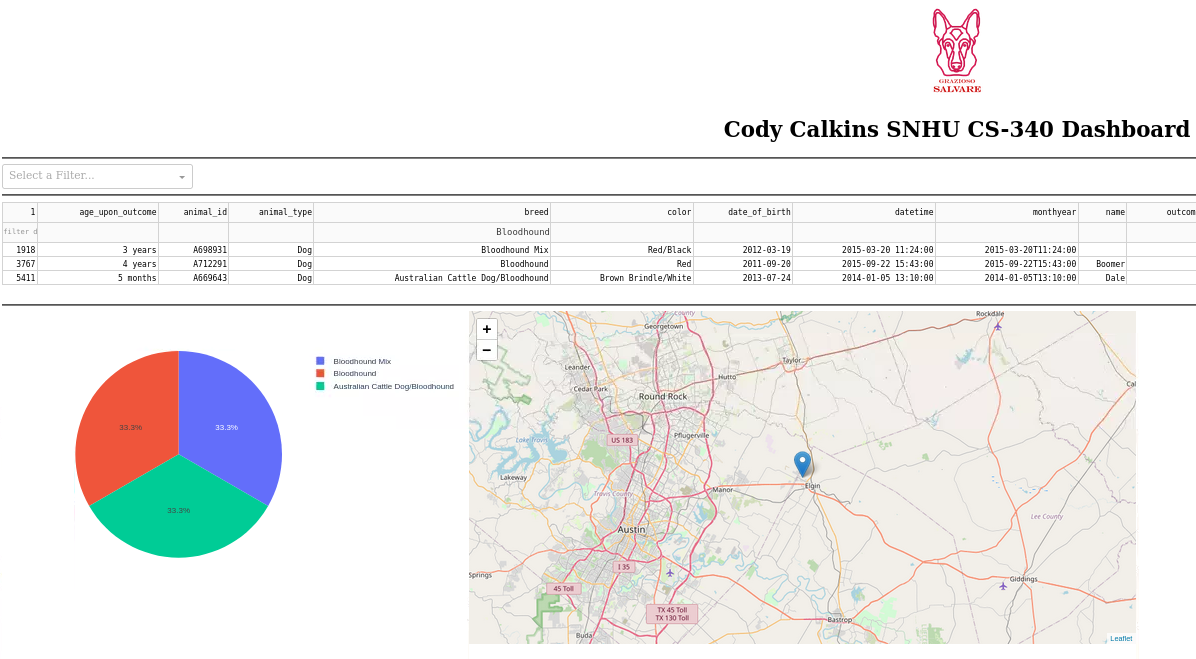




At any point, these filter sets can be cleared with a simple click. In the same dropdown, an “X” will appear once a filter set is selected. If this is clicked, the current filter set will be cleared, and the data table will be returned to its default state of loading all available data. This can be seen below:



Individual filters can also be used by typing values in the row right below the column names. Simply type in the appropriate values in as many boxes as you desire, press ‘Enter’, and the dashboard will update to correctly reflect the newly filtered data. The image below shows a data table filtered for any male dogs that are purebred or mixed Bloodhound:



**Tools Used**

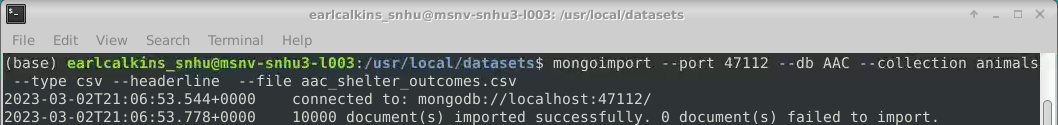
To use this software, you will need the following tools:

* Operating system of your choice, this dashboard was made within Linux
* MongoDB, installation instructions can be found [here](https://www.mongodb.com/try/download/community)
* Python 3.6, downloads can be found [here](https://www.python.org/downloads/)
* The code for this project is ran through a Jupyter Notebook, which can be accessed through [Anaconda](https://www.anaconda.com/products/distribution). Once Anaconda is installed, you will need to create a Python 3.6 environment with at least the following packages installed (this can all be done within the “Environments” tab in the Anaconda navigator.):
  + pymongo, version 3.10.1
  + dash, version 1.10.0
  + dash-html-components, version 1.0.3
  + dash-table, version 4.6.2
  + jupyter-dash, version 0.2.1.post1
  + plotly, version 4.6.0
  + pandas, version 0.25.1
  + numpy, version 1.17.2

For this dashboard, MongoDB is used in conjunction with Python due to the ease of communication between the two with pymongo. The tools available to Python provide a good framework for user authentication, as well as dashboard creation and database querying within the dashboard. Dashboard creation within Python is facilitated by Dash, which provides a simple yet effective means of creating a functional dashboard. Using Dash html, core, and plotly components, we are able to create a dashboard that provides simple data visualization. Once combined with pymongo and pandas, user authentication and data parsing can be performed to populate the dashboard with the appropriate data.

**Reproduction**

Along with the tools mentioned above, a MongoDB database that includes users and appropriate permissions will need to be created. This can be done by importing your desired dataset into MongoDB; The documentation for this process can be found here: <https://www.mongodb.com/docs/database-tools/mongoimport/>. An example of this is shown here (your port number will be different):



Once the database is created, you can then create any number of desired users. Steps for MongoDB user creation and authentication can be found here: <https://www.mongodb.com/docs/v4.2/tutorial/enable-authentication/>. Once the steps are followed and users are created, you can manually login to ensure that they are set up correctly:

Text

Description automatically generated

Once these steps are done, ensure that you have the provided files placed within the correct folder to run through Jupyter (likely in your system’s user folder). The CRUD.py file will need to be changed to correctly run through your own local database. This can be done within this function:



Adjust the indicated fields to reflect your port number, the authSource used to create the user, and the database being used. Once this is done, you will also need to change the following lines within the ProjectTwoDashboard.ipynb file to reflect the correct username and password:

A picture containing text, font, screenshot, guide

Description automatically generated

After the database and users are created and the appropriate code has been changed, this dashboard will be usable on your personal system.

**Challenges**

One challenge that I encountered while creating this dashboard was loading the data from MongoDB into the data table. Initially, the output of the dashboard only showed as “Loading…” and there were no error messages that indicated why this was happening. After some research, I found that the cause of this was due to the ‘ObjectID’ field passed from MongoDB. The issue was that this filed was unable to be serialized through JSON and used within the data table. The solution to this was to simply remove this field from the dataset once it was pulled from MongoDB, as shown below:



Another challenge that I encountered was related to how the data table interacts with the filter sets in the dropdown. Specifically, if you started on a page outside of the bounds of the filter set then changed to that filter set, nothing would be shown on the data table. For example, say a filter set contains two pages worth of data and you tried to switch to that filter set while on page 30 of the unfiltered data, no data would show since the page is still set to page 30. For a case like this, one could just keep changing the pages until they get to the right page, but if a filter set only contains one page of data then the page couldn’t be changed at all since the page changing arrows wouldn’t be present. To prevent this from happening, I simply created a callback to reset the data table back to page one when the value of the dropdown is changed.

**Contact**

Cody Calkins

Earl.calkins@snhu.edu