Brian Engel

CS-340 Project 2 Readme

The functionality of this project was set by Grazioso Salvare, who identifies dogs that are good candidates for search-and-rescue training. Grazioso Salvare is seeking a software application that can work with existing data from the animal shelters to identify and categorize available dogs. Global Rain has contracted for a full stack development of this application that will include a database and a client-facing web application dashboard. Grazioso Salvare will use this dashboard to interact with and visualize data from a MongoDB database. The dashboard must be a user-friendly, intuitive interface that will reduce user errors and training time.

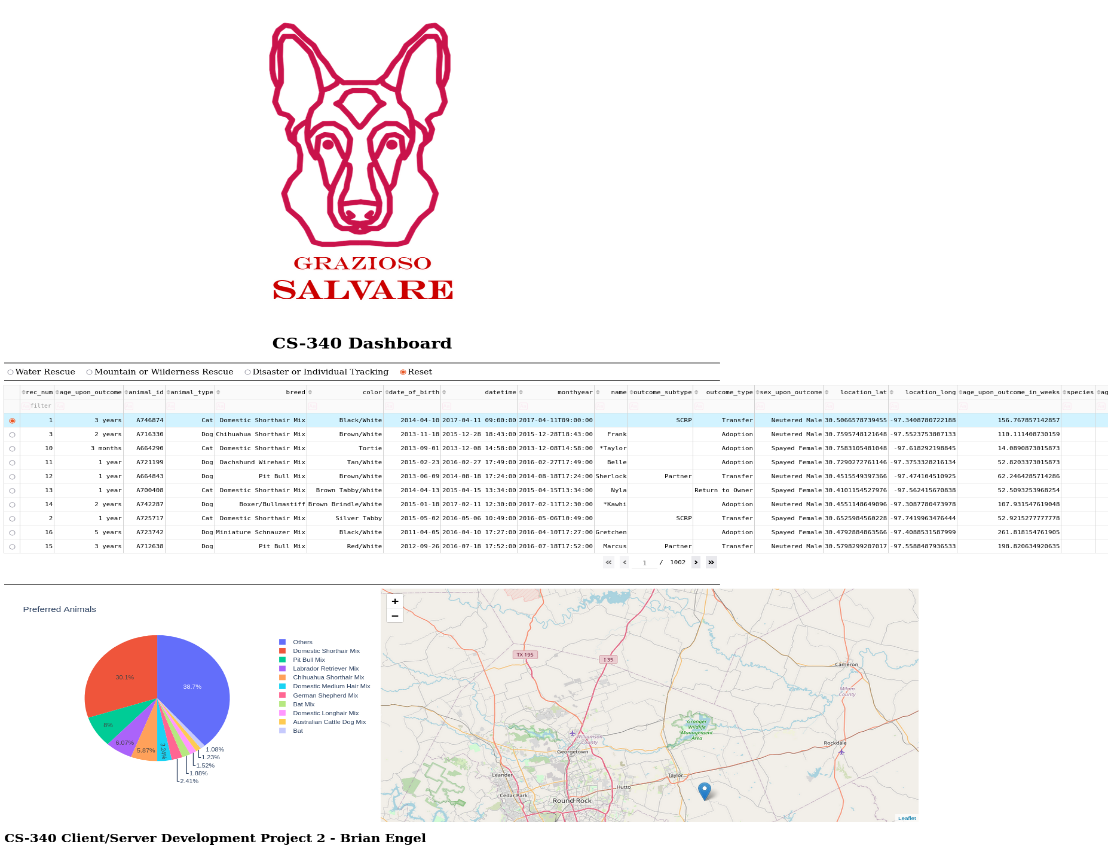
In addition to this, they would like their company logo and have that link to their home page: [www.snhu.edu](http://www.snhu.edu). They also want a unique identifier to credit the creator of the dashboard. For functionality they want filter options to include: Water Rescue, Mountain or Wilderness Rescue, Disaster Rescue or Individual Tracking, and Reset (returns all widgets to their original, unfiltered state), a data table which dynamically responds to the filtering options, a geolocation chart and a second chart of your choice (such as a pie chart) that dynamically respond to the filtering options.

For water rescue, the breeds include just: Labrador Retriever Mix, Chesapeake Bay Retriever, and Newfoundland, the sex should be just intact female, and the training age should be 26 weeks to 156 weeks.

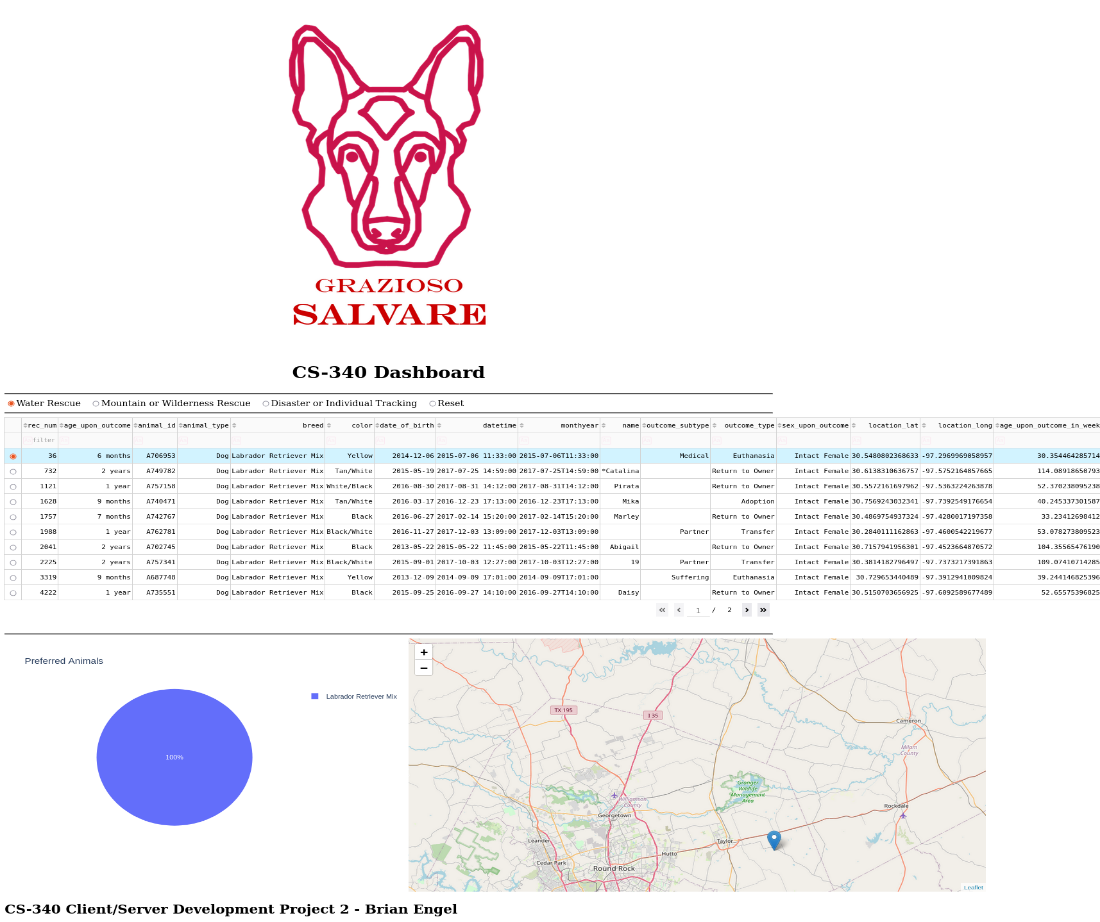
For mountain or wilderness rescue, the breeds include just: German Shepherd, Alaskan Malamute, Old English Sheepdog, Siberian Husky, Rottweiler, the sex should be just intact male, and the training age should be 26 weeks to 156 weeks.

For disaster or individual tracking, the breeds include just: Doberman Pinscher, German Shepherd, Golden Retriever, Bloodhound, Rottweiler, the sex should be just intact male, and the training age should be 20 weeks to 300 weeks.

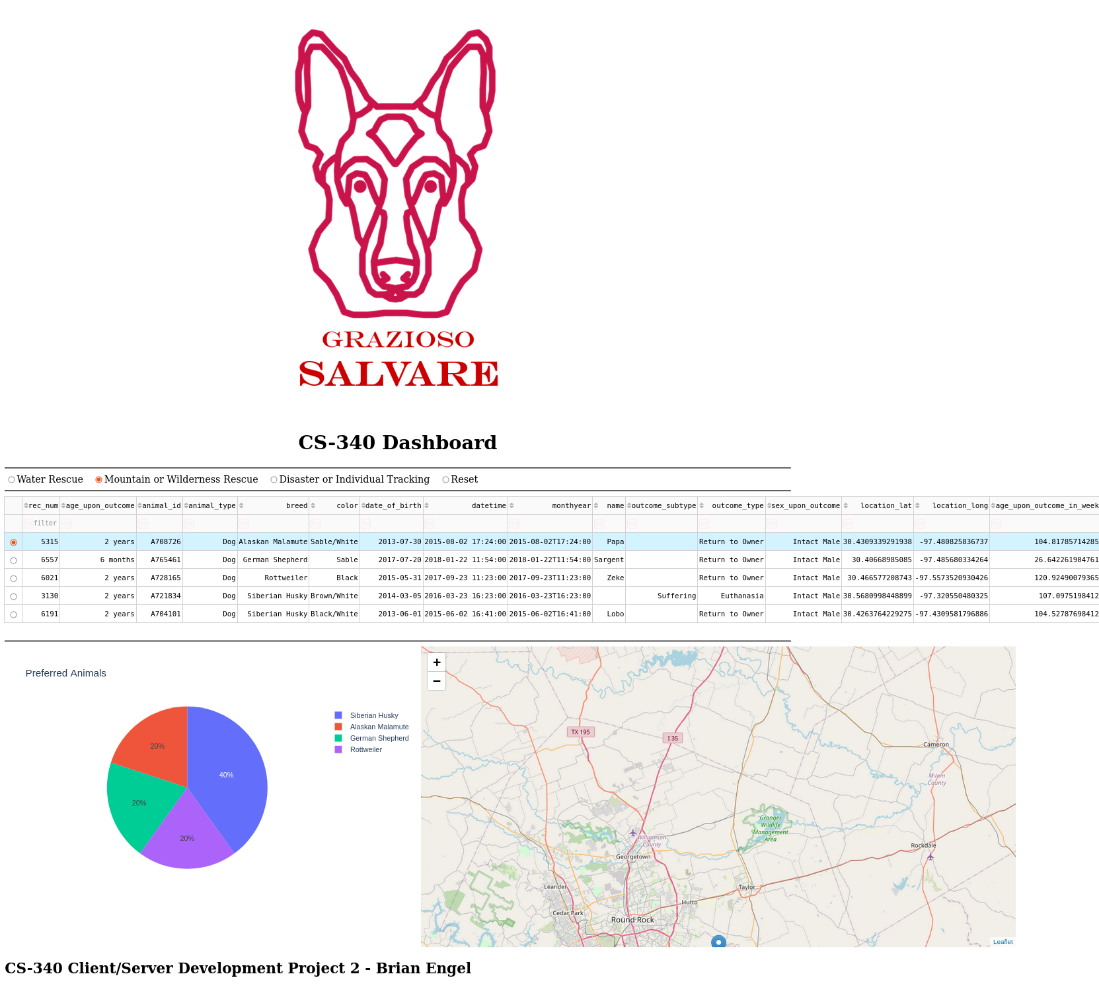
Starting state



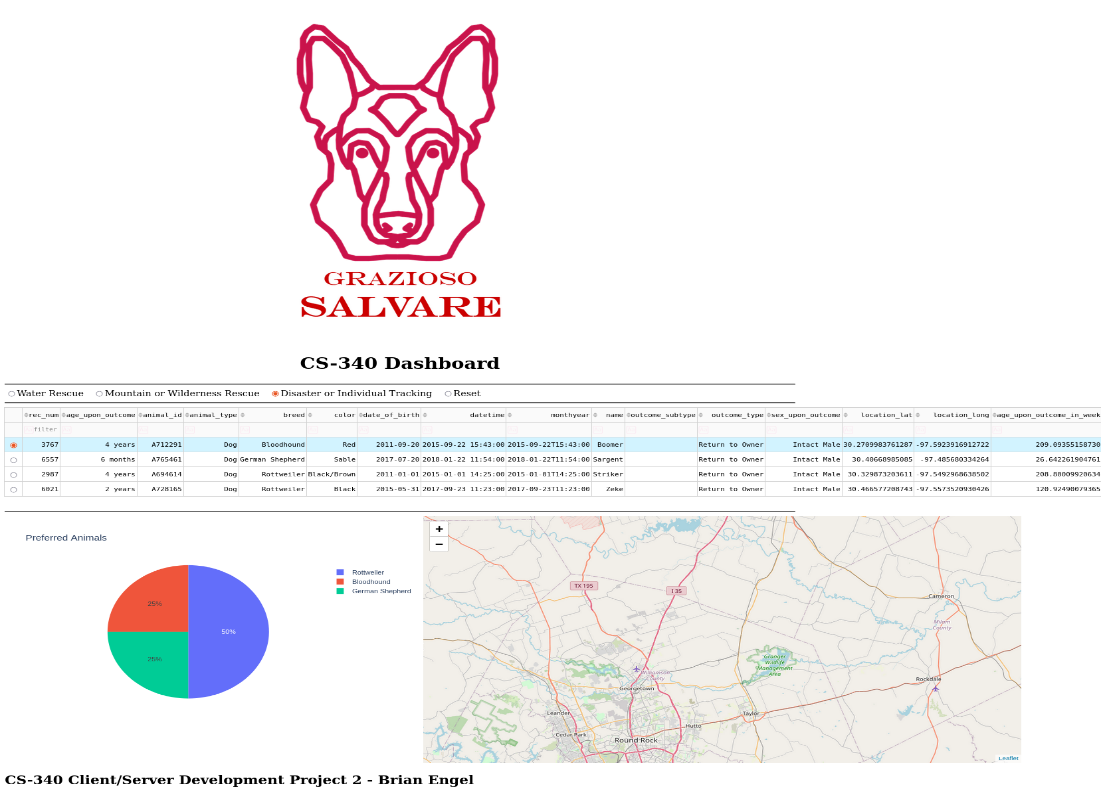
Water rescue



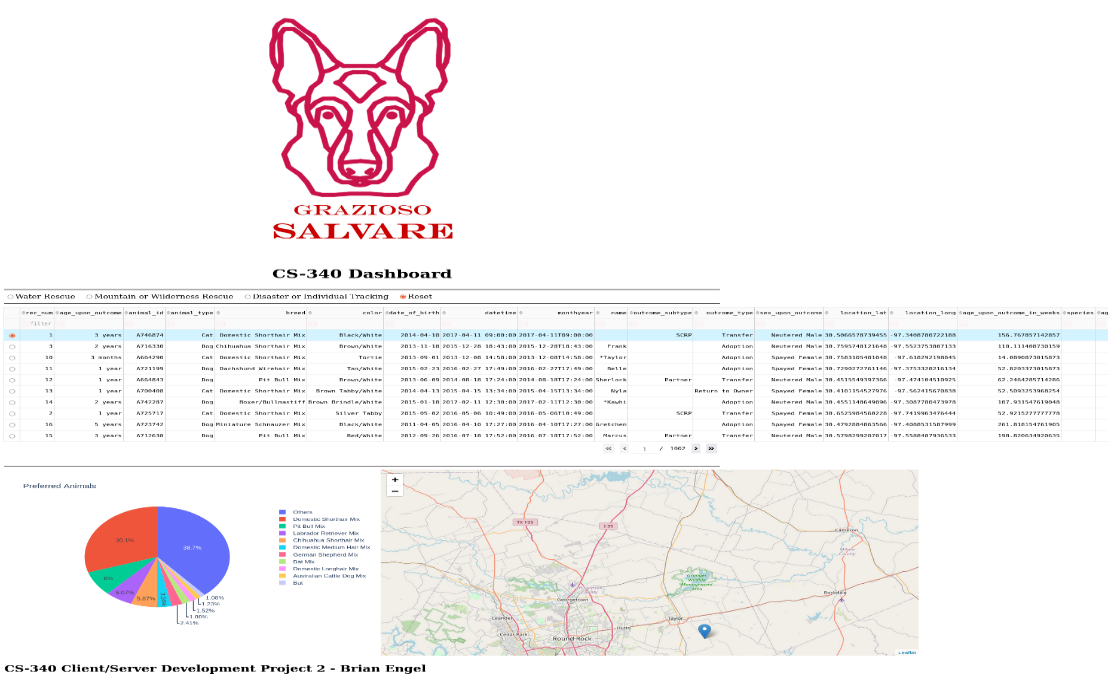
Mountain or wilderness rescue



Disaster or individual tracking



Reset



There are a few tools that I used to achieve the desired functionality of this application. First off, I created this in a Jupyter notebook. I used MongoDB as the database since it has an easy-to-use API with Python. I also used the Dash framework since it can create a web application using only Python and is also easy to use. NumPy is a math library for Python. For creating the pie chart and the map Plotly, Matplotlib, and Pandas were used. To use or learn more about any of these you can get them from these sites:

[Jupyter](https://jupyter.org/)

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

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

[NumPy](https://numpy.org/)

[Plotly](https://plotly.com/)

[Matplotlib](https://matplotlib.org/)

In order to get the required functionality out of this application, the first thing is to import the database into a dataframe. We have to create an instance of our python module AnimalShelter to convert the csv file and allow us to read the information and then transfer it to a dataframe.

A screenshot of a computer program

Description automatically generated

After this we set up the web pages html using dash. We add the company logo and link with this:



A close up of a computer screen

Description automatically generated

The unique identifier is done with this:



The filter options are radio buttons handled like this:

A screenshot of a computer code

Description automatically generated

The dataTable:

A screen shot of a computer code

Description automatically generated

The pie chart and map:

A screen shot of a computer code

Description automatically generated

This is just the general setup for the web page and does not include any of the interactivity. For making it interactive we have to set up the callback functions. The most important one is the one filtering the animals for the different rescue options. This also sets the row that is selected to the first row when a new radio button is clicked.

A screenshot of a computer program

Description automatically generated

A screen shot of a computer code

Description automatically generated

A screenshot of a computer program

Description automatically generated

Next is a callback function to update the graph to what is in the datatable. I took the liberty of grouping any breed that came back as 1% or less into a group called “Others” because the graph was next to unreadable with the amount of breeds that are in the table for the unfiltered option.

A screen shot of a computer code

Description automatically generated

The next function is a callback to highlight the selected row or rows:

A screen shot of a computer code

Description automatically generated

The last callback function is to update the map to the selected animal’s location.

A screenshot of a computer code

Description automatically generated

I didn’t run into too many challenges while creating this application except for I was getting quite a few errors when the page was first initializing. This was due to the fact that I didn’t really have a starting state defined for anything. Once I figured out how to fix that by setting the initial row to the first row, and having the initial tab be the reset button everything started working beautifully.