Harrison Bergeron

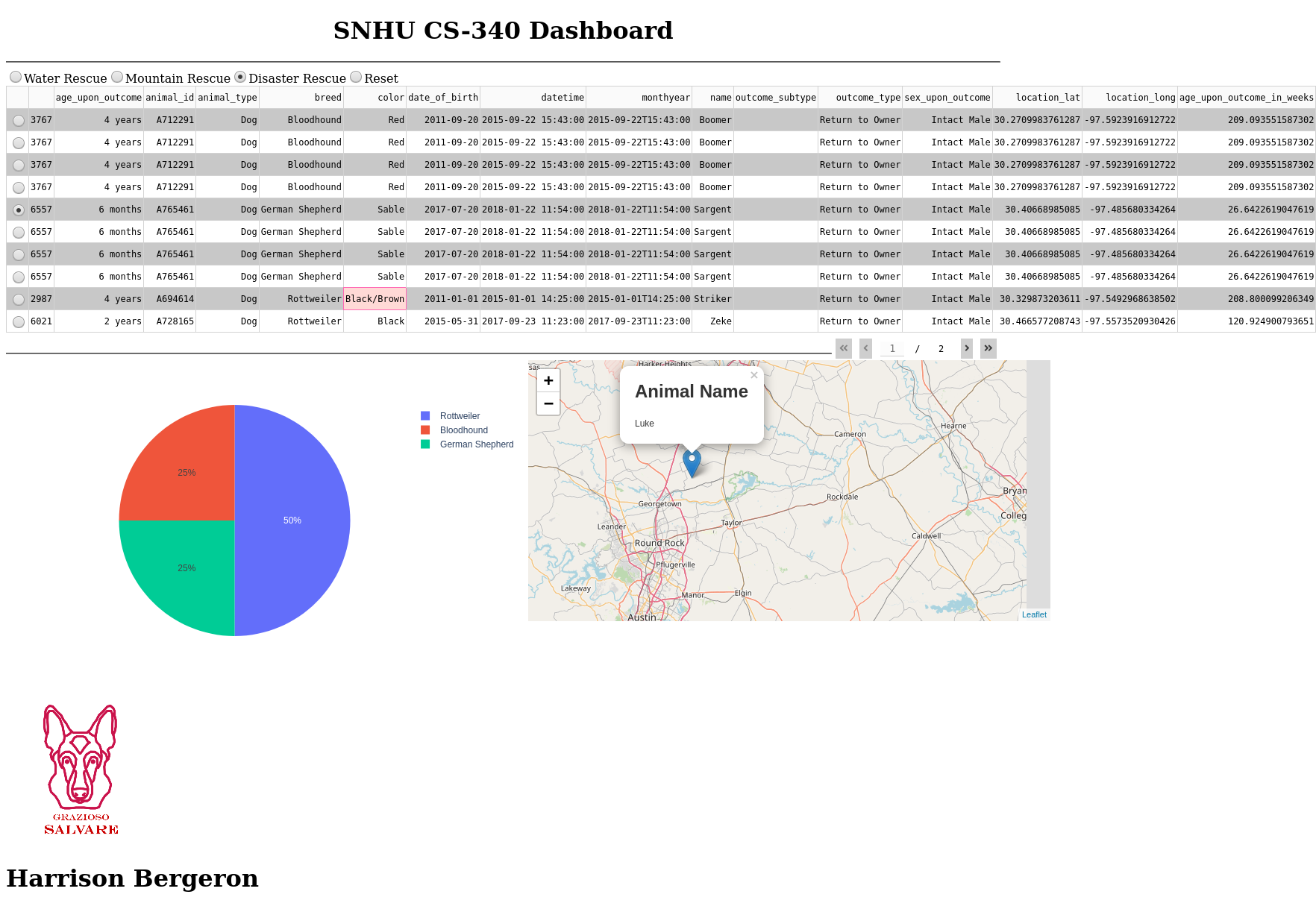
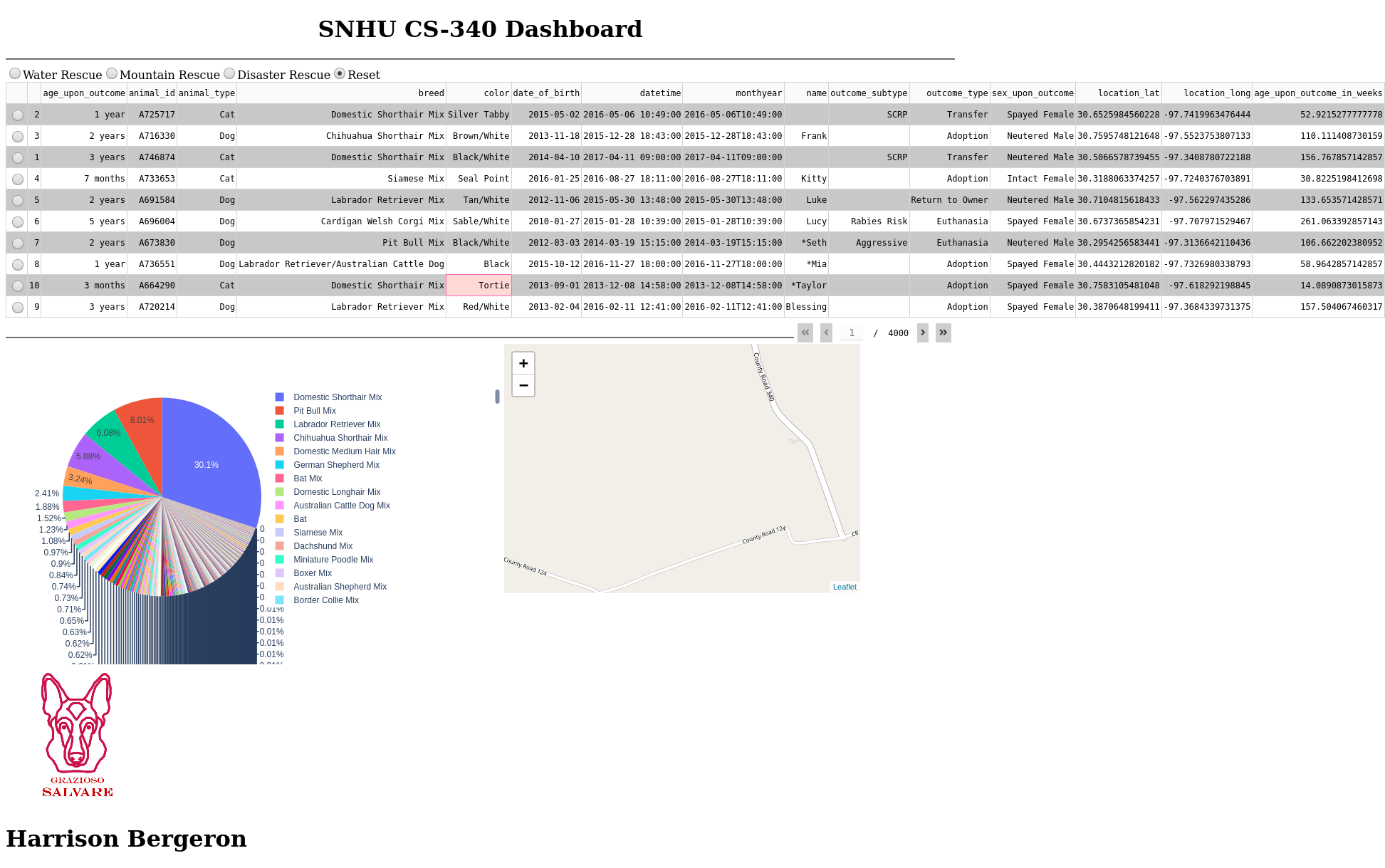
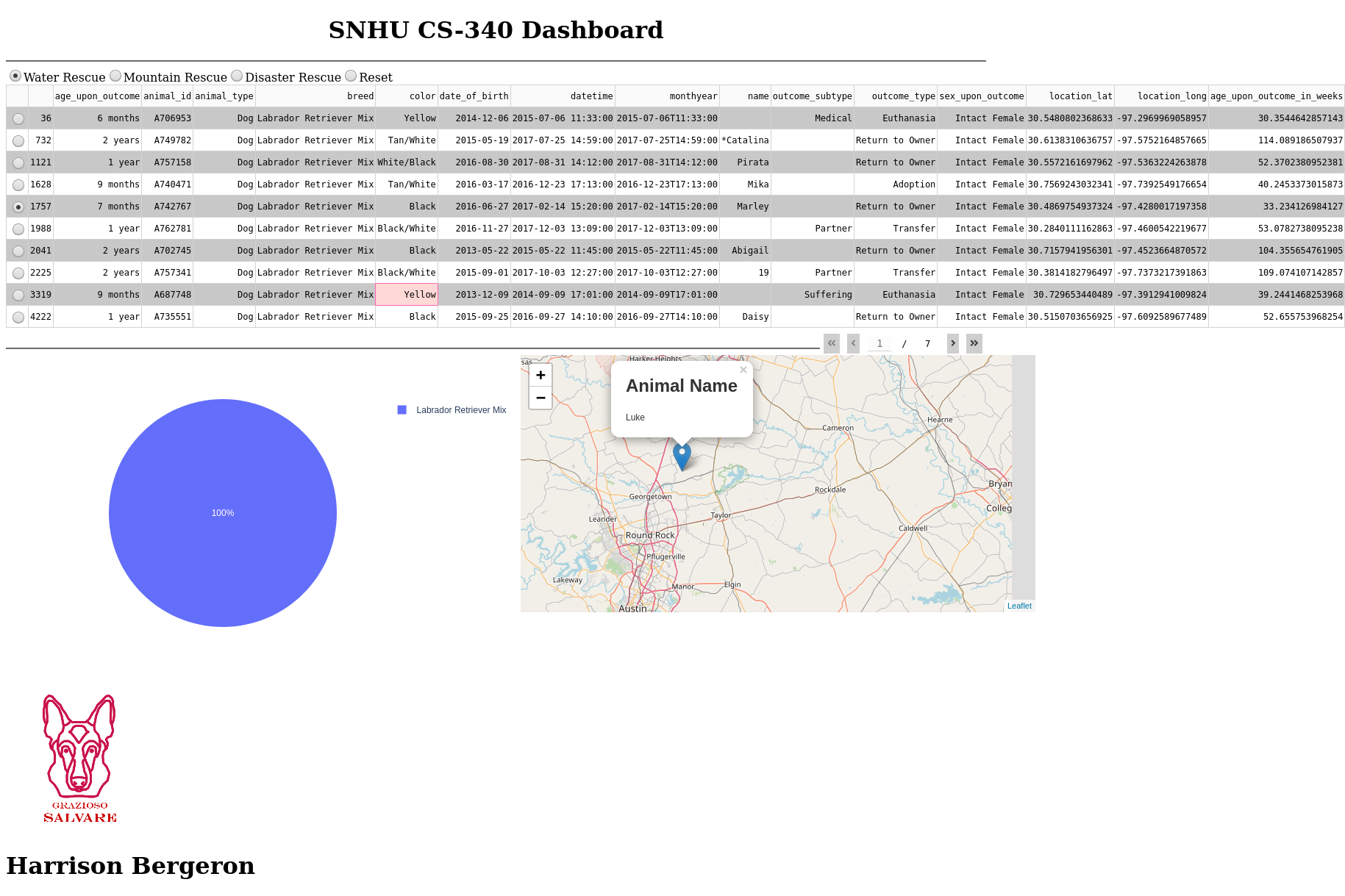
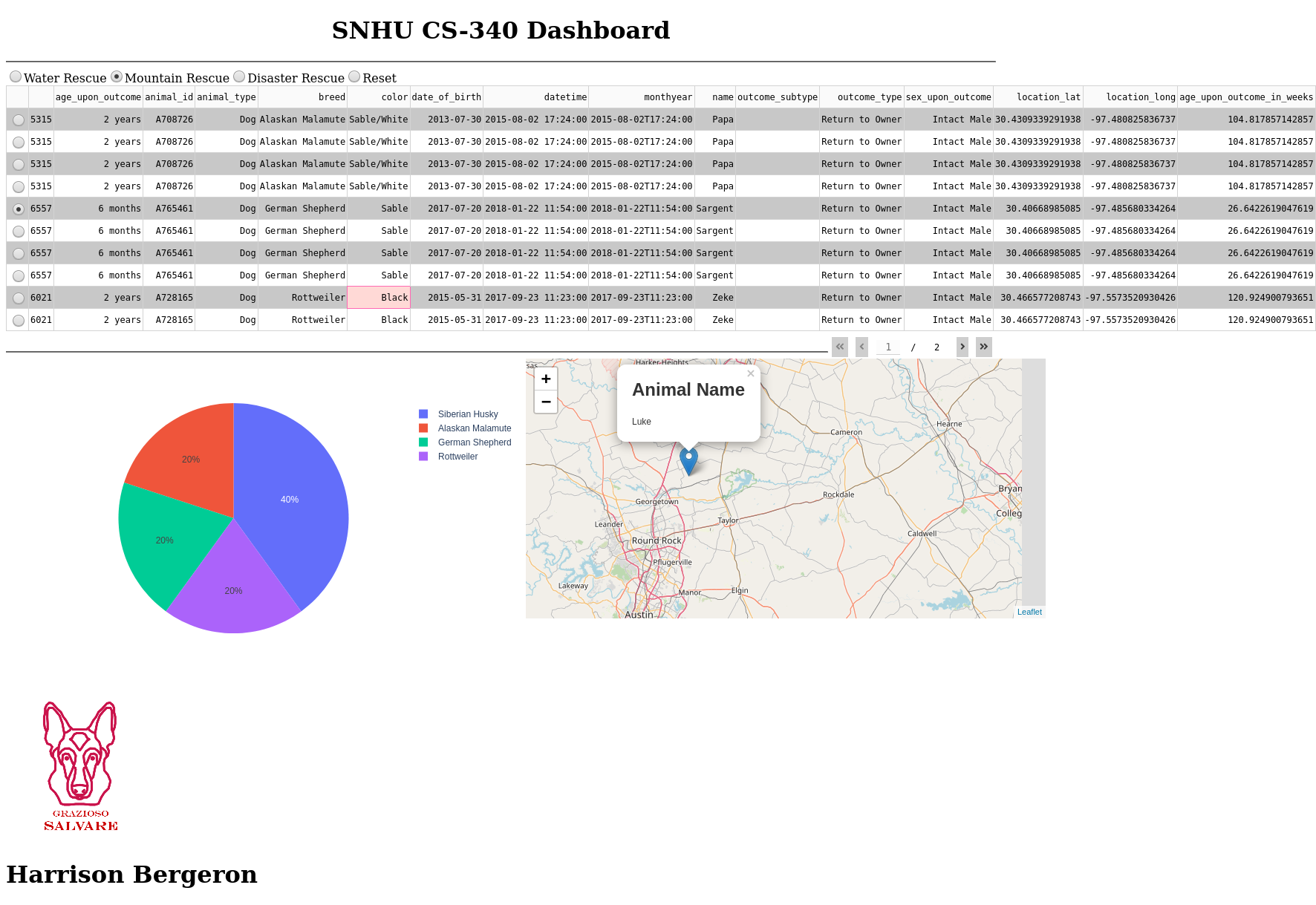
CS 340

Project Two Grazioso Salvare Readme

October 20, 2021

**Functionality**

The Grazioso Salvare dashboard required a data table for the Animal Shelter Outcomes dataset with a filter option to limit which rescue types were displayed. In addition, a geolocation map that updates to show the location of the selected animal was required to display a navigable map using GPS markers. Lastly, another chart was required to display data visually from the dataset. All of the widgets required the ability to update dynamically based on the rescue filter selected.



**Tools**

MongoDB was used for this project because of its ability to easily retrieve data from python queries using PyMongo. A very simple Python CRUD module allows for an interface between the Dash application and PyMongo. MongoDB provides data through queries that can be programmatically defined in a Python script, which makes it the perfect tool for a Python and Dash data visualizing application. Dash provides an easy way to create visual displays for data like graphs, as well as interactive UI components, and is easy to use with Python since it combines the functionalities provided by HTML, CSS, and JavaScript. The following resources were utilized to assist the completion of the Grazioso Salvare application:

<https://dash.plotly.com/dash-core-components>

<https://plotly.com/python/>

**Steps**

The first step in developing this project for Grazioso Salvare was to create a data table that would display all documents from the Animal Shelter Outcomes dataset in MongoDB. This involved creating a data table in Dash and using Panda data frames to allow for easy loading of the dataset from the MongoDB CRUD module. The next step was to create custom Python CRUD queries that would filter the data based on the type of rescue animal. By using the appropriate compound queries like $or and $gte, the data could be filtered by passing the appropriate Python dictionary queries. After this, the Dash data table, graph, and geolocation map had to be updated to accept the data from the new query using the Dash interactive components as inputs. The Dash components were updated using callbacks that referenced the component ID receiving the new data as well the component ID causing the event with a change in the desired attribute.

**Challenges**

One challenge when implementing the dashboard application was creating a component layout that was intuitive. For example, the pie chart and geolocation map would not adopt a horizontal side-by-side layout automatically, and there was a great deal of trial and error with component styling options before the desired layout was achieved. Another issue was getting components to respond to their inputs properly, especially when retrieving updated data from new queries. Using separate Python functions, the data frame was to be updated per component, but since the data frame was not being treated as a global variable, the new data from the query was not carrying over. By returning the new data frame instead of using a global variable, the issue was resolved and all components were able to respond to changes in the rescue filter.