# CS 340 README Template

About the “Grazioso Salavare Animal Rescue Dashboard”:

The project is a web-based dashboard app that is designed to assist Grazioso Salvare identify suitable dogs for different types of rescue operations. The dashboard app gives the user interactive data tables and tools that will help filter out animals to identify suitable candidates from the Austin Animal Center (“AAC”) database.

## Motivation

This project was created to support Grazioso Salvare in their mission to find suitable rescue dogs efficiently. By using data visualization and filtering, the dashboard helps to streamline the process of identifying dogs for specific types of rescue.

## Getting Started

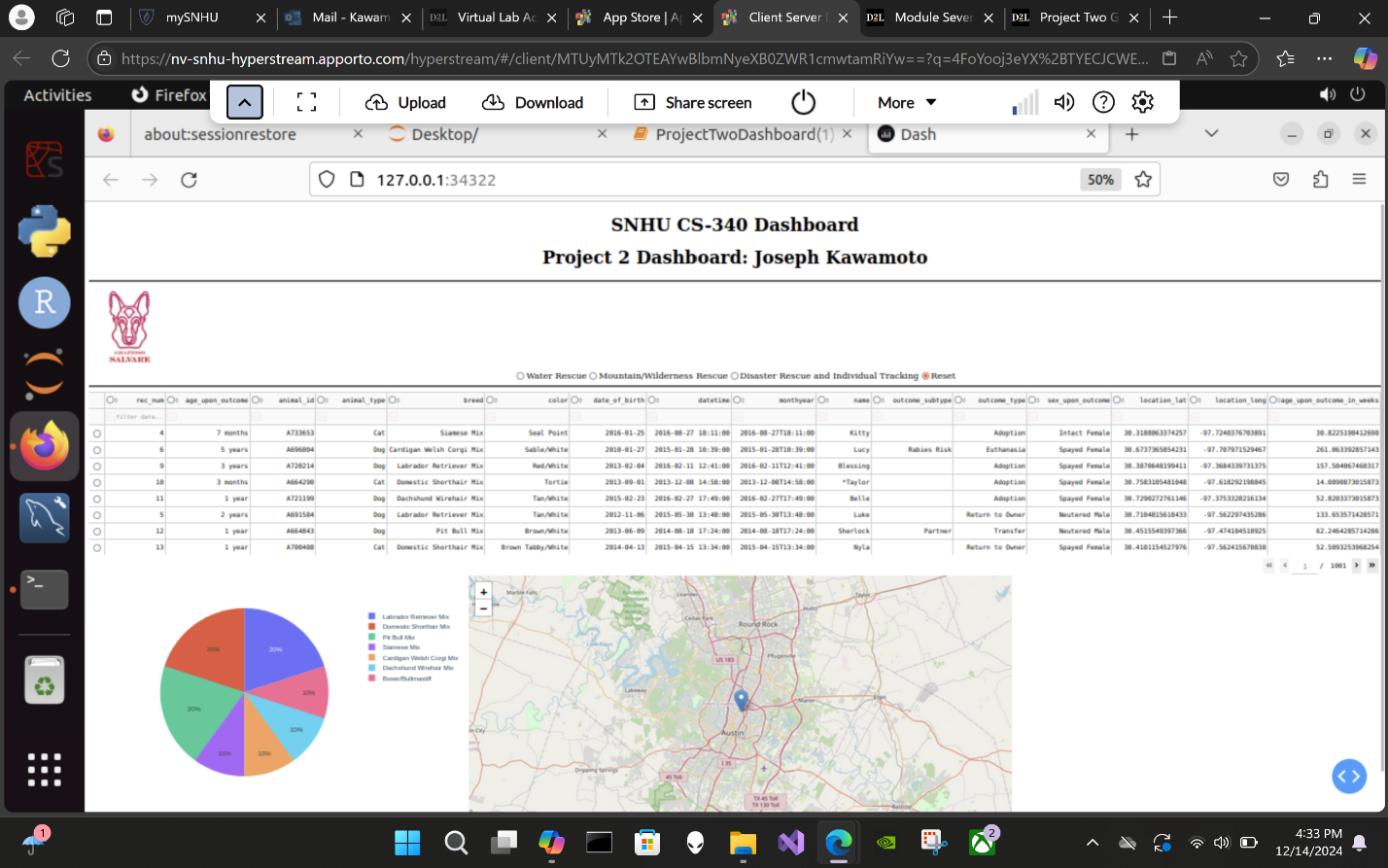
To get a local copy running the user would need to clone the repository, navigate to the project directory, and install the required packages. Next, you will need to upload the AAC database outcomes CSV file into MongoDB. The user will also need to create a MongoDB username and password to access the information and run queries on it.

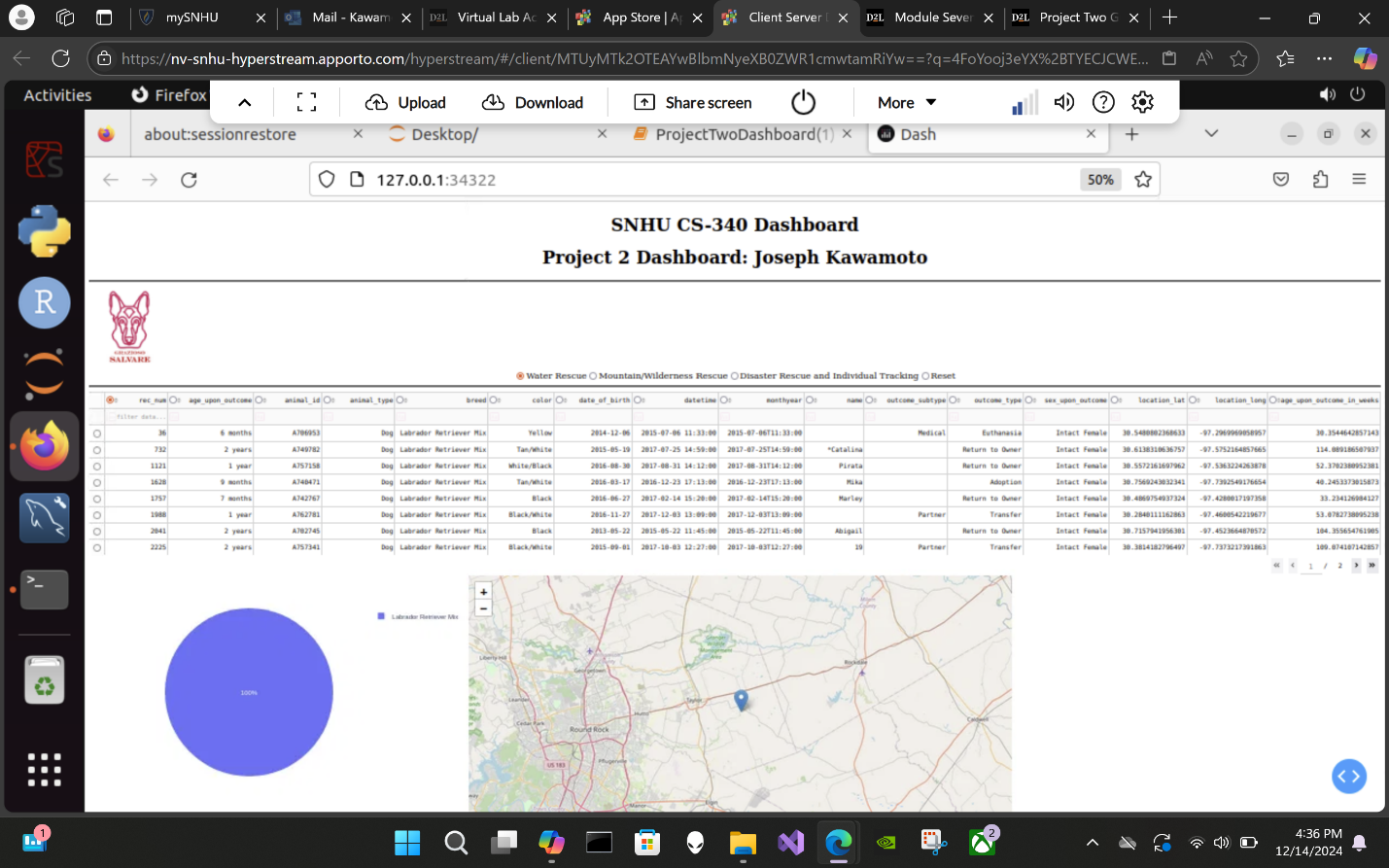
## Usage

The Grazioso Salvare Animal Rescue Dashboard Application allows the user to filter through animals depending on the type of rescue operation they are trying to fulfill. The filters are broken up into three categories: water rescue, mountain/wilderness rescue, and disaster rescue/individual tracking. Each filter type is linked to a key word, and each key word has suitable dogs associated with it. For example, the water rescue filter will bring up dogs of the following breeds: Labrador Retrievers, Chesapeake Bay Retrievers, and Newfoundlands. The mountain/wilderness rescue filter will display breeds like: German Shepherds, Alaskan malamutes, Old English Sheepdogs, Siberian Huskies, and Rottweilers. And the disaster rescue/individual tracking filter displays breeds like: Dobermans, German Shepherds, Golden Retrievers, Bloodhounds, and Rottweilers. The filters will also display dogs of the desired breed and age.

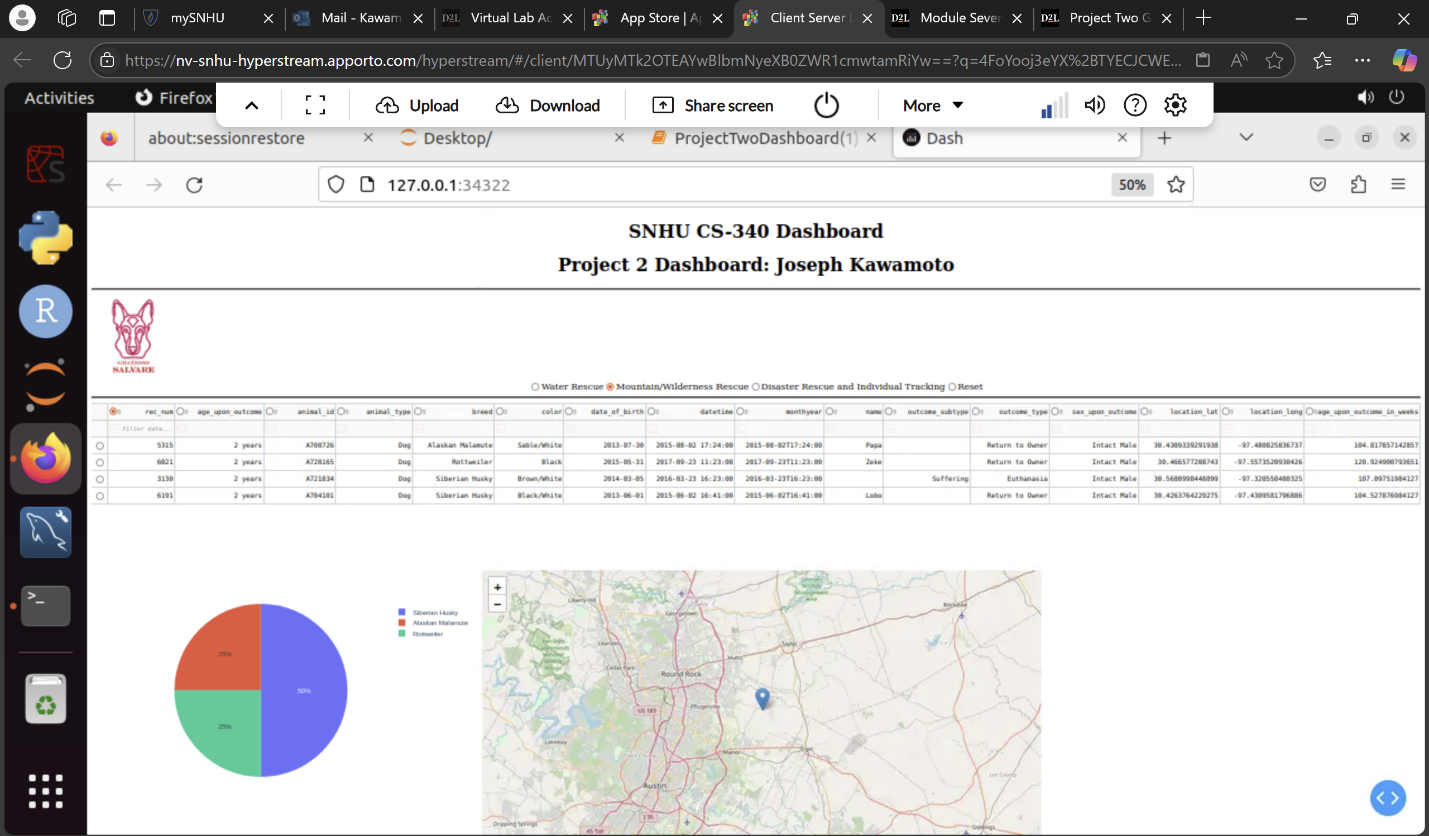


The filtered animals will be reflected in the pie chart on the bottom left-hand side of the dashboard. This will show the number of each breed of dog that fits the chosen filter. All animals will be displayed in the data table while the database remains unfiltered. If the database is filtered, only the animals that correspond to each filter will be displayed on the data table. Each animal can be selected by clicking the bubble next to each data entry. Selecting an animal will display its geolocation on the map in the bottom-right hand side of the dashboard.

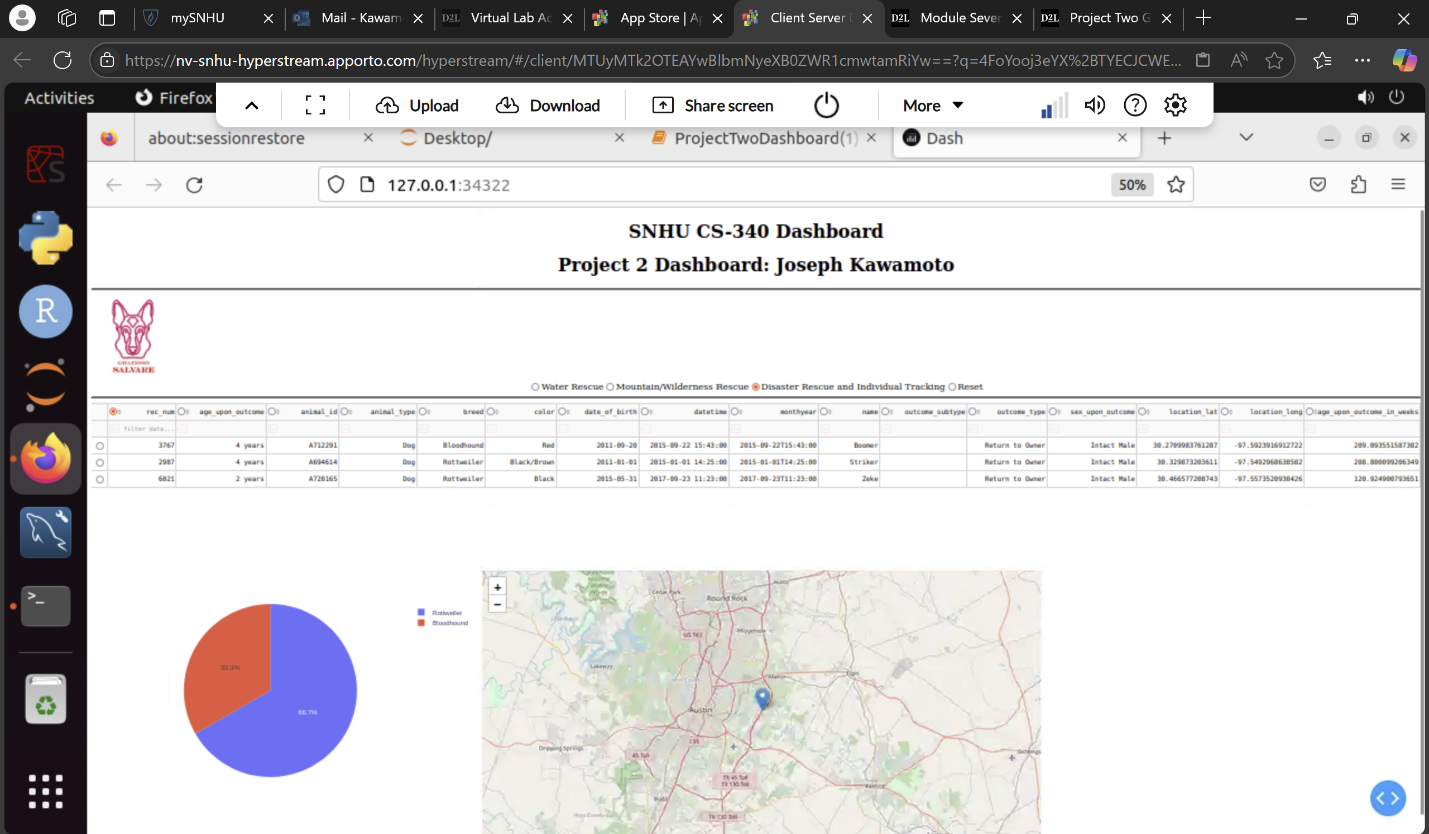


While the water rescue filter is selected, the dashboard will display the following data:

While the mountain/wilderness rescue filter is selected, the dashboard will display the following data:



While the disaster and individual tracking filter is selected, the dashboard will display the following data:



## Tools Used and Rationale:

Dash: This was used for creating the web application interface. Dash was used because it provided a way for me to build the interactive web application that was required for the project.

Plotly: I imported plotly so that I would be able to implement interactive charts and graph widgets. I was able to use this to implement each of these into my dashboard.

Dash-leaflet: This is how I was able to add an interactive geolocation map.

MongoDB: Mongo was used as the database that would be manipulated and acted upon for this project and all previous iterations that I did for each of the previous modules. MongoDB is known for being file based and its ability to handle unstructured data. It is also able to be integrated into Python applications using pymongo. MongoDB allows for storage and manipulation of large mounts of data, which made it perfect for managing the animal data that was used in this project.

Challenges faced:

The biggest challenge that I faced while working on this project was authenticating username and password while attempting to instantiate the AnimalShelter() class from my CRUD\_Operations.py file. It took me a while to discover what I was doing wrong, but it turned out that I had accidentally created a username called “accuser” instead of “aacuser.” This small type kept me from being able to connect to the MongoDB to run my code. This project has highlighted the biggest challenge that I have while coding: not being diligent when typing out each line of code. I have found that I need to be very aware of each keystroke so that I can avoid these small errors in the future. By being diligent when typing code, and being aware of exactly what I am doing, I can create better projects that work as intended.

## Contact

Joseph Kawamoto