

**Grazioso Salvare Animal Shelter Web Dashboard**

**CS-340: Client / Server Architecture**

**Connor Bailey**

# Screen Shot 2025-06-29 at 1.46.10 PM.pngAnimal Shelter Web Dashboard

The purpose of this project was to help Grazioso Salvare find suitable canine candidates for search and rescue training by aggregating data from five different animal shelters in the area surrounding Austin, TX and creating a web dashboard to filter results from these databases. This application has the following features:

* A web dashboard where a user can come to begin their search
* Table containing aggregated data from the participating animal shelters
* Filtering options, allowing a user to quickly begin their search for the right dog
* Pie chart that visualizes breed distributions and updates based on the selected filters
* Interactive map displaying animal locations

# Tools

**MongoDB**

MongoDB runs the back-end data model for our web application. MongoDB’s Python interface allows us to interact directly with the database in our scripts. We can access the database on the command line with the *mongosh* command, and through the *pymongo* Python library that we import into our script.

**Dash Framework**

Dash is a Python framework used to build the front-end and handle the application logic. This framework allows us to create our UI using html and enables reactive callback functions that respond to user input, making it ideal for dashboard applications.

# Steps Taken to Complete the Project

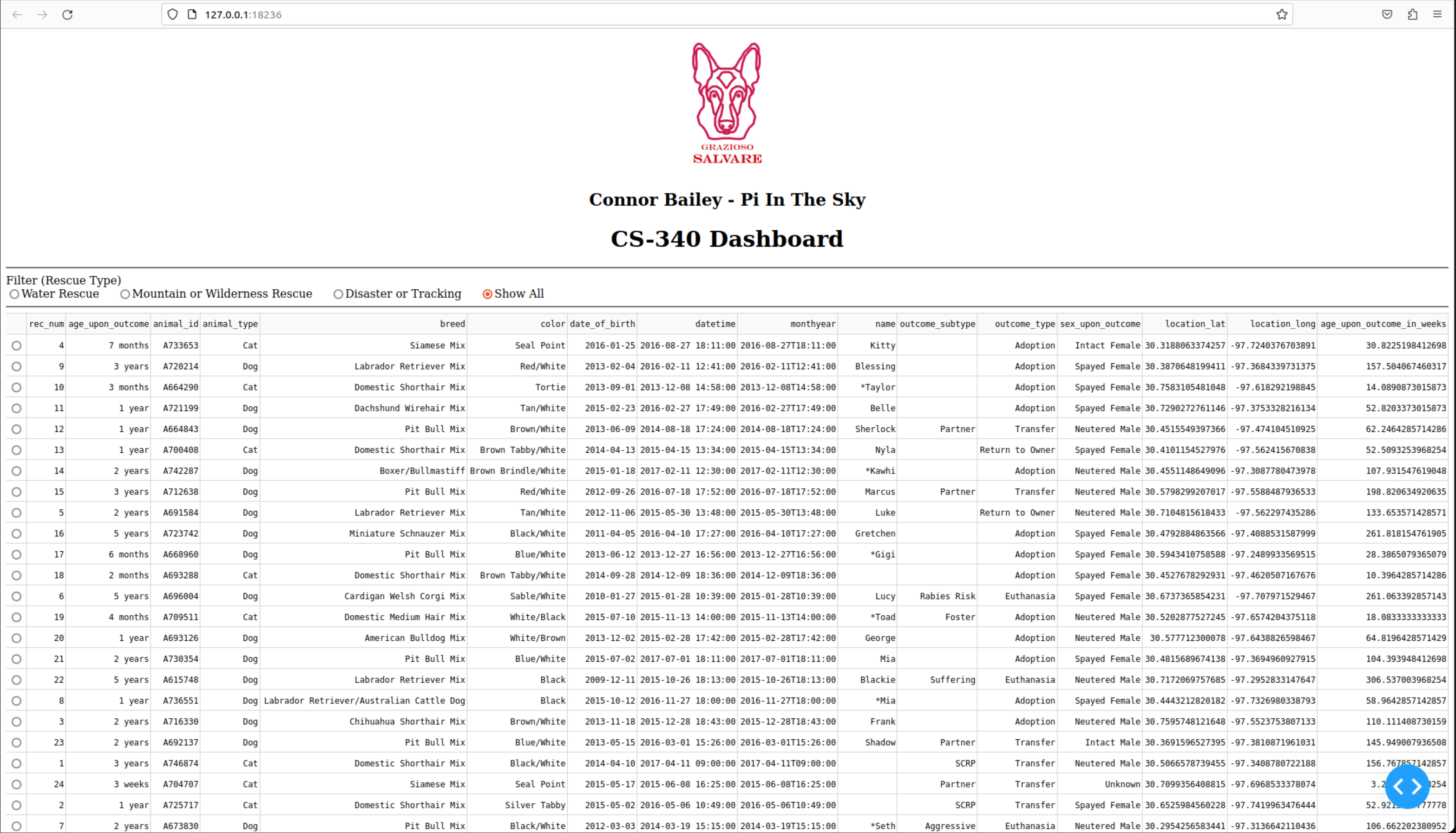
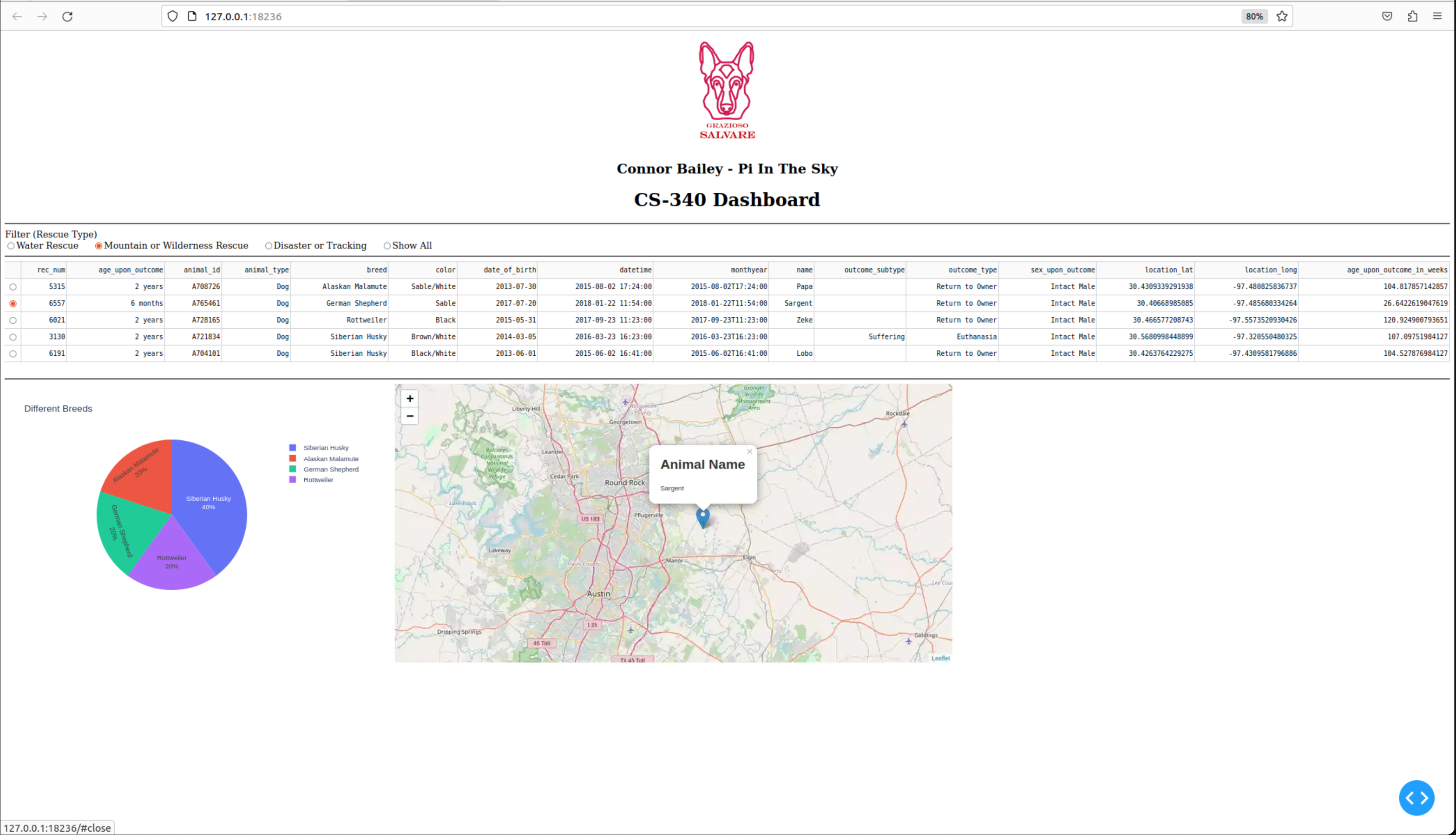
1. Set Up MongoDB
   1. Created the accuser account with appropriate permissions
   2. Imported the animal shelter data
2. CRUD Module
   1. Developed the AnimalShelter class that leverages pymongo to perform create, read, update, and delete operations
3. Dash App Layout
   1. Created web dashboard layout with the company logo, filtering tools, data table, map, and pie chart
4. Callback Functions
   1. Wrote callback functions to update the map and pie chart based on user input
5. Testing/Debugging
   1. Conducted multiple rounds of functional testing to verify the business requirements were met
   2. Resolved remaining errors and removed debugging logging

# Challenges and Solutions

During the development of the CRUD module, I was encountering an issue where the database authentication was failing when we were initializing the AnimalShelter module. In testing Python files we were able to authenticate, but when integrating the service with our Dash dashboard, authentication would always fail. The root cause of the issue came down to a small parameter naming issue. In the init() of AnimalShelter, we are expecting the parameters *user* and *password,* however, in the Dash application we were sending *username* and *password* which caused authentication to always fail, regardless of if the data contained in these parameters was correct or incorrect. By fixing the parameter naming issue and moving forward with a consistent naming scheme, I was able to overcome this challenge.

During the development of the dashboard I found another challenge. When launching the dashboard, the Dash framework was generating multiple errors regarding a NoneType. After some troubleshooting and issue isolation, I determined this issue was happening during the callback to the update\_graph() function. If an item in the table had not been selected, the function would immediately return an error. I was able to resolve this by adding additional conditional checks, and updated the return call in the update\_graph() function.

# Screenshots

****

