# CS 340 README Template

## About the Project

This project implements create and read functionality for a MongoDB database utilized by the Austin Animal Center. Developed in Python, the application allows users to insert and retrieve documents using the PyMongo library. PyMongo serves as the interface between Python applications and the MongoDB database, enabling efficient execution of CRUD (Create, Read, Update, Delete) operations. These operations are fundamental to database management and essential for maintaining and interacting with records in any modern application.

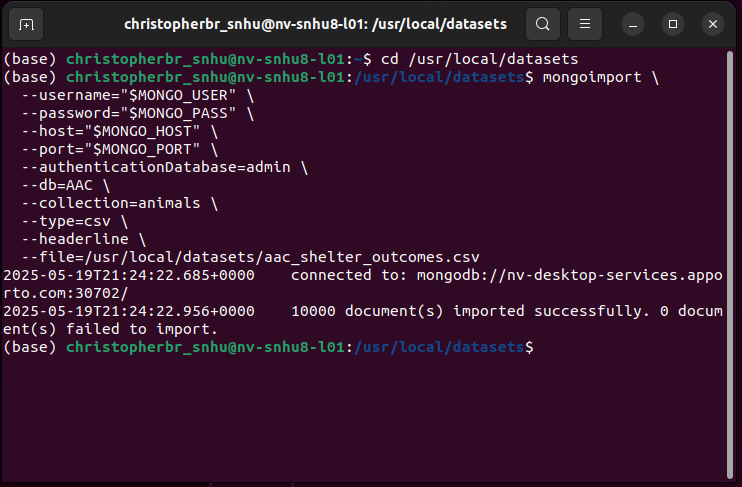
The project also features a dashboard built using the Dash framework, which provides a breed distribution chart, geolocation data for each animal, and interactive filters. These filters include water rescue, mountain or wilderness rescue, and disaster or individual tracking.

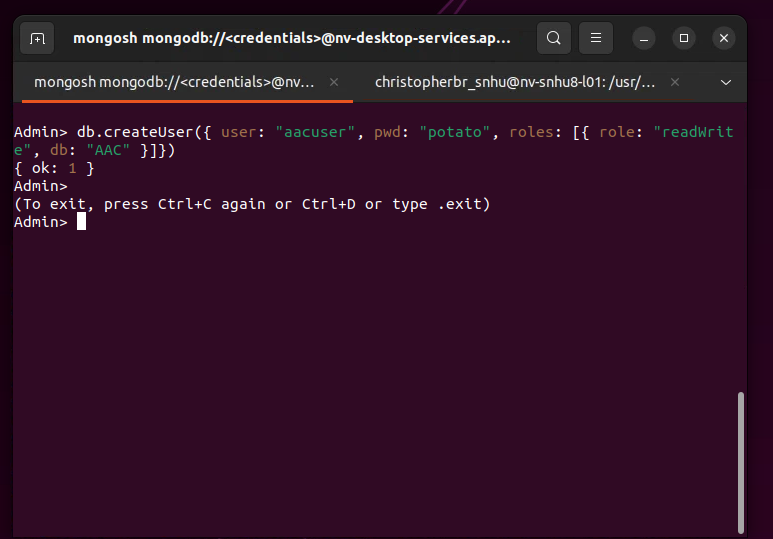
## Motivation

## This project was developed as part of an assignment focused on modular Python design and secure database connectivity. The core objective was to apply CRUD principles to interact with the animals collection within the AAC (Austin Animal Center) MongoDB database. By building a reusable and maintainable module, the project demonstrates practical usage of database abstraction in Python.

## Getting Started

1. Use the mongoimport utility to import the AAC dataset into your MongoDB instance.



1. Create a dedicated MongoDB user account with appropriate permissions for the AAC database.
2. Connect to the AAC database using the newly created credentials.
3. Place the AnimalShelter.py module in the same directory as your Jupyter Notebook.
4. Import the AnimalShelter class, and use the create() method to insert documents and the read() method to query based on attributes like name or animal type.
5. Open the notebook file, run all cells to launch dashboard.
6. Use filters to verify different breeds.

**Project Steps**

1. Set up MongoDB and user authentication.
2. Developed the AnimalShelter class for access to the database.
3. Loaded AAC shelter data into MongoDB.
4. Verified the create and read functionality via Jupyter Notebook.
5. Designed dashboard layout using Dash.
6. Implemented radio item filters and corresponding MongoDB queries.
7. Created callbacks for updating table, chart, and map.
8. Debugged layout, styling, and geolocation issues.
9. Tested and captured screenshots for each filtering scenario.

**Challenges**

Case sensitivity is important. Ensure you are using the correct upper or lower case when connecting to the database or entering your login credentials. Some of the breeds are listed as “Mix,” like “Newfoundland Mix,” which caused some filtering errors. Adding “Mix” to the search opened more options.

## Installation

Python 3.8+ – Required for running the CRUD class and Dash application

PyMongo – MongoDB driver for Python

MongoDB – NoSQL database system used to store shelter animal records

Jupyter Notebook – Interactive development environment for running and testing the application

Dash – Enables the use of interactive dashboards using Python

Dash Leaflet – Geolocation mapping with Python coding

Pandas – Used to convert MongoDB data into filtered and visualized form

Plotly Express – For creating pie or distribution charts

**Why MongoDB?**

MongoDB was utilized for its ability to scale, integration with PyMongo and Python applications, and compatibility with semi-structured data.

**Why Dash?**

Dash was utilized for its ability work create effective dashboards using Python. Using the MVC structure, dashboards can be created quickly using modular components.

## Usage

from AnimalShelter import AnimalShelter

shelter = AnimalShelter()

sample = {

"name": "Dog Test",

"animal\_type": "Dog",

"breed": "Beagle",

"age\_upon\_outcome": "2 years",

"outcome\_type": "Adoption"

}

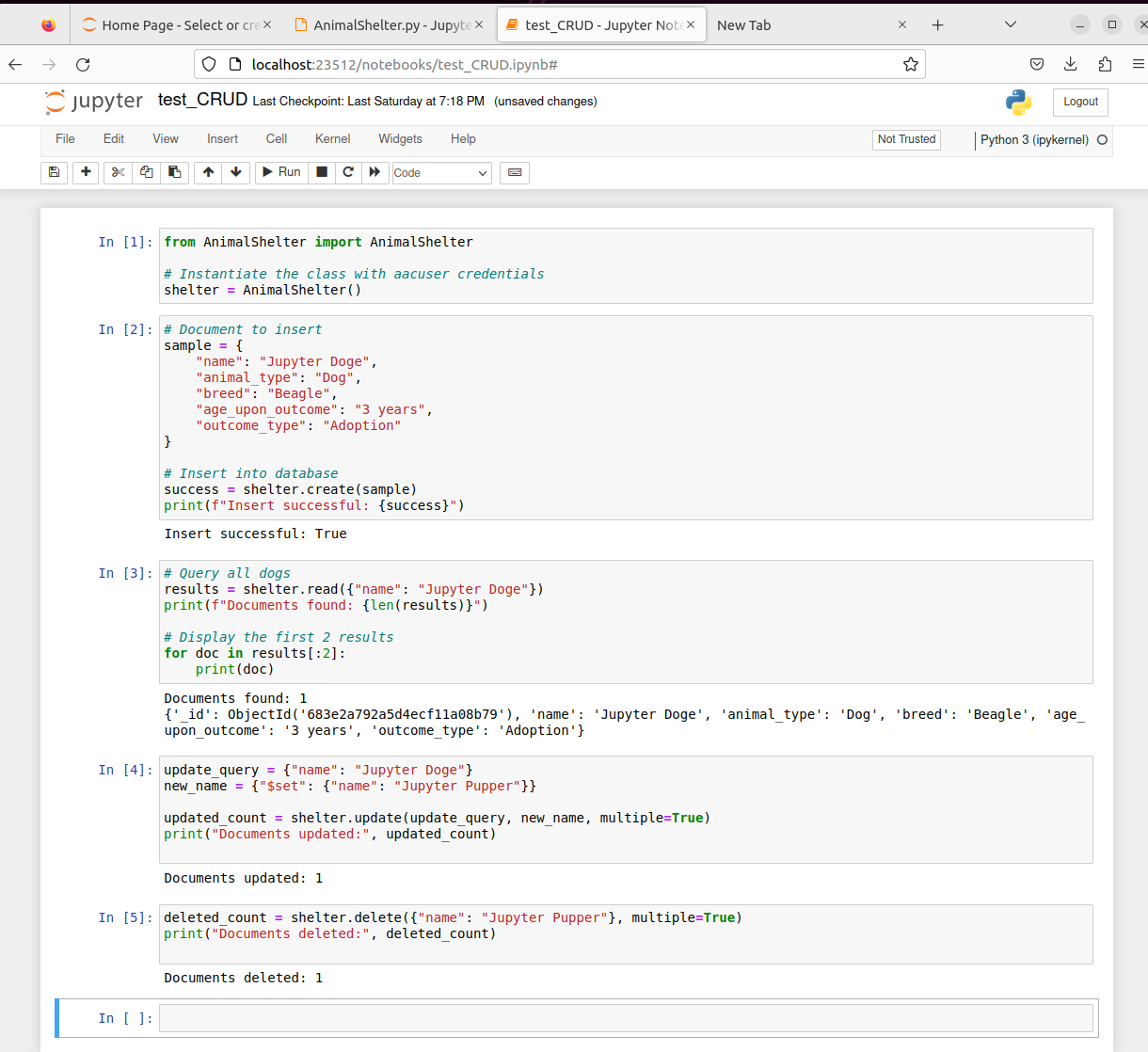
shelter.create(sample)

results = shelter.read({"name": "Dog Test"})

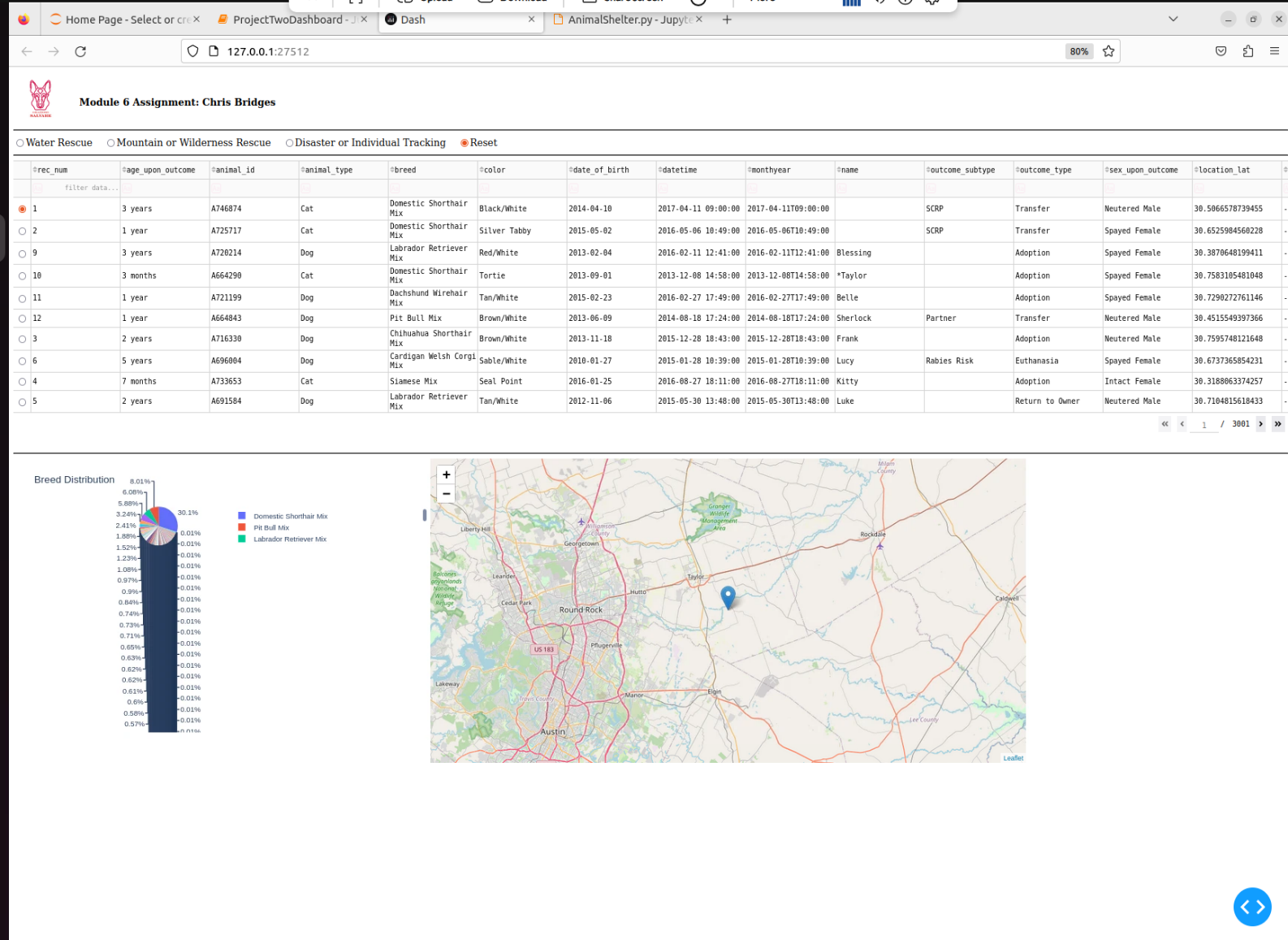
for doc in results[:2]:

print(doc)

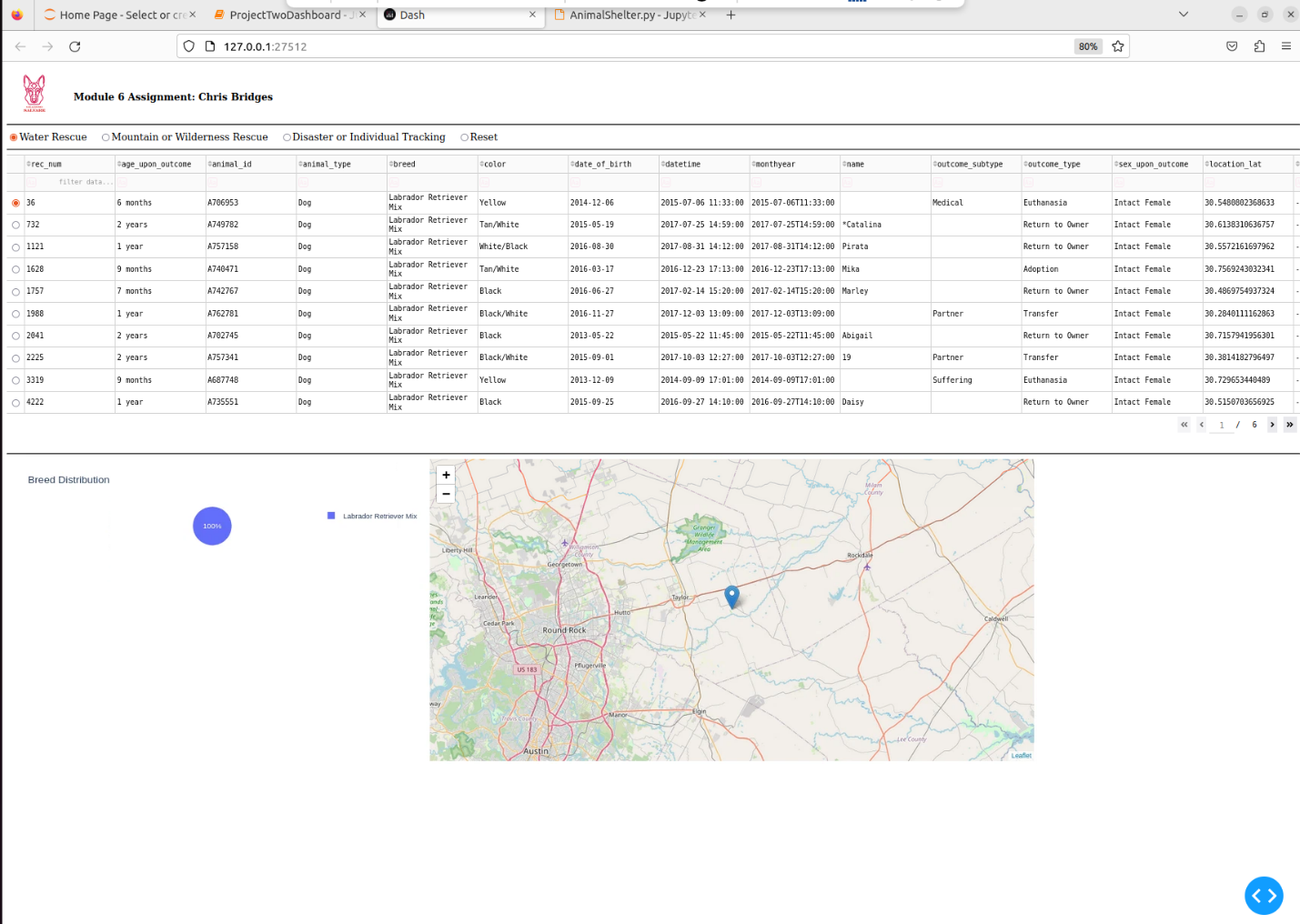
This script demonstrates how to add a test record to the database and verify its existence. Filtering by different attributes such as animal\_type or breed may return a broader set of results.



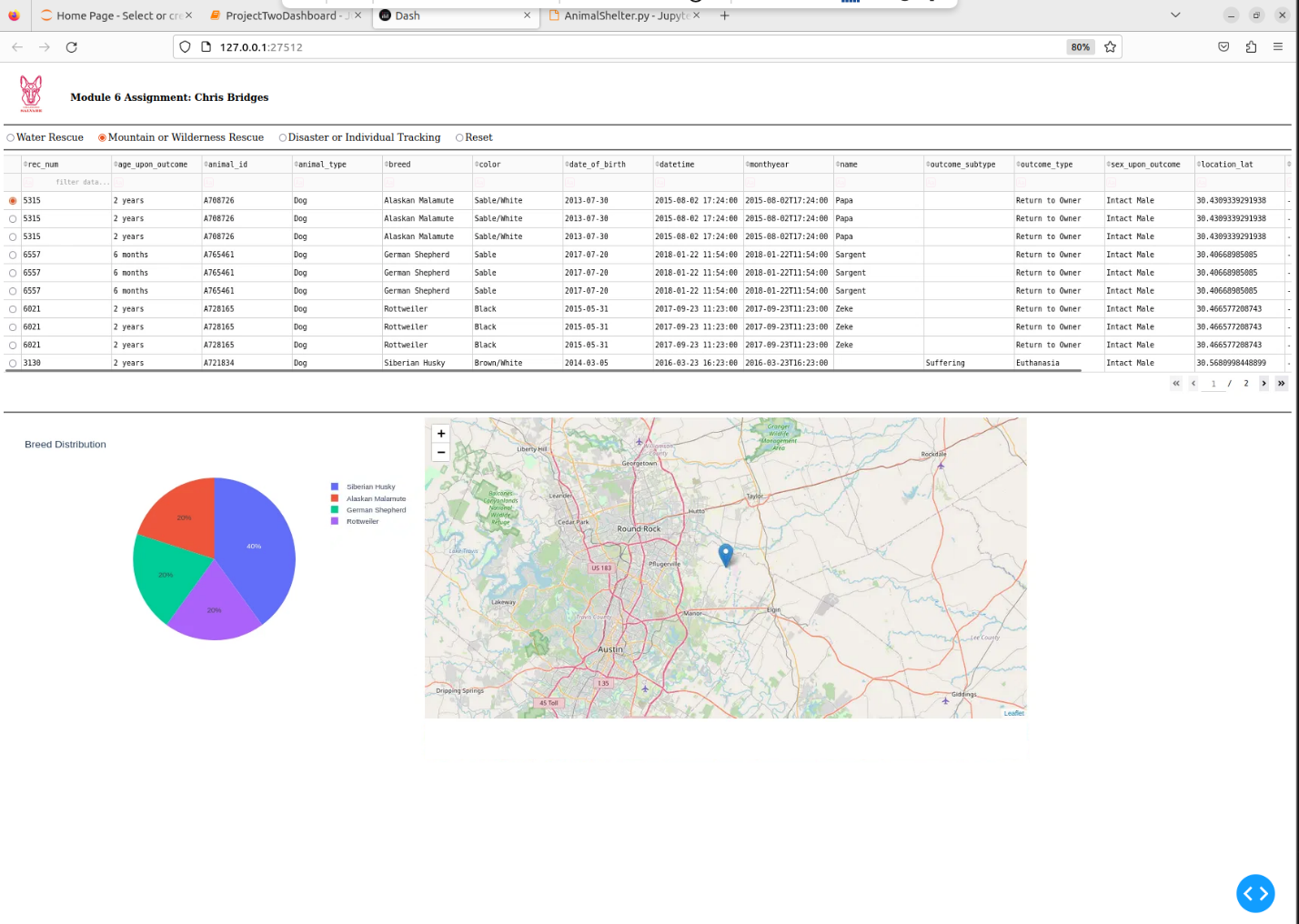
Below is an example of the unfiltered or reset state.



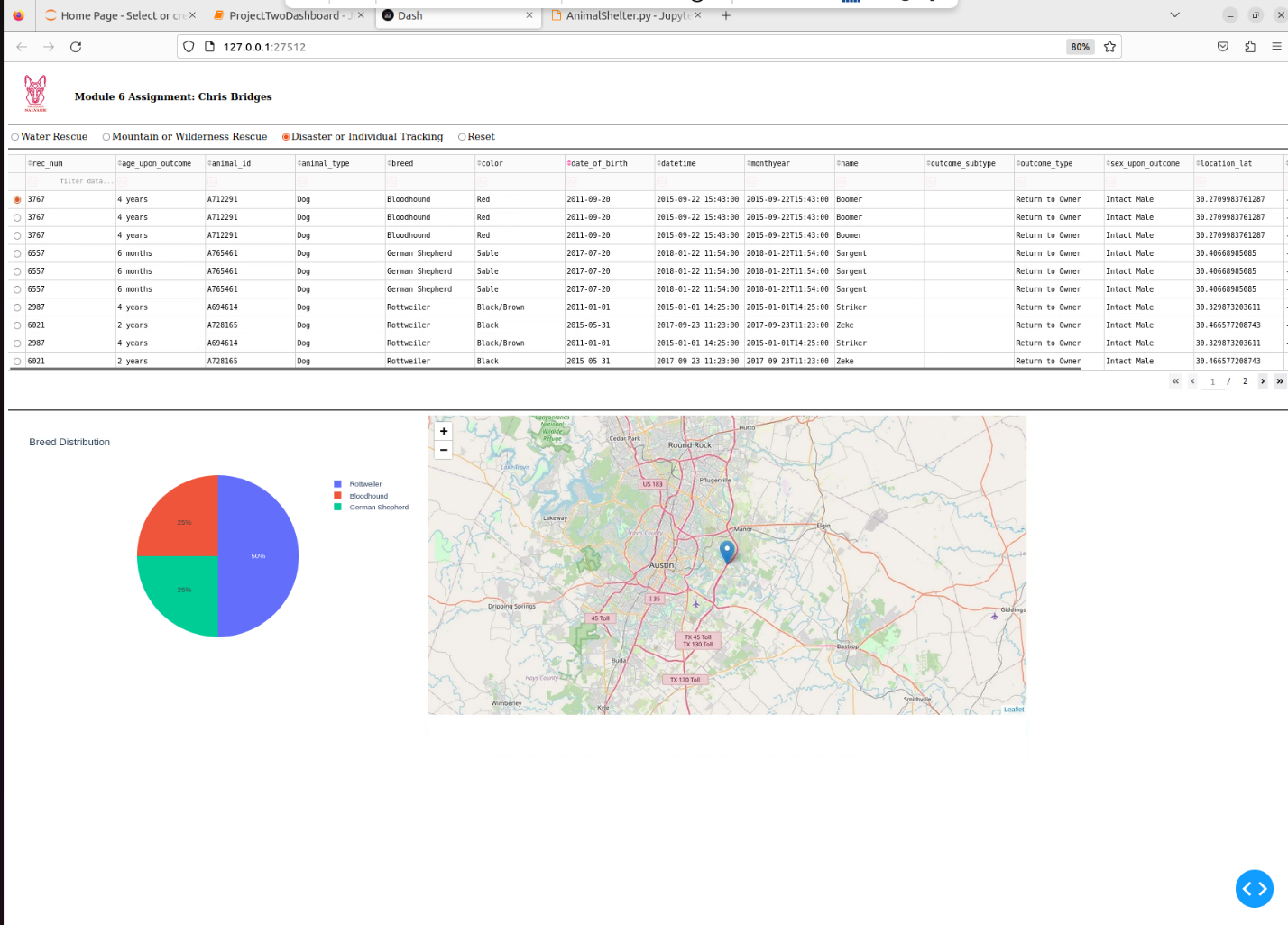
Below is an example of “Water Rescue” selected as a filter.



Below is an example of the “Mountain or Wilderness Rescue" filter.



Below is an example of “Disaster or Individual Tracking” selected as a filter.



**Resources**

**Dash -** [**https://dash.plotly.com/dash-core-components**](https://dash.plotly.com/dash-core-components)

**Dash Plotly -** [**https://dash.plotly.com/datatable**](https://dash.plotly.com/datatable)

**Jupyter Notebook -** [**https://docs.jupyter.org/en/latest/**](https://docs.jupyter.org/en/latest/)

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

**Pandas -** [**https://pandas.pydata.org/docs/**](https://pandas.pydata.org/docs/)

**Python -** [**https://docs.python.org/3.13/whatsnew/3.13.html**](https://docs.python.org/3.13/whatsnew/3.13.html)

## Contact

Chris Bridges