# CS 340 README

## About the Project/Project Title

This project is an interactive web application dashboard to help Grazioso Salvare manage its animal rescue. The dashboard integrates a create, read, update, delete, functionality with MongoDB to store and retrieve animal records. It also allows for filtering of records, data visualization, and geolocation mapping. By providing a user friendly interface Grazioso Salvare can efficiently manage and analyze its rescue operations.

## Motivation

The goal of this project is to develop a scalable and efficient data management system for Grazioso Salvare. Grazioso Salvare aims to train and rescue dogs, having a structured database is crucial in managing all records. The system allows for easy to learn filtering, optimized queries and secure authentication helping Grazioso Salvare rescue and train animals efficiently.

## Usage

The system provides full CRUD functionality for managing animal records. It can insert a new record. Retrieve existing records. Update and modify an existing record including deletion. Along with this record management it has an interactive dashboard for filtering and visualization. This dashboard includes an interactive data table that dynamically updates based on filtering that allows users to filter records based on rescue type. A pie chart is used to display breed distribution for filtered data while a geolocation map shows the location of selected animals.

## Installation

Required Tools

Python <https://www.python.org/downloads/>

MongoDB <https://www.mongodb.com/try/download/community>

PyMongo <https://pypi.org/project/pymongo/>

Jupyter Notebook <https://jupyter.org/>

Dash<https://dash.plotly.com/>

**Tools Used**

1. MongoDB was used as the database for this project due to its ability to store large amounts of data with varying attributes. It is also easy to scale and adaptable. Due to the data not having to be structured in MongoDB it was great for handling animal records with varying attributes while staying efficient. Finally, it is Python compatible allowing for seamless interaction for the CRUD operations.
2. Dash framework was chosen as the framework to develop the web based interactive dashboard. It is a python based framework that has tools needed for the interactive table, graph, and pie chart. Dash only relies on python making it accessible due to no other language needed. The addition of callbacks and reusable components like dcc.RadioItems allows for user interaction and real time updates creating a user interface that is easy to learn.

**Reproduction Instructions**

To reproduce this project you would first install Python, MongoDB, PyMongo, Jupyter Notebook, Dash and the requirements which can be done with (pip install -r requirements.txt).

You would then set up MongoDB you may need to update user credentials depending on factors like local host, username, and password. You would then import the data set into AAC.animals.

Next, run Jupyter notebook and open ProjectTwoDashBoard.ipynb and execute all the cells. A link will pop up at the bottom of the page which will take you to the interactive dashboard. You can then select different rescue filters, sort by attributes, while viewing real time updates to the table, pie chart, and map locations.

### Code Example

This is the code for creating a record:

dog = {

"animal\_id": "X123456",

"animal\_type": "Dog",

"breed": "American Bulldog",

"color": "Black"

}

insertion\_result = crud.create([dog])

print("Insertion Result:", insertion\_result)

This is the code for reading a record:

query\_result = crud.read({"breed": "American Bulldog"})

print("Query Result:", query\_result)

Code for updating a record

update\_data = {"color": "Brown"}

update\_result = crud.update({"animal\_id": "X123456"}, update\_data, multiple=True)

print("Update Result:", update\_result, "document(s) updated")

Code for deleting a record

delete\_result = crud.delete({"animal\_id": "X123456"}, multiple=True)

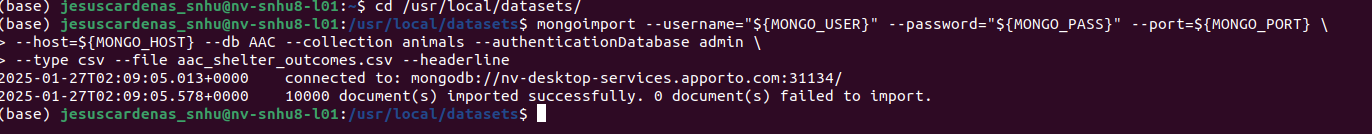
print("Delete Result:", delete\_result, "document(s) deleted")

### Tests

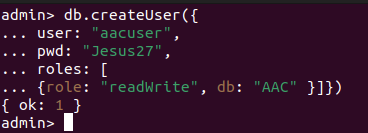
For the test I first created an animal record into the database. I then verified successful insertion by checking the return value True. I then Queried for an animal by breed and confirmed the results. Afterwards I updated an existing record and changed its color from black to brown. I then verified the field was updated. Finally I deleted the record for the animal I created to test. I then tested for functionality of interactive data table, pie chart, and geolocation map.

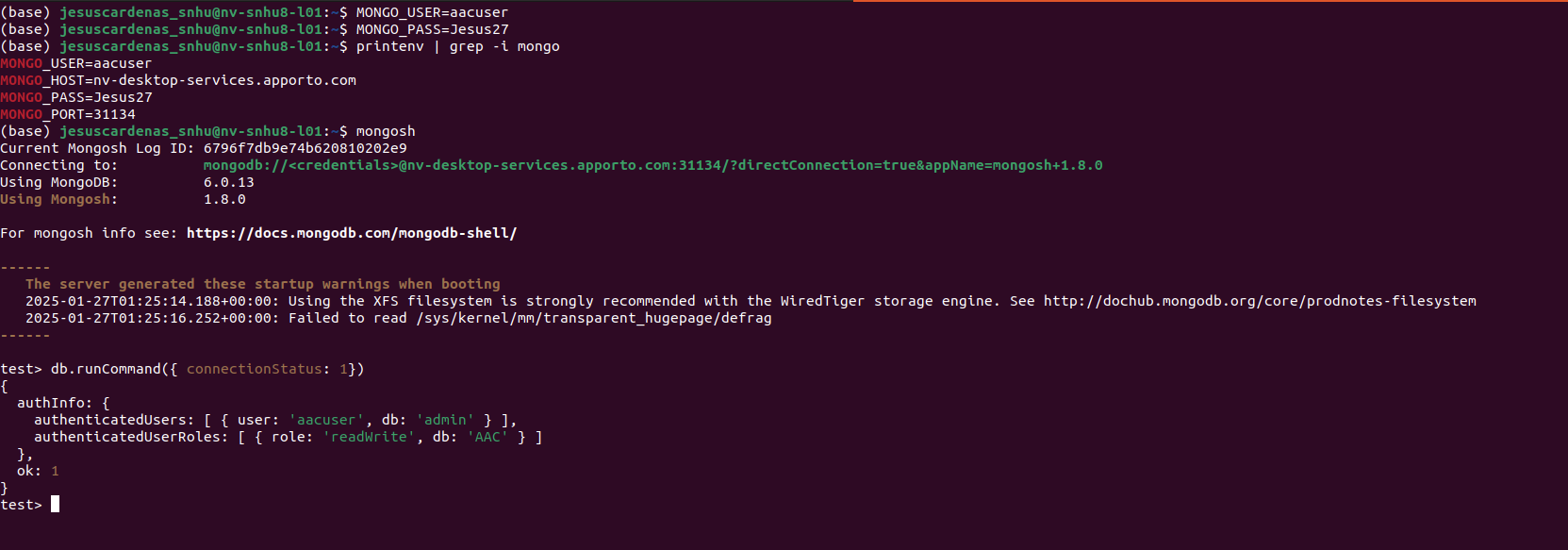
### Screenshots

1. MongoDB Import command execution



1. User Authentication in the mongo shell

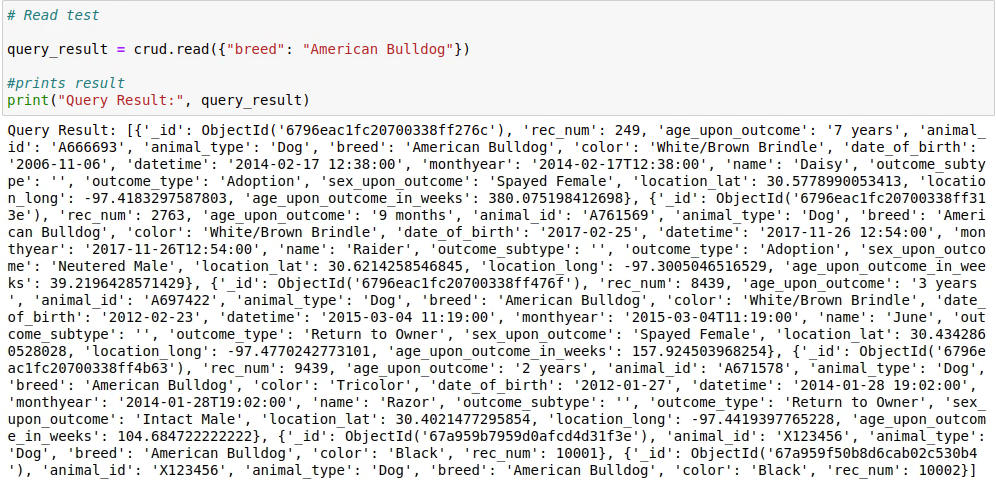




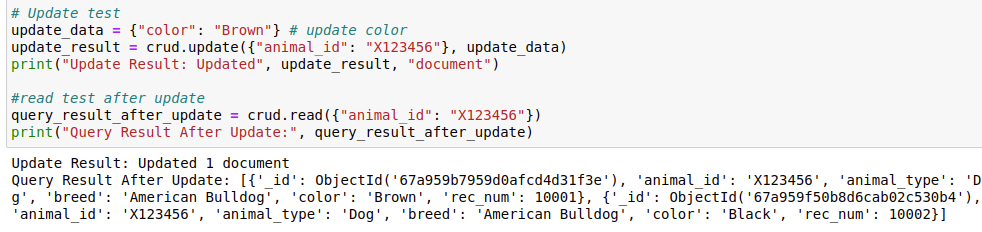
1. Create functionality test



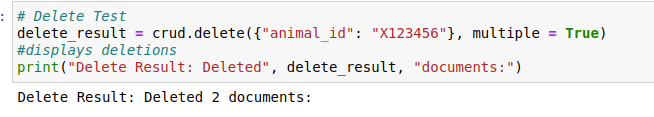
1. Read functionality test



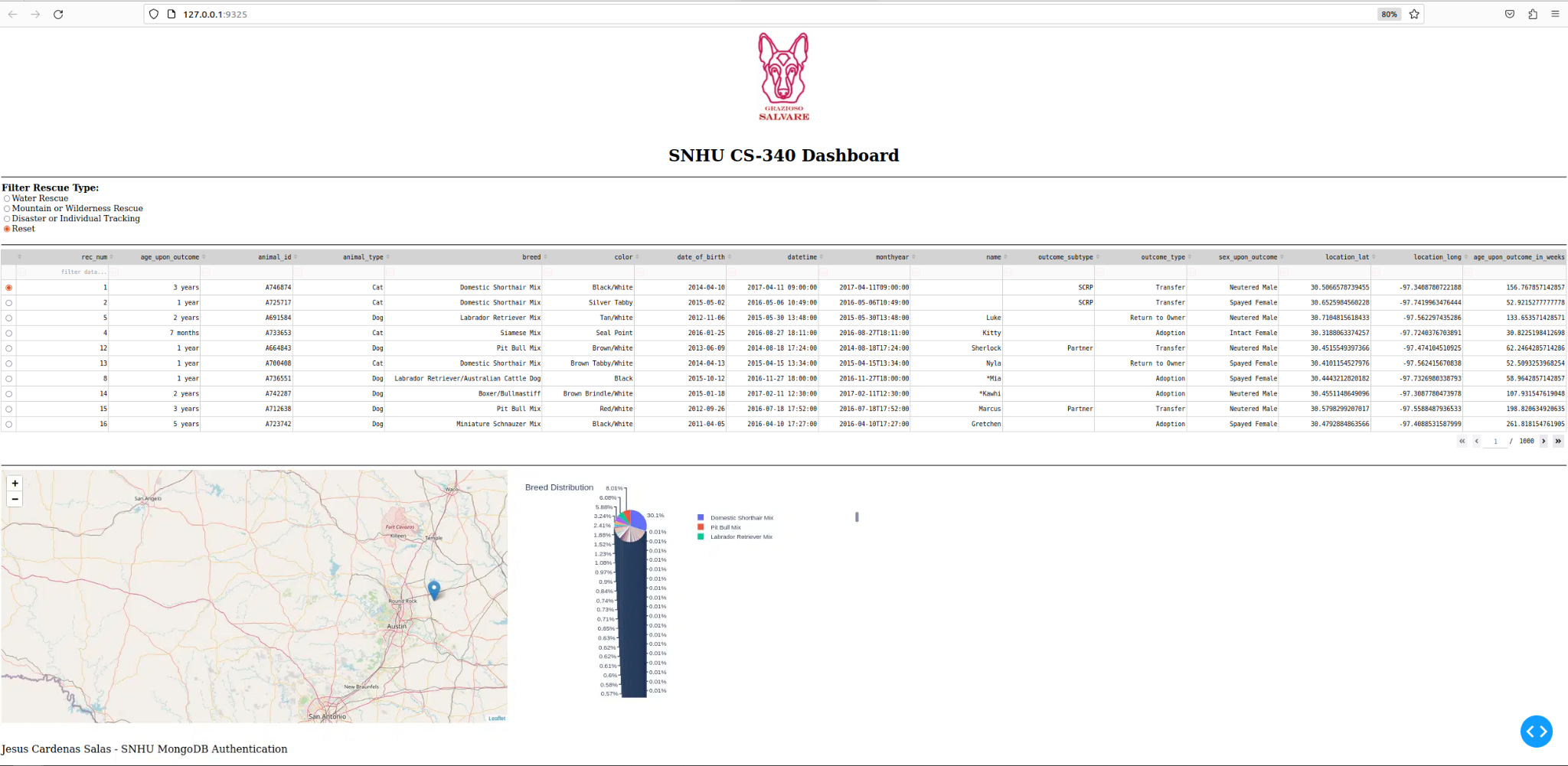
1. Update functionality test



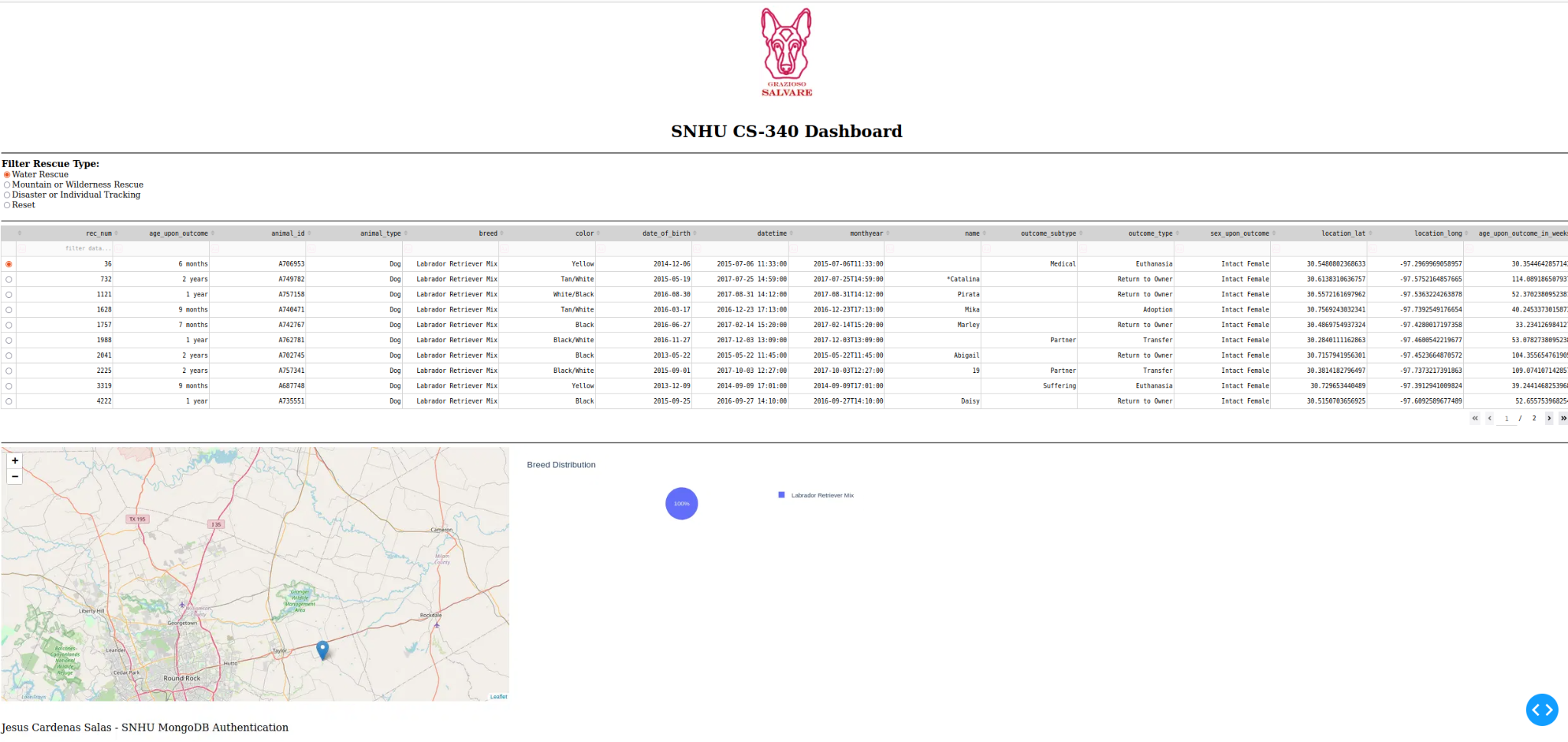
1. Delete functionality test

**

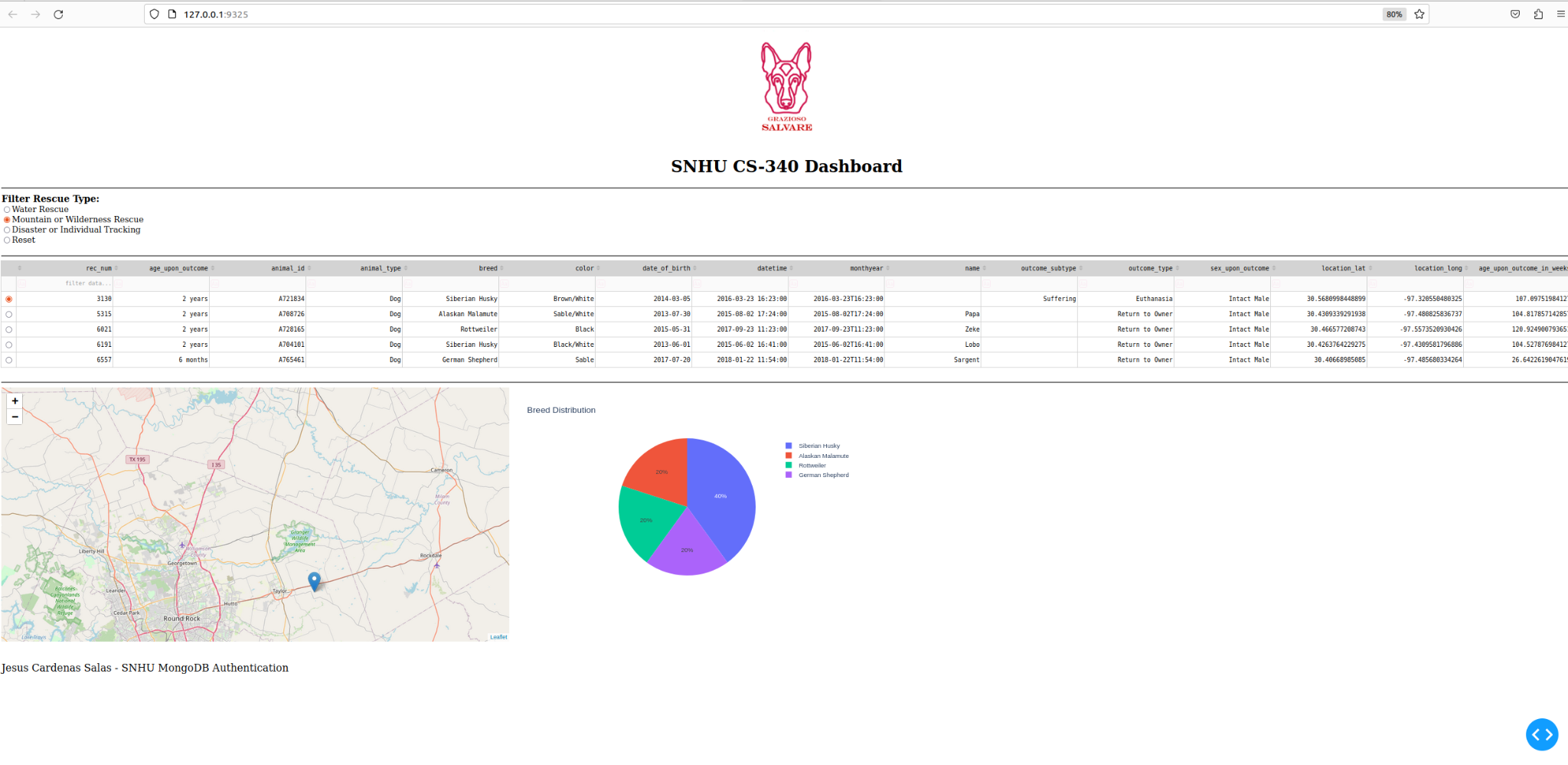
1. Dashboard initial view



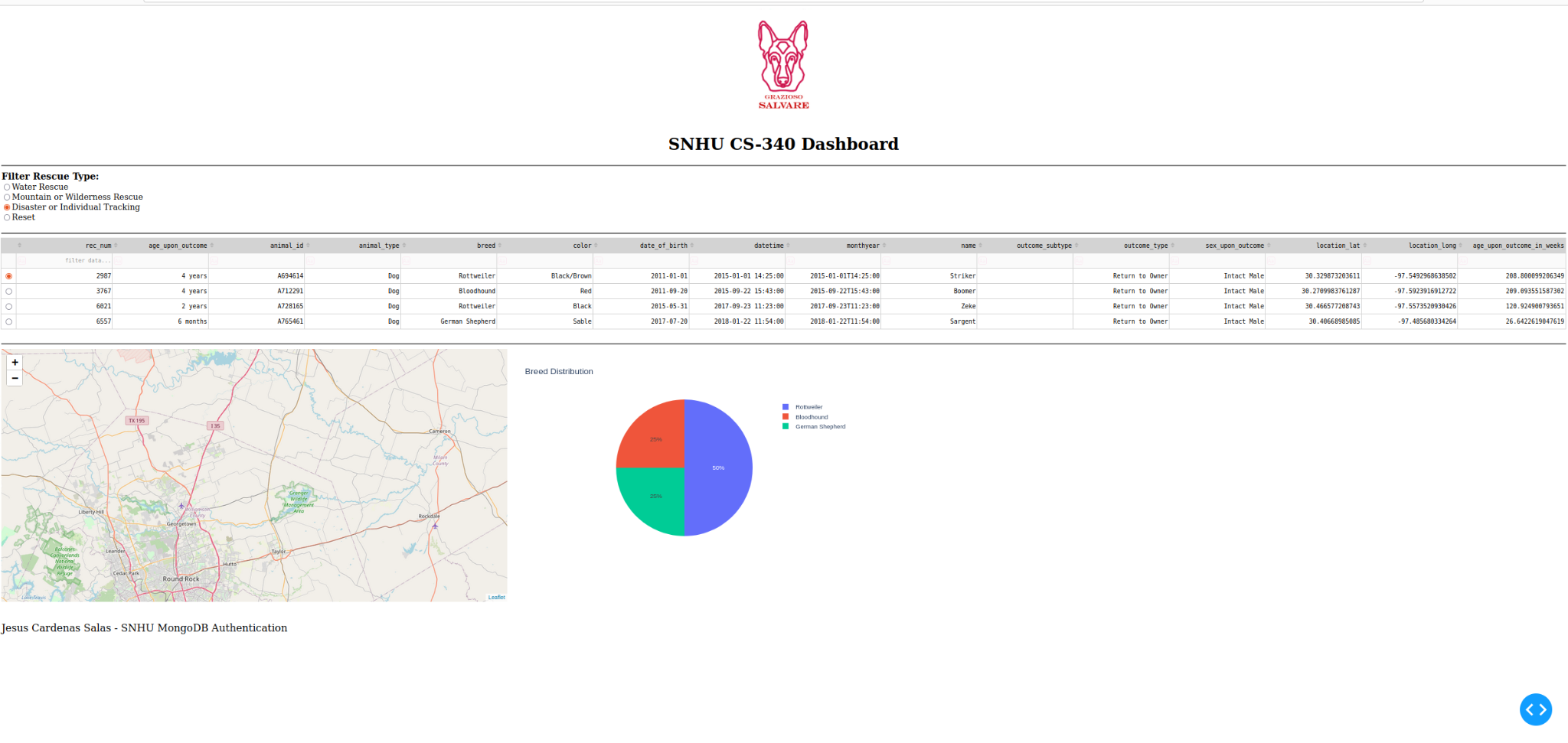
1. Water rescue filter applied



1. Mountain rescue applied



20. Disaster rescue applied



**Challenges Faced**

Initially the filtering options were not updating property, however, implementing a callback function to dynamically query and update the table using derived\_virtual\_data

When selecting reset the table would display all records, however, selecting a different filter would break the functionality of reset. The solution to this issue was adjusting the logic in the callback to return all records when reset is selected.

**Reproduction Instructions**

To reproduce this project you would first install Python and the requirements which can be done with (pip install -r requirements.txt).

You would then set up MongoDB you may need to update user credentials depending on factors like local host, username, and password. You would then import the data set into AAC.animals.

Next, run Jupyter notebook and open ProjectTwoDashBoard.ipynb and execute all the cells. A link will pop up at the bottom of the page which will take you to the interactive dashboard. You can then select different rescue filters, sort by attributes, while viewing real time updates to the table, pie chart, and map locations.

## Contact

Your name: Jesus Cardenas Salas

For questions or contributions, reach me @ cardenasjesus003@gmail.com