# CS 340 README Template

## About the Project

This full-stack project is a continuation of an application for Grazioso Salvare, a company that scouts and trains rescue dogs. The Objective is to provide a flexible and easy to use web application that will help Grazioso associates in identifying suitable candidates to serve as rescue dogs using real world shelter data.

The final deliverables include:

* A MongoDB database model
* A CRUD Python module to access the database
* A user-friendly dashboard using Dash and Plotly
* Interactive filtering and geolocation for deeper analysis

## Motivation

This project aims to provide Grazioso Salvar a robust system to help identify and train young dogs and manage this process by importing data from multiple shelters and providing CRUD (Create Read Update Delete) based access through a user-friendly Python interface. This dashboard centralizes operations and helps to optimize Grazioso Salvare’s business analysis and expand rescue efforts.

**Functionality Overview**

The complete dashboard project allows users to view an interactive data table containing all animals in the Austin Animal Center Outcomes database. Radio buttons can be used to filter dogs for different types of rescue operations which include water rescue, mount or wilderness rescue, disaster and individual tracking.

A pie chart to visualize rescue breeds provides a clear representation of table data as the user works. A dynamic map is also provided to display any animals’ location, allowing for more convenient scouting.  
The table, map, and respective pie chart update based on your currently selected data, providing interactive experience.

**Tool Used**

MongoDB

https://www.mongodb.com/

MongoDB was used as the model component because it is a popular and well documented NoSQL database which will allow the flexible storage of Animal Shelter Records in a document-oriented format. It is well suited for the semi-structured data we might encounter, allowing for diverse fields for properties like breed, age, and sex, which might vary naturally across different entries.

MongoDB also supports the most popular programming languages, including Python through the PyMongo library. This library enables efficient database queries through the friendly language of Python, making it perfect for a small and fast team.

PyMongo

https://pymongo.readthedocs.io/en/stable/

PyMongo gives us a simple Python API for interacting with MongoDB. In other words, it lets Python talk to the database. It allows the CRUD module to perform database operations like reading all the documents or applying filtered queries to get useful and specific data.

Dash Framework

https://dash.plotly.com/

Dash was selected to build the view and controller portion of this project. This is because it combines the frontend and backend development portion with Python, making it easier for small teams to maintain powerful dashboards. Dash automatically binds different components like graphs and tables to the backend logic through a callback system.

JupyterDash was used inside a Jupyter Notebook to prototype and run the dashboard. This provided a convenient environment to iteratively progress through the project while being able to quickly see test results.

## Steps Taken to Complete the Project

1. Make sure you have a working development environment that supports Python and MongoDB.
2. Import the shelter data using the following command:

mongoimport --username=aacuser –pasword=letmelearn1 –host=${MONGO\_HOST} –port=${MONGO\_PORT} –authenticationDatabase=admin –db=AAC –collections=animals type=csv –headerline aac\_shelter\_outcomes.csv

Where ${MONGO\_HOST} and ${MONGO\_PORT} are the host and port values for your Mongo DB instance.

1. Connect the CRUD module to the dashboard:
   1. Retrieved full documents use the read({}) method.
   2. Dropped MongoDB’s \_id field to avoid issues when loading the data into Dash tables.
2. Design the dashboard layout:
   1. Added a data table component and hooked it up to the search navigation with paging and sorting.
   2. Embedded the Grazioso Salvare logo and my name.
3. Implement dynamic filter controls:
   1. I created a radio item for each rescue category.
   2. Applied corresponding MongoDB queries such as $and and $in based on breed, age, and gender.
4. Build interactive components:
   1. Linked the radio buttons to the data table, pie chart, and map via Dash callbacks. This allows the dashboard to become dynamic and interactive.
   2. Updated the pie chart dynamically based on the selected dog’s coordinates.
5. Debugging and adjustments:
   1. Fixed pie chart issues by calculating the breed counts.
   2. Adjusted map handling to avoid crashes when data was missing or not selected.
   3. Base64-encoded the image to fix the broken image display.
6. Tested all filters:
   1. Verified the correct functioning of each rescue type selection.
   2. Verified reset behavior properly reset table and visualizations.
   3. Took screenshots of each outcome.

## Challenges

Along the way I experienced some challenges, such as MongoDB crashing from the \_id field. To fix this, I dropped the \_id after retrieval. Another challenge I faced was empty callbacks on startup of the application. This was because I was not doing proper validation checks on the viewData.

Another common challenge I faced was a glitch with empty or filtered datasets. My pie chart would begin to glitch and become unreadable. This was a combination to my implementation as well as the speed of the virtual environment. I overcame this by pre aggregating breed counts using .valuue\_counts().

The final challenge was just making sure all the moving pieces were connected properly. This can be a struggle the bigger a project gets. Poor connections can cause components to become out of sync or stop rendering all together. To overcome this, I made sure to diligently ensure my callbacks were populated and bug free.

## Installation

Tools and libraries required for this assignment:

* MongoDB – a document based NOSQL database which will store the animal data

https://www.mongodb.com/

* PyMongo – a python-based driver to interact with MongoDB

https://pymongo.readthedocs.io/en/stable/

* Dash – Python framework for building web apps

https://dash.plotly.com/

* Dash Leaflet – Geolocation chart library based on Leaflet.js

<https://www.dash-leaflet.com/>

* Plotly Express - A simplified graphing library for easier visualization

https://plotly.com/python/plotly-express/

* Jupyter Notebook – used to test the functionality of the script

https://jupyter.org/

* Terminal – to execute basic commands

**Usage**

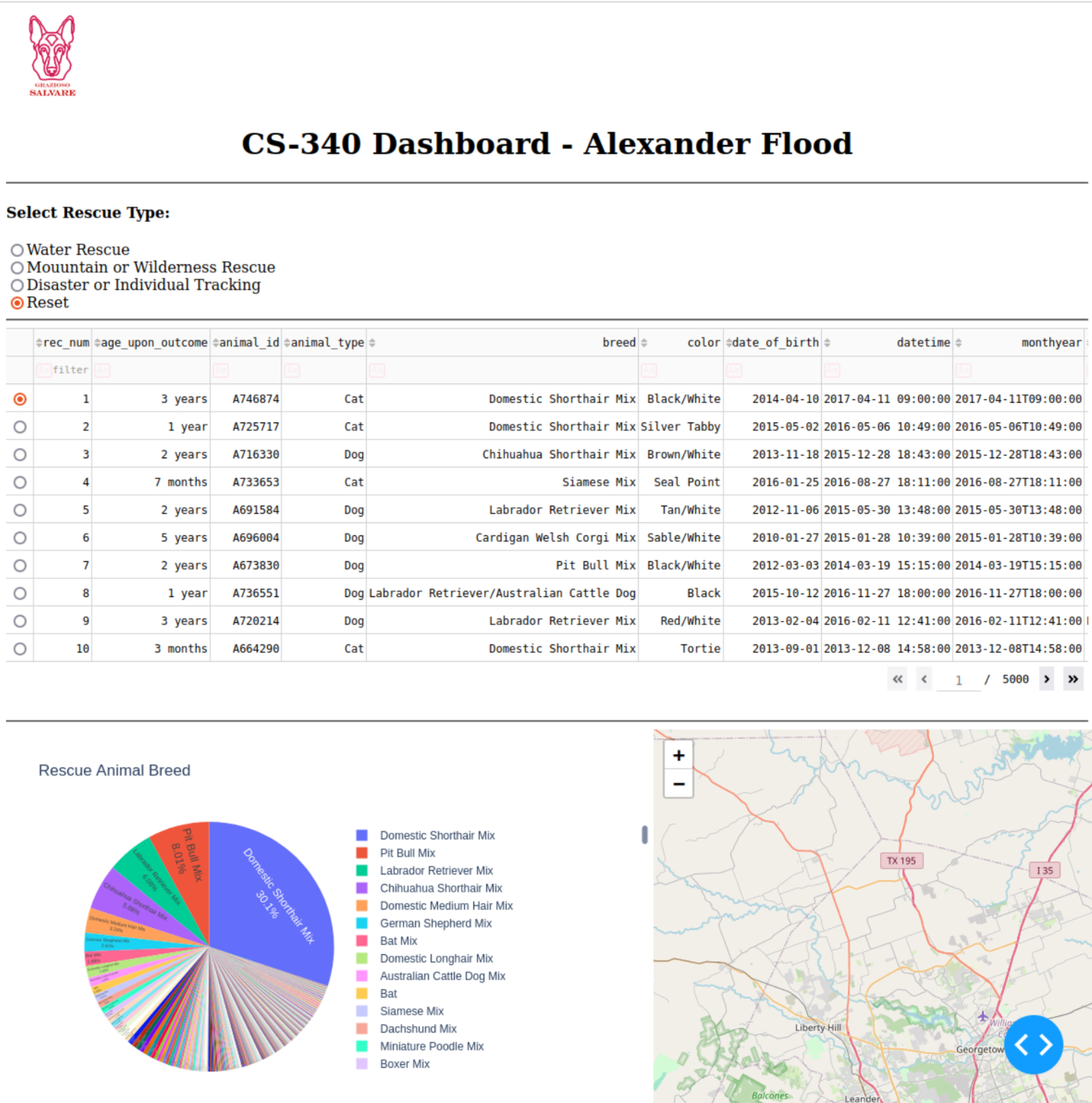
1. Launch MONGO DB and ensure the database and collection are accessible. If you need to import the data, you can do so by using the following command:

mongoimport --username=aacuser –pasword=letmelearn1 –host={$MONGO\_HOST} –port={$MONGO\_PORT} –authenticationDatabase=admin –db=AAC –collections=animals type=csv –headerline aac\_shelter\_outcomes.csv

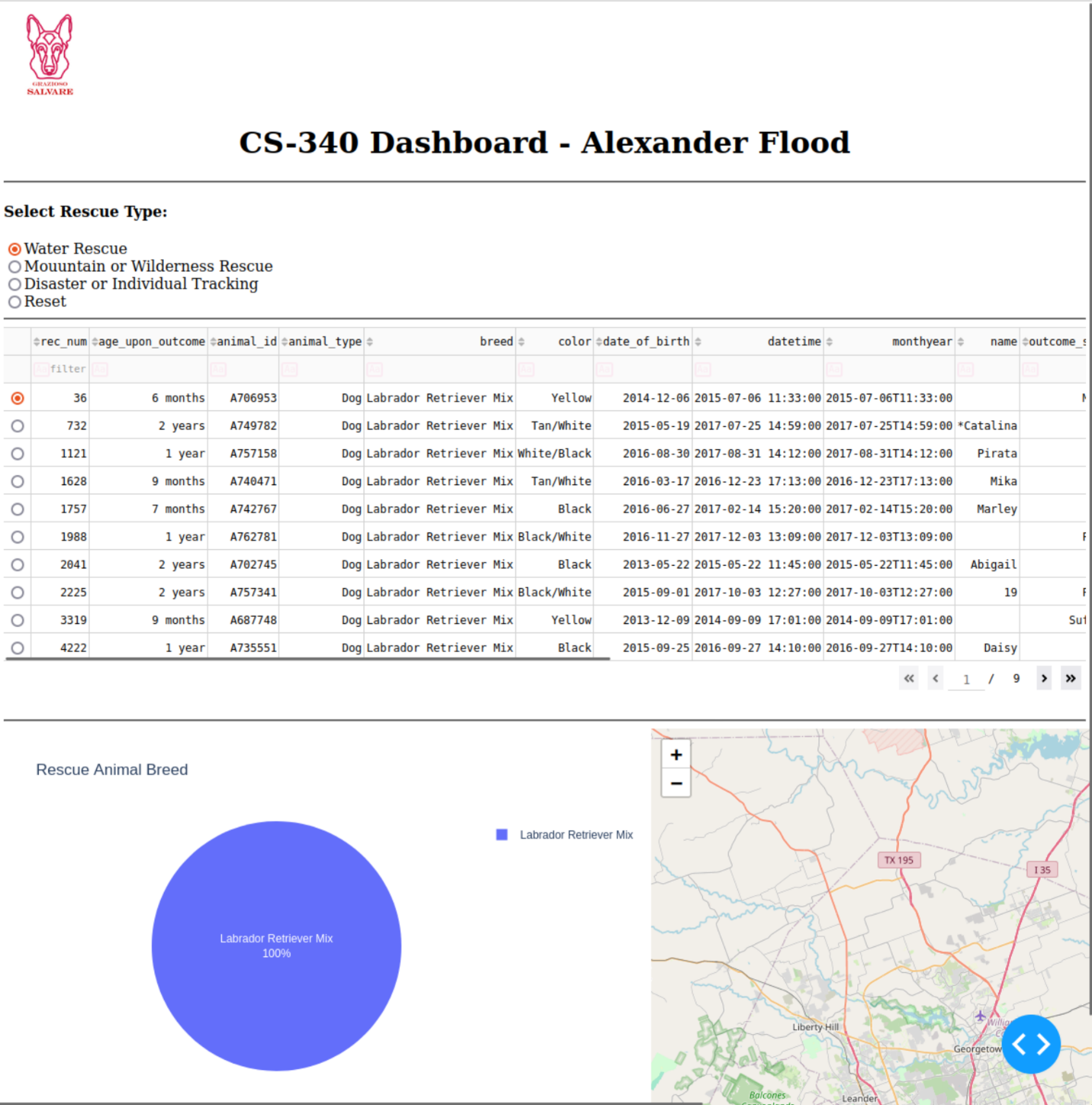
1. Open the ProjectTwoDashboard.ipynb
2. Ensure the AnimalShelter class is imported from the my\_project module
3. Run all cells to start the dashboard and begin analysis.
4. Use the radio buttons to filter by rescue type:
   1. Water Rescue
   2. Mountain or Wilderness
   3. Disaster or Individual Tracking
   4. Reset
5. Interact with the data table, pie chart, or map.

The Dashboard will automatically refresh when new input occurs.

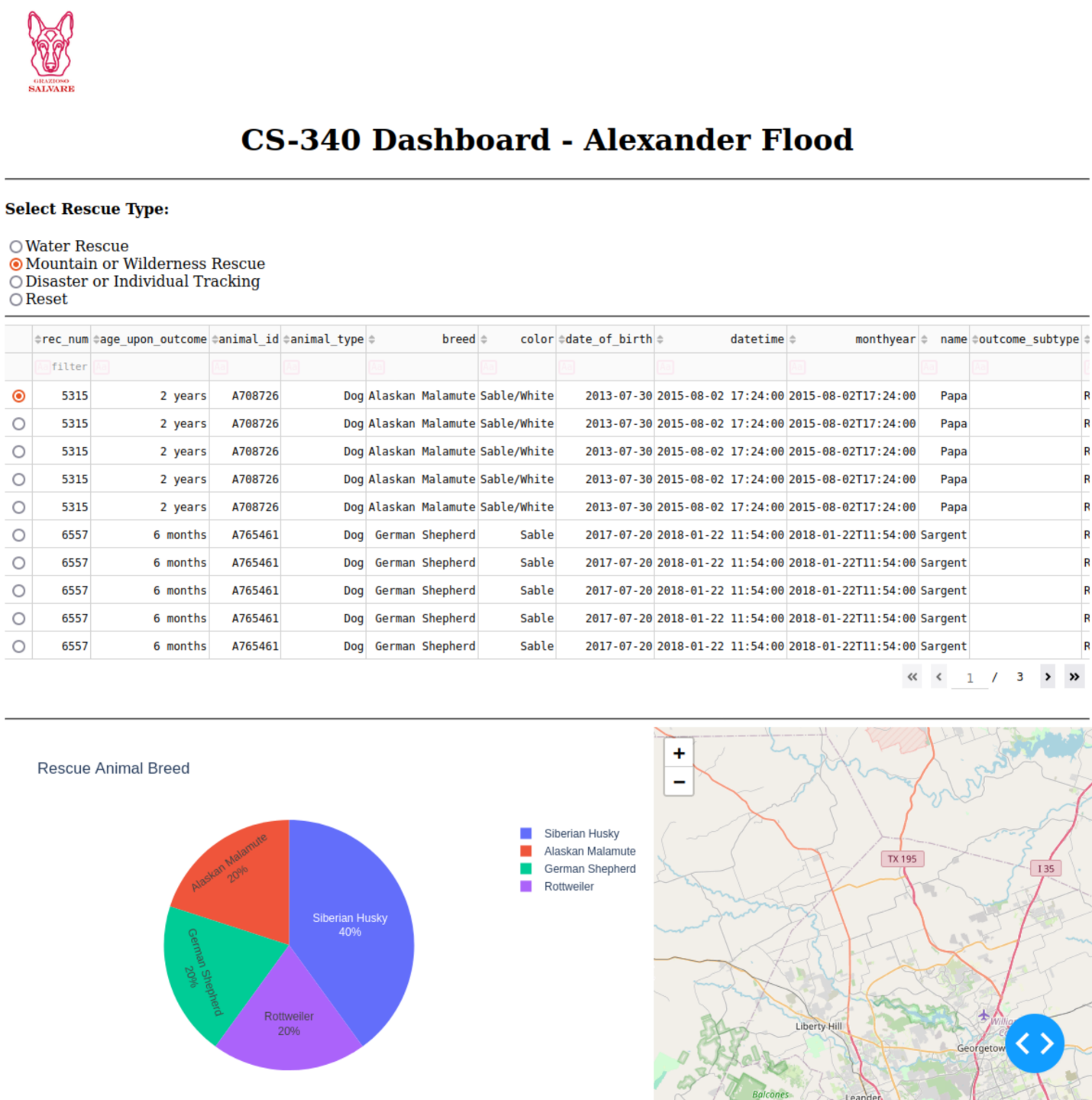
### Tests (Screenshots)

**Unfiltered**

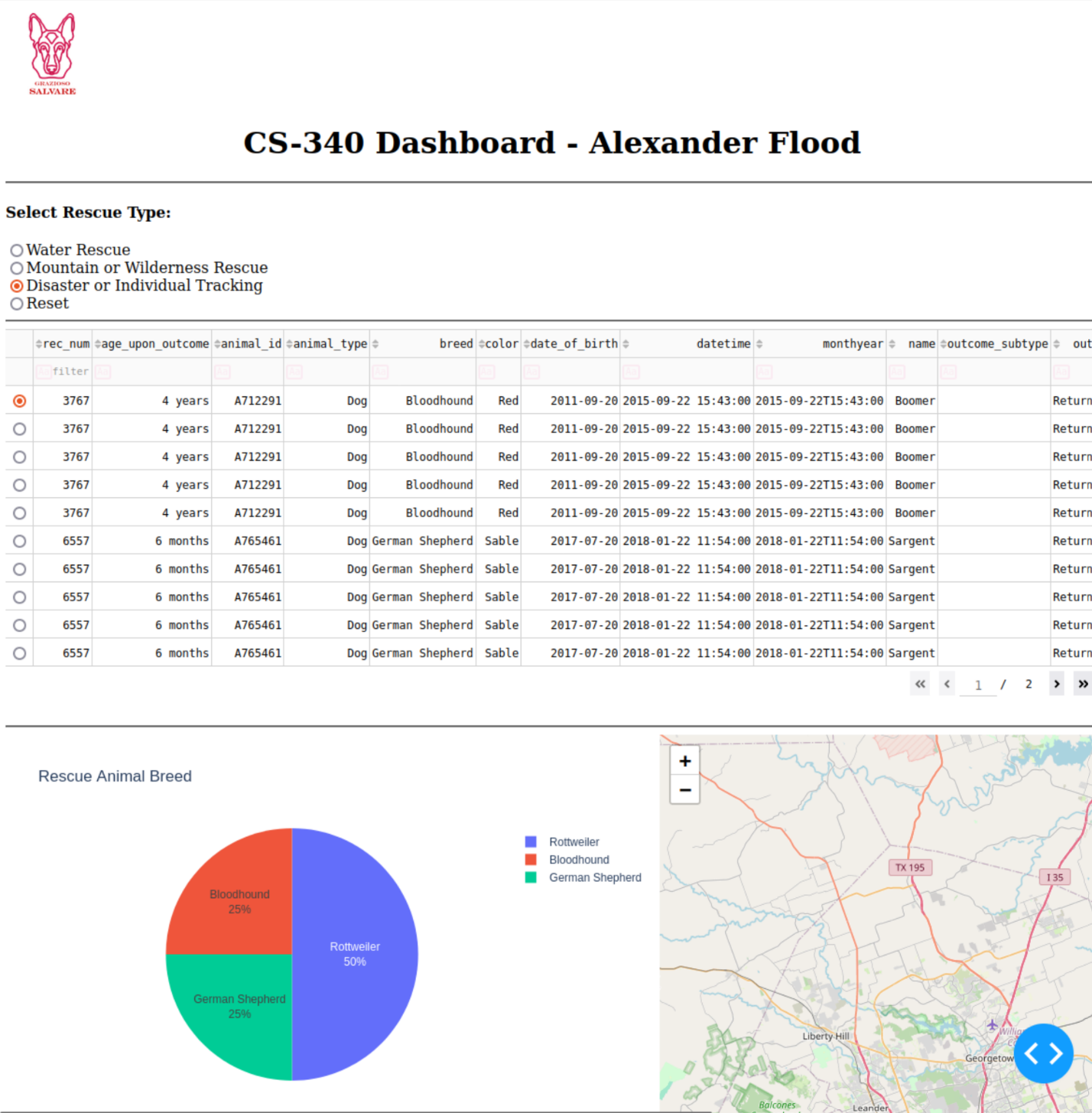
**Water Rescue**

****

**Mountain or Wilderness Rescue**

****

**Disaster or Individual Tracking**

****

## Contact

Your name: Alexander Flood