# **READ ME**

## **About the Project/Project Title**

This project is a full-stack dashboard application for Grazioso Salvare, built with Python, MongoDB, and Dash. It integrates the previously developed animal\_shelter.py CRUD module (Project One) with an interactive dashboard (Project Two) that allows the client to filter, visualize, and explore the Austin Animal Center (AAC) animal outcomes dataset.

Refer to project one READ ME for the update, add, remove functions.(included in file)  
  
 The dashboard provides:  
 - Interactive filtering options for identifying search-and-rescue candidate dogs  
 - An interactive data table connected to the MongoDB backend  
 - A dynamic geolocation map showing the selected animal’s location  
 - A bar chart visualization of breed distribution within the filtered dataset  
 - Branding with the Grazioso Salvare logo and developer identifier

## **Motivation**

The motivation behind this project is to provide Grazioso Salvare with a user-friendly, intuitive interface to identify and categorize dogs suitable for search-and-rescue training. By combining MongoDB with a Dash-based dashboard, the solution reduces training time, minimizes errors, and enables quick insights into the available shelter data.

## **Getting Started**

1. Import the AAC dataset into MongoDB using the mongoimport tool.  
  
 2. The database name is AAC, and the collection name is animals.  
  
 3. Two user accounts exist for authentication:  
  
 • aacuser with readWrite permissions (password: snhu1234)  
 • An admin account (separately managed).  
  
 4. The Python module animal\_shelter.py provides CRUD operations on the AAC collection.  
  
 5. The Jupyter Notebook file ProjectTwoDashboard.ipynb runs the dashboard.

## **Installation**

The installation requires:  
 • Python 3.x  
 • MongoDB (server running, with AAC dataset imported)  
 • Jupyter Notebook / JupyterLab  
 • Required Python libraries:  
 - pymongo  
 - pandas  
 - dash, jupyter-dash, dash-leaflet, plotly  
 - matplotlib, numpy

## **Usage**

The dashboard supports both CRUD functionality and data visualization.  
  
 1. CRUD Module (animal\_shelter.py)  
 - Create, Read, Update, and Delete methods for interacting with MongoDB.  
  
 2. Dashboard (ProjectTwoDashboard.ipynb)  
 - Load the dashboard in Jupyter Notebook.  
 - Use the radio buttons to filter by:  
 • Water Rescue  
 • Mountain or Wilderness Rescue  
 • Disaster or Individual Tracking  
 • Reset (all dogs)  
 - View the filtered results in the interactive data table.  
 - Explore the bar chart for breed distribution.  
 - Select a row in the table to highlight the dog’s location on the map.

## **Code example**

Filter Query Building

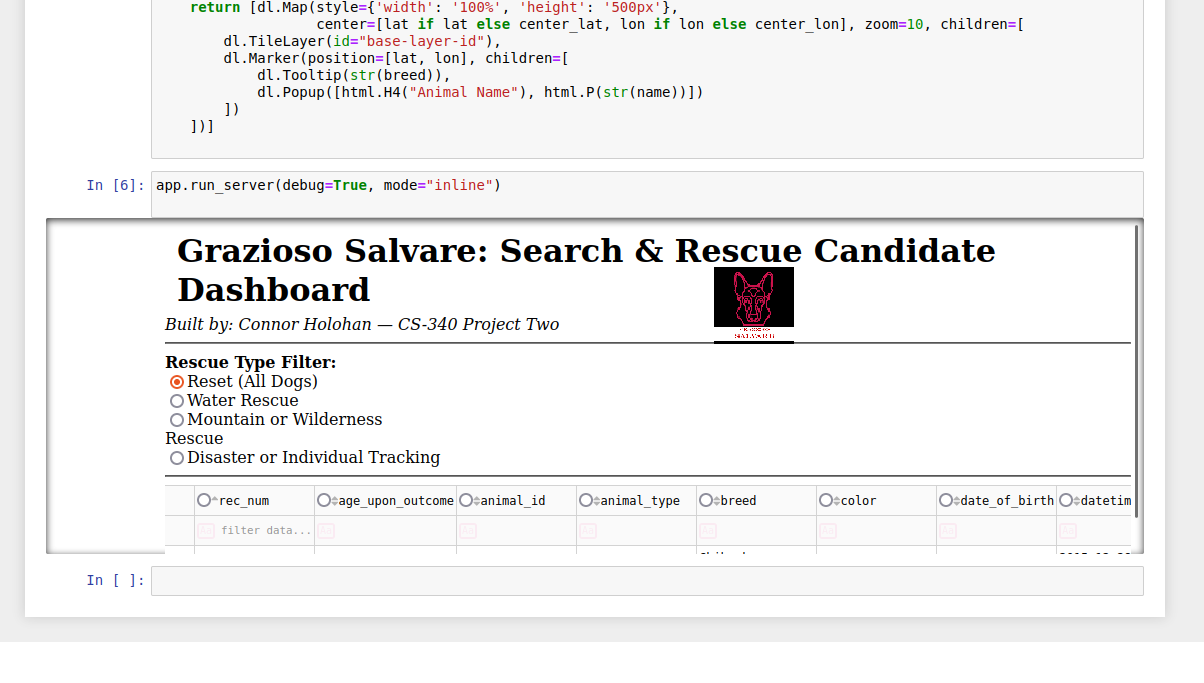
def build\_mongo\_query(rescue\_type):  
 if rescue\_type is None or rescue\_type == "Reset":  
 return {"animal\_type": "Dog"}  
 spec = RESCUE\_SPECS.get(rescue\_type)  
 breeds = spec["breeds"]  
 sexes = spec["sex"]  
 lo, hi = spec["age\_weeks"]  
 query = {  
 "$and": [  
 {"animal\_type": "Dog"},  
 {"breed": {"$in": breeds}},  
 {"sex\_upon\_outcome": {"$in": sexes}},  
 {"age\_upon\_outcome\_in\_weeks": {"$gte": lo, "$lte": hi}}  
 ]  
 }  
 return query

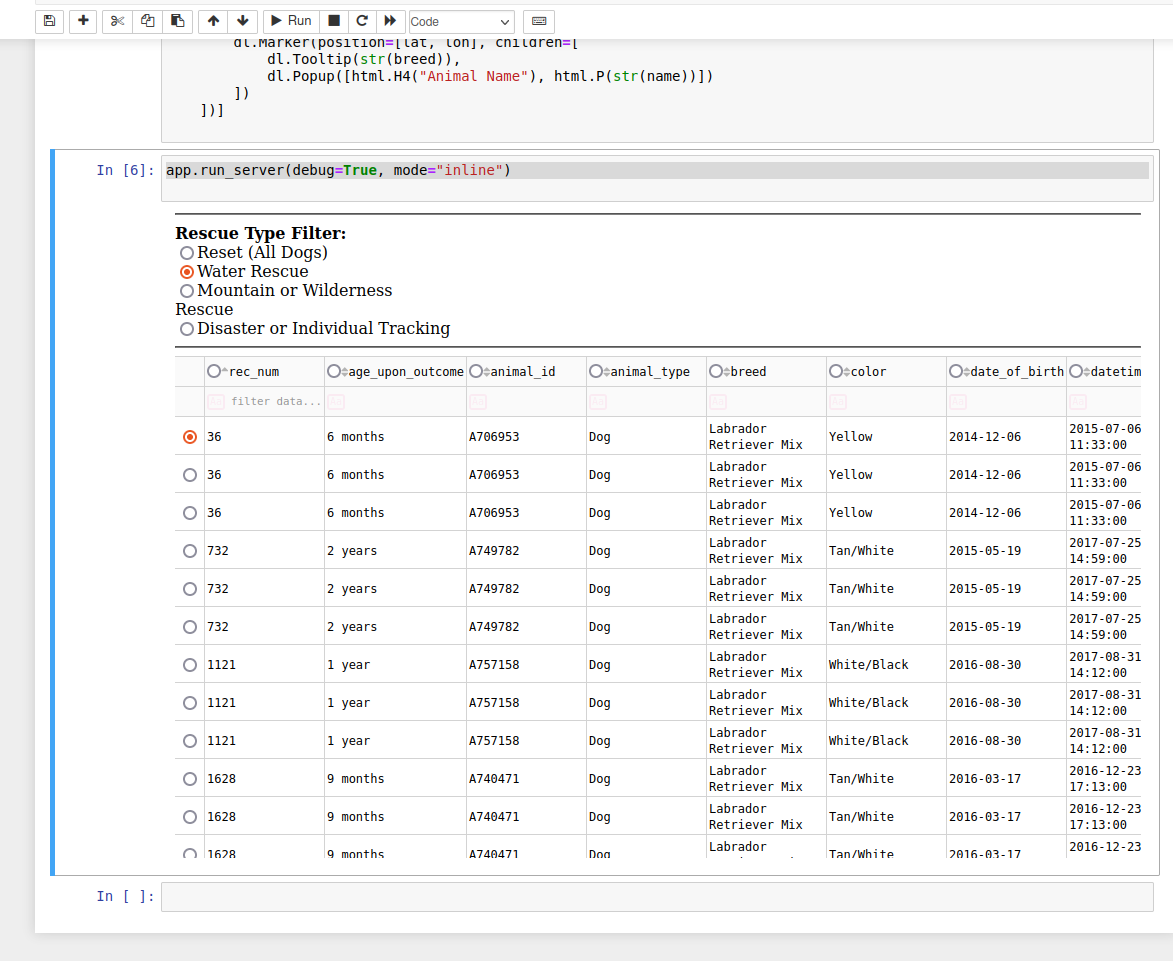
Interactive Graph

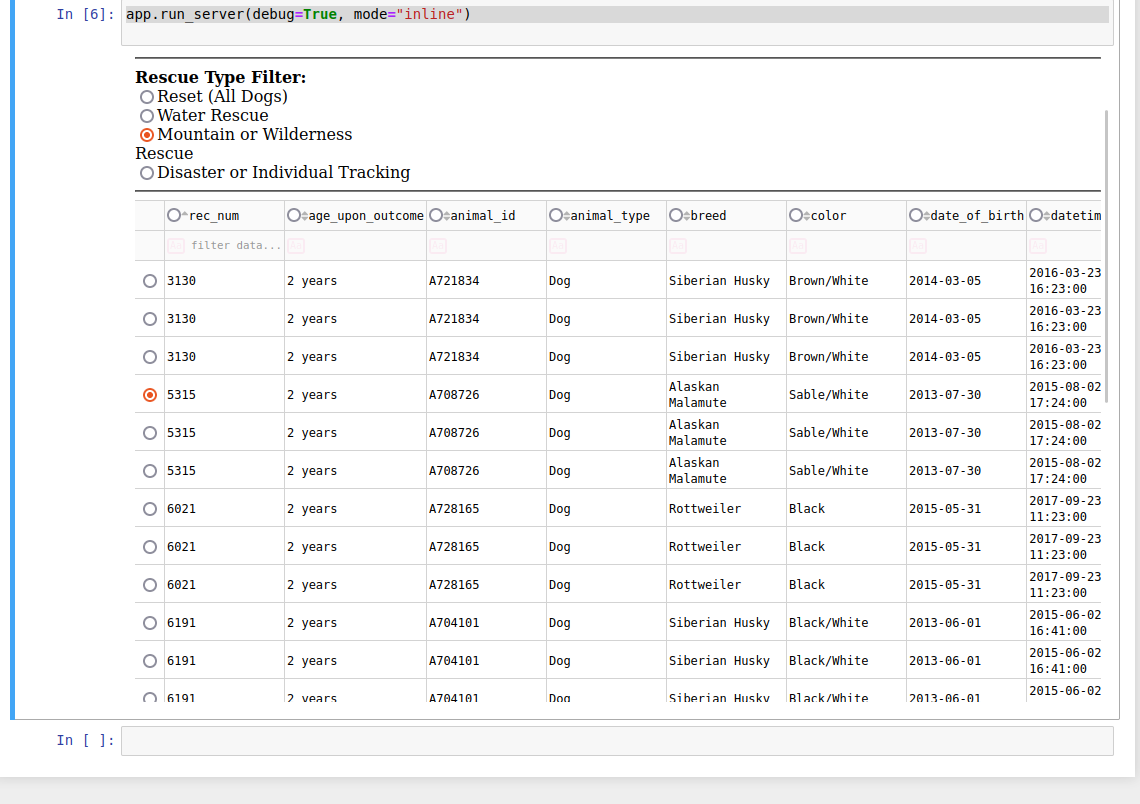
@app.callback(Output("graph-id", "children"),  
 Input("datatable-id", "derived\_virtual\_data"))  
 def update\_graphs(viewData):  
 dff = pd.DataFrame.from\_dict(viewData) if viewData else pd.DataFrame()  
 if dff.empty or "breed" not in dff.columns:  
 return [html.Div("No data to plot.")]  
 top = dff["breed"].value\_counts().reset\_index().rename(columns={"index": "breed", "breed": "count"}).head(15)  
 fig = px.bar(top, x="breed", y="count", title="Breed distribution of current results")  
 return [dcc.Graph(figure=fig)]

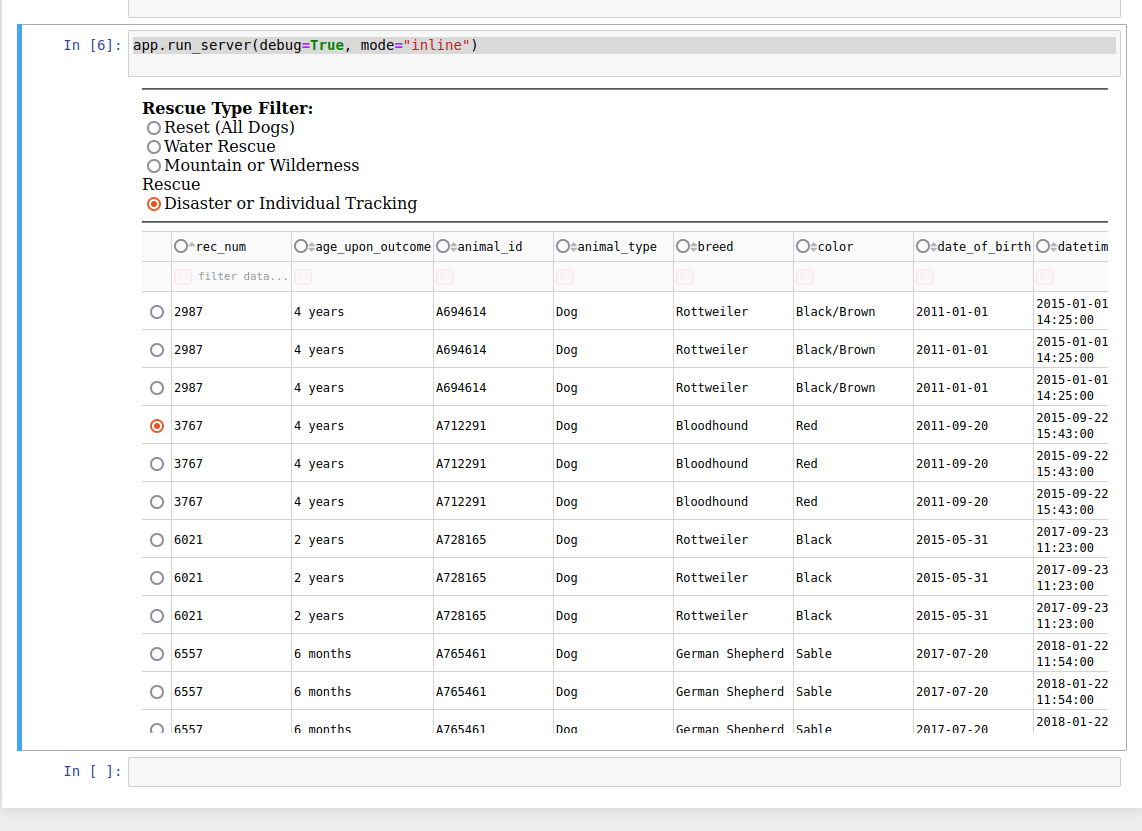
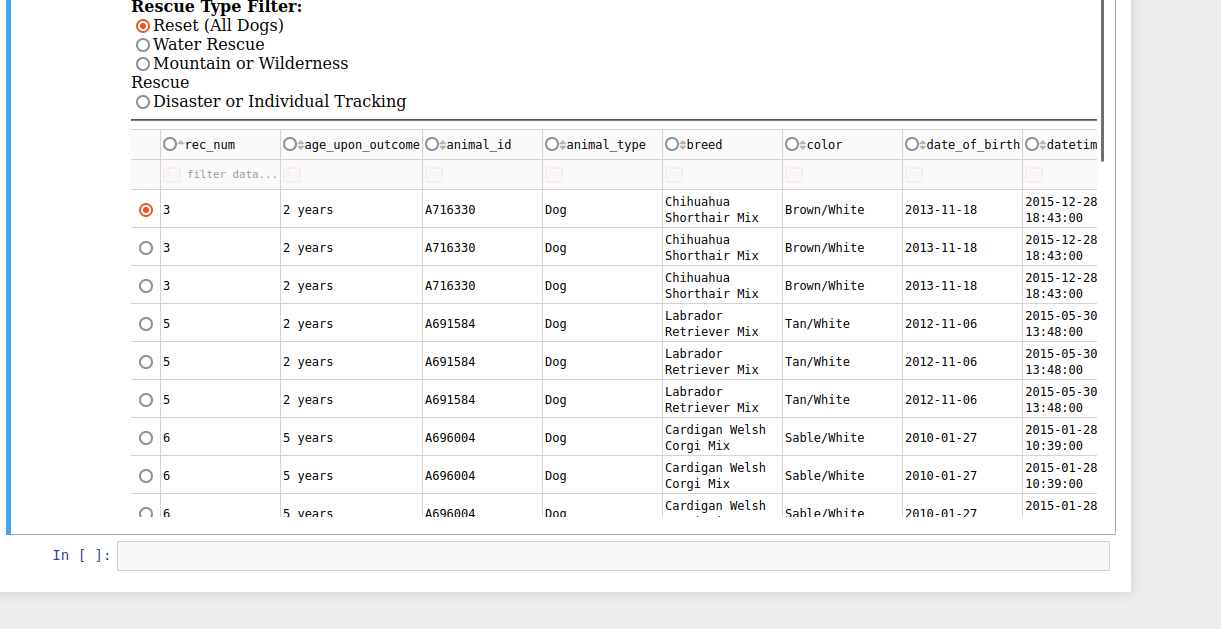
## **Tests**

Required Screenshots (or screencast) included in submission:  
 1. Dashboard starting state

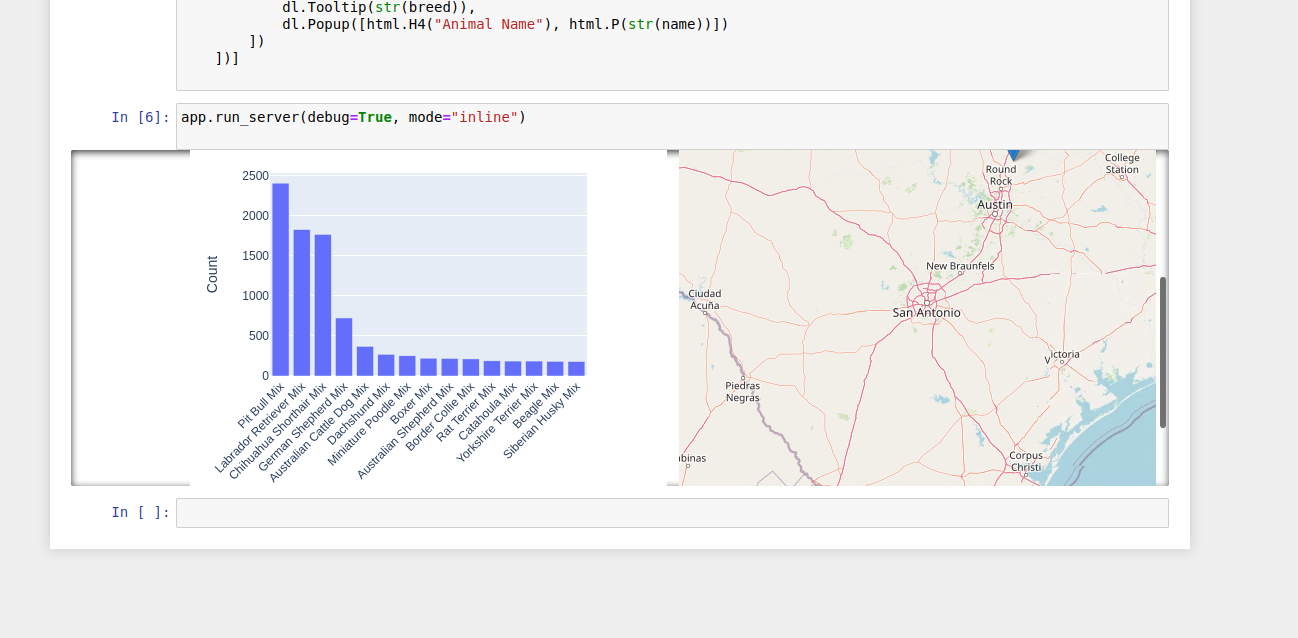
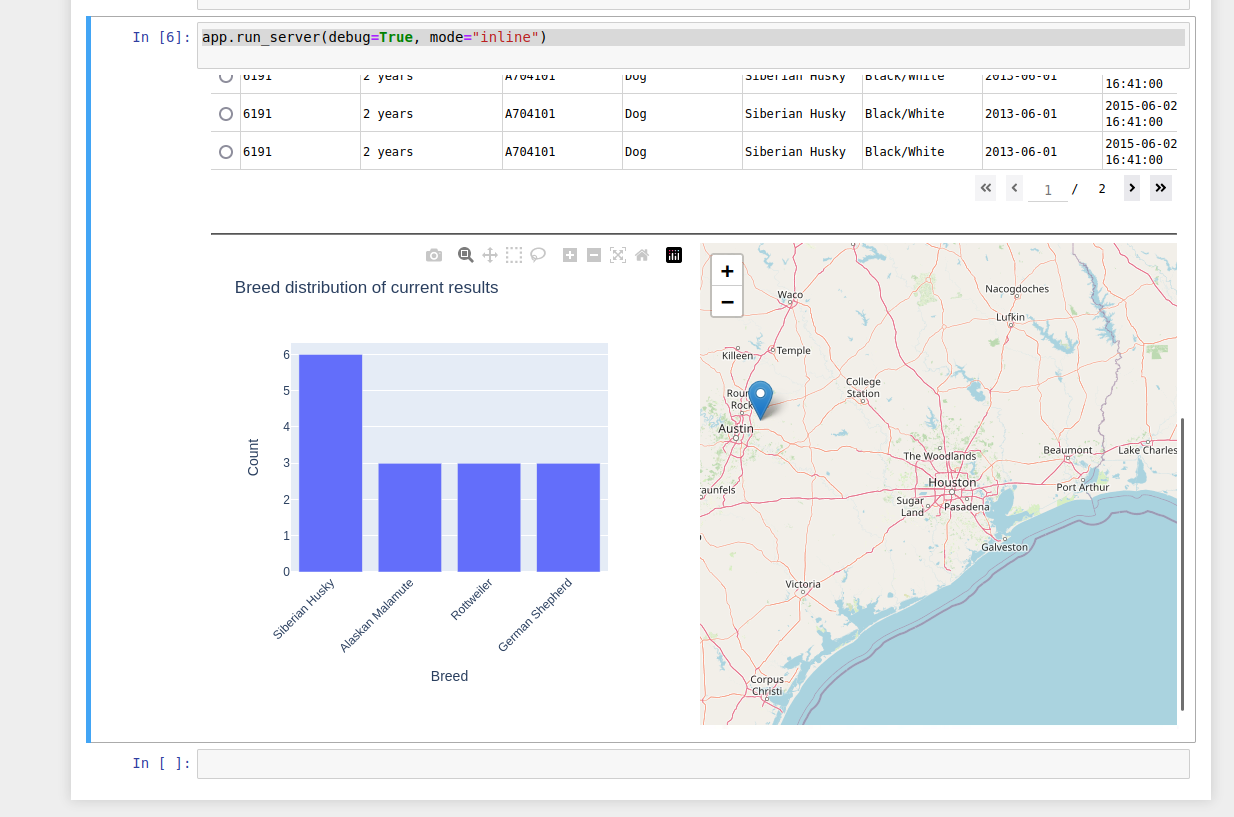
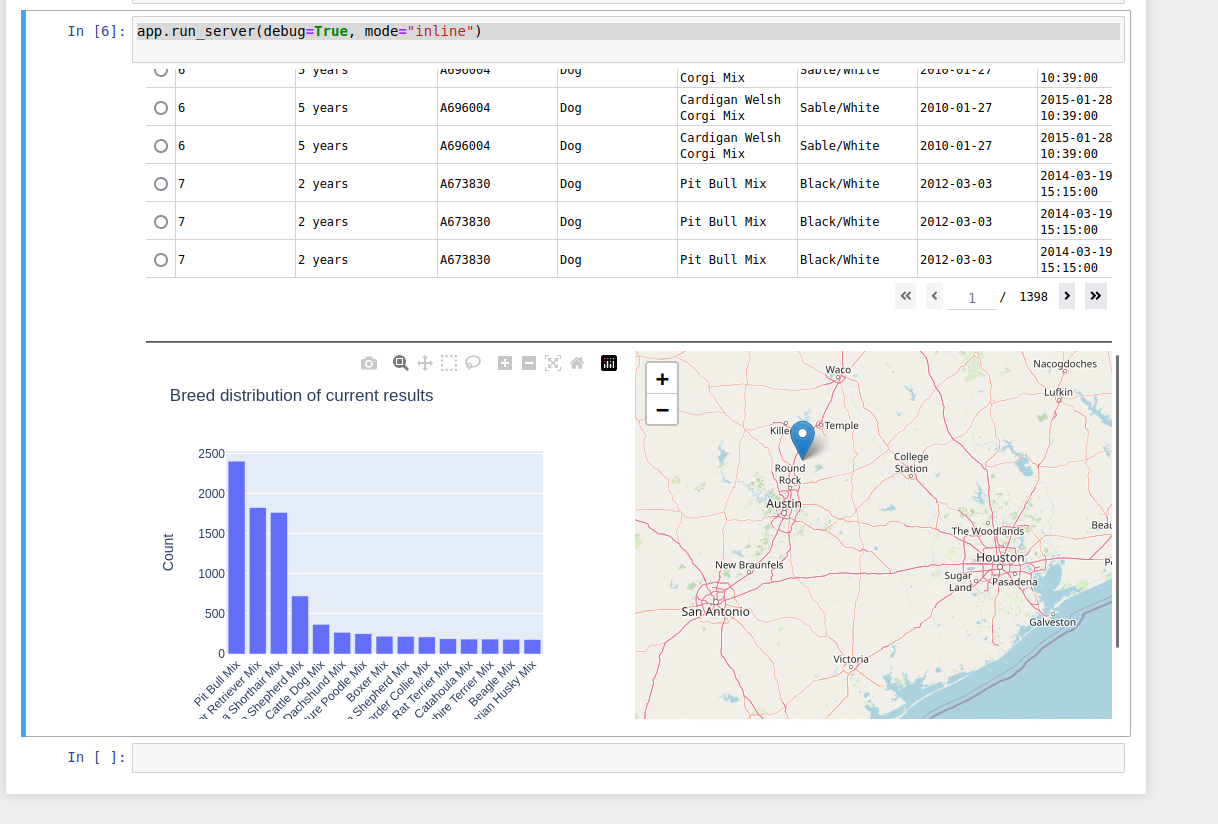
  
 2. After applying Water Rescue filter.

  
 3. After applying Mountain or Wilderness Rescue filter.

  
 4. After applying Disaster or Individual Tracking filter.

  
 5. After applying Reset filter.  


Displaying graphing.



## **Tools Used**

- MongoDB: Model component for flexible schema storage and fast querying.  
 - PyMongo: Interface for CRUD operations in Python.  
 - Dash / JupyterDash: Framework for building interactive dashboards.  
 - Dash Leaflet: For mapping and geolocation visualization.  
 - Plotly Express: For interactive charts.  
 - Pandas / NumPy / Matplotlib: For data handling and preprocessing.

## **Challenges**

- Handling age fields stored as strings required converting them into numeric weeks.  
 -Serious issues getting my images to work correctly.

## **Contact**

Connor Holohan  
 connor.holohan@snhu.edu