# CS 340 README

## About the Project/Project Title: Grazioso Salvare Dashboard

## This project is a Dash-based interactive dashboard that connects to the Austin Animal Center (AAC) MongoDB database. It allows users to filter dog records by rescue mission type, view the results in a table, see a pie chart of breed counts, and map the selected animal’s location. It’s meant to help Grazioso Salvare quickly find dogs that meet specific training and mission needs.

## Motivation:

## The purpose of this project was to get hands-on experience with Dash and build a full interactive web dashboard using real animal shelter data. I wanted to build something that not only connects to a live MongoDB database but also makes the data easier to view and use. It builds off the CRUD work from Project One and adds more complexity through visualizations and interactivity.

## Getting Started:

## I first created a CRUD module (AnimalShelter.py) that connects to MongoDB and supports create, read, update, and delete operations. Then I used Dash and JupyterDash to build out the dashboard interface. I added callbacks to update the data table, pie chart, and map when filters or rows are selected.

## Installation:

* MongoDB – stores the ACC animal data
* PyMongo – Allows Python to connect to MongoDB
* JupyterDash – used to build and display the dashboard
* Pandas- for manipulating and filtering data
* Plotly Express- used for the pie chart
* Dash Leaflet- used for the interactive map
* Apporto – this was the environment everything was developed and tested

## Usage

To run the dashboard:

1. Make sure MongoDB is running in Apporto
2. Open ProjectTwoDashboard.ipynb in Jupyter Notebook
3. Run the cell to load the dashboard
4. Use the radio buttons to filter for a specific mission
5. Click a row in the table to see the dog’s location on the map

### Code Example

**Filtering the Data Table**:

if filter\_type == 'Water Rescue':

query = {"$and": [{"animal\_type": "Dog"},

{"breed": {"$in": ["Labrador Retriever Mix", "Chesapeake Bay Retriever", "Newfoundland"]}},

{"sex\_upon\_outcome": "Intact Male"},

{"age\_upon\_outcome\_in\_weeks": {"$gte": 26, "$lte": 156}}]}

**Pie Chart Callback:**

if filter\_type == 'Reset':

top\_breeds = dff['breed'].value\_counts().nlargest(10).index

dff = dff[dff['breed'].isin(top\_breeds)]

fig = px.pie(dff, names='breed', title='Preferred Breeds by Filter')

**Map Marker Callback:**

dl.Marker(position=[dff.iloc[row,13],dff.iloc[row,14]], children=[

dl.Tooltip(dff.iloc[row,4]),

dl.Popup([html.H1("Animal Name"), html.P(dff.iloc[row,9])])

])

### Tests:

I tested each part of the dashboard after building it:

* Verified filters returned the correct animals
* Made sure the table updates on filter change
* Confirmed the pie chart adjusts based on the current dataset
* Checked that the map displays the right location and dog info when row is clicked
* Also tested the CRUD methods again and updated them based off previous feedback

**What’s Next**:

If this dashboard were to be expanded, it could be turned into a full web app with login features, more filtering options, and a cleaner layout. It could also be connected to a larger system for tracking adoptions, training progress, or other mission details.

### Screenshots

|  |
| --- |
| A screenshot of a computer |

|  |
| --- |
| A screenshot of a map  AI-generated content may be incorrect. |

|  |
| --- |
| *A screenshot of a map  AI-generated content may be incorrect.* |

|  |
| --- |
| *A screenshot of a computer  AI-generated content may be incorrect.* |

## Contact

Your name: Matthew Tampon