**Grazioso Salvare – Rescue Dog Dashboard**

# About the Project

## Project Description

Global Rain, a software engineering company, required the development of a web application dashboard for Grazioso Salvare, an international rescue-animal training company. This dashboard enables Grazioso Salvare to interact with and visualize data from animal shelters in the Austin, Texas region, helping them identify suitable dogs for search-and-rescue training.

**Project Requirements**

Allow Grazioso Salvare to view data from the Austin Animal Center Outcomes data set. Additionally, required functionality included:

* Displaying a data table with all the data from the Austin Animal Center Outcomes data set which dynamically responds to filtering options
* Interactive filter options to filter the data set by; Water Rescue, Mountain or Wilderness Rescue, Disaster Rescue or Individual Tracking, and resetting the data set
* A geolocation chart that would display the location of a selected animal
* A data chart that dynamically responds to the filtering options
* Logo for Grazioso Salvare that would link to their website

## Tools used to complete the project

* MongoDB: Used as the database to store and retrieve animal shelter data
  + MongoDB was chosen as the model component due to its flexibility, scalability, and compatibility with Python. It aligns well with Python's data handling capabilities and offers an efficient and developer-friendly way to manage and query data for the Grazioso Salvare project as well as a python specific interface utilizing PyMongo.
* Dash Framework: Provides the view and controller structure for the web application.
* Python: Programming language used for backend development.
* HTML/CSS: Frontend web development.

## Getting Started

Follow these instructions to reproduce the project on your local machine:

**Prerequisites**

* Python 3.x
* MongoDB
* Dash Framework
* Python CRUD module
  + This project utilizes a CRUD module that has already been developed. You can find the module and its documentation here: https://github.com/CodeBando/CS340\_PyMongo\_CRUD

**Installation**

1. Clone the repository to your local machine
2. Ensure you have Python 3.x installed
3. Install the required packages (See installation notes):
4. Initialize an instance of the ‘AnimalShelter’ class
   1. You must provide the MongoDB Credentials for username and password
   2. All other connection details are hard-coded for the “AAC” Database and “animals” collection
5. Utilize methods of module to complete Create and Read operations on the MongoDB. Example executions can be found in the usage section below

## Installation notes

In order to utilize the Python module, you will need the following tools installed:

* Python 3.x
  + Visit <https://www.python.org/downloads/> to download

## Usage

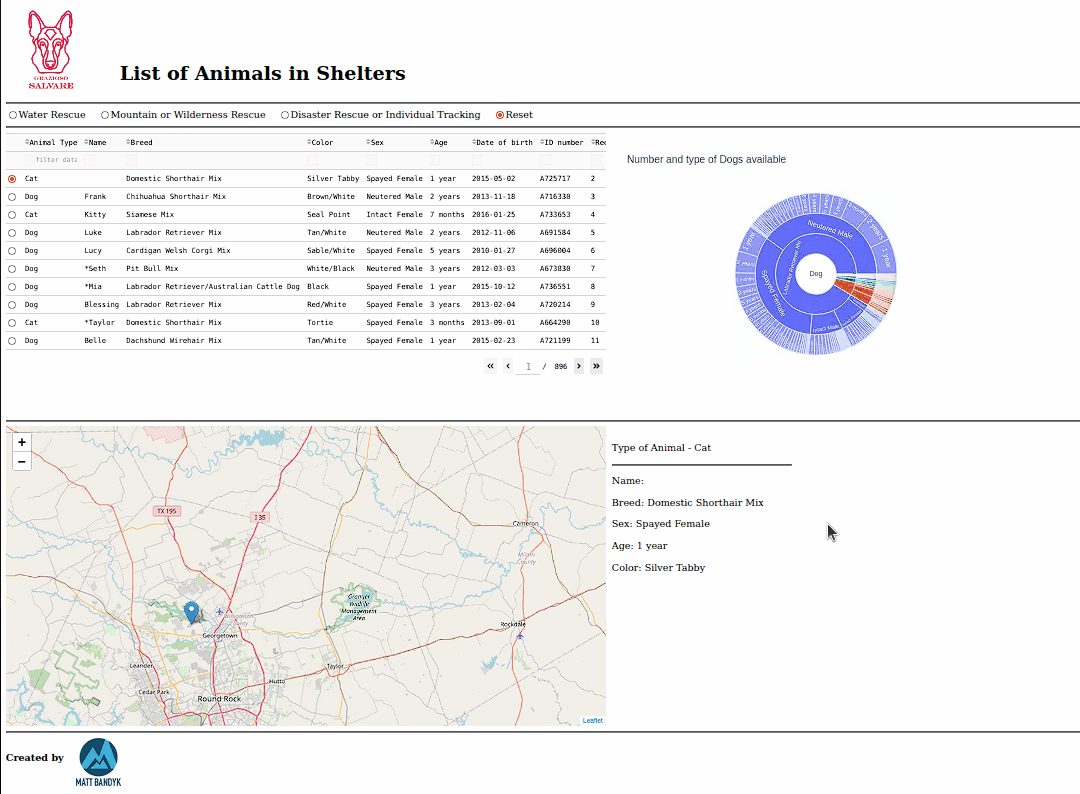
Below are some examples of how to utilize the dashboard functionality, as well as examples of the codebase and how to manipulate it to utilize for your own project needs.

### Dashboard Examples

This dashboard allows for the utilization of 5 key features, which can be seen below: Data Table, Sunburst Chart, Geolocation information, Document details, and sorting and filtering options.

*Dashboard:*

*------------------------------------------*

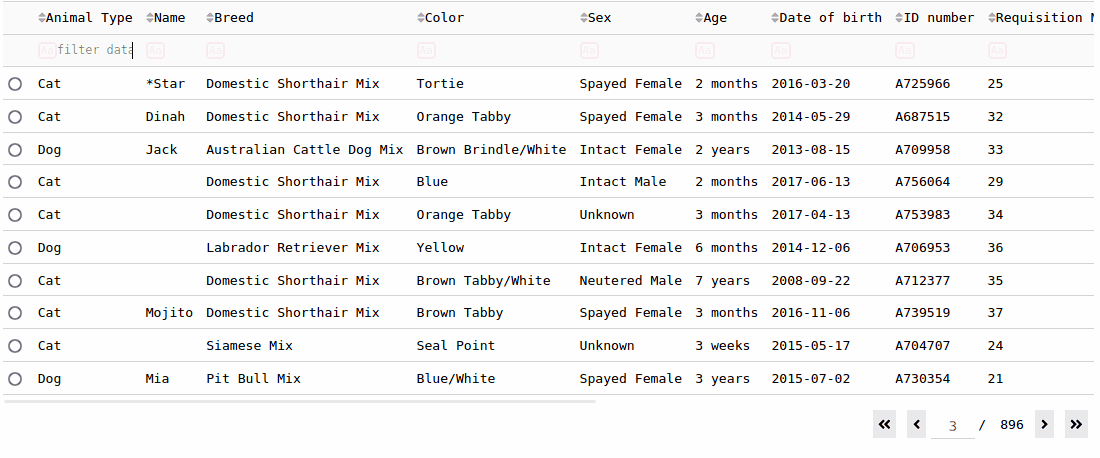


The Dashboard provides a view to the user that displays the interactive data table, an interactive sunburst graph, a geolocation map indicating the animals location, and an html display showing key information about the selected animal. Users can utilize the filtering radio buttons to filter the elements based on the type of rescue dog they are looking for as seen in the gif above.

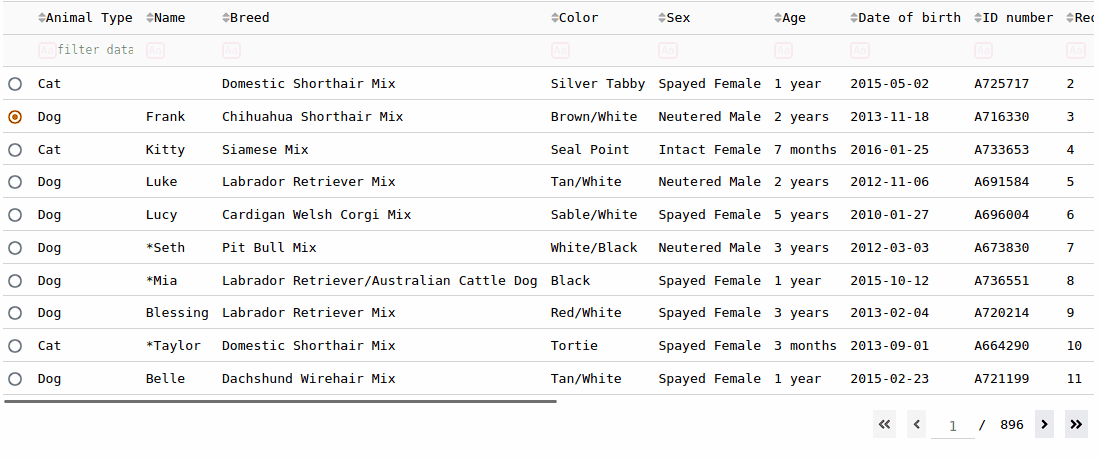
*Data Table functionality:*

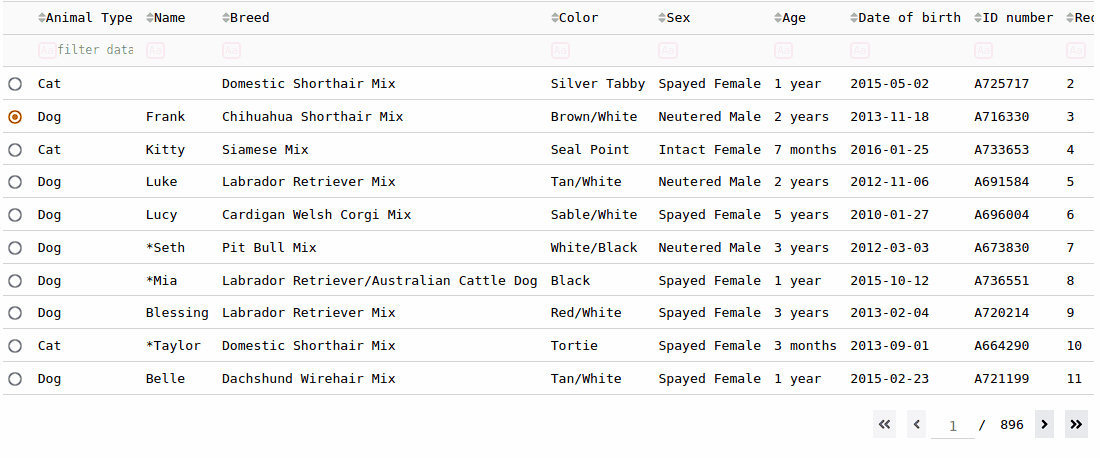
*------------------------------------------*

The data table offers many unique features to help the user navigate it. To increase usability and user experience the table is limited to displaying 10 lines at a time and allows the user to navigate through the pages as needed. They are also able to scroll horizontally to view the entire documents information, with the most critical being in the first columns. (See gif below)



Users are also able to sort the data within the data table by column as well. (See gif below)

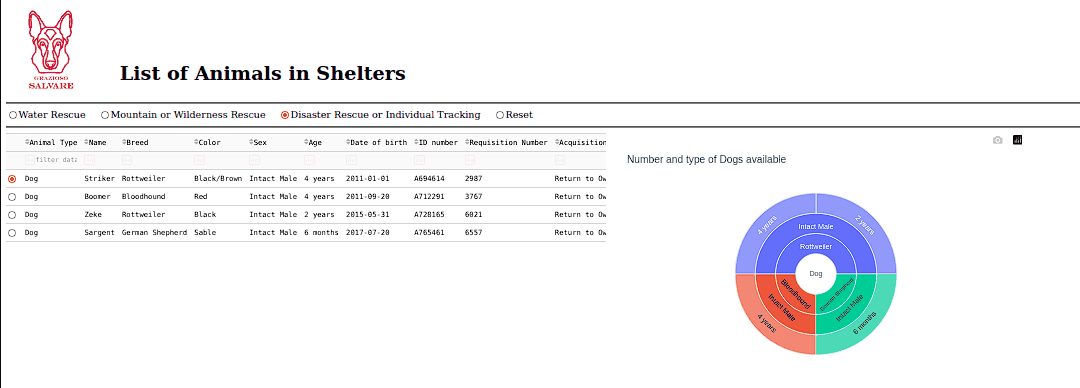
 Lastly, the user is also able to filter the information shown based on parameters in any of the columns by typing what they are looking for into the filter data row. (See gif below)



*Sunburst Graph functionality:*

*-----------------------------------------*

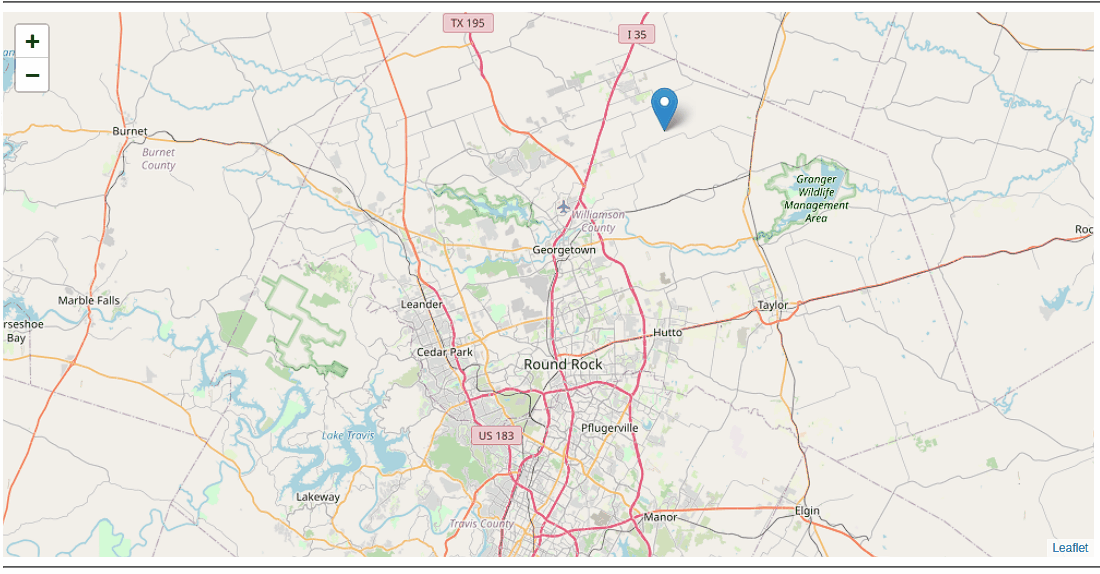
The sunburst graph allows the user to see the available trainable dogs based on the preferred breeds for the different types of training, water rescue, wilderness rescue, and individual tracking. The sunburst graph also updates when the radio button filters are selected. It can be utilized by selecting the second level of information to then display only that breed of dog which will then show the user how many of each sex and then how many by age. (See gif below)



*Geolocation map functionality:*

*-----------------------------------------*

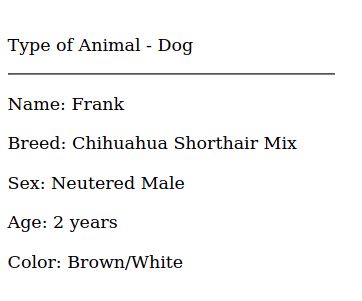
The dashboard also boosts a geolocation map that displays the currently selected animals location. It will indicate it with a location marker that can be selected to show the animals name and hovering over it will show the breed of the animal selected. (See gif below)



*Document details:*

*-----------------------------------------*

The last section of the dashboard displays key information about the animal that has been selected within the data table and will update when a new row is selected.



### Dash framework overview

Below are details regarding the Dash framework utilized to create this program along with tips to manipulate it to be utilized for your own specific needs.

***Layout (view):***

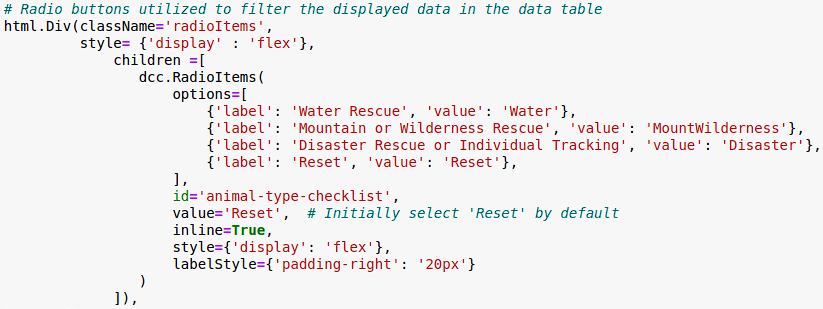
*-----------------------------------------*

The layout section of the dash framework structures the appearance of the web application.

1. **Logo and Title**

Creates a container to hold the logo and title at the top of the page. Loads the .png file of the logo, which can be updated to use your own image. Utilizing header 1 for the title. Update the title of the dashboard you are creating.

1. **Radio Buttons**

Creates a container to hold the radio buttons. You can update the radio buttons to your own needs. Update ’label’ to be what you want visible in the dashboard and ’value' is what you want to pass to the callback. Add or remove options as needed. For more information on radio buttons and styling, see https://dash.plotly.com/dash-core-components/radioitems

1. **Data Table and Sunburst Chart**

Creates a container to hold the data table and the graph being utilized, putting them inline with each other. I have set the data table to display unique headers to help usability which can be updated as needed. Additional styling for the data table is also included. To learn more you can visit https://dash.plotly.com/datatable

1. **Geolocation and Document Details**

Creates a container to hold the Geolocation map and a container to hold html driver data displaying information about a selected record.

***Callbacks (Controller):***

*-----------------------------------------*

1. **Radio Button Filtering**



I have utilized variables to hold the various filtering requirements provided by Grazioso Salvare to utilize when completing queries on the database. This allows for easier updating in the future if additional breeds, sexes, or differences in age occur. If you wish to create your own filtering options, you can add your own specific variables to utilize when completing queries.

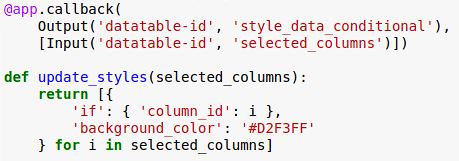


The callback will then read the passed value which is assigned by the user selecting a radio button in the dashboard. Based on the selection, a data frame is created by filtering only the animals available that fit the criteria. You can create your own queries or update these based on your specific needs.

1. **Sunburst Graph**

I decided to utilize an interactive sunburst graph to display all the animals in the animal shelters data base that are considered trainable dog breeds based on Grazioso Salvare’s requirements. The graph will also update when a filter is selected utilizing the radio buttons. It will allow the user to quickly and easily see how many of each breed there are, their specific sexes, and their ages. If you would like to utilize a different graph, you can see the many options available through Ploty and how to utilize and implement them at https://plotly.com/python/

1. **Cell Highlighting**



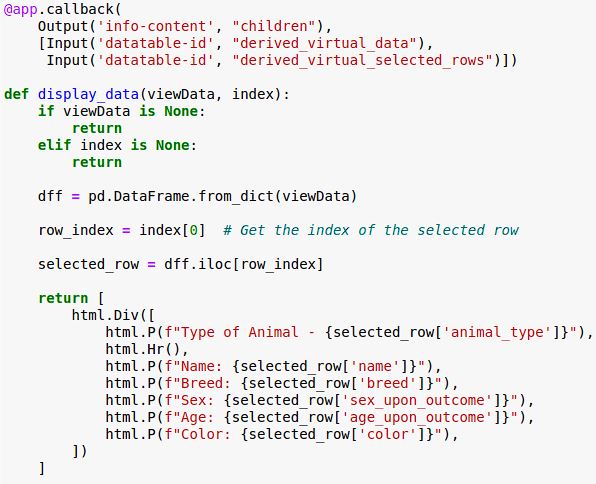
This callback will highlight a cell if the user clicks on it. The color can be updated to whatever you would like it to be by updating the Hex Color value in ‘background\_color’.

1. **Geo-location Map**



This callback will display a map based on the lat/long values you provide. These can be updated if needed to set the map to a different location. Currently, the map marker indicating the location of the selected animal will display the animals name when selected and if hovered over, will show the animals breed. These can be updated if you would like to display different information.

1. **Display Document Details**



The last callback displays specific information regarding the selected animal, displaying it in a vertical format, and giving the user the key information about the animal. If you would like to capture additional or different information, you can update what is printed and the selected\_row column title to display it.

**Challenges and Solutions**

Below are a few challenges I faced when completed this project and how I was able to solve them.

***HTML/CSS Usage:***

*-----------------------------------------*

It had been a long-time since I had worked with HTML/CSS and I was having difficulty getting the layout of the dashboard to execute how I wanted it to look. To assist, I did a bit of a refresher and reference a great resource for HTML/CSS “A Simple Guide to HTML” which can be found here: <http://www.simplehtmlguide.com/>

***Graph Functionality:***

*-----------------------------------------*

I wanted to utilize a graph that would provide relevant information to the user that would improve usability and the user experience, which is why I chose the Sunburst graph. To utilize it effectively though, I needed to create additional information and add it to the data in order to display it correctly, which was the counts of each of the different data types I wanted to utilize, breed, sex, and age. To accomplish this, I utilized the Pandas reference documentation which can be found here https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.count.html to develop a solution.

## Contact Information

Your name: Matthew Bandyk