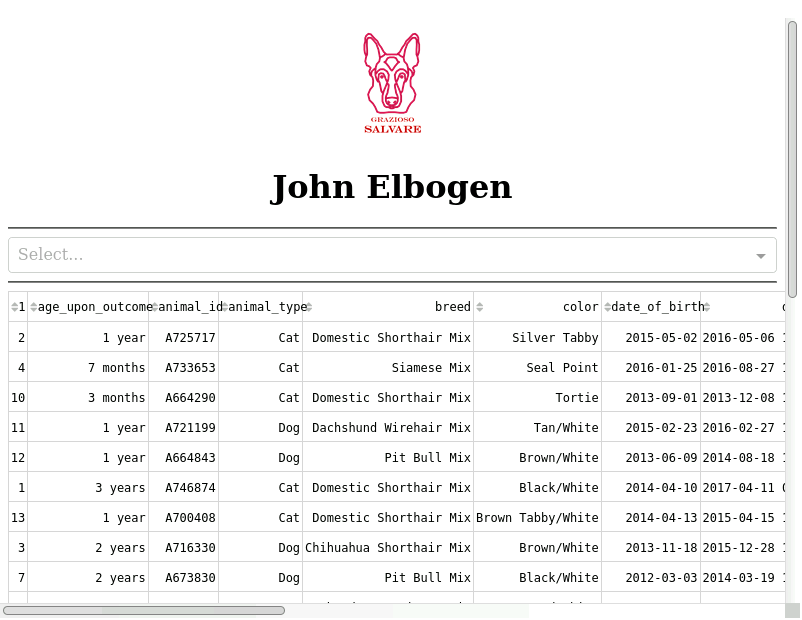
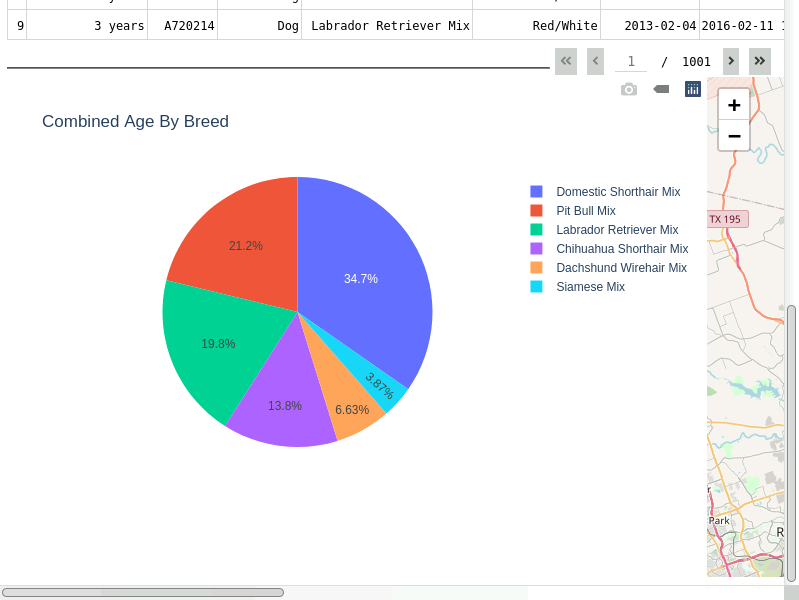
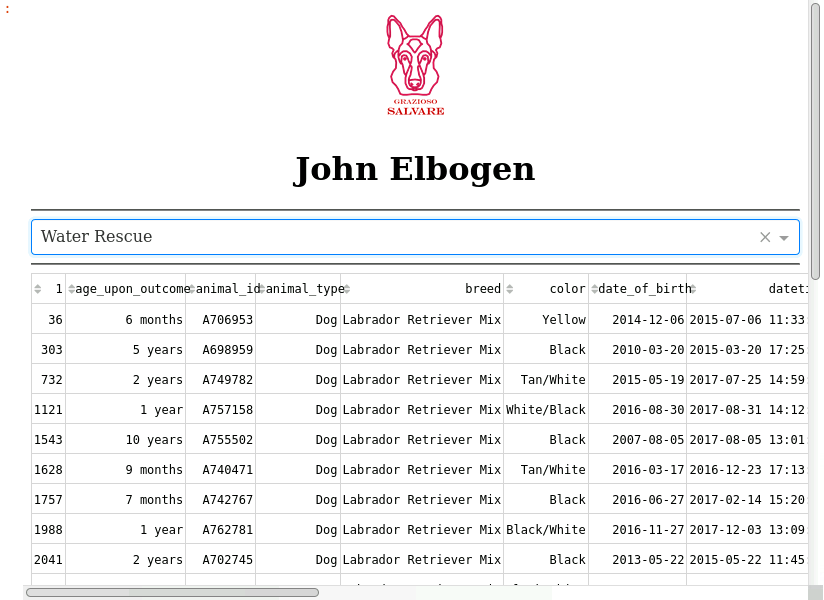
**Project 2 ReadMe**

Author: John Elbogen

Version: 1.0







**Pre-Note:**

To begin, I have included the screenshots along the left side which show the several interactable functionalities and filtering.

**Getting Started:**

The purpose of this project was to create a full stack development, allow users from Grazioso Salvare to identify dogs that are good candidates for certain conditions. The main purpose is to create an easily viewable and filterable table, with accompanying charts.

**Installation:**

*Necessary Files* –

ProjectTwoDashBoard.zip

aac\_shelter\_outcomes.csv

<https://jupyter.org/>

<https://www.mongodb.com/>

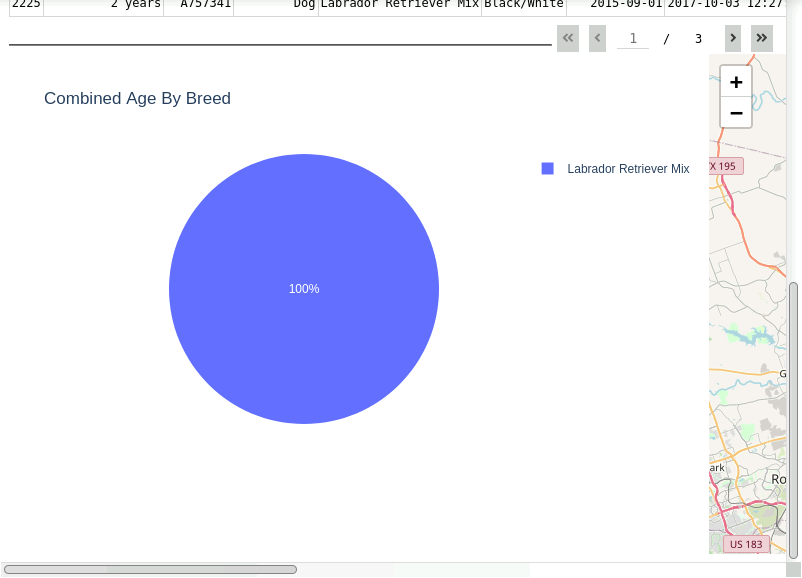
Jupyter Notebook can be used to run the ProjectTwoDashBoard Final.IPYNB, and the other files in the zip must be in the same directory.

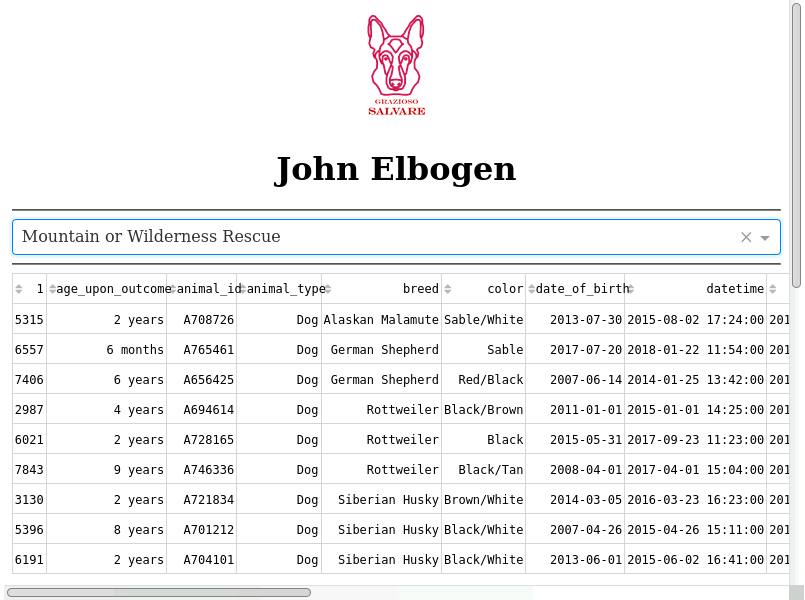
MongoDB must be active and contain the AAC database.

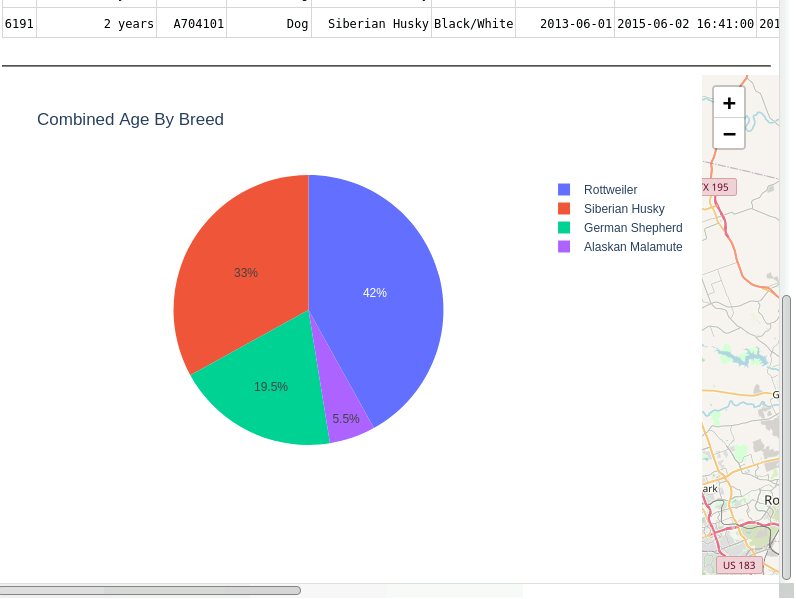
**Required Functionality:**

Dashboard branding, with logo, and unique identifier.

Interactive filter options, for Water Rescue, Mountain or Wilderness Rescue, Disaster Rescue or Individual Tracking, and a Reset option to return to the base state.







A data table which dynamically responds to filtering.

Finally, a geolocation chart and second chart that dynamically responds to filtering.

**Recreation:**

The basic processcompleting this process is as follows.

1. Setup up a functioning MongoDB database. This includes creating user account, with privileges for reading and writing files within the database.

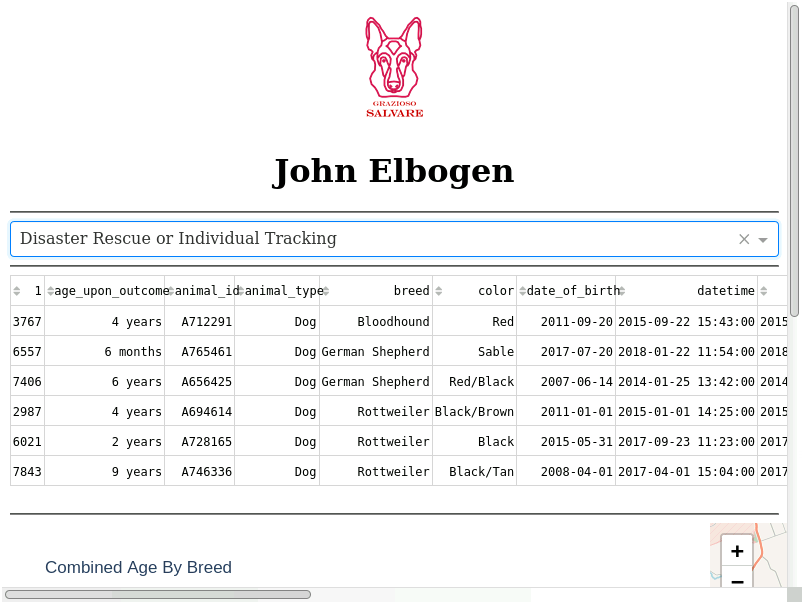
MongoDB provides easy query functionality, and database control for users, to search and filter easily.

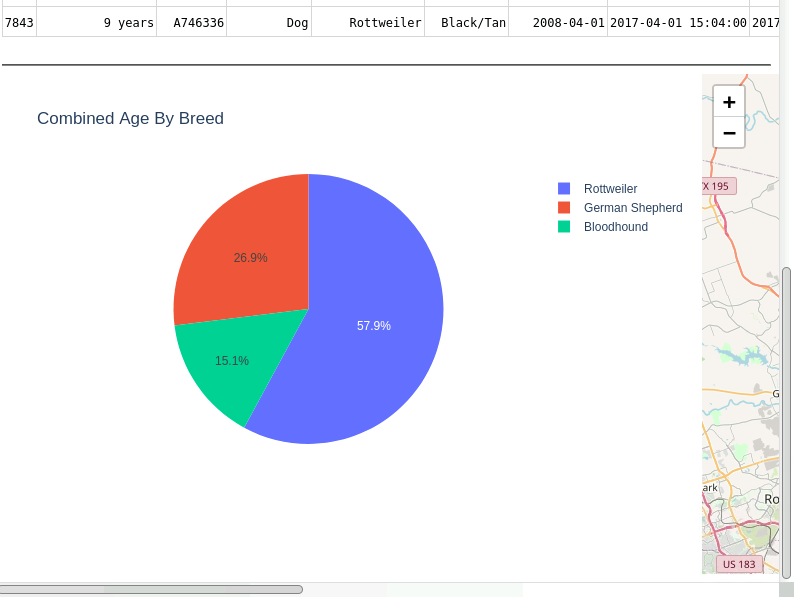
<https://www.mongodb.com/what-is-mongodb>

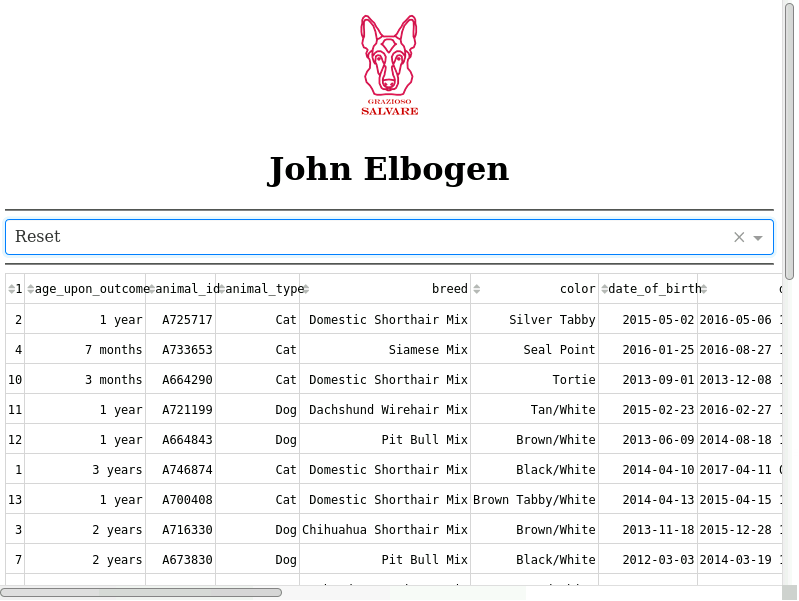
1. Next a script is written in python for the CRUD functionalities. The Create, Read, Update, and Delete functions are the basis of handling the database. This was written in python and stored as a .py script later imported into our web application dashboard.

<https://www.codecademy.com/articles/what-is-crud>

1. We begin creating our dashboard using several core components. We use plotly dash libraries for many of the components. Plotly dash basically provides the glue for all the components needed within our web application. For example, we use dash html components, which allows the application to implement html components easily in our dashboard code. Also, we use Plotly express to create our interactive data chart. In the







case of this project, a pie chart was created. Dash core components also allow us to implement listener components such as a drop-down menu for selecting the filter choices from. <https://plotly.com/dash/>

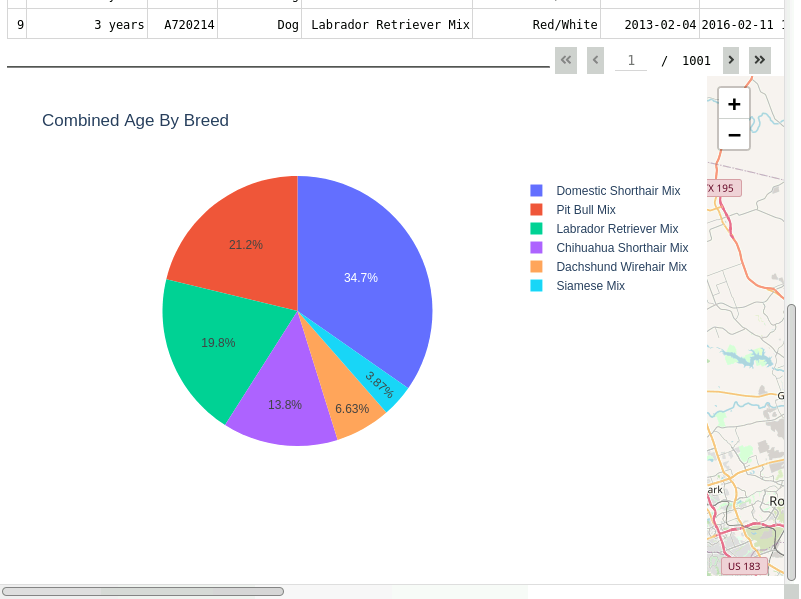
1. The basic structure of the dashboard is created through several html components. We start with our header and logo, followed by data table, and finally our charts and geolocation. This is mainly created with the dash html components, dash core components, and dash table. Html mainly provides the structure and style of each component.
2. Next, we implement our basic interactive functions using callbacks. Our main functions include a callback for selection upon interacting with the dropdown, selecting columns, and data updates for the graph and map. In the dropdown function, we include our queries for each function, to filter the data according to the specifications for each type of animal. Finally, the query is returned and updated in the data table.

<https://docs.mongodb.com/guides/server/read_queries/>

1. With these steps we have created our final working dashboard.

**Challenges:**

Throughout the project several challenges were faced. One main challenge was trying to implement the data table update and filter.



The filter had several problems, for example, when a user changes to a new page and filters the results that display less pages than the number currently on, the database essentially cannot be reloaded and displays no information. One solution to this was resetting the current page to zero after the callback.

However, this solution did not stick. It caused issues with updating the graph. Another challenge was working with the queries in general. Sometimes errors would occur, and it was unclear whether the query was incorrect or not. For example, I was getting filter results that made no sense, and reviewed my query, realizing it was a coding error elsewhere. The main solution to this problem was just to constantly review and debug my code while working on it.

I tested in small iterations and made sure each feature was functional as I finished it. I went back and checked how it worked in relation to other parts of the program, to stay on top of newly created errors.