# Grazioso Salvare Data Table Interface

## About the Project

This is a user interface for a MongoDB database used for a client that identifies and trains dogs for search-and-rescue training. The functionality of this project is the ability to filter through the database for specific dogs that could qualify for search-and-rescue training with Grazioso. In addition, this project displays a pie chart of the popularity of different types of breeds for a selected rescue type, as well as an interactive map that shows a marker on the location of a selected dog from the database with a tooltip containing basic information about the dog.

## Motivation

This project exists to help Grazioso to identify and categorize potential search-and-rescue dogs from local animal shelters in Austin, Texas. Grazioso has identified that specific breeds are often better suited for a specific type of rescue, so Grazioso’s use of this project is to develop a quick reference to identify what dogs in the local area would be good for different types of rescues, as well as provide a forecast and estimate for different health factors of the dogs. For example, this project predicts the age a given dog would be after training and contains the current age of each dog currently in a shelter.

## Getting Started

## Installation

List of the tools you need to use the software and how to install them.

MongoDB:

Can be downloaded and installed here: (https://www.mongodb.com/docs/manual/installation/).

Python:

Can be downloaded and installed from (python.org/downloads/).

Pymongo:

Can be downloaded and installed from python package index (https://pypi.org/project/pymongo/) or through pip using “pip install pymongo”.

Plotly:

Can be downloaded and installed using pip, “pip install plotly”.

Dash-leaflet:

Can be downloaded and installed using pip, “pip install dash-leaflet”.

Dash:

Can be downloaded and installed using pip, “pip install dash”

Jupyter Notebook:

Can be downloaded and installed from their website at (jupyter.org/install) or through pip, “pip install jupyterlab”.

**Setup**

To begin, start MongoDB via the terminal. Once done there’s two steps that need to be completed. First is to import the data for the database. For this project that will be a .csv file of animal shelters in Austin, Texas imported to a database named “AAC” under a collection named “animals”.  
**Text

Description automatically generated**

Next a user account needs to be made for authentication purposes later. To do this, create a user that has “readWrite” privileges in the database used for the application, for this project this will be the database “AAC”.



For scripting, this project uses Python on Jupyter Notebook, but other IDEs with other languages work as well as long as they’re compatible w/MongoDB. This project uses the pymongo library to interact with MongoDB as it’s developed by the team of MongoDB and well supported to provide quick and easy functionality to managing Mongo documents.

For Jupyter Notebook there are two different files that are used to execute CRUD functionality with MongoDB. First is a (.py) file that contains the code for performing CRUD actions with Mongo and uses pymongo. Make sure to create each function for CRUD functionality under a class object so that it can be easily imported later into the codebase of the application. For this project’s class that will be a class object named “AnimalShelter”, with the name of the file being “animalShelter”.

The second file will be a Jupyter file (.ipynb) that houses the code for the interface of the application. This file will import all the dependencies listed under the installation section. The most prominent dependencies here will be: Plotly, Dash and Dash-Leaflet. Plotly is a free and open-source graphing library for Python, Dash is a open-source framework for interfaces using data visualization, and Dash-Leaflet is a wrapper for React-Leaflet. Dash will be used primarily for the creation of the data table & data table interactivity sections of the application, Plotly will be used to make a pie-chart that shows the breakdown of the popularity of a specific breed given the filters selected, and Dash-Leaflet will be used to display an interactive map & map markers of selected dogs from the data table along with a tooltip of some basic information about the dog selected (name & breed).

## Usage

**CRUD (Python module):** To make the application work, there is a pymongo module file that we created earlier that includes CRUD operations to interact with the Mongo database. When passing the read functions from the filter buttons of the application interface, make sure to write the request as one argument, either using ($and) or using ({}) in the correct manner to pass the query as a single argument. The following are the different functions of that file that are used in the final application (Initialize, Read):

*A picture containing text

Description automatically generated*

*Text

Description automatically generated with medium confidence*

Future developments of the project could include the other functions of the CRUD module to allow a user to import/delete shelters and their data, or update information regarding specific entries.

**Filters:** Filters trigger once a button is pressed. The timestamp that the button was pressed is saved and compared against the timestamps of the other buttons. Depending on which is most recent, that button queries the database with a statement and the end of the callback will set the data of the data table to the result of the query.

Text

Description automatically generated

*Text

Description automatically generated*

**Data Table:** The data table will take the read information from the data import using the user account we created earlier and create columns using the data read from each document inside of the collection “animals” in the “AAC” database using the “AnimalShelter” CRUD class we created. Other settings are used to make the data table interactive, limiting or expanding the input a user can have with the element, as well as the display of the table. In my development I attempted to rename the columns to a new set of names that were more user friendly, but I encountered issues with the data table accepting an altered “columns” list, future development could include a localized list of column names.

Text

Description automatically generated

**Graphical user interface, text

Description automatically generated**

**Pie Chart:** The pie chart takes the keys of the breed column and counts the occurrences of each breed in the current updated data table (viewData) into two different lists. Then the pie chart is updated with the information retrieved from those two lists. Future development could include a drop-down menu that allows the user to filter for different columns instead of breed. There could also be localization so that the tooltip when hovered over individual sections of the pie chart don’t display things such as “value=[result]”

Text

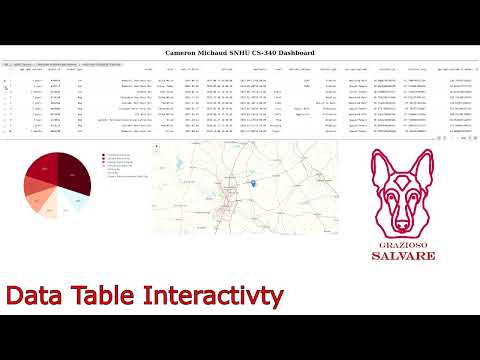
Description automatically generated

**Map:** The map takes the current updated data table (viewData) and the selected row via the radio buttons on the left-most row of the data table (rows). If there are no selected rows, then the rows list must be initialized to keep the map from disappearing, so the default row that is shown will be the first of current data table (0). Then ideally for each row the columns with the required information such as latitude/longitude will be taken for placing the marker, as well as the name & breed of the animal for the tooltip. I say ideally because attempts were made to be able to show a multitude of animals at a time depending on how many rows were selected from the data table, but this resulted in whatever row was lowest in the order would be shown while the rest would be hidden. To preserve functionality the data table attribute “row\_selectable” was changed from “multi” to “single”. There are no complications with the leftover code.

*Text

Description automatically generated*

### Video Demonstration

*[](https://www.youtube.com/embed/wNS2JfujClQ?feature=oembed)*

Link: <https://youtu.be/wNS2JfujClQ>

## Roadmap/Features

* Localize column names & pie chart values.
* Include option to change data import for the pie chart (select different columns).
* Increase the number of markers placed on the map at a time to be able to display multiple different rows.
* Increase functionality of the data table to be able to perform the other functions of the CRUD module such as create, delete, and update.
* Hide the display of certain columns that are of little use to the user (location data).
* Optimize the display of the application to resize dynamically to the display of the window to always display elements at a reasonable size.

## Contact

Cameron Michaud