# Grazioso Salvare Dashboard README

## About the Project

Grazioso Salvare’s Rescue Dog Dashboard is a full-stack application using a Dash web application and intermediary Python modules to access and display data from a MongoDB database. The database contains information about animals that have been in any of five nonprofit shelters around Austin, TX. The Python modules perform CRUD functions on the database, and the Dash application provides an intuitive data display in a data table that can be filtered, a map locating any selected animal, and a pie chart displaying the proportions of either breeds or sexes in the filtered results. It is designed according to Grazioso Salvare’s requirements.

## Motivation

The application’s purpose is to identify dogs that may be trainable to perform searches and rescues of humans and other animals. This identification is done with criteria provided by Grazioso Salvare.

**Background**

This project was completed in three broad stages. First, development of the MongoDB database on Ubuntu, using the terminal and mongosh, including importing the dataset to a new collection and creating a read/write user for that database.

Second, creating the Python modules using PyMongo to make the database accessible to a controller.

Third, creating the model-view-controller application in Dash to display database data to the user and send callbacks from the user to the database, thereby updating the display.

Challenges were relatively few. The main difficulty was simply learning the syntaxes for using MongoDB and the involved Python libraries (see **Installation** below), and there is excellent documentation on the tools’ respective websites to make this possible. Getting acclimated to Dash was the most substantial hurdle, but Dash also has some of the most extensive and user-friendly documentation on their website and it is commonly used enough that even Googling tends to get great results.

## Getting Started

To use the application, it is necessary to already have MongoDB set up locally with the database you wish to use, and a user set up with read/write authorization in that database. Read more information on setting up a MongoDB server here: <https://www.mongodb.com/docs/v6.0/tutorial/>

## Installation

**1.0**

Tools:

* MongoDB. MongoDB provides a flexible schemaless database, making it possible to work with imperfect data or data from various sources, which are both the case in the Austin Animal Center dataset. <https://www.mongodb.com/docs/manual/installation/>
* Mongosh to set up the MongoDB database/user. <https://www.mongodb.com/docs/mongodb-shell/install/>
* Python (latest stable release recommended). Python is good for this sort of application because of its many pre-built libraries offering high-level functionality. <https://www.python.org/downloads/>
  + PyMongo is a full and stable Python library giving us access to a MongoDB database by loading the database in a local Mongo client and defining CRUD functions to make calls to the database. <https://pypi.org/project/pymongo/>
  + Dash provides view and control functions for the database accessed through the PyMongo modules. It has built-in data visualization, which it loads through a local port for web browser access, and allows callbacks to make calls to the PyMongo module, thereby filtering the database. <https://dash.plot.ly/>
* Optionally, Jupyter Notebook. <https://jupyter.org/install>

Instructions:

* Install MongoDB locally. In a command prompt, navigate to the directory containing the dataset in a csv file and run the command mongoimport --username="${MONGO\_USER}" --password="${MONGO\_PASS}" --port=${MONGO\_PORT} --host=${MONGO\_HOST} --db AAC --collection animals --authenticationDatabase admin --type csv --headerline --file "filename.csv" Replace user, pass, port, and host variables with local variables – the host and port values must match those in AnimalShelter.py. Replace “filename.csv” with the csv’s actual filename.
* Install mongosh. In the command prompt, start mongosh and use the admin database (use admin). Then run the command db.createUser({user:"aacuser", pwd:"aacpwd1"}), roles: [{role:"readWrite", db:"AAC"}]. The username and password set here must match those in AnimalShelter.py.
* Save MongoCrud.py, MongoConnect.py, and AnimalShelter.py in a directory with a driver file (e.g. a .py file, a Jupyter Notebook .ipynb) that will contain the Dash module that drives the application. In this repository, the Dash application is contained redundantly in two files: GraGraziosoSalvareDashboard.ipynb and app.py. In the same directory, create a folder called “assets” containing the Grazioso Salvare logo with the filename “logo.png.” (This folder and filename must match its reference in the Dash application’s html.Img object which has the source attribute src="assets/logo.png")
* Configure the AnimalShelter.py database connection variables to the host and port of your server, and the username and password of your read/write-authorized database user.
* Start the Dash application. This can be done with an .ipynb file by installing and starting Jupyter Notebook, navigating in-browser to the containing folder, and opening and running the file. If a .py file is used for the Dash application, navigate in a command prompt to the containing file and run python filename.py (replacing “filename.py” with the actual filename). Then, go to the URL provided in the terminal being used and the Dash application will load.

## Usage

Filter results by:

* Using the “Filter Data Table” dropdown to apply filters according to Grazioso Salvare’s recommended breed/sex combinations for particular rescue scenarios.
* Typing search queries into the “filter data…” cell in the second row of any column, and pressing enter. These queries are case insensitive and may be partial, but may not be inaccurate (e.g. “germ” in the “Breed” column will return all results involving German Shepherds, but “grem” will return no results.)

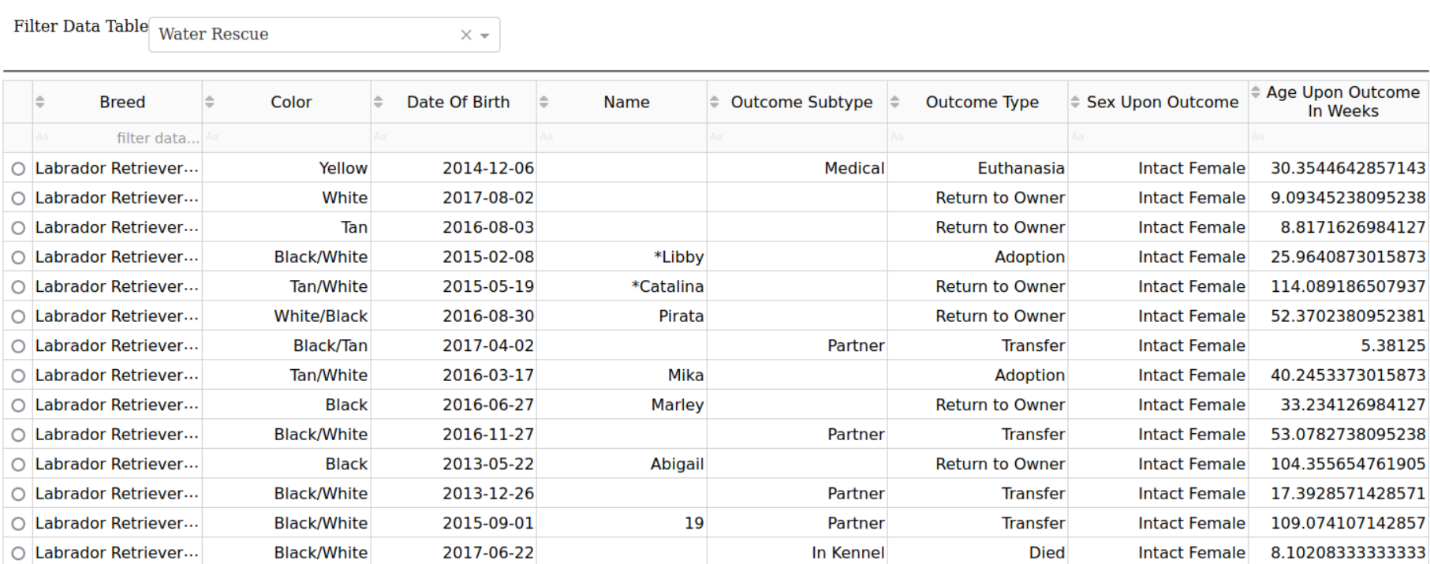
The data table may be sorted alphanumerically by any column’s values. It is paged, and each page contains 20 results. Change pages with navigation buttons or by entering a page number at the bottom-right of the table.

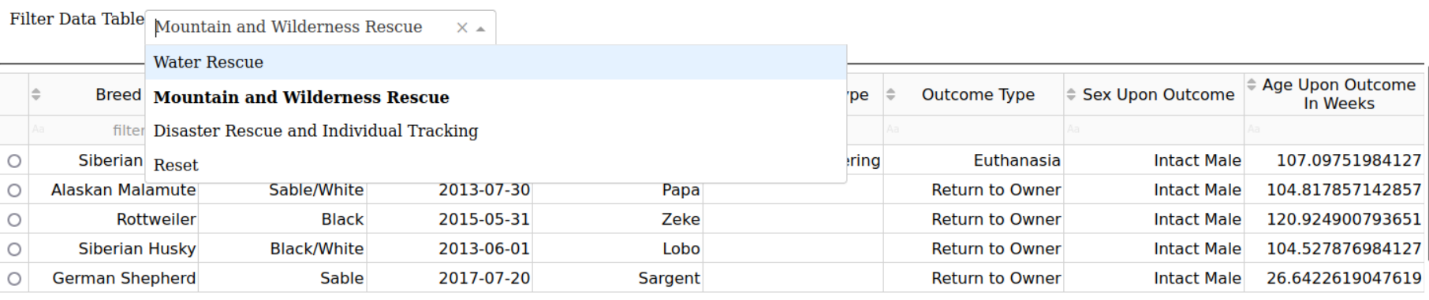
Selecting a row in the table, by clicking the radio button at its left, will center the map on the selected dog’s location. Clicking on the map pin will open a tooltip displaying the dog’s name and breed.

The pie chart below displays proportions of the filtered results’ breed or sex; select either of those options with the “Filter Pie Chart” dropdown menu. There are many breed values, so make sure your filter is fairly specific before using the pie chart, or it will not be too useful. A key to the right shows the values’ color codes, and hovering over any segment in the pie chart will also show the associated breed or sex.

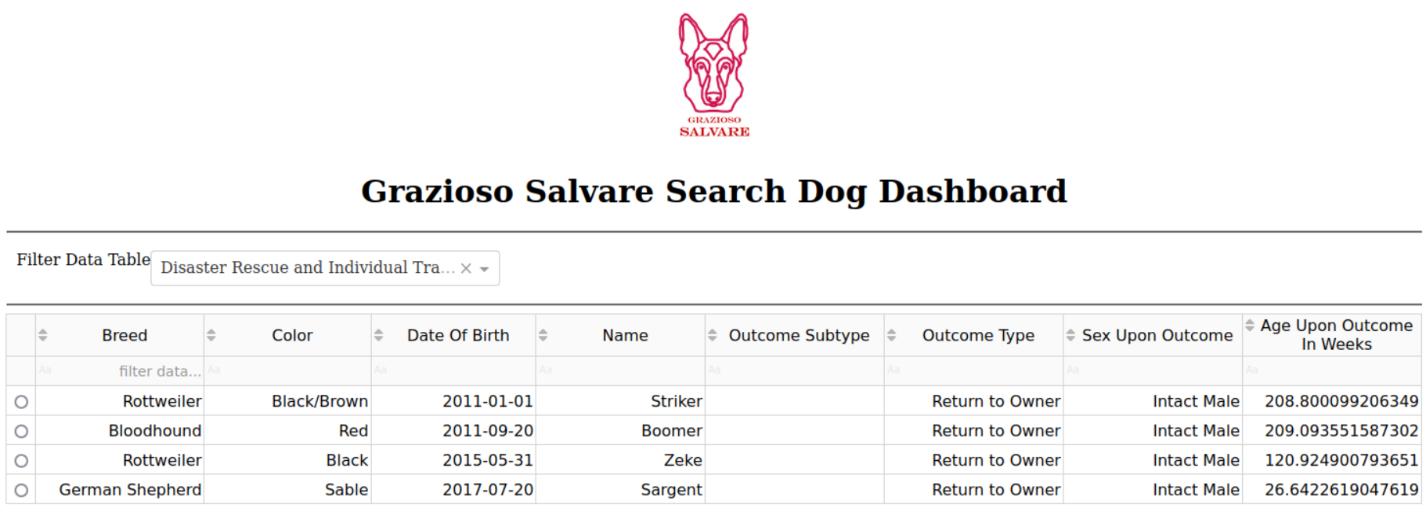
### Screenshots

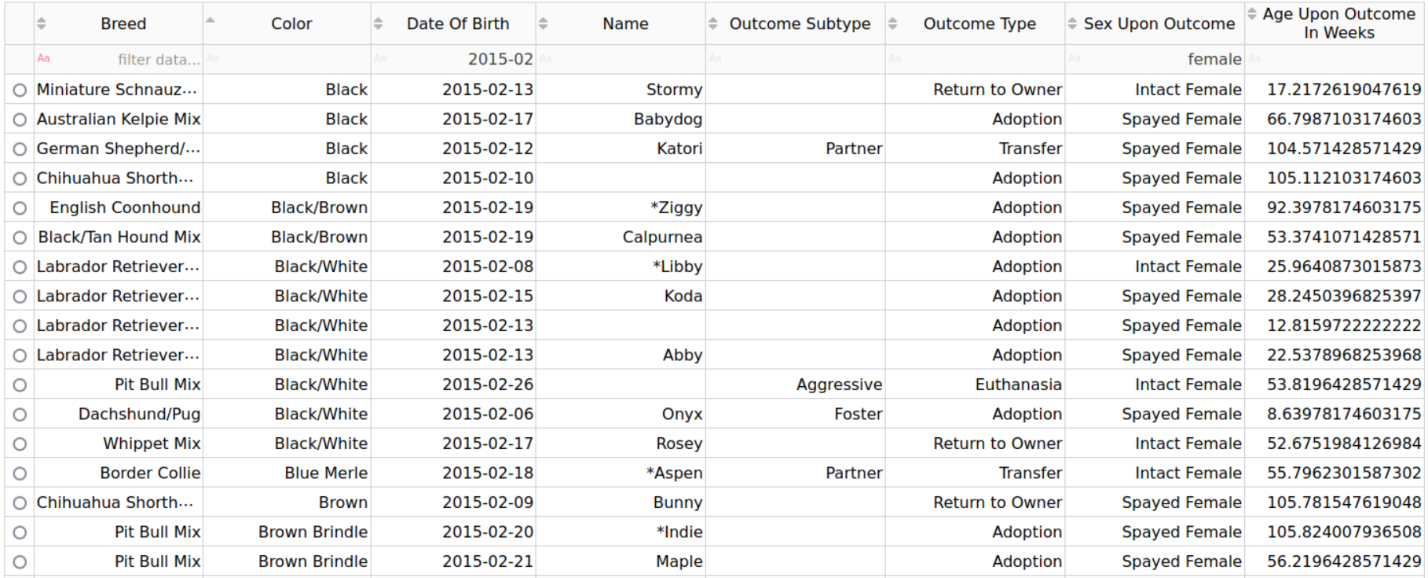
## The head of the application, showing the logo, title, and top of the data table.

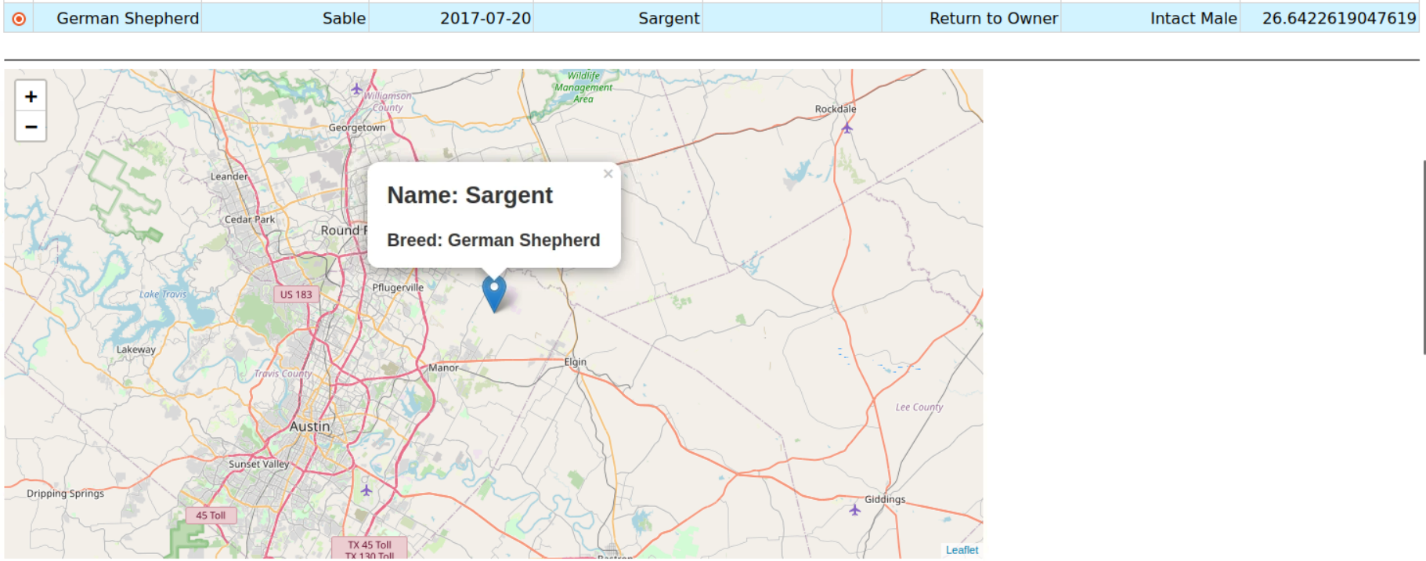
“Water Rescue” filter applied to the data table. Note that all results on the first page have the same breed, and all results have the same sex upon outcome.

The filter dropdown, with all results for the “Mountain and Wilderness Rescue” filter.

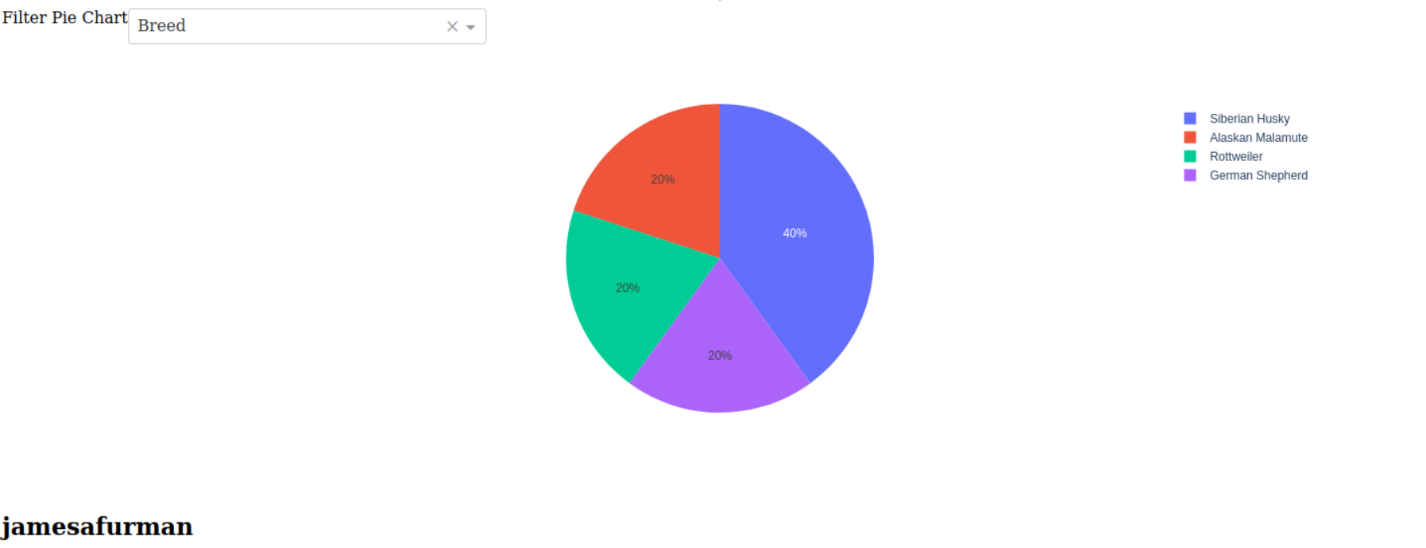
The “Disaster Rescue and Individual Training” filter.

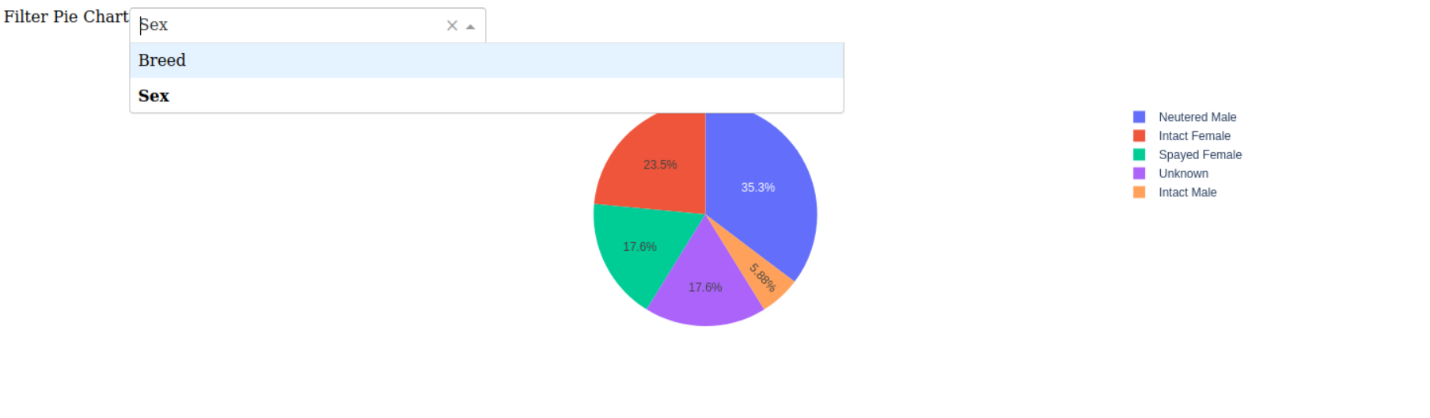


The data table display when no rescue filter is set in the dropdown menu, but text filters narrow the results down to female dogs born in February of 2015.

The German Shepherd named Sargent selected (note the radio button on the left of the row) and his location displayed on the map.

The pie chart displaying the proportions of breeds while the “Mountain and Wilderness Rescue” filter is applied. (The identifier at the bottom showing the dev’s name was requested by Grazioso Salvare.)



The pie chart displaying the sex proportions when another filter is applied. 

## Roadmap/Features

Future features:

* Clean up display formatting.
* Deal with messy pie charts when filter includes many breed values.
* Deal with accidental map navigation when mouse scrolling.
* Fully implement exception handling in CRUD modules (AnimalShelter, MongoConnect, MongoCRUD).
* Implement remote client access with SSL for authorization.
* Add security measures for connection variables.

## Contact

James Furman