# CS 340 PROJECT TWO README

## About the Project

## The purpose of this project was to create a web application dashboard for Grazioso Salvare - an international rescue-animal training company. The user-friendly dashboard serves as a way for Grazioso to work with existing data from the local Austin Animal Shelter database, identifying and categorizing specific dogs available for rescue training. Additional features, like the pie chart and map, were added to visualize results.

## Motivation

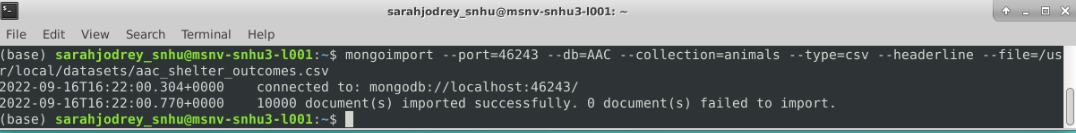
The motivation for this dashboard was to enable users to easily search for rescue dogs at the Austin Animal Center. The user filters by a desired rescue type and is provided with a visual representation of the breeds, and quantities, available. The map provides additional information regarding the specified animal’s location and basic details.

## Getting Started

To get a local copy up and running, please follow the below example steps.

1. Create and import the database AAC in MongoDB.

*mongoimport --port=”your port #” --db=AAC --collection=animals --type=csv --headerline --file=”location of AAC csv file”*

****

1. Create an ‘admin’ account with authentication.

*Text

Description automatically generated*

1. Create a user with read/write privileges specific to the AAC database and enable authentication.
2. Navigate to desired database - use AAC
3. *db.createUser({user: “username”, pwd:passwordPrompt(), roles[{role:”readWrite”, db: “AAC”}]})*
4. Enter password when prompted
5. Authenticate:

*Text

Description automatically generated*

\*\*Alternately, you can use MongoDB’s tutorial <https://www.mongodb.com/docs/manual/tutorial/create-users/> \*\*

1. Create a .py file for CRUD and initialize the MongoClient:

Company name

Description automatically generated

Text

Description automatically generated

A picture containing timeline

Description automatically generated

1. Run tests in Jupyter Notebook to ensure the functionality of the AnimalShelter() class.
2. Create a Dash web application by importing the appropriate classes and libraries.

Text

Description automatically generated

1. Initialize your AnimalShelter class and use Pandas to create the DataFrame from your read() function.

Text

Description automatically generated

1. Write the app.layout to format the web applications design including the radio button options, pie chart location, map location, and DataTable. Images can also be included, like the Grazioso logo in this project.
2. Define update\_dashboard using database logic for the radio options.

Graphical user interface, text

Description automatically generated

1. Define update\_graphs to display the filter options as a pie graph using Plotly and Pandas.

Graphical user interface, text, application

Description automatically generated

1. Define update\_map to display the Map based on the selected filter. It displays the data for the first animal in the table using Plotly and Pandas. This can be modified to show the location/Tooltip for more than one animal on the table. The Popup can also be modified to display desired information.

Text

Description automatically generated

1. Write the @app.callback() for each feature to be displayed on the application.
2. Upon completion of these steps, you should successfully run your own animal rescue dashboard. For any issues, please refer to the .py and .ipynb files.

## Installation

The tools required for this project include Jupyter Notebooks (<https://jupyter.org/install>) for testing, Python (<https://realpython.com/installing-python/>) for the CRUD module, and MongoDB(<https://docs.mongodb.com/manual/installation/>) as the database. Dash (https://dash.plotly.com/installation) was the framework used to build the web application, while Plotly and Panada were used to create the maps and charts to visualize the data. and Plotly were used to These can all be installed by following the instructions on their associated websites.

## Usage

This web application meets Grazioso Salvare’s needs via four essential functions - data table, radio button filters, pie chart, and map. First off, is the data table that allows the user to filter through all the animals currently available at the AAC. This is achieved through the search bar below each header on the table. Second, is the use of radio buttons to filter by rescue type - water, mountain/wilderness, disaster/individual tracking, and reset/clear. The results displayed correlate with the dog sex, breed, and age best suited for the designated rescue type. The reset/clear option returns the data to the starting, unfiltered, state. The pie chart displays the percentage of breeds available in response to the chosen rescue filter. Finally, the map will display a marker with the animal’s location along with a few essential details.

The images provided below exemplify the four functions in action.

***Filter Data Table by Header***

***A picture containing timeline

Description automatically generated***

***Water Rescue Filter***

***A picture containing text

Description automatically generated***

***Mountain/Wilderness Rescue Filter***

***Text

Description automatically generated with medium confidence***

***Disaster Rescue and Individual Tracking Filter***

***A picture containing text

Description automatically generated***

***Reset/Clear Filter***

***Text

Description automatically generated with low confidence***

## Roadmap/Features

Future releases may include more filter options, photos of animals, and rescue success rates.  
Features:

* Responsive map of animal location
* Responsive visual aid of available breeds by rescue type
* Interactive filtering of animals by breed, sex, age, location, etc.

## Contact

Sarah C Jodrey