# Joseph Langley CS 340 Project 2 README

## About the Project/Project Title

*This software application works with existing data to identify and categorize dogs with certain profiles. The web application is designed to provide insights and visualization for data from an imported database. It offers features for filtering data based on rescue types, interactive data viewing, and displaying geo-location information for selected entries. The database is referenced through python CRUD functionality with MongoDB. For this README, the Austin Animal Center (AAC) database is utilized.*

*Project Title: Grazioso Salvarete Animal Shelter Dashboard*

## Motivation

*Grazioso Salvarethe, the international rescue-animal training company requesting this application creation, would like to identify dogs that are good candidates for search-and-rescue training. Identifying dogs that meet specific requirements for candidacy (i.e. less than two years old or certain breeds for certain rescue training operations) will help the company establish better training and dog locating. By visualizing key information, the dashboard is user-friendly and intuitive helping to reduce user error and training time.*

## Installation

*To run the Animal Shelter Dashboard, the following tools and dependencies need to be installed:*

* *Python (version 3.12 found at* [*https://www.python.org/downloads/*](https://www.python.org/downloads/) *)*
* *Jupyter Notebook (latest version found at* [*https://jupyter.org/install*](https://jupyter.org/install) *)*
* *MongoDB (latest version found at* [*https://www.mongodb.com/docs/manual/installation/*](https://www.mongodb.com/docs/manual/installation/) *)*

*Once these are installed, you can install Dash and Pandas through the terminal by following the instructions listed here:* [*https://dash.plotly.com/installation*](https://dash.plotly.com/installation) *. Dash installation through the terminal includes the Plotly graphing library, also essential for running the web application.*

## Getting Started

*The software design pattern used for this application is the Model-View-Controller (MVC) pattern. The ‘M’ represents the data logic being used, the ‘V’ is the user interface, and the ‘C’ is the intermediary between the two. In this application, the model is the most developed, being the ‘AnimalShelter’ class that encapsulates the logic to interact with MongoDB and holding the methods to implement CRUD operations. The view is developed using Dash and Plotly. And the controller, currently, is the python testing done through the Jupyter notebook.*

*To access the application, the following steps need to be performed first through the terminal.*

1. *Enter the terminal and change the directory to the location of the file that is going to be used as the database, in this case, a ‘.csv’ file is used. Others could be used but the input would need to be modified appropriately.*
2. *Enter the terminal and import the csv file “aac\_shelter\_outcomes.csv”.*

A screenshot of a computer

Description automatically generated

1. *This code uses an authenticated user “aacuser” and password but could be rewritten for a different user inside the “animal\_shelter.py” file.*

A screenshot of a computer screen

Description automatically generated

1. *Ensure the port and host are correct by accessing the current ones in the terminal.*

*A computer screen with text

Description automatically generated*

*Following these steps and after proper installation as discussed, the beginning of the python code established a connection to the MongoDB database opened through the terminal. The ‘\_\_init\_\_’ method acts as the constructor for the ‘AnimalShelter’ class to ensure it’s ready to perform CRUD operations.*

*A computer screen shot of a computer

Description automatically generated*

1. *Creating the Dash application using Jupyter notebooks that connects to the python code above. Starting this is done by importing essential libraries (i.e. pandas, plotly, etc.)*

*A screenshot of a computer

Description automatically generated*

1. *Connect to the database.*
2. *Implement any desired stylistic changes to the Dash application. I prioritized visual appeal which took time to go through documentation to determine the appropriate styles that I wanted and how to implement. For example, changing the names of the header columns and hiding ones that were irrelevant, importing the company logo as a link and creating a good title next to that, and changing the color schemes of the dash table.*
3. *Implement the required callbacks for easy visualization. This included radio items that filter data based on the requirements of the company, a pie chart that displayed information about the data currently being displayed in the table, and a geo-location interface that displayed information about a currently selected animal. For me, this by far took the longest time. Ensuring that syntax and logic were correct took detail review of Dash and Plotly documentation.*

## Usage

*The dashboard has several distinct areas that each required their own unique coding and styling.*

1. *The header includes the title of the dashboard, the author, and the logo of the company with a link to* [*https://www.snhu.edu/*](https://www.snhu.edu/) *.*
2. *The radio items display the buttons that can change the data being displayed in the table and in the pie chart.*
3. *The data table adjusts according to the input from the radio items or the sorting option.*
4. *And finally, the map displays geolocation information about a selected animal.*

### Header Code

*A screen shot of a computer code

Description automatically generated*

### Header Display

*Stylistically, I decided to make the logo smaller and appearing all the way to the left of the screen with the title and author appearing directly next to it.*

*A close-up of a white background

Description automatically generated*

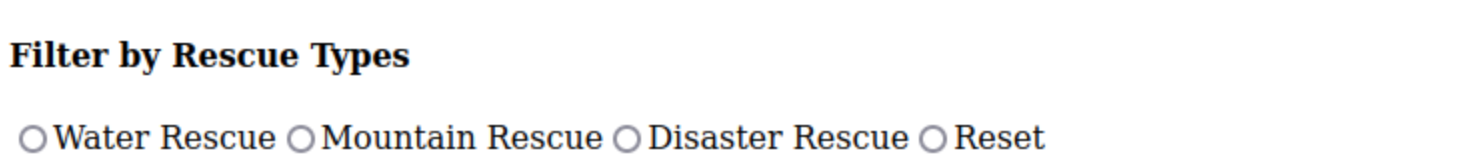
### Radio Items Code

*This code showcases the callback where the radio items are given their functionality to change the data based on certain filter conditions. The testing done to ensure this was accurate was rigorous. Beyond making sure the buttons actually filtered data, I also wanted to make sure it was correct so went to the original database to make sure I was correctly spelling the information that needed to be filtered. Especially in the water rescue filter, the Newfoundland dog had several variations that I wanted to make sure were included (even though after filtering the other information, only Labradors fit all the requirements from this database).*

*A screenshot of a computer program

Description automatically generated*

### Radio Items Display

**

### Dash Table Code

*The dash table I wanted to style slightly differently colored than the rest of the area to make it easier to view and add some better visualization to the page. I also changed the names of the columns being displayed and hid columns that weren’t essential. There’s also functionality for sorting each of the columns and selecting a row (for geolocation usability). I also made the decision to only display 5 results at a time on each of the pages to hopefully make the interface load faster.*

*A screen shot of a computer code

Description automatically generated*

### Dash Table Display (before filters applied)

*A black and white table with white text

Description automatically generated*

**Pie Chart Code**

*I spent the longest time on the pie chart implementation. The code was mostly straightforward, but getting it to display only percentages that fit inside the box and update each time the radio filter buttons were toggled took some syntactical understanding that I had to delve to find.*

*A computer screen shot of a computer code

Description automatically generated*

**Geo-location Code**

*A screenshot of a computer code

Description automatically generated*

### Full Display (water rescue filter applied)

### A screenshot of a computer Description automatically generated

### Full Display (mountain rescue filter applied)

### A screenshot of a computer screen Description automatically generated

### Full Display (disaster rescue filter applied)

### A screenshot of a computer screen Description automatically generated

## Contact

Your name: Joseph Langley