# CS 340 README Template

## About the Project/Project Title

Grazioso Salvare is an animal-rescue training company that has partnered with the local animal shelter to find dogs that meet specific criteria which can be trained to be rescue animals. This project is an interactive dashboard that can be used by Grazioso Salvare staff to search the Austin Animal Center database for qualified dogs. It includes a filterable data table and two widgets (a geolocation map and histogram) that can be used to analyze the data and aid in their selection.

## Motivation

The Austin Animal Center houses a large number of animals (over 10,000 in its database) which can be very difficult to sift through to find potentially trainable dogs. The database is built in MongoDB which is a powerful tool for storing, filtering, and sorting data but it can be difficult to use for those who are less technologically inclined. A user-friendly dashboard that can obtain, filter, and sort the data then present it is an easy to read format is a better option. This project fills that need by granting access to the information in the database while also making it easy to use.

## Getting Started

Use the following information to get started using this project:

## Installation

Required tools:

* Python 3.6 or higher – This project is built using the Python language and needs it in order to run.
* Pymongo 3.12 or higher – A library of tools designed to work with MongoDB. Allows for interaction between Python and MongoDB
* MongoDB – A NoSQL database. In this case, the animal shelter’s animal database is a MongoDB.
* Plotly (5.18.0 or higher) – A Python library that eases the process of data visualization. It helps to create graphs and charts from data.
* Dash (2.14.2 or higher) – A Python library that makes it possible to easily build interactive web pages in Python. It integrates well with Plotly to create web pages that interactively display data in various formats (e.g., tables, charts, graphs, etc.).
* Pandas (2.1.3 or higher) – A Python library that allows for easy conversion of various data formats into a data table used by Plotly and Dash.
* Jupyter Notebook (7.0.6 or higher) – A web-based development environment for creating, testing, and running projects like this.
* AnimalShelter CRUD module – The middleware CRUD software that creates, reads, updates, and deletes information from the MongoDB based on interactions on the web-facing user interface.

For installation instructions for the above tools, please refer to those sources:

* [Python](https://www.python.org/downloads/)
* [Pymongo](https://pypi.org/project/pymongo/)
* [MongoDB](https://www.mongodb.com/docs/manual/installation/)
* [Plotly](https://pypi.org/project/plotly/)
* [Dash](https://pypi.org/project/dash/)
* [Pandas](https://pypi.org/project/pandas/)
* [Jupyter Notebook](https://jupyter.org/install)

Installation and use information for the AnimalShelter CRUD module can be found in its ReadME file.

Final Setup:

After installation and setup of the prior tools, the final setup of this project is as simple as opening and running the .ipynb file within Jupyter Notebook. A link to a locally hosted web page will be shown in the Jupyter Notebook output. Clicking this link will bring the user to the functioning tool.

## Usage

After opening the dashboard, the user will be presented with a filterable data table, a geolocation map, and a histogram. There are 4 preset filters that the user can select:

* Water Rescue
* Mountain Rescue
* Disaster Rescue
* Reset

Clicking on any of the first three will filter the data table to the animals that meet the specific criteria for those rescue animal types. The last filter will reset the data table to the entirety of the Austin Animal Center database.

Each column in the data table is sortable (by clicking on the sort icon next to the column name) and filterable (by adding filter text in the cell below the column name). By default, the filtering is case-sensitive. To turn this off, click the case sensitivity icon next to the filter text.

The geolocation map shows the location of the selected animal. To select a new animal, click the circle at the left of the row for the animal in question. Doing so will update the geolocation map to the newly selected animal. Hovering over the pin on the map will display the animal breed while clicking on the pin will display its name.

The histogram will display the count of each breed that is in the data table based on the currently selected filter. Changing the filter will update the histogram to the animals in that filter. The histogram can be zoomed in on to change the view and focus on specific breeds if needed.

**Screenshots**

Default data filtering:

A screenshot of a computer

Description automatically generated

Water Rescue filtering:

A screenshot of a computer

Description automatically generated

Mountain Rescue filtering:

A screenshot of a computer

Description automatically generated

Disaster Rescue filtering:

A screenshot of a computer

Description automatically generated

## Roadmap/Features

This project is currently deployed locally. Future updates would change it so it is deployed in the cloud so it is accessible from anywhere. Additionally, future updates would also improve the speed of the filtering and data table updating.

This project was built using many small iterative steps. First, the background database was constructed and setup to house data from the Austin Animal Center. Next, the Python middleware was built to allow for interacting with the database from a client facing dashboard. Next, the dashboard was built and connected to the middleware. This was done in a step-by-step fashion focusing on each part of the dashboard (i.e., data table, geolocation chart, and histogram).

There were a few challenges encountered while building this project. First, the dashboard would throw an error upon initial load that there was no data for the histogram. Despite this error, the histogram loaded correctly and functioned as expected. Upon reviewing the code, it was discovered that the callback used to update the histogram was running upon loading the dashboard. This was happening prior to the data table being fully populated so it was actually receiving no data on initialization. After the data table was fully populated, the histogram callback was then run again which would then populate the chart with the correct data giving the appearance that there was no issues despite the thrown error. To fix this, an if statement was added that would check the data table data before doing anything. If it was empty, the callback would end and return nothing. This prevented the error from happening again.

The second issue was more cosmetic in nature. The dashboard originally was built with a pie chart instead of a histogram and showed the number of each breed in the currently filtered data. Because there is a very large number of different breeds in the entire database, this was causing a huge number of pie slices which was rendering strangely on the screen and made it difficult to use. This was dealt with by changing the chart to a histogram. While the histogram is still showing the same data, it does so in a way that is more conducive to displaying a large number of distinct items.

## Contact

Your name: Quintin B. Rozelle