**CS-340 README**

**Required Functionality:**

The client, Grazioso Salvare, identifies dogs meeting certain conditions that might be suitable for search-and-rescue training. To support this endeavor, a non-profit agency operating five animal shelters around Austin, Texas has agreed to provide the client with data from these shelters. A dashboard that will present the information in a clean, logical manner is necessary to make good use of the data being provided. Filtering options provided by the client can be enabled with a single mouse click and the map and graph widgets updated based on the data being presented.

Below, the starting state of the dashboard (Figure 1) can be seen:

A screenshot of a computer

Description automatically generated

A screenshot of a map

Description automatically generated with medium confidence

(Figure 1)

The screenshots below (Figures 2-4) display the other filters being used:

A screenshot of a computer

Description automatically generated

(Figure 2)

A screenshot of a computer

Description automatically generated

(Figure 3)

A screenshot of a computer

Description automatically generated

(Figure 4)

**Technology Used:**

MongoDB: <https://www.mongodb.com/>

Python: <https://www.python.org/>

PyMongo: <https://www.mongodb.com/docs/drivers/pymongo/>

There are several reasons that MongoDB was chosen as the model component in this project. The primary reason is flexibility. Relational databases leverage predefined schemas that might make mid-project changes too difficult to perform, limiting development options throughout the process. Also, the ease of interfacing with Python with a library like PyMongo makes “spinning up” a project extremely fast. Once a database is deployed (a very simple process with MongoDB), creating the basic methods for interacting with that database takes mere minutes. Security and authentication can also be handled through the Python interface thanks to PyMongo.

Dash: <https://dash.plotly.com/>

Dash is a Python framework that facilitates the development of dashboard web applications with a strong emphasis on data visualization. For this specific project, Dash provided multiple benefits. For one, the primary way data is viewed is through a data table that breaks out key data attributes for each record in the database. With only a few lines of code, this data table was implemented and customized to create a wonderful user experience. Easily accessible queries were created to refine the data per specific customer requirements, and additional filters can be created by the user. Widgets, specifically charts and graphs, further illustrate key points about the data being viewed while even something that looks more complex (such as a geolocation widget) is easily implemented.

**Project Completion:**

This project was completed in several phases. Before work began on the dashboard, a CRUD (create, read, update, delete) module was created in Python. This was utilized in the dashboard code to run a “retrieve all” query that brought all of the data in from the MongoDB database.

The first area of focus was the primary data table that displayed records for the user. Some customization was done to enhance the user experience, such as the ability to select a row of data, the first row of data displayed was selected by default, filtering, hiding unnecessary attributes, and reducing the number of records displayed at once.

To make searching more efficient, queries were developed so that a user could simply select one to limit the data. These selections were tied to radio buttons near the top of the data table, and limited results based on animal type, breed, sex, and age.

Dashboard widgets were created to provide additional features for the user. One widget in particular was very interesting – the interactive map. Whenever a row of data was selected, the pin on the map would update to display that animal’s location. A histogram was also added which broke down the breeds of animals into buckets and displayed a count for each one.

**Challenges and Solutions:**

During this project, I encountered several challenges. One of the first was implementing the filtering options that were tied to specific radio buttons on the dashboard. This project was the first time I worked with the Dash framework, so understanding that application’s layout and logic were separated was confusing at first. Fortunately, the template file provided great examples and the documentation for the framework filled in any gaps. From there, it was fairly simple to refer back to the documentation on MongoDB queries and Python functions to make sure that the correct documents were being read.

A challenge that I was unable to overcome due to time constraints was the apparent duplication of query results in my application. It appears that, for some reason, a result would appear three times when there should only be a single appearance for each. I hope to do some additional research to understand why that is occurring.