**Project Two - README**

Katrina S. Gilliland

Global Campus, Southern New Hampshire University

CS-340-R4865 Client/Server Development

Dr. Reuben Wilson

April 21, 2024

**Project Two - README**

**Description of Required Functionality**

The purpose of this software application is to supply Grazioso Salvare, an international rescue-animal training company, with an interactive dashboard they can use to identify and categorize available dogs that might be ideal candidates for a comprehensive search-and-rescue training program they offer. The client-facing web application includes and connects to a database that stores existing data from animal shelters within the Austin, Texas area. Grazioso Salvare’s branding requirements and dashboard widget requirements were thoroughly analyzed and considered throughout the development of this application. Firstly, as far as branding goes, Grazioso Salvare required that their logo be included on the dashboard along with a URL anchor tag to their home page (‘www.snhu.edu’). A screenshot of this addition can be found here: **A red line drawing of a dog

Description automatically generated**

Second, Grazioso Salvare requested that a unique identifier be added to the dashboard to credit myself, the author. This addition can be seen in the following screenshot:

**A screenshot of a computer

Description automatically generated**

Once these branding requirements were satisfied, the dashboard widget requirements were developed. The first dashboard widget to be built was an interactive filter option so users can have the ability to filter through the Austin Animal Center Outcomes data set stored within the database. Specifically, radio buttons were added to allow users to filter through data set by ‘Water Rescue,’ ‘Mountain or Wilderness Rescue,’ ‘Disaster Rescue or Individual Tracking.’ When a filter is selected, the data table dynamically responds and shows the results from that search filter (i.e., if the user selects “Mountain or Wilderness Rescue’ the data table will show any existing data associated with that filter). Both the geolocation chart and pie chart created for visualization purposes, included underneath the data table, will adjust as well.

Here is a screenshot of the data table before a filter selection is made. By default, the ‘Water Rescue’ radio button is selected: **A screenshot of a computer

Description automatically generated**

Here is how the data table changes when the ‘Mountain or Wilderness Rescue’ filter option is selected instead:

**A screenshot of a computer

Description automatically generated**

As noted, when a filter option is selected, the pie chart and geographical chart updates as well. To show this in action, here is a screenshot of the pie chart and geographical chart for the ‘Mountain or Wilderness Rescue’ filter:

**A screenshot of a computer

Description automatically generated**

Watch what happens when we change the filter to ‘Disaster Rescue or Individual Tracking,’ as anticipated, the pie chart and geographical chart update once again:

**A screenshot of a computer

Description automatically generated**

**Tools used to Achieve Functionality**

Many tools were vital for achieving the required functionality for this project. MongoDB, the model component for this application, can be credited for providing a streamlined approach to data manipulation thanks to its use of JSON-like documents for data storage. Since JSON closely resembles ‘dictionaries’ and ‘lists’ in Python and native Python data structures can be directly used in MongoDB, it is much more intuitive to manage data. Further, MongoDB provides pymongo, the official driver for Python. Pymongo offers incredibly useful APIs for interacting with MongoDB databases; for example, pymongo allowed this application to have CRUD functionalities, such as permitting users the ability to ‘read’ data from the stored data set.

The Dash framework was also essential for the success of this program. The Dash framework enabled user interaction with the database through its support of callback functions. User interactions, such as clicking a radio button to select a search parameter, triggers a callback function. In the case of the Grazioso Salvare application, when a user selects a search filter such as ‘Water Rescue,’ the callback function triggers the webpage to display a data table, pie chart, and geographical chart specific to that filter. Additionally, the Dash framework integrates with Plotly, a Python visualization library that enables the creation and use of interactive charts and graphs, such as the pie chart developed for this program.

The most useful resource for tackling this project was referring to the Dashboard Specifications document. The Dashboard Specifications document helped me align my work with the business needs of Grazioso Salvare and gave me crucial insight into how the interactive filter options should look and operate. I also utilized a digital resource for Python’s matplotlib library to aid in making my application more colorful and physically appealing. Using the link below, I utilized a handful of CSS colors to personalize the program.

**Digital Resource:** https://matplotlib.org/stable/gallery/color/named\_colors.html

**Steps Taken to Complete Project and Challenges Encountered**

To complete the project, I created a IPYNB file in Jupyter Notebook. Once the file was created, I set my environment up to include the necessary Python libraries and modules and established a connection to the MongoDB database. I was able to access MongoDB by importing the ‘pymongo’ module and initializing the AnimalShelter class I created previously. After the system was successfully set up to retrieve data from MongoDB, I used the Dash Framework within Jupyter Notebook to define the application layout; through the Dash Framework, I was able to add in HTML components, create an interactive chart and map (powered by callback functions), and ultimately, run the application to ensure it is working as expected.

I encountered many challenges while working on this project, however, the biggest difficulty I had was getting my geographical chart/map to appear. The issue kept stemming from my callback function being incorrectly set up. After awhile of troubleshooting, I found the issue was that my update\_map callback function was not set to update based on the users’ ‘selected\_rows.’ Once I made this change, my map appeared on the webpage without error and began functioning properly.