**Project Two README**

**Functionality**

This project, made for Grazioso Salvare, helps identify dogs that may be good candidates for a variety of rescue situations using data provided by a local animal shelter. The application was built using an MVC framework to allow for further customization and the easy addition of extra functionality. The model used in the application consists of a Python module that is used to communicate with the MongoDB database that houses the data from the animal shelter. The view used in the application was made using Plotly Dash, a multi-language framework used to build interactive and responsive HTML pages. In addition to providing a toolset for the easy manipulation of HTML elements, Plotly Dash also includes several useful tools for data analysis – like creating the pie chart and map used in the dashboard for this application. The controller logic used in this application comes in the form of callback functions the Plotly Dash module. These callback functions enable the real time updating of the pie chart when search filters are applied, of the geolocation map when different rows are selected, and of individual data table cells when clicked by the user. The controller allows for more specific uses of basic CRUD functionality as provided by the model.

Below are several screenshots of the application in action – displaying the several different search filters being used and how the pie chart and map change with the filters.

Default dashboard state:

A screenshot of a computer

Description automatically generated

Filtering by animals eligible for Water Rescue:

A screenshot of a computer

Description automatically generated

Filtering by animals eligible for Mountain Rescue:

A screenshot of a map

Description automatically generated

Filtering by animals eligible for Disaster Rescue:

A screenshot of a computer

Description automatically generated

Using the Reset option to return the dashboard to its default state:

A screenshot of a computer

Description automatically generated

**Rationale of Tooling Used**

For the model portion of this application, MongoDB was chosen to store the animal shelter data paired with a Python module used to programmatically communicate with the database. This Python to MongoDB communication is enabled by the pymongo library – the official Python driver for MongoDB. MongoDB was chosen because it uses a non-relational model for storing data because the simple syntax used to interact with the data enables rapid development and because the schema-less nature of MongoDB means that even if some of our data is malformed or is missing values in specific fields, our database can easily adapt and store the data despite any flaws in formatting or completeness. This is important because our animal dataset includes animals that may not have a name, a known location, or a determinate outcome. In a SQL database, these potential variations in data formatting might require the developer to set up a precise schema, which may break if some edge case is missed. In contrast, the NoSQL structure of MongoDB allows us to store our data without fear and leaves it to us to ensure that any necessary formatting standards are met.

For the view and controller portions of this application, Plotly Dash was chosen to create the user dashboard and to add interactivity and responsiveness to the dashboard using callback functions. For the view segment, Plotly Dash was chosen because it allows the quick construction of elegant HTML pages in a format that may feel more familiar to a non-web developer as HTML elements and their children are created using dot notation and function nesting. Each HTML element can be assigned a unique ID which can be used to enable the use of a callback to perform some action(s) with that element. For the controller segment, Plotly Dash framework allows the use of callback functions that operate using data from one to several HTML elements or an external data source as inputs and that output results to another HTML element. Each HTML element used in a callback function is identified by its unique ID and is classified as an Input or Output in the callback decorator’s signature which defines the variables used in the callback and where the output should be placed. In this application, the callback functions are used to highlight selected cells, update the geolocation map based on which row is selected by the user, and update the pie chart based on which search filter (if any) is selected by the user.

**Steps Taken to Complete the Project**

To complete this project, I began by adding the HTML elements for all the widgets necessary for the client’s desired functionality. In practice, this meant adding the Grazioso Salvare logo as an image element at the top of the dashboard that can be clicked to direct the user to [www.snhu.edu](http://www.snhu.edu). As the UI designer, I chose to scale the logo to 25% as it would otherwise take up most of the view height on most monitors and browsers. Beneath the logo, I added my name as a unique identifier to mark the work as completed by me, as requested by the client.

Beneath my name and between two horizontal rules (horizontal dividers), I placed four radio buttons, each with a corresponding label, to enable filtering the dashboard’s data table based on criteria defined by the client.

While the data table was provided in the base code, my design limits the displayed results to just ten results per page, splitting the results into multiple pages rather than displaying all results on a single page which would sometimes require a lot of scrolling to reach the bottom.

The base code allocated space below the data table for a chart and a geolocation map side-by-side. However, neither the pie chart nor the geolocation map was displayed until I added functionality in the associated callback functions.

After completing the front-end design for the dashboard, I moved to constructing the logic for the callback functions. For its interactivity and responsiveness, this application uses four callback functions, each of which is associated with a different HTML element on the dashboard. The update\_dashboard callback function uses the radio buttons as input and updates the data table with the data returned by pymongo queries that were constructed based on animal requirements for each filter type as defined by the client. The update\_graphs callback function uses the data table data to create and update a pie chart which displays the names and counts of each breed on the data table. The update\_styles callback function changes the background of a data table cell to a light red when the user clicks on the cell. The update\_map callback function uses the data table’s selected\_rows parameter to extract the latitude and longitude associated with the selected data entry and then place a teardrop marker on the geolocation map. When the marker is placed, the map automatically re-centers itself on the marker. When the user hovers over the map marker, the animal’s breed is shown as a tooltip. Likewise, when the user clicks the map marker, the animal’s name is shown as a popup dialogue.

With the front-end design and the backend functionality complete, the project was finished.