**Grazioso Salvare**Allow customer to identify dogs that are candidates for search-and-rescue training.

**Required Functionality**

The required functionality is to provide the customer with a working application to which they can easily find what animals of a certain species are acceptable to be used for rescue groups. This functionality is achieved by adding a dropdown menu to the dashboard. When selecting an option from the dropdown menu it will update the table shown, it also updates the map and pie chart below which is shown by breed. After selecting a particular rescue group from the dropdown menu selecting and will update the map to show the location of that animal. Choosing “reset” from the dropdown menu will reset the table back to defaults. The dashboard also features the filter and sort functionality which is not shown in the screenshots as it was explicitly required in the rubric.

Below are the screenshots showing different results from the functionality. The requirements called for including the customers logo and unique identifier in all screenshots, so it was required that the browser be zoomed out. I have also included a screenshot show a close-up of the dashboard, but it does not include the whole entire view. Each screenshot is labeled showing the applicable functionality.

|  |
| --- |
| Default Screen |
| A picture containing graphical user interface  Description automatically generatedWater Rescue |
| Graphical user interface, application  Description automatically generated with medium confidenceMountain Rescue |
| Graphical user interface  Description automatically generated with medium confidenceDisaster Rescue |
| A picture containing application  Description automatically generatedReset View |
| A picture containing table  Description automatically generatedClose-Up View |

**Tools**

There were a few tools used to achieve the desired functionality required by the customer. Below is the list of tools required, a description of the tool and reason for its use. Each tool served a specific purpose and had advantages which is explained below.

**MongoDB –** The MongoDB is an open-source document database that provides an easy way for storing

data. It is scalable and integrates very well with Python via the PyMongo library. In the case of this dashboard PyMongo made it very easy to interact with data stored in the database. Setting up CRUD methods using PyMongo was no problem at all, which paved the way for retrieving data for the dashboard. With the use of PyMongo, setting up the interaction between the dashboard and the database would have been a complete nightmare.

**Dash –** Dash is an open-source framework used for building data centered interfaces. Dash assists data

scientists with building analytical web applications without the need for advanced web development knowledge. Dash is made up of three key components: Flask - Provides web server functionality, React.js – Builds user interfaces for web pages and Plotly.js – Generates charts used in dashboards (Castillo, n.d.). Dash is used in Python for building interactive dashboards, leveraging Dash with Plotly allows a developer to create dashboards in fractions of the time it used too. It only requires a basic knowledge of web technologies.

In our dashboard, using Plotly made it fairly easy to develop the dashboard with the pie chart, with Dash creating the data table was incredibly simple, then when you couple that with the PyMongo, creating a data centric dashboard was quite easy.

**Jupyter -** Jupyter is a web-based development environment providing data scientists an easy way of

interacting with data, it is sort of like a IDE and is used very heavily in education for teaching students a particular skill set. It is also very good for testing an application with the need of setting up a development environment, in other words it is a quick way of getting up and running. For our application it was used to develop the code for running our dashboard and sharing our code with the instructor (Science, 2020).

**Completing The Project**

The steps to complete the dashboard were straightforward, it involved a series of research, implementing the functionality and testing the code. I skip the creation of the database and AnimalShelter class, I start out with the creation of the data table. Below are the steps used to complete the project.

1. To create the data table, I used PyMongo to read the data from the MongoDB database, then use Dash to render the data table.
2. Added features to the data table such as making it filterable, sortable and pagination. Each column a user-friendly name and using callbacks the selected row is highlighted in orange.
3. Using Dash Leaflet incorporated a map into the dashboard which identifies the location of the selected animal using a marker. The first row in the data table is selected by default and the location of the animal is shown on the map. Selecting the marker will show the name of the animal.
4. Added the dropdown menu above the data table for displaying animals meeting the required criteria. Researched to find the best way for executing a data lookup for each of the rescue groups. Then implemented and tested the code in the MongoDB console until the correct data was shown.
5. Created the callback functions that tied the dropdown menu to the data table and double checked that after filtering for the different rescue groups the map still worked.
6. Research the best way of implementing a pie chart into a Dash dashboard, using the Plotly website found a way to render a pie chart and created the container for the chart adding the appropriate styling to render it on the same row as the map.
7. Using chainable callbacks tied the pie chart callback to the data table callback so that whenever the data on the table changed it would update the chart accordingly.
8. Tested the functionality of the callbacks and corrected a few issues with getting the chart to correctly render. There we some issues on how to select the criteria and render the chart correctly.
9. After figuring out how to make the data show correctly tested the functionality of the components. Selecting a particular rescue group renders the data correctly, updates the pie chart which shows the breeds, then selecting a row from the data table would show the location on the map.
10. Finally, added the image to the dashboard above the unique identifier, centered the image and scaled the size of the image down to a size that was more in line with the dashboard.

**Challenges**

The challenges I encountered during this project was mainly centered around getting the pie chart

to render correctly. I was able to determine the correct syntax to use early on but could not really find any documentation on how to render information from a database after being filtered. Once I found how to get the pie chart to render correctly, I had an issue with it showing an error message on initial page load. I corrected that by using an if statement which essentially would not run the function on page load.

The other issue I had was with the dropdown menu, I tested running the filter queries inside the MongoDB database and was able to get results from the database. I originally assigned the queries to variables but whenever the variables were added to the shelter method it would not allow the queries to execute. I finally figured it out by adding the queries straight into the methods and then everything worked correctly.

**Code**

Below are the screenshots showing the code used to generate the dashboard, it didn’t say it was a

requirement in the instructions but in the rubric, it mentions a screenshot of the code. The images below start at the top of the Jupyter file and proceeds down.

|  |
| --- |
| Graphical user interface, text, application, email  Description automatically generated |
| Graphical user interface, text, application  Description automatically generated |
|  |
| Text  Description automatically generated |
| Graphical user interface, text  Description automatically generated |
| Graphical user interface, text  Description automatically generated |
| Graphical user interface, text, application, email  Description automatically generated |
| Graphical user interface, text, application, email  Description automatically generated |

# References

Castillo, D. (n.d.). *Develop Data Visualization Interfaces in Python With Dash*. Retrieved from realpython.com: https://realpython.com/python-dash/#:~:text=in%20this%20tutorial.-,What%20Is%20Dash%3F,requiring%20advanced%20web%20development%20knowledge.

Science, O. -O. (2020, Jul 15). *Why You Should be Using Jupyter Notebooks* . Retrieved from medium.com: https://odsc.medium.com/why-you-should-be-using-jupyter-notebooks-ea2e568c59f2#:~:text=The%20Jupyter%20Notebook%20is%20an,text%20in%20a%20single%20document.