**Grazioso Salvare Dashboard README**

**Overview:**

At Global Rain, we partner with organizations to deliver custom software solutions as well as provide supplementary maintenance and support. As such, a partnership with Grazioso Salvare made sense and their request to bring us on board for the noble and necessary cause of sourcing rescue animals is an honor. This deliverable will allow for Grazioso to easily search and categorize shelter animals in the area to ease the burden of finding available animals. Their organization will be able to place more resources into the training and development of the rescue dogs through the added efficiency we are able to provide with this interactive application.

**Background:**

When first approached with the project, it was imperative that we find a solution that creates both a powerful, and easy to use tool for the staff at Grazioso. The main source of information for the dashboard will stem from local animal shelters who have agreed to allow our client to utilize their logs. In this case, the most logical tool was through MongoDB where the information can be effectively stored and modified while allowing for the flexibility to develop and link programs in a variety of languages. After the data was uploaded and linked with a Python script containing standard CRUD functions for the database, it became time to develop the main UI for the staff members to utilize. Again, with many options available at our disposal, we had to choose the path that would integrate best with the existing database infrastructure. Our client had also requested certain functionalities as well as an easy-to-use GUI to view the database entries. The most easily integrated framework to achieve these goals was the Dash Framework by Plotly. This framework is based in Python and integrates many elements from their data modeling API’s as well as linking many HTML elements to allow for the creation of an accessible online platform. With these components, the most logical way to integrate them comes from the Model, View, Controller or MVC structure of client/server interactions. MongoDB serves as the model component due to the powerful querying and accessibility it provides, while the Dash framework provides both the View and Controller aspects. In order to properly format the data for Dash, we have used the Pandas Dataframe structure which Dash can then access and modify. Dash gives unparalleled integration with multiple styles and languages of programing and interaction, which makes it the perfect tool to control and view the data contained on MongoDB.

**Installation:**

As this dashboard will be accessed through the network, minimal installation is required. The device must support modern HTML elements in addition to a stable network connection. Any maintenance to the Mongo Database will require access to a Linux-based terminal with the latest versions of the Python API, MongoDB API, and Dash Framework API installed. It will also require the importing of the Pandas module. These API’s can be found on the respective websites of each provider and installation will be done through the command prompt using certain commands such as “pip install”. All documentations for these frameworks are widely available online as they are incredibly popular and highly utilized structures.

**Usage:**

The goals of this project were laid out in detail by our client, Grazioso Salvare. Their team requires a tool to allow for the seamless filtering of data provided from local animal shelters. Specifically, their organization specializes in training rescue dogs to serve in a variety of public roles. The three overarching goals of the project were to create useful queries, visual modeling, and ease-of-access.

Queries:

Through their work, Grazioso has identified certain breeds of dogs which are better suited for the various rescue and safety duties where the animals are needed. It was necessary to build a Dash Datatable which receives the Dataframe structure as an input. This gives a spreadsheet style display of the results of the queries. This utility allows for built-in filtering options, but to achieve the best results for the client, our approach was to develop custom callbacks to give the results a truly tailored feel. While they will still posses basic alphabetical and numeric sorting of the chart, the bulk of the filters are built into a single large callback function. Dash provides elements called Radio items which act as buttons which a user can toggle on and off. We have implemented four of these items directly correlating to the traits identified for each rescue category. There are also other secondary and tertiary filtering options should any other queries be needed in the future. Within the callback function, these methods will have access to the CRUD module created earlier on in the development process, and more specifically this callback will use the read() portion of the module. As the user sends input data to the dashboard, it can then send the correlated queries to the CRUD class which can then be relayed to the MongoDB with the results being subsequently returned.

Visual Modeling:

The inclusion of the Dash framework immediately gave us access to the entire Plotly graphing library and its large selection of data modeling tools. One has already been mentioned previously in the Datatable structure which will show the contents of a Pandas Dataframe. While this tool is very powerful and useful when it comes to viewing filtered data, the users at Grazioso require visual elements to help identify areas of concern or focus. Therefore, we implemented a Plotly express pie chart which shows the number of each specific breed as a percentage of the total available for a given rescue role. We believe this can allow our client to assess the availability of the animals as well as the direction in which to proceed with training. The pie chart is complimented by a map which displays the geolocation of a selected animal within the Datatable which delivers a visual representation of the shelter to contact about the animal.

Ease-of-access:

While keeping our client’s need for a tool which is accessible from a multitude of locations and devices, it was only natural to use a client/server structure. This way, the users at the rescue organization will be able to utilize this tool from any device with a network connection as long as they possess valid login credentials. Not only does the Dash framework operate under these conditions, but the integration it contains with HTML provides an application that operates much like nearly every website or web-based application. The format should be easy to maintain and follow, while still giving the users easy access to the information they will need to complete their work.

**Screenshots:**

Highlighting selected rows

**Graphical user interface, text, application

Description automatically generated**

Filtering by “Disaster” type**Graphical user interface, text, application

Description automatically generated**

Dynamic Pie Chart and Map**Graphical user interface

Description automatically generated with low confidence**

..More Visual Data Examples**Graphical user interface, application, map

Description automatically generated**

Default unfiltered view of data**Graphical user interface, application

Description automatically generated**

Toggle columns visibility**Graphical user interface, application

Description automatically generated**

Update and Delete portions of CRUD module**Text

Description automatically generated**

Logo, image, and developer information present**Graphical user interface, text, application

Description automatically generated**

**Challenges:**

While the end goal was achieved for our client and we have delivered an easy-to-use tool to supplement their operations, the creation of this application was anything but easy. Numerous setbacks occurred along the way ranging from incorrect outputs to the application not running at all. There were two specific features that proved especially taxing to implement, mainly the layered filtering options and the Plotly visual chart. The original intention was to use Button elements for the main filtering options, but I was unsuccessfully able to get them to work with the Radio items within the same function. Unfortunately, in the Dash Framework there can only be a singular callback for any given output which means that exporting the filtered data to the table in two separate callbacks for each style would not be possible. The compromise that was created was to utilize the same element for all filtering options while adjusting the size and appearance of the different layers. This solution may not be as aesthetically pleasing, the functionality is still fully present for the client. Development was relatively smooth after that hang-up until it came time to integrate a Plotly express Pie Chart. This visual model made the most sense for the dashboard as our client will likely need to identify which breeds are more readily available than others. However, the API did not seem to like the data it was receiving from the Pandas Dataframe module. To overcome this, I entered a testing phase where I would run the application after changing the input variables within the px.pie() function. Many inputs resulted in a blank screen output, but every now and then one would run and display some information. While the data wasn’t very useful, through observations, pattern recognition, and cross-referencing the Pandas and Dash documentation, I was able to tune the chart to output useful information about the ratio of each breed to the total number of animals available. This application introduced many new concepts and APIs into my vocabulary which will certainly prove incredibly valuable in future endeavors.

**Resources:**

<https://docs.mongodb.com/>

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

<https://pandas.pydata.org/docs/>

<https://docs.python.org/3/>

<https://www.w3schools.com/css/default.asp>

https://www.w3schools.com/python/pandas/default.asp

**Contact**

Gary Clark

[gary.clark@snhu.edu](mailto:gary.clark@snhu.edu)

Southern New Hampshire University