**Project 2 Dashboard README**

Name: Matthew Rix

Date: 10/13/2022

**Functionality Description:**

Starting State:

The following screenshot shows the starting state of the dashboard in which all the animals are populated in the data table. The pie chart below shows the breakdown of all the different breeds in the current data table. This can be changed via the dropdown menu on the left below the data table. The Geographic map next to the pie chart shows a drop pin of the latitude and longitude of which ever animal’s radial button is selected.

A picture containing graphical user interface

Description automatically generated

Water Rescue:

The following screenshot shows the dashboard when the ‘Water Rescue’ radial button is selected. This radial activates a filter that removes any animal that is not of a preferred breed as defined by the specifications, is not an intact female, and is not between the ages of 26 to 156 weeks. This filter also removes any animal whose status is euthanized. You will also notice that the pie graph is set to show the breakdown of all the animals listed in the now filtered data table by age.

Graphical user interface, application

Description automatically generated

Mountain or Wilderness Rescue:

The following screenshot shows the dashboard when the ‘Mountain or Wilderness Rescue’ radial button is selected. This radial activates a filter that removes any animal that is not of a preferred breed as defined by the specifications, is not an intact male, and is not between the ages of 26 to 156 weeks. This filter also removes any animal whose status is euthanized. You will also notice that the pie graph is set to show the breakdown of all the animals listed in the now filtered data table by breed.

Graphical user interface, application

Description automatically generated

Disaster or Individual Tracking:

The following screenshot shows the dashboard when the ‘Disaster or Individual Tracking ‘ radial button is selected. This radial activates a filter that removes any animal that is not of a preferred breed as defined by the specifications, is not an intact male, and is not between the ages of 20 to 300 weeks. This filter also removes any animal whose status is euthanized.

Graphical user interface, application

Description automatically generated

Reset:

The following screenshot shows the dashboard when the ‘Reset’ radial button is selected. This radial clears the other filters by repopulating the data table with all the animals. You will also notice that the pie graph is set to show the breakdown of all the animals listed in the now filtered data table by the third option of outcome type.

Graphical user interface, application

Description automatically generated

**Tools:**

While I did not choose any of the tools in this project, I do understand why they were selected in that they compliment each other very well for the development of full stack application. The most obvious and helpful example I can think of is that MongoDB has collections that are essentially the same thing as a dictionary in the Python language which makes handling data a breeze. Not having to reformat data between tools really cuts down on the runtime and processing required to run the application. The other tools used were numpy for math operations, pandas for data manipulation, and dash for the visualizations.

**Framework:**

This project uses an MVC (model view controller) framework. The model portion of this project is the data set contained and accessed in MongoDB. The view portion is through the dashboard and widgets which are created run using the Python code in the Project\_2\_Dashboard.ipynb file. These widgets include a data table with radial buttons to activate the filters, a pie graph of one of three variables in the filtered data table, and a map with a marker geolocated to the latitude and longitude of the selected animal in the data table. This code also enables the login authentication process using hard coded values and placing the Graziosa Salvare logo. The controller portion of this framework comes through the modules in the animal\_shelter.py file in which all the transfer and interaction between the model and view happen. The controller has six modules that create the functionality of the application. The first is the initialization module that authenticates the session and establishes the database to be used. The rest are all the CRUD modules; create for creating a new animal and adding it to the dataset, readOne for finding and returning a single animal based on passed in search criteria, readAll which simply returned all the entries in the dataset, update which updated entries based on passed in input, and delete which deletes entries based on passed in inputs.

**Timeline:**

There were many steps that had to be completed to get this project finished. The first of these steps were completed over the previous seven weeks in which I imported the dataset to the mongoDB, created an authenticated user and admin for the database, and began to create the controller and viewer in python in the Jupyter app. When it came time to really start in on this project though I began by reading through the prompt for the project along with the specifications given by the customer to make sure I knew what was to be expected from this application. I had already had a dashboard populated with the dataset but I had not developed any way to filter it and so I had to do a lot of reading and trail and error in getting the filtering working. I then had to implement the radial buttons, drop down menu and graphs/maps to add some interactivity. This was easier since I had already figured out how to filter the data, so it just came down to applying it via widgets. While testing did happen throughout the process, I was able to most of it through minor unit testing on my local machine. The final testing really came in when I brought my developed code in to check the visualizations and apart from a few typos the dashboard worked as designed.

**Directions:**

To use the application simply open a command prompt and start up the mongoDB using the command “mongod\_ctl start-noauth”. Then open the Jupyter notebook, open the Project\_2\_Dashboard.ipynb, and run the code. This will populate the dashboard and allow you to use the radial buttons and drop-down menu to filter the data to find whatever you are looking for. The user interface should be intuitive enough that it is self-explanatory.