Caio Mauro

CS340

**DASHBOARD README**

**CLIENT REQUIREMENTS**

* Display the data from the Animals database on the front page in a grid format.
* Provide users with interactive components to filter the data they view based which should then affect other comments on the page.
* Give users a second visual component to act as a data visualizer to go alongside the geolocation map.
* Allow users to filter data based on rescue type and dog breeds.
* Include the company logo at the top of the main page.

**TOOLS USED**

**Dash:** Dash is a Python framework for building web applications. It's used here to create the main page as well as the interactive components of the dashboard.

**Pandas:** Pandas is used for data manipulation, for this application we used it for handling data in the DataFrame format.

**Plotly:** Plotly is used for creating interactive data charts and here we used it to create the pie chart.

**dash\_leaflet:** This library provides the map component for Dash applications, allowing us to show the location of a selected animal.

**CRUD**: CRUD is a proprietary python module designed to facilitate MongoDB interactions. It leverages PyMongo to make connection and perform actions on the database.

* **MongoDB:** 
  + MongoDB is chosen as the database due to its flexibility and compatibility with Python through libraries like PyMongo.
  + MongoDB's document-based structure works well with Python dictionaries and Pandas DataFrames, simplifying storing and manipulating data.
* **Dash Framework:**
  + Dash provides a structured framework for building web applications in Python.
  + It allows defining layouts using HTML components and interactivity through Python callbacks.
  + Dash's reactive architecture allows changes in user inputs to trigger updates in the page content without needing to reload the page every time.

**PATH TO COMPLETION**

First, we developed the CRUD module, which allows easy access to any MongoDB. The module provides the main application with all the tools needed to give users full interactivity with the available data. After developing CRUD, we moved on to developing the Dash HTML layout. It was important to set up the page to be user-friendly and capable of displaying all the needed data. Through pagination, the visible data is limited, making its consumption much easier. It was also important to set up the data visualizers and dropdowns for filtering. Afterwards, we implemented callback functions to update the dashboard elements based on user interactions. This allows the displayed data to react to being filtered and elements like the pie chart to respond to the new data being displayed after a user interaction. Once all of that was complete, I could then run the Python script, and Dash would handle hot reloading the page whenever the user interacted with the data.

**CHALLENGES FACED**

* Specifying the file path for the client’s logo was not as clear since we were using Jupyter. At first, they were both in the same directory, and the image would not show up on the dashboard. Eventually, we had to create an assets folder and set the image path to the folder, which fixed everything.
* Setting up the filtering callbacks to work properly was time-consuming because we had to preserve the data's structure so it could plug into the table. Thankfully, the dataframes helped in keeping everything correct, which allowed the filter query to substitute in for the data variable used by the table.
* It was a challenge to get the pie chart to react properly to the displayed data. There were many bugs that caused the data not to load properly or caused it to be stuck after setting one filter. After thoroughly investigating the callbacks, everything was resolved.

**SETUP AND RUN**

* Download and setup Jupyter notebook on your machine.
* Create a “Dashboard” folder to serve as the root directory for this project.
* Inside of the file we made create an “assets” folder to store images used by the project.
* Place the company logo inside “Dashboard/assets” with the name “logo.png”
* Inside the root directory place the “dashboard.ipynb” and “CRUD.py” files.
  + Please note user credentials are hard coded into the dashboard file.
* Once everything has been setup and your MongoDB instance is running with a proper Animals collection you can run the dashboard file in Jupyter.

**PREVIEW**

**A screenshot of a computer

Description automatically generated**Landing Page

Water Rescue Filter

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**Mountain or Wilderness Rescue

**A screenshot of a computer

Description automatically generated**Disaster or Individual Tracking Rescue

**A screenshot of a computer

Description automatically generated**Reset Filter Option

A screenshot of a computer

Description automatically generatedFilter by Breed Option (Ex: Pit Bull Mix)