Katie Stapleton

CS 340

Final Project

07/2021

**ReadMe**

**Overview**

The web application is created to find potential animals to train as search and rescue. The types of animals are dogs suitable for water rescue, wilderness rescue, and tracking. It connects data provided by the affiliated animal shelters to the criteria for each type of rescue dog. A sense of humor has been included to make reviewing the code more entertaining.

The repo focuses on creating a web application dashboard through the usage of Pymongo, Dash, Plotly, and Jupyter. It is considered a full-stack application, focusing on both back-end and front-end development. The back-end uses MongoDB to host the data. MongoDB is considered a NoSQL database, allowing the storage of unstructured data. The csv files used contains numerous cells with inconsistent data formatting and empty cells. Communication between the client and server is established through the usage of Python 3 functions tailored to the MongoDB (pymongo). Python is commonly used to create and manipulate databases including NoSQL and SQL. The client-server application has basic CRUD capabilities. However, users are only granted the permission to view and interact with the dashboard/interface. Users denied permission to add, alter, or delete the information.

The front-end uses Plotly and Dash to establish the user interface. Dash is a python framework designed for building web analytic applications and interactive data visualizations. Dash and Plotly only requires knowledge of Python instead of multiple languages. Jupyter Notebook is used to run interface, test the functionality, and apply changes to the interface. Jupyter offers the ability to host python-based web applications. It offers programmers the ability to visualize and interact with the interface as its being developed.

**Development**

First, the mongoDB was created and tested. The csv was uploaded into a MongoDB and assigned a collection by using the computer terminal and MongoDB methodology. Currently an connection to the MongoDB needs to be established locally for the Jupyter Notebook to work locally (go to data folder, start mongo, login into Mongo). Next, the Python CRUD queries were created in the IDE (.py). The queries can also be created in Jupyter, but must be saved an a python file. Jupyter was used to call upon and test the Python queries (.ipynb). Then the user interface was created and tested with Jupyter (.ipynb). The interface started with basic text and visual unfiltered database as its minimal viable product. The interface was updated to become interactive. For the data table, callbacks were established to query and/or filter certain datasets, accessible by selecting a given tab. (Tabs are not required. Another styles can be used such as radio items and dropdown menus). A basic layout of a geomap was added. The map was customized to respond to the same callback used for the data table. The map was stylized to provide live location markers with tooltip and hover data. A scatterplot was also added to compare the dog breed to their age. The scatterplot was programmed to also respond to the callback implemented for the data table.

The layout of the user interface starts with a heading, title, and brief description. Next, a data table is available for users to interact with. The user can sort the files based on the data columns or using tabs with customized queries to review specific animals. Following the table, a scatterplot compares the animal breed to its age. This provides users a quick glance at the available animals without reading all the information. A geomap (using Leaflet) is available for users to view the exact location of the animal with a short summary of data available in the hover tag. For a visual of the user interface, please navigate to the “preview” folder, containing screenshots when the application is live.

**Troubleshooting**

Other materials used included resource documentations, stackoverflow, and tech forums. Many sites and sources were used, but some specific links have also been documented/provided. The Jupyter Notebook was used to constantly test the coding as it was being developed. Issues were addressed individually as they occurred. Improvements was built upon top of each other to ensure functionality and minimize trouble-shooting.

**Resource Links**

* **MongoDB**: <https://www.mongodb.com/what-is-mongodb>
  + This is not a link to the database. It is the official MongoDB website.
  + The CSV file is available to upload into the server. You will need to review the names of the databases and collections used by this application. You can either copy the names or use your own name by editing the coding.
  + A command terminal was initially used to establish the DB locally, but other methods are applicable
* **Python**: <https://www.python.org/about/> Currently utilizing Python 3. Python files can be created using Jupyter or an IDE of your choosing.
* **Jupyter Notebook**: <https://jupyter.org/>
* **Plotly/Dash Open Source:** <https://dash.plotly.com/> Installation and documentation
* **Leaflet:** <https://leafletjs.com/> Leaflet library for creating mobile-friendly interactive maps

**Notable issues (add into GitHub Issues)**

This is designed as an MVP (minimum viable product). At this time, the scatterplot is not responding properly to its callback. A graph is displayed, but contains no visual data or customizations. The source of the problem is currently unknown. The geomap is not fully responsive. It only contains the location/data for the first animal list in the selected datatable. The user is not able to view a selected or multiple animals on the map. If you have suggestions (methods, errors, corrections, improvements, etc), please submit into “Issues” in the Github repo. The application is not loaded into Heroku yet. Any UI/UX improvements were not considered essential for this phase.