**CS-340 Dashboard Project README**

**Project Overview**

This project implements a data dashboard designed to display and filter animal shelter data, with a focus on different types of rescue animals and their attributes. The dashboard enables users to:

* Filter animal records based on rescue types (Water Rescue, Mountain or Wilderness Rescue, Disaster Rescue or Individual Tracking).
* View filtered data in a table.
* View animal breeds in a pie chart.
* See the geographic location of a selected animal on a map.
* Reset filters to view the complete dataset.

**Proof of Functionality**

*The Screenshots will be zoomed out to show a better understanding of the whole project. Each screenshot will include a selected animal to show the location*

* **Screenshot 1:** Initial dashboard layout with all animals displayed.

A screenshot of a computer

AI-generated content may be incorrect.

* **Screenshot 2:** Filter applied (Water Rescue) showing filtered data in the table and pie chart.

A screenshot of a map

AI-generated content may be incorrect.

* **Screenshot 3:** Filter applied (Mountain or Wilderness Rescue) showing filtered data in the table and pie chart.

A screenshot of a computer

AI-generated content may be incorrect.

* **Screenshot 4:** Filter applied (Disaster Rescue or Individual Tracking) showing filtered data in the table and pie chart.

A screenshot of a computer

AI-generated content may be incorrect.

**Tools and Technologies Used**

The main reason that the technologies that were used were chosen for this class. We are expected to use what is required for the course. I have also provided some reasons below, as to why someone would choose these.

**1. MongoDB**

MongoDB was used as a backend database to store and manage animal shelter data because of its flexible, document-oriented model. Key reasons include:

* **Schema Flexibility**
* **Powerful Query Language**
* **Seamless Python Integration**
* **Scalability**

**2. Dash Framework**

Dash was selected to build the dashboard interface due to:

* **Full-stack Python Framework**
* **MVC (Model-View-Controller) Architecture:**
* **Integration with Plotly**
* **Dash Leaflet**
* **JupyterDash**

**Resources and References**

Most of the resources used for this project were from the prior weeks of our class. Over these last weeks, we have been adding more to this database. I was able to take parts and pieces of this to make project 2.

**Project Completion Steps**

1. **Setup Environment:** Installed required libraries including pandas, dash, dash-leaflet, jupyter-dash, and pymongo.
2. **Database Integration:** Implemented the AnimalShelter CRUD class to connect with MongoDB and perform queries.
3. **Data Retrieval:** Loaded initial dataset from MongoDB into a pandas DataFrame, cleaning it by removing the \_id field.
4. **UI Layout Design:** Constructed the dashboard layout with buttons for filtering, a data table for viewing records, a pie chart for breed distribution, and a map for geolocation.
5. **Interactivity Implementation:** Developed Dash callbacks for button filtering, data table updates, pie chart generation, and map marker updates based on user selection.
6. **Testing and Debugging:** Verified all components work together interactively, fixing issues such as callback errors, data filtering, and visual updates.
7. **Deployment:** Ran the app within JupyterLab using JupyterDash and validated it through various use cases.

**Challenges Encountered**

* **MongoDB \_id Field Handling:** The MongoDB \_id field caused errors when converting records to a pandas DataFrame for display. This was resolved by explicitly dropping the \_id column before rendering.
* **Callback Context Management:** Determining which button triggered the filter callback required proper use of dash.callback\_context to avoid applying multiple filters or conflicts.
* **Dash DataTable Configuration:** Initial errors with incorrect DataTable properties (column\_selectable expecting specific strings) were corrected by adhering to Dash's documented options.
* **Map Marker Coordinates:** Ensuring map markers used the correct columns for latitude and longitude required careful indexing and validation.
* **Indentation Errors:** Several Python indentation issues were resolved by careful code structuring to avoid syntax errors.
* **Running in Jupyter Environment:** Configuring JupyterDash with infer\_jupyter\_proxy\_config() was necessary to enable inline deployment within JupyterLab.

**Conclusion**

This project successfully demonstrates a full-stack Python data dashboard application integrating MongoDB as the data model and Dash as the web framework for the view and controller components. It highlights interactive filtering, data visualization, and geospatial mapping, leveraging the strengths of the chosen tools to build a user-friendly analytical dashboard.