**Project 2 README**

**Project Functionality**

The Grazioso Salvare Dashboard is designed to assist in the selection of dogs suitable for various types of rescue operations. The dashboard provides a user-friendly interface to filter and view data according to specific rescue categories such as Water Rescue, Mountain Rescue, and Disaster Tracking. It allows the users to interact with the data dynamically, presenting the information in both tabular and visual formats, including a geolocation map and a pie chart for breed distribution.

**Screencast**

[CS340 Project 2 - YouTube](https://www.youtube.com/watch?v=e3YFh-_D4bc)

**Tools and Technologies**

**MongoDB**

MongoDB is utilized as the backend database system for its flexibility with unstructured data and its scalability. Its NoSQL nature allows for quick iterations during development, a key feature for evolving project requirements. MongoDB also provides powerful query capabilities, which are instrumental in filtering data based on the criteria defined by Grazioso Salvare.

**Dash Framework**

The Dash framework is employed to create the web application interface. Dash is ideal for building data visualization apps with custom user interfaces in pure Python. It's particularly suited for this project due to its ability to create reactive applications, where user interactions can trigger updates to the dashboard without requiring a page reload.

**Python**

Python serves as the programming language for this project due to its simplicity and the extensive ecosystem of data manipulation and visualization libraries available.

**Resources**

* Dash: https://dash.plotly.com/
* MongoDB: <https://www.mongodb.com/>
* Plotly: https://plotly.com/python/

**Steps to Reproduce**

1. Install the necessary Python packages including **dash**, **pandas**, **plotly**, and **dash\_leaflet**.
2. Clone the repository containing the dashboard code and CRUD module.
3. Ensure MongoDB is running and accessible with the appropriate credentials.
4. Run the dashboard application using JupyterDash within a Jupyter notebook environment or as a standalone Dash app.

**Challenges and Solutions**

* **Data Integration:** Initially faced difficulties in integrating MongoDB data with Dash. Solved by using **pandas** to convert data into a format compatible with Dash DataTable.
* **Dynamic Filtering:** Implementing filters that react to user input required careful state management. Utilizing Dash callback functions addressed this challenge.
* **Map Visualization:** Encountered issues with map responsiveness, which was resolved by proper state management and ensuring the map component is updated only when necessary.

**Conclusion**

The development of the Grazioso Salvare Dashboard provided insights into the capabilities of Python for web application development and data visualization, particularly when combined with MongoDB for data storage and management. Despite challenges, the final product successfully meets the project requirements and offers an intuitive tool for the client.

Top of Form