**Grazioso Salvare Dashboard README**

**Project Overview**

The Grazioso Salvare Dashboard is a web application designed to provide an interactive and visual representation of animal rescue data. The dashboard allows users to filter and analyze rescue data based on various criteria, such as rescue type and preferred animals.

**Required Functionality**

* *Interactive Filtering*: Users can filter data based on rescue types like Water Rescue, Mountain or Wilderness Rescue, and Disaster or Individual Tracking.
* *Data Visualization*: The dashboard displays a data table containing detailed rescue information and interactive charts, such as a pie chart showing preferred animals.
* *Geolocation Mapping*: Users can select a row in the data table, and the corresponding location is highlighted on a geolocation map.

**Programming Languages and Libraries**

* *Python:* The primary programming language for backend development and data manipulation.
* *Dash*: A Python framework for building analytical web applications.
* *Dash Leaflet*: A Dash component for rendering interactive maps.
* *Plotly Express*: Used for creating interactive charts and visualizations.
* *Pandas*: Used for data manipulation and analysis.

**Database**

* *MongoDB*: Chosen as the model component due to its flexibility, scalability, and seamless integration with Python through the PyMongo library.

**Dash Framework**

* Dash is a productive Python framework for building web applications. It follows the MVC (Model-View-Controller) architecture, where:
* *Model (M*): MongoDB serves as the data model, storing and retrieving rescue data.
* *View (V):* The front end of the application is built using Dash HTML components for layout and visualization components like Dash Leaflet and Plotly Express.
* *Controller (C):* Callback functions in Python link the backend data (Model) with the front end (View), ensuring interactivity.

**Steps to Reproduce**

* *Clone Repository*: git clone https://github.com/your-username/grazioso-salvare-dashboard.git
* Install *Dependencies: cd grazioso-salvare-dashboard, then pip install -r requirements.txt*
* *Set* Up MongoDB: Ensure MongoDB is installed and running. Update MongoDB connection details in CRUDing.py.
* *Run the Application*: python your\_dashboardsname\_app.py
* *Access the Dashboard*: Open your web browser and navigate to http://localhost:8050.

**Challenges and Solutions**

* **Challenge 1:** Data Filtering Complexity
* *Solution*: Created specific callback functions for each filtering option, ensuring accurate data filtering.
* **Challenge 2**: Geolocation Map Integration
* *Solution*: Implemented callback functions to dynamically update the geolocation map based on the selected row.
* **Challenge 3**: MongoDB Integration
* *Solution*: Leveraged PyMongo library for seamless communication between Python and MongoDB.

Resources

*DASH Documentation & User Guide | Plotly*. (n.d.). <https://dash.plotly.com/>

*MongoDB documentation*. (n.d.). MongoDB Documentation. <https://www.mongodb.com/docs/>

*Plotly*. (n.d.). <https://plotly.com/python/plotly-express/>

*Dash*. (n.d.). <https://dash-leaflet.herokuapp.com/>

Screenshots:

Default set to reset to show full database

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedFiltering on Disaster or Individual Tracking

Filtering on Mountain or Wilderness Rescue

A screenshot of a computer

Description automatically generated

Filtering on Water RescueA screenshot of a computer

Description automatically generated

Geolocation on map from row selection

A screenshot of a computer

Description automatically generated

Map update with new row selection

A screenshot of a computer

Description automatically generated