# **Grazioso Salvare Animal Rescue Dashboard**

## **Overview**

This project is a web-based dashboard designed for Grazioso Salvare, a client specializing in animal rescues. The dashboard provides essential insights into animal types, rescue locations, and other relevant data sourced from a MongoDB database. It allows filtering based on specific rescue operations (Water Rescue, Mountain/Wilderness Rescue, Disaster/Individual Tracking) and visualizes the results through an interactive map and chart.

## **Functionality**

The dashboard meets the following key requirements:

* **Data Table**: Displays information about animals, including breed, location, and age. The table can be filtered based on rescue type and is paginated for easy navigation.
* **Interactive Map**: Shows the geographical location of the selected animal on a map.
* **Dynamic Charts**: Displays bar charts based on the filtered data, providing a visual overview of the animal distribution.

**Screenshots of the dashboard during testing and deployment:**

### **All Filter Active**

### **Water Rescue Filter Active**

### **Mountain or Wilderness Rescue Filter Active**

### **Disaster or Individual Tracking Filter Active**

## **Tools and Rationale**

### **MongoDB**

MongoDB is used as the database system for storing and querying animal data. The choice of MongoDB was driven by its:

* **Flexibility**: MongoDB's schema-less design makes it ideal for handling diverse data types.
* **Scalability**: Its capacity to handle large datasets makes it well-suited for the volume of data generated by Grazioso Salvare.
* **Python Integration**: The pymongo library allows seamless integration with Python, making it easier to perform CRUD operations directly from the application.

### **Dash Framework**

The Dash framework is employed for developing the dashboard because of its:

* **Simplicity**: Dash abstracts much of the web development complexity, enabling the creation of responsive web applications with minimal boilerplate.
* **Integration**: It natively supports Plotly for visualizations and Pandas for data manipulation, ensuring smooth data processing and presentation.
* **Interactivity**: Dash's callback mechanism allows for dynamic updates to the UI based on user interactions, enhancing the user experience.

### **Additional Libraries**

* **Plotly**: Used for generating the bar chart visualizations.
* **Dash Leaflet**: Facilitates the integration of interactive maps.
* **Pandas**: Aids in data processing and cleaning.
* **Matplotlib**: Employed for plotting within the Jupyter environment.

## **Steps Taken**

1. **Data Retrieval**: Connected to MongoDB using the pymongo library, fetched data from the 'animals' collection, and cleaned it for use in the dashboard.
2. **Dashboard Construction**: Developed a responsive layout using Dash components, including data tables, radio buttons, and graphs.
3. **Interactivity Implementation**: Implemented Dash callbacks to enable real-time updates of the data table, map, and chart based on user selections.
4. **Testing and Deployment**: Thoroughly tested the dashboard in a Jupyter environment to ensure proper functionality and responsiveness.

## **Challenges and Solutions**

1. **Data Cleaning**: Inconsistent data formats and missing values were addressed using Pandas to ensure smooth filtering and visualization.
2. **MongoDB Connection**: Encountered issues with MongoDB connection timeouts, which were resolved by optimizing connection parameters and handling exceptions in the CRUD class.
3. **Interactivity Delays**: Experienced delays in updating the map when switching filters. This was mitigated by refining the query and data processing logic to reduce overhead.