**Grazioso Salvare Dashboard Project - README**

**Project Overview**

The Grazioso Salvare Dashboard is a web application designed to help the organization identify and categorize available dogs for search-and-rescue training. The dashboard provides an interactive interface that allows users to filter, view, and analyze data related to dogs at various animal shelters in the Austin, Texas region. The dashboard includes several key features:

1. **Interactive Filtering Options**: Users can filter data by rescue type (Water Rescue, Mountain/Wilderness Rescue, Disaster/Individual Tracking) using radio buttons.

2. **Dynamic Data Table**: The data table updates in real-time based on the selected filter, displaying relevant dog data.

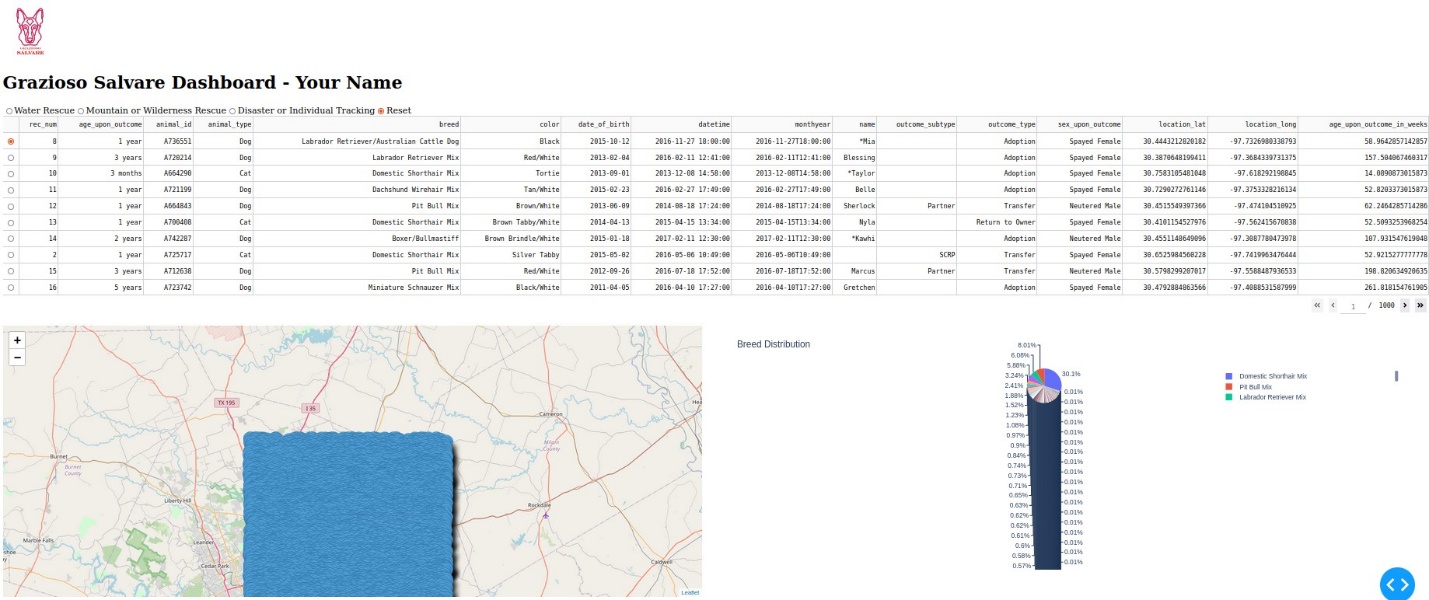
3. **Geolocation Map**: A map displaying the locations of dogs, which also updates dynamically based on the selected filter.

4. **Pie Chart**: A chart that visualizes the distribution of dog breeds, dynamically updated according to the selected filter.

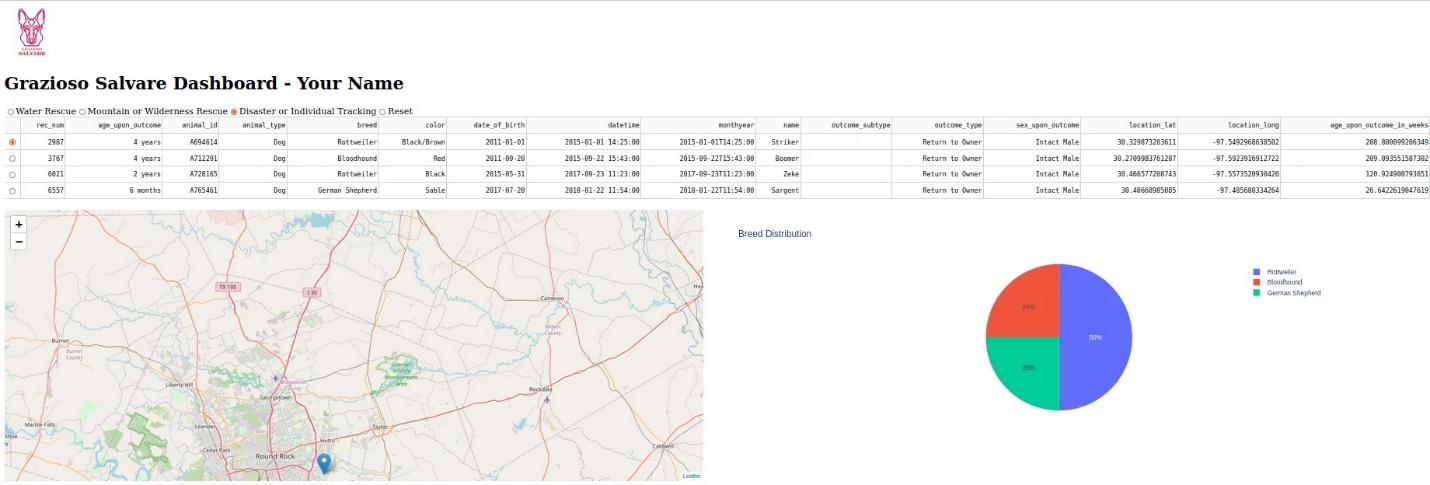
5. **Branding**: The Grazioso Salvare logo and a unique identifier with the developer's name are displayed on the dashboard.

**Proof of Functionality**

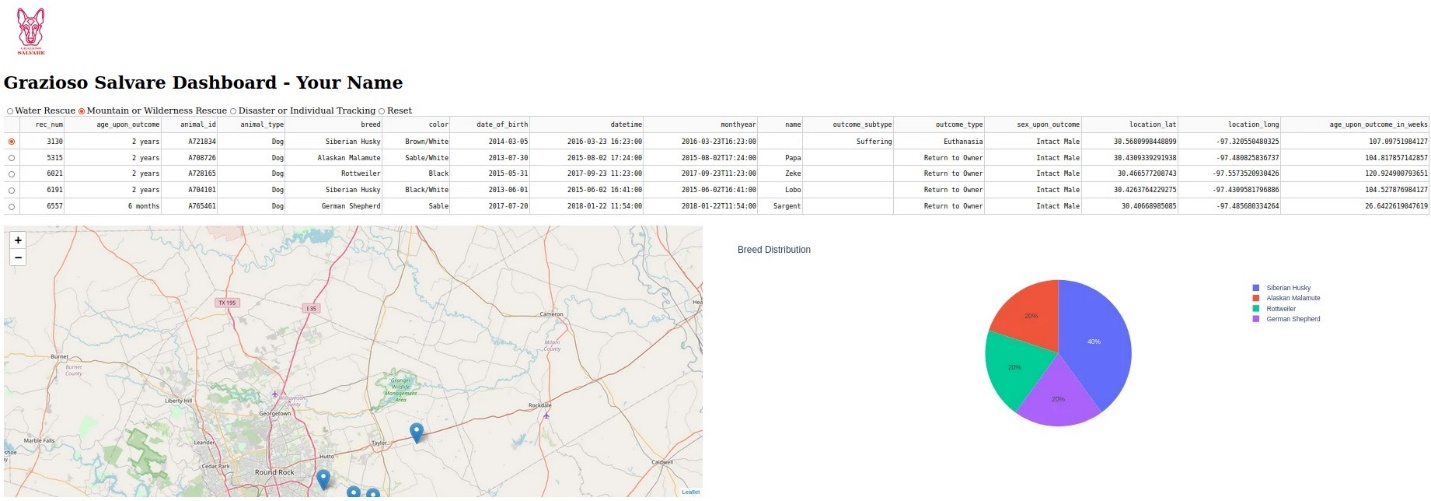
* The initial state of the dashboard / Reset



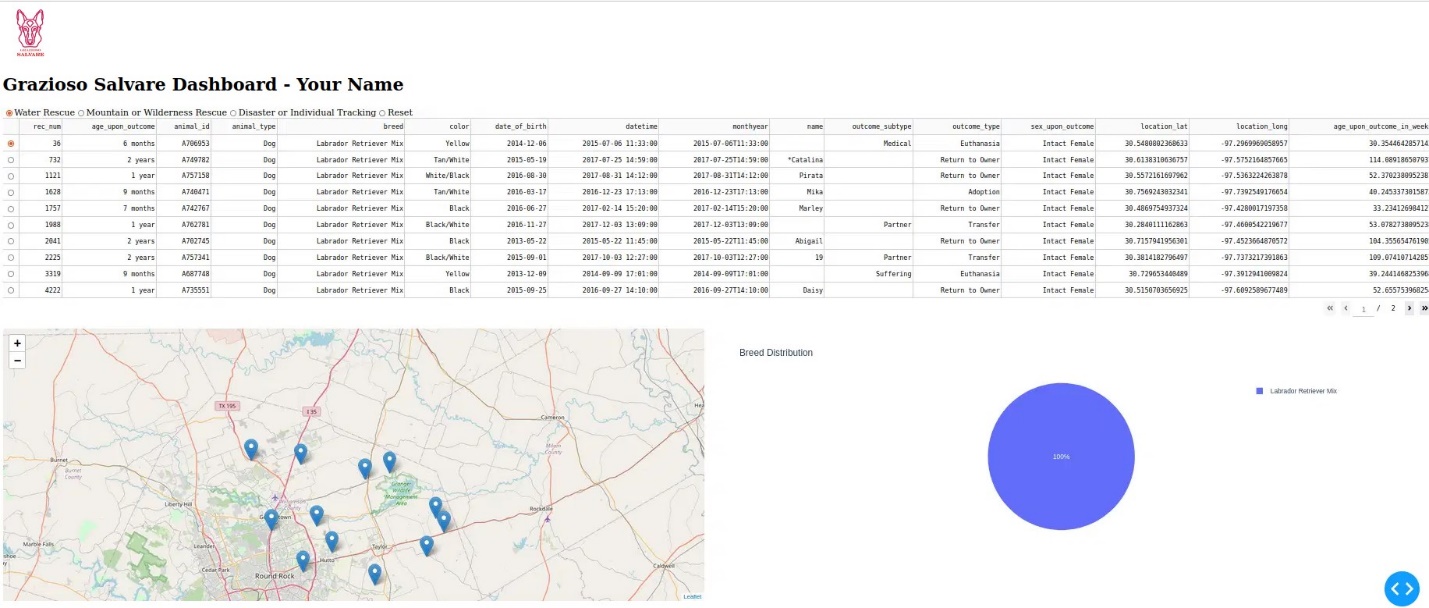
* Disaster or Individual Tracking



* Mountain or Wilderness Rescue



* Water Rescue



**Tools and Technologies Used**

**1. MongoDB**

- **Purpose**: MongoDB was used as the database to store and retrieve data about the dogs. It was chosen because it is a NoSQL database, which allows for flexible storage of data without a fixed schema, making it easier to work with diverse datasets.

- **Interfacing with Python**: MongoDB's compatibility with Python through libraries like `pymongo` allows seamless integration and easy manipulation of the data using Python scripts.

**2. Dash Framework**

- **Purpose**: Dash was used to build the web application. It provides a powerful, yet simple, way to create interactive web applications purely in Python, making it ideal for data visualization and interaction.

- **View and Controller**: Dash handles both the view and controller aspects of the web application. It allows for easy integration of HTML, CSS, and JavaScript elements into the Python environment, enabling the creation of a responsive and dynamic user interface.

**3. Pandas**

- **Purpose**: Pandas was used for data manipulation and analysis. It provides data structures like DataFrames that are ideal for handling and analyzing structured data.

**4. Plotly**

- **Purpose**: Plotly was used to create interactive charts, specifically the pie chart that displays breed distribution. Plotly integrates well with Dash and provides a wide range of customizable visualization options.

**5. Dash Leaflet**

- **Purpose**: Dash Leaflet was used to create the geolocation map that displays the locations of dogs. It integrates seamlessly with Dash, allowing the creation of interactive maps within the web application.

**6. Jupyter Dash**

- **Purpose**: Jupyter Dash was used to run the Dash application within a Jupyter Notebook environment, making development and testing more convenient.

**7. VS Code or Jupyter Notebook**

- **Purpose**: Used as the primary development environment for writing and testing the Python code.

**Project Setup and Installation**

1. **Set Up MongoDB**

- Ensure MongoDB is installed and running on your machine or a remote server.

- Import the data into the `AAC` database with the `animals` collection.

- Adjust the connection details in the `crud.py` file to match your MongoDB setup.

**2. Run the Dashboard**

- Open the `Josh Perry ProjectTwoDashboard.ipynb` file and run the cells to start the Dash server.

**3. Access the Dashboard**

- Open your web browser and navigate to the port specified (default: `http://127.0.0.1:8051`).

**Project Development Process**

**1. Initial Setup**

- Established the MongoDB database and imported the Austin Animal Center Outcomes dataset.

- Created the CRUD module to handle interactions with the MongoDB database.

**2. Building the Dashboard Layout**

- Developed the basic layout using Dash components, including the logo, data table, and placeholders for the map and chart.

**3. Interactive Functionality**

- Added radio buttons for filtering by rescue type.

- Created callbacks to update the data table, map, and pie chart based on the selected filter.

**4. Testing and Debugging**

- Tested the dashboard to ensure all components responded correctly to user inputs.

- Debugged issues related to data retrieval, layout alignment, and interactivity.

**5. Deployment**

- Documented the setup and deployment process.

- Captured screenshots for inclusion in the README.

**Challenges and Solutions**

**1. Data Integration with MongoDB**

- **Challenge**: Ensuring data from MongoDB was accurately reflected in the dashboard components.

- **Solution**: Used comprehensive testing to confirm data integrity and adjusted MongoDB queries as needed.

**2. Responsive Layout**

- **Challenge**: Achieving a responsive layout that displayed the map and pie chart side by side.

- **Solution**: Utilized CSS styling within Dash's `html.Div` elements to control the layout and ensure proper alignment.

**3. Handling Empty Data Sets**

- **Challenge**: Preventing errors when the dataset returned by MongoDB was empty.

- **Solution**: Implemented checks within the callback functions to handle empty data and display appropriate messages or default states.

**Resources**

- **Dash Documentation**: https://dash.plotly.com/

- **MongoDB Documentation**: https://docs.mongodb.com/

- **Pandas Documentation**: https://pandas.pydata.org/

- **Plotly Documentation**: https://plotly.com/python/

**Conclusion**

This project successfully implemented a user-friendly dashboard that allows Grazioso Salvare to filter and visualize data related to potential search-and-rescue dogs. The combination of MongoDB, Dash, and various Python libraries provided a powerful platform for creating this interactive web application. The challenges encountered during development were effectively addressed, resulting in a robust and functional tool that meets the client's needs.