## **README**

The **Grazioso Salvare Dashboard** is a powerful tool for analyzing animal shelter data to identify rescue dogs best suited for specific missions. The application is designed to help users filter data, visualize information through graphs, and locate animals using an interactive map.

This dashboard leverages MongoDB for data storage, Dash for front-end functionality, and Python libraries for backend integration and data manipulation.

## **Dashboard Functionality**

The dashboard provides a user-friendly interface with the following key functionalities:

### **Filter: All**

Displays the complete dataset of animals from the database.

Ideal for browsing all available animals without any specific rescue mission in mind.

### 

### **Filter: Water Rescue**

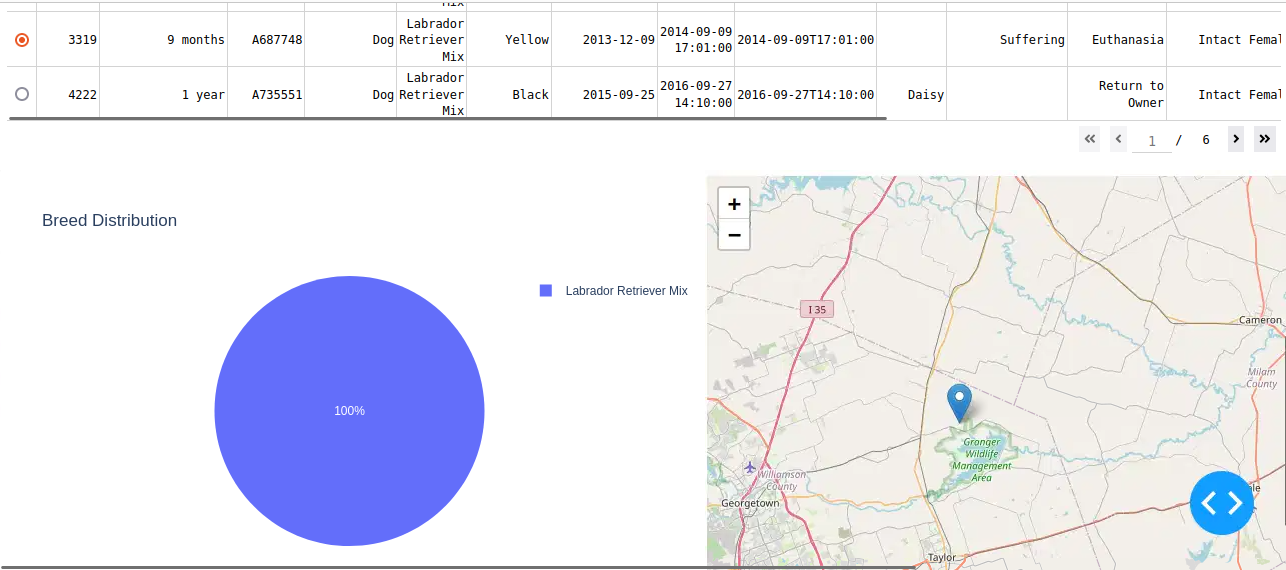
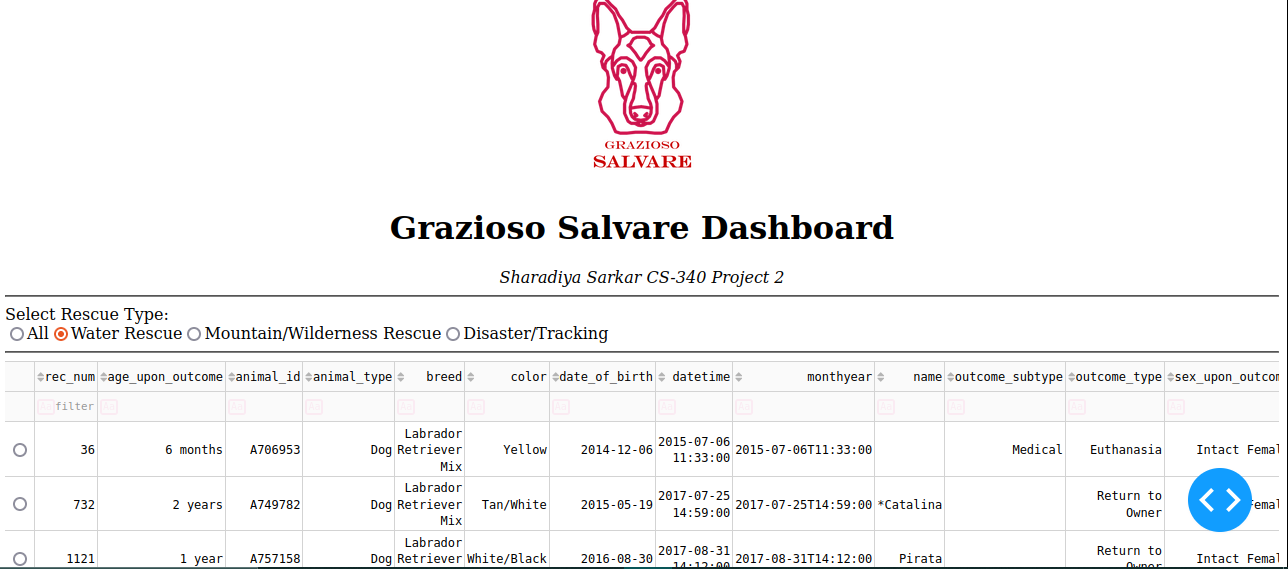
Filters for dogs suitable for water rescue missions. Example:

Breeds: Labrador Retriever Mix, Chesapeake Bay Retriever, Newfoundland.

Sex: Intact Female.

Age: 26 to 156 weeks (6 months to 3 years).

This filter helps water rescue teams identify potential candidates.

.

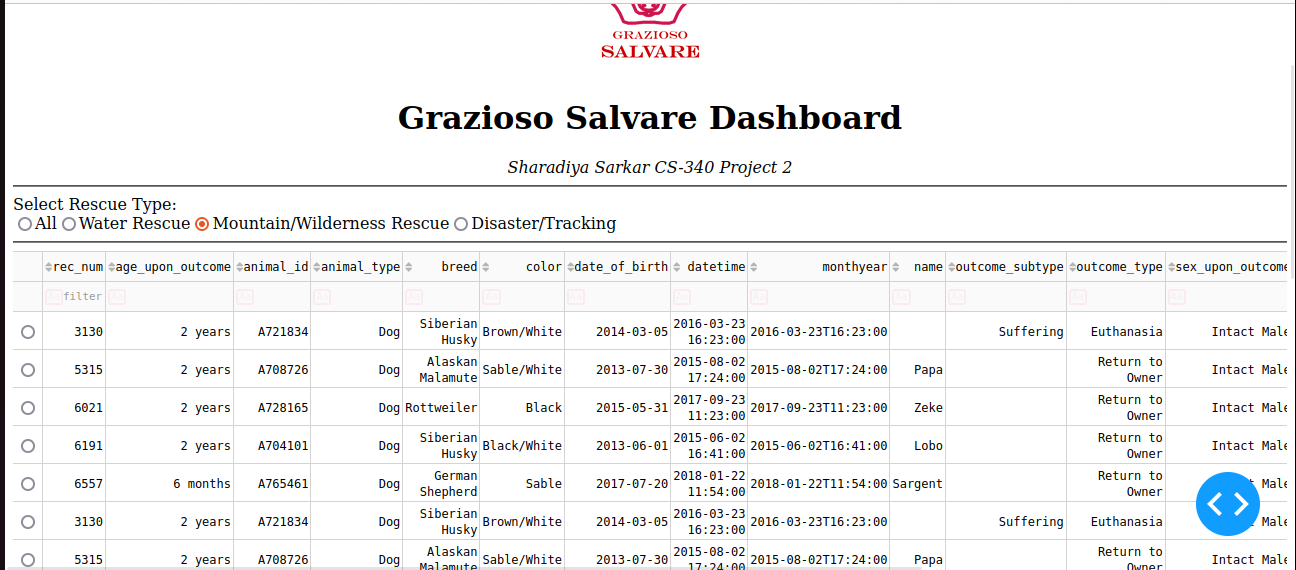
### **Filter: Mountain or Wilderness Rescue**

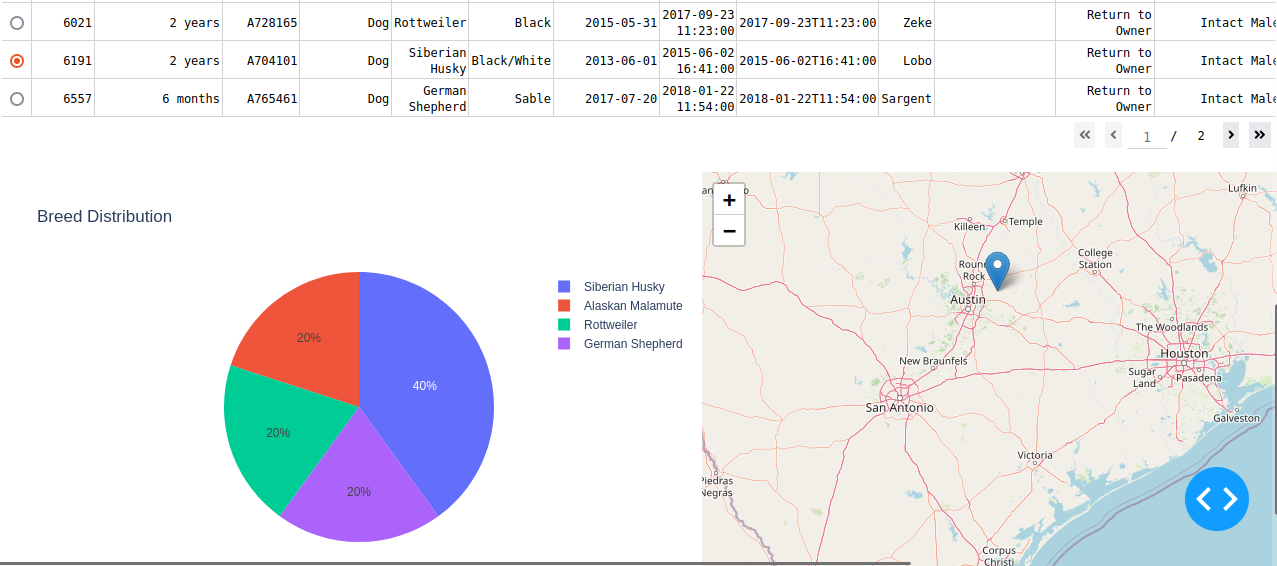
Filters for dogs suitable for mountain or wilderness rescue missions. Example:

Breeds: German Shepherd, Alaskan Malamute, Old English Sheepdog, Siberian Husky, Rottweiler.

Sex: Intact Male.

Age: 26 to 156 weeks (6 months to 3 years). This filter assists in identifying dogs capable of performing well in rugged terrains.





### **Filter: Disaster or Individual Tracking**

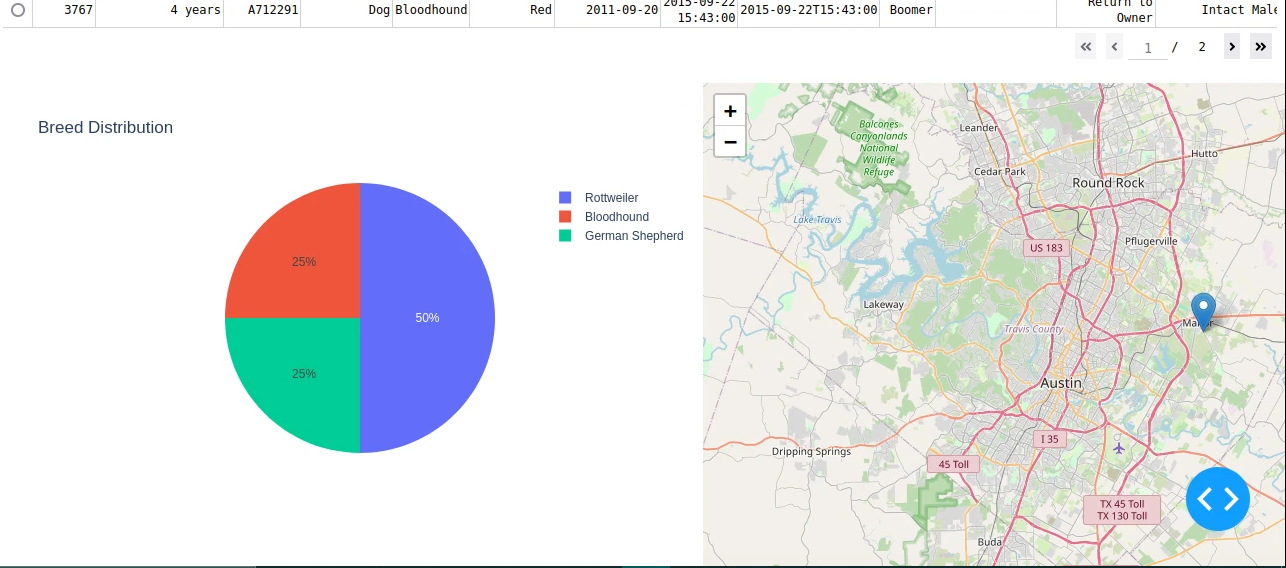
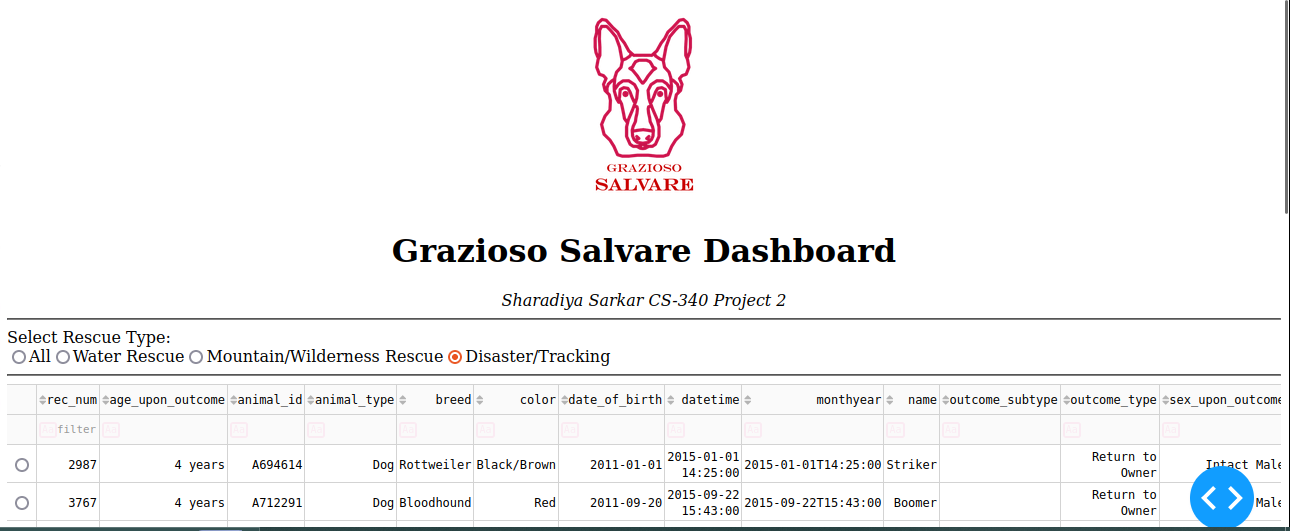
Filters for dogs suitable for disaster rescue or individual tracking missions. Example:

Breeds: Doberman Pinscher, German Shepherd, Golden Retriever, Bloodhound, Rottweiler.

Sex: Intact Male.

Age: 20 to 300 weeks (5 months to 6 years).

This filter helps disaster response teams and trackers find suitable candidates.



## **Tools Used**

### **MongoDB**

It's a database that stores animal shelter data in a NoSQL format. It is a flexible schema for dynamic data.

### **Python**

It is a backend programming language for data processing and dashboard functionality. The libraries used were pandas, plotly, dash-leaflet.

### **PyMongo**

PyMongo Facilitates communication between the Python application and MongoDB database. And provides an easy-to-use interface for querying and updating MongoDB collections.

### **Dash Framework** builds the web-based dashboard interface. It supports interactive graphs, tables, and maps.

## **Steps Taken**

1. **Data Preparation**:
   * Extracted animal data from MongoDB.
   * Cleaned and preprocessed the data using pandas.
2. **Dashboard Layout Design**:
   * Created a user-friendly layout with filters, graphs, and a map.
3. **Integration of Filters**:
   * Added radio buttons for rescue type selection.
   * Configured callbacks to update the data table, pie chart, and map dynamically.
4. **Visualization**:
   * Used Plotly for pie charts.
   * Integrated Dash Leaflet for map functionality.
5. **Testing and Debugging**:
   * Tested with sample data to ensure accurate filtering and visualization.
   * Resolved issues related to callback conflicts and missing map markers.

## **Challenges Encountered**

**Callback Conflicts**: I encountered an issue where multiple callbacks attempted to update the same component. I resolved it by rewriting that part

**Map Rendering**: Addressed missing or invalid latitude and longitude values. And added default coordinates for cases with missing location data.

**Dynamic Data Updates**: Added real-time updates to the map and graphs when filters were applied.

**Visualization Overlap**: I improved the layout to prevent overlapping of components on smaller screens.