

Grazioso Salvare Dashboard Project

About the Project

I developed the Grazioso Salvare Dashboard to support the company in its mission to identify and train rescue dogs. The dashboard is an interactive data visualization tool that integrates data from five animal shelters in the Austin, Texas, area. It allows users to filter the dataset to find suitable dogs for different types of search-and-rescue training. These filters include Water Rescue, Mountain or Wilderness Rescue, and Disaster or Individual Tracking. The dashboard provides insights into the breed composition of selected dogs and displays the geographic location of each dog on a map.

The project is built with user experience in mind, offering a seamless and intuitive interface for Grazioso Salvare's staff to interact with the data, filter based on training criteria, and visualize the results in real-time.

Required Functionality Achieved

The dashboard meets all required functionalities specified by Grazioso Salvare:

1. Interactive Filter Options:

- The dashboard includes radio button filters for:
 - **Water Rescue:** Displays dogs suitable for water-based rescue training.
 - **Mountain or Wilderness Rescue:** Filters for dogs ideal for rugged terrain rescues.
 - **Disaster or Individual Tracking:** Shows dogs suitable for locating individuals or navigating post-disaster environments.
 - **Reset:** Clears the filters to show the entire unfiltered dataset.

2. Dynamic Data Table:

- A table displays records matching the selected filter criteria. It is interactive, allowing sorting, pagination, and selection.

3. Charts and Visualizations:

- A **pie chart** dynamically visualizes the distribution of dog breeds based on the filtered dataset, helping to understand the breed composition for each rescue type.

- A **geolocation map** updates based on the selected entry in the data table, showing the dog's location.

4. **Branding Requirements:**

- The dashboard features the **Grazioso Salvare logo**, linked to the company's homepage, and includes my name as the developer for credit.

Screenshots Provided:

- The README contains screenshots showing:
 1. The dashboard after applying the "Water Rescue" filter.
 2. The dashboard after applying the "Mountain or Wilderness Rescue" filter.
 3. The dashboard after applying the "Disaster or Individual Tracking" filter.
 4. The dashboard after resetting the filters (unfiltered state).

These screenshots are located in the "Screenshots" section of the README, providing visual proof of the functionalities achieved.

Tools Used

- **Python:** The primary programming language for the backend development, data manipulation, and visualization.
- **MongoDB:** Chosen as the database solution for the project. It stores the animal data and supports real-time querying, making it suitable for interactive applications like this dashboard.
 - **Why MongoDB?**
 - **Flexible Data Model:** MongoDB's document-based storage is ideal for handling the semi-structured data from the animal shelters.
 - **Seamless Integration with Python:** Using the pymongo library, I could easily connect to MongoDB and perform database operations directly within the dashboard code.
 - **Scalability:** The database can handle large datasets efficiently, which is useful when dealing with a growing number of animal records.

- **Dash Framework (Plotly):** Provides the user interface for the web application, enabling me to create an interactive dashboard using pure Python.
 - **Why Dash?**
 - **User-Friendly:** Dash enables rapid development of interactive data visualizations without requiring extensive front-end web development skills.
 - **Interactivity:** It allows for the creation of components that react to user input, making the dashboard more engaging and functional.
 - **Plotly Integration:** With built-in support for Plotly charts, Dash made it easy to create high-quality, interactive graphs.
 - **Dash Leaflet:** Utilized for integrating geolocation functionality, allowing users to see the location of each dog on an interactive map.
 - **JupyterDash:** Enabled development and testing within the Jupyter environment, which made iterative coding and debugging easier.
-

Steps Taken to Complete the Project

1. Setting Up the MongoDB Database

- I used the mongoimport tool to upload the Austin Animal Center dataset to a remote MongoDB instance. Authentication was set up with the "aacuser" account to secure the connection.

2. Developing the CRUD Python Module

- I created a Python module to handle Create, Read, Update, and Delete operations for the MongoDB database.
- I implemented a read() method in the module to support filtering data based on specific criteria, such as breed, age, and sex.

3. Building the Dashboard Using the Dash Framework

- **Layout Design:** I set up the dashboard layout, including the radio button filters, data table, pie chart, and map components.
- **Geolocation Integration:** Using Dash Leaflet, I added a map to visualize the geographic location of selected dogs.

4. Adding Interactive Filtering Functionality

- I created callback functions to filter the dataset based on the selected rescue type and update the data table, pie chart, and map accordingly.
- Each filter option runs a query to retrieve the relevant subset of data from MongoDB using the CRUD module.

5. Testing and Validation

- I manually tested each filter option to ensure the data table, chart, and map updated correctly based on user interactions.
- Screenshots were taken to demonstrate the achieved functionality, which are included in the README.

Challenges Encountered and Solutions

1. Filtering Data Efficiently

- **Issue:** Crafting MongoDB queries to filter data based on multiple conditions (e.g., breed, age range, and sex) was complex.
- **Solution:** I utilized MongoDB's advanced query operators (\$in, \$gte, \$lte) to refine data retrieval, ensuring the dashboard remained responsive.

2. Ensuring Synchronization Between Components

- **Issue:** Keeping the data table, pie chart, and map in sync when applying filters was challenging.
- **Solution:** I used Dash's callback mechanisms to link the components, ensuring that the visualizations updated simultaneously in response to user inputs.

3. Handling Missing Location Data

- **Issue:** Some records did not have latitude and longitude values, which could cause the map to display incorrectly.
- **Solution:** I added error handling to check for missing location data and display a user-friendly message when necessary.

Explanation of Key Components

1. MongoDB Integration

- MongoDB serves as the data source for the dashboard, storing the animal records in a NoSQL database. It was chosen because of its flexibility in handling diverse

and semi-structured data, making it suitable for managing animal records that may not all follow the same structure.

2. Dash Framework

- Dash provides the front-end interface, linking user interactions to backend data manipulations. The dashboard layout is constructed using Dash components such as `dcc.RadioItems` for the filters, `dash_table.DataTable` for displaying tabular data, and `dcc.Graph` for plotting interactive charts.
- The **callback functions** in Dash connect the front-end components to the backend logic, ensuring the dashboard reacts dynamically to user inputs.

3. Callback Functions


- Callback functions link the interactive elements (radio buttons, data table, pie chart, and map) to the data retrieval and filtering logic. For example, when a user selects "Water Rescue," a query fetches the relevant data from MongoDB, updating the displayed table and visualizations.
-

Screenshots Section

Include the following screenshots:

- 1. **Water Rescue Filter Applied:** Shows the results after filtering for water rescue.

127.0.0.1:30980



GRAZIOSO
SALVARE

CS-340 Dashboard - Zainab Lowe

☐ Reset ☒ Water Rescue ☐ Mountain or Wilderness Rescue ☐ Disaster or Individual Tracking

	rec_num	age_upon_outcome	animal_id	animal_type	breed	color	date_of_birth	datetime	monthyear	name	outcome_subtype
<input type="radio"/>	36	6 months	A706953	Dog	Labrador Retriever Mix	Yellow	2014-12-06	2015-07-06 11:33:00	2015-07-06T11:33:00		Medical
<input type="radio"/>	732	2 years	A749782	Dog	Labrador Retriever Mix	Tan/White	2015-05-19	2017-07-25 14:59:00	2017-07-25T14:59:00	*Catalina	
<input type="radio"/>	1121	1 year	A757158	Dog	Labrador Retriever Mix	White/Black	2016-08-30	2017-08-31 14:12:00	2017-08-31T14:12:00	Pirata	
<input checked="" type="radio"/>	1628	9 months	A740471	Dog	Labrador Retriever Mix	Tan/White	2016-03-17	2016-12-23 17:13:00	2016-12-23T17:13:00	Mika	
<input type="radio"/>	1757	7 months	A742767	Dog	Labrador Retriever Mix	Black	2016-06-27	2017-02-14 15:20:00	2017-02-14T15:20:00	Marley	
<input type="radio"/>	1988	1 year	A762781	Dog	Labrador Retriever Mix	Black/White	2016-11-27	2017-12-03 13:09:00	2017-12-03T13:09:00		Partner
<input type="radio"/>	2041	2 years	A702745	Dog	Labrador Retriever Mix	Black	2013-05-22	2015-05-22 11:45:00	2015-05-22T11:45:00	Abigail	
<input type="radio"/>	2225	2 years	A757341	Dog	Labrador Retriever Mix	Black/White	2015-09-01	2017-10-03 12:27:00	2017-10-03T12:27:00	19	Partner
<input type="radio"/>	3319	9 months	A687748	Dog	Labrador Retriever Mix	Yellow	2013-12-09	2014-09-09 17:01:00	2014-09-09T17:01:00		Suffering
<input type="radio"/>	4222	1 year	A735551	Dog	Labrador Retriever Mix	Black	2015-09-25	2016-09-27 14:10:00	2016-09-27T14:10:00	Daisy	

Distribution of Breeds

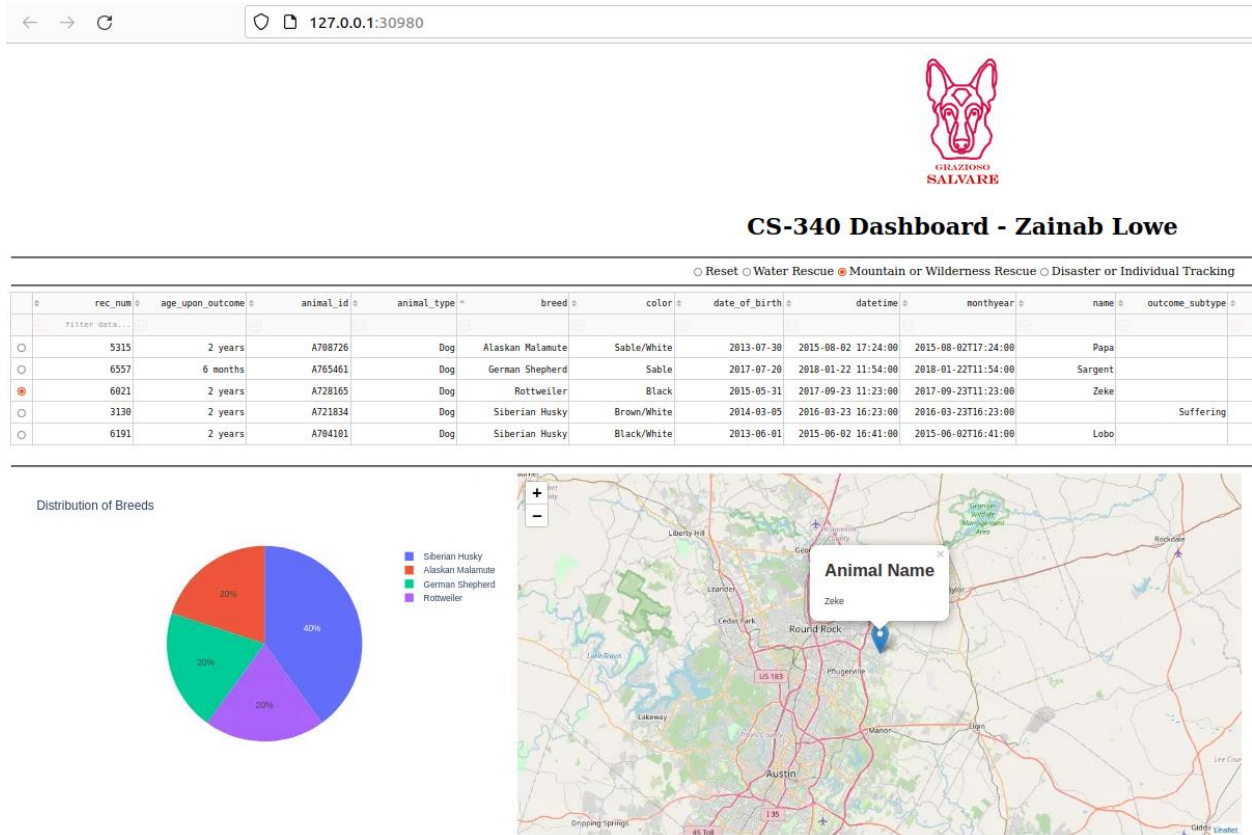
100%

■ Labrador Retriever Mix

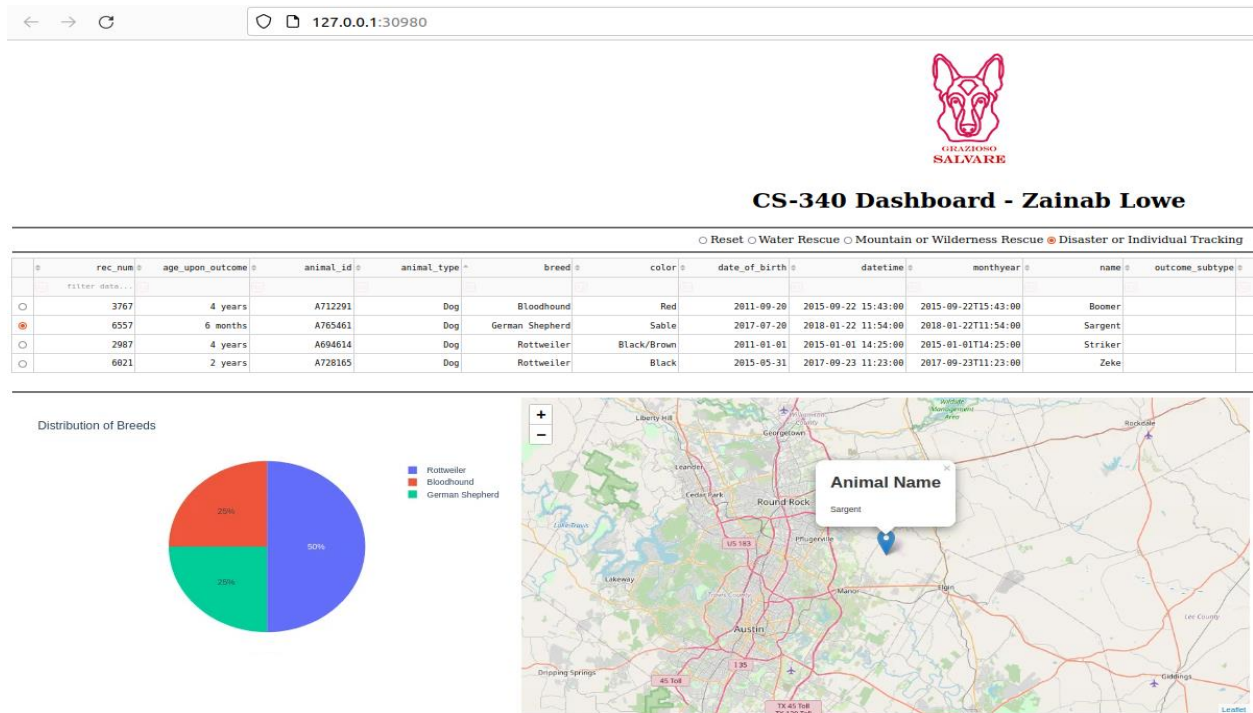
Animal Name

Mika

2. Mountain or Wilderness Rescue Filter Applied: Displays the filtered data for mountain or wilderness rescue.



3. Disaster or Individual Tracking Filter Applied: Shows the dashboard after applying the disaster rescue filter.



4. Reset State: Demonstrates the dashboard in its unfiltered state.

