**Grazioso Salvare Interactive Dashboard**

**Author:** George Saxby

**Project Overview**

This project implements an interactive data dashboard for Grazioso Salvare, developed using Python, Dash, and MongoDB. The dashboard visualizes the Austin Animal Center Outcomes dataset and allows the client to filter, explore, and analyze animal records through an intuitive web interface.

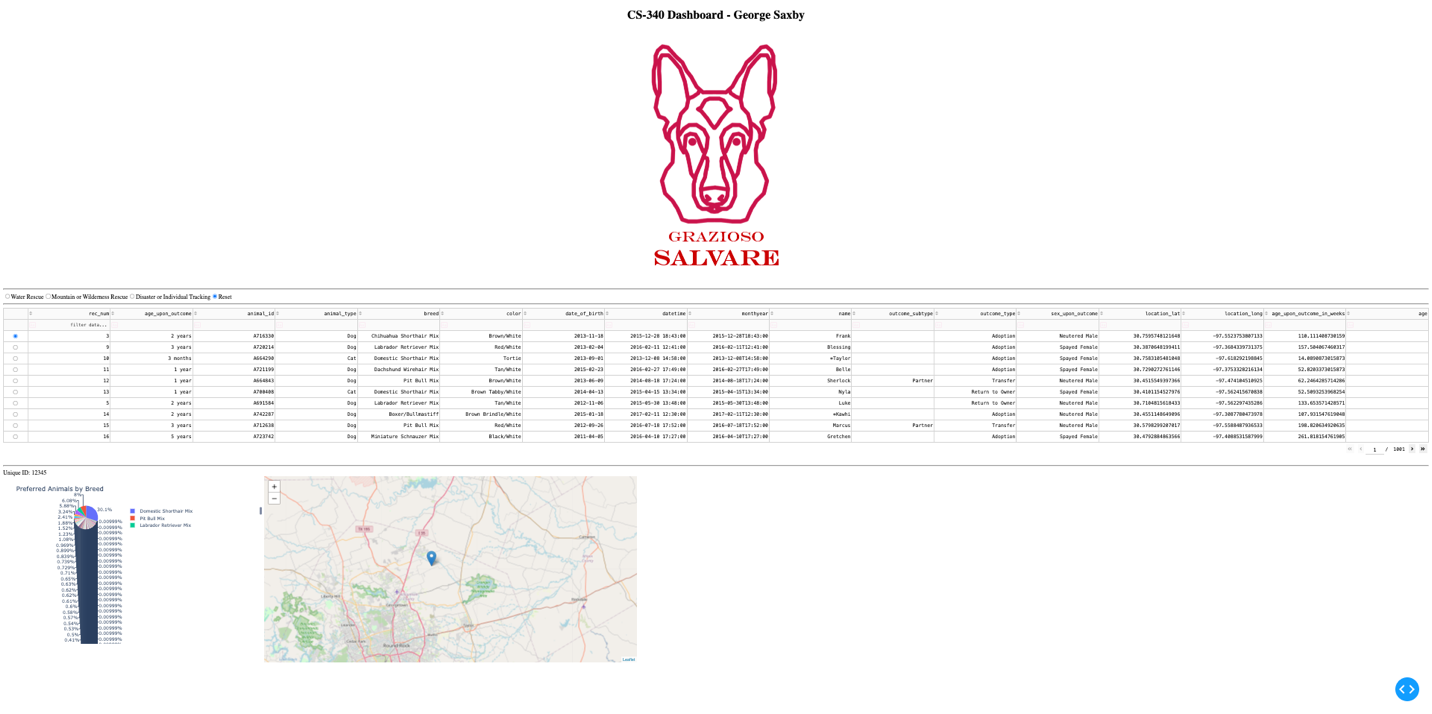
The completed dashboard provides:

* Interactive filter options based on Rescue Type and Preferred Dog Breeds
* A data table that dynamically responds to filter inputs
* A geolocation map of outcomes
* An additional bar chart (or chart of choice) that dynamically responds to filters
* The Grazioso Salvare logo and a unique identifier (author’s name) embedded in the interface

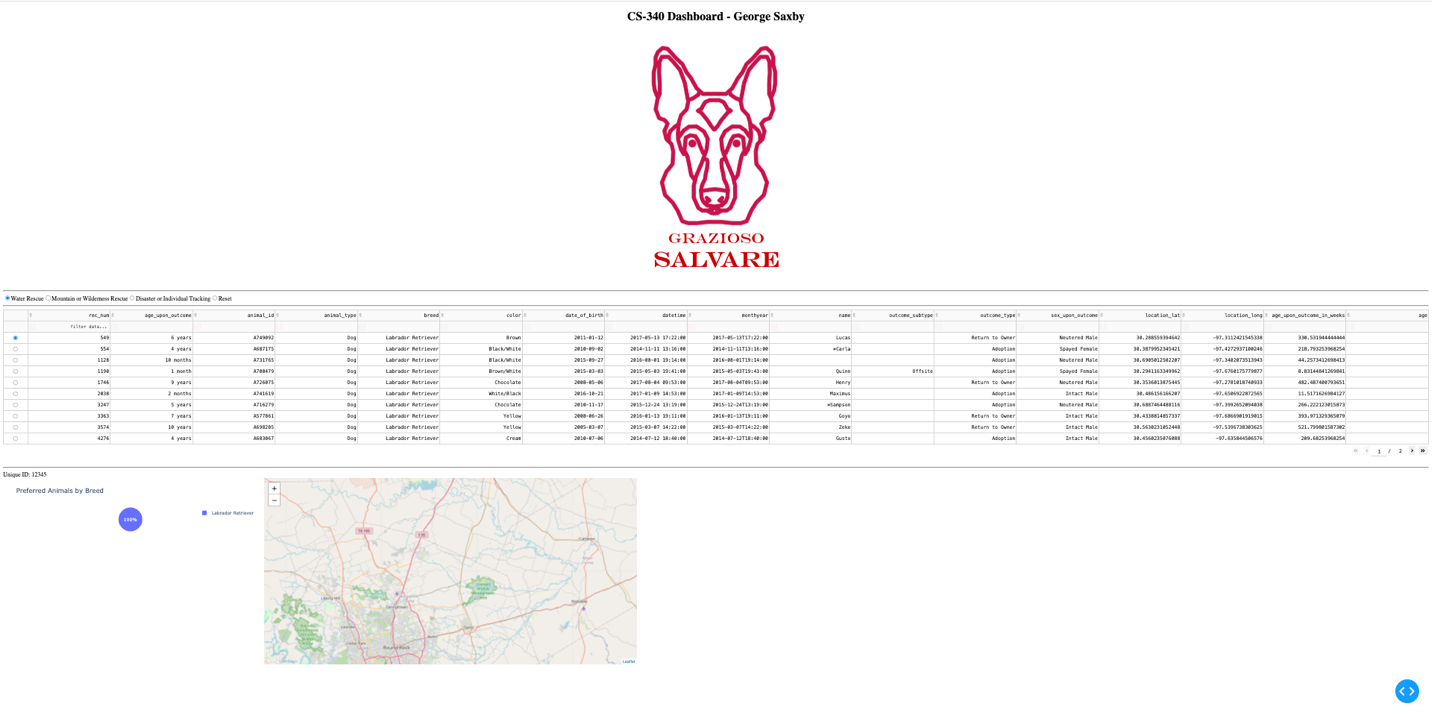
**Required Functionality**

The dashboard meets all functional requirements:

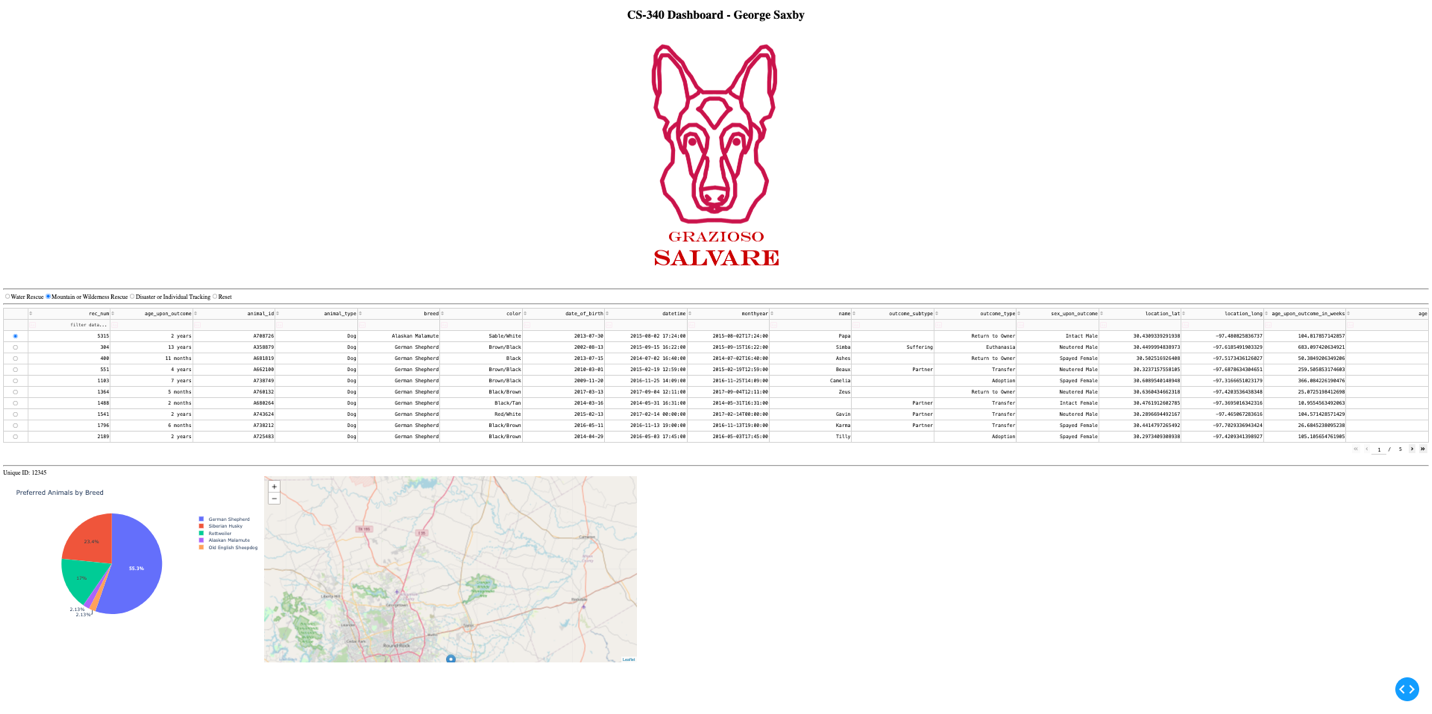
1. **Interactive Options**
   * Radio buttons/dropdown menus allow filtering by Water Rescue, Mountain/Wilderness Rescue, Disaster/Individual Tracking, or resetting to view all data.
2. **Data Table**
   * Displays the Austin Animal Center Outcomes dataset.
   * Dynamically updates in response to the selected filter.
   * Includes usability features such as pagination and sorting.
3. **Charts**
   * Geolocation Chart: Displays animal locations using outcome coordinates.
   * Bar Chart (or alternative chart): Provides additional insights based on selected filters.
4. **Branding and Identifier**
   * The Grazioso Salvare logo is displayed at the top of the dashboard.
   * The developer’s name, *George Saxby*, is included as a unique identifier.
5. **Proof of Functionality (Screenshots)**
   * Dashboard initial state with logo, filters, and widgets



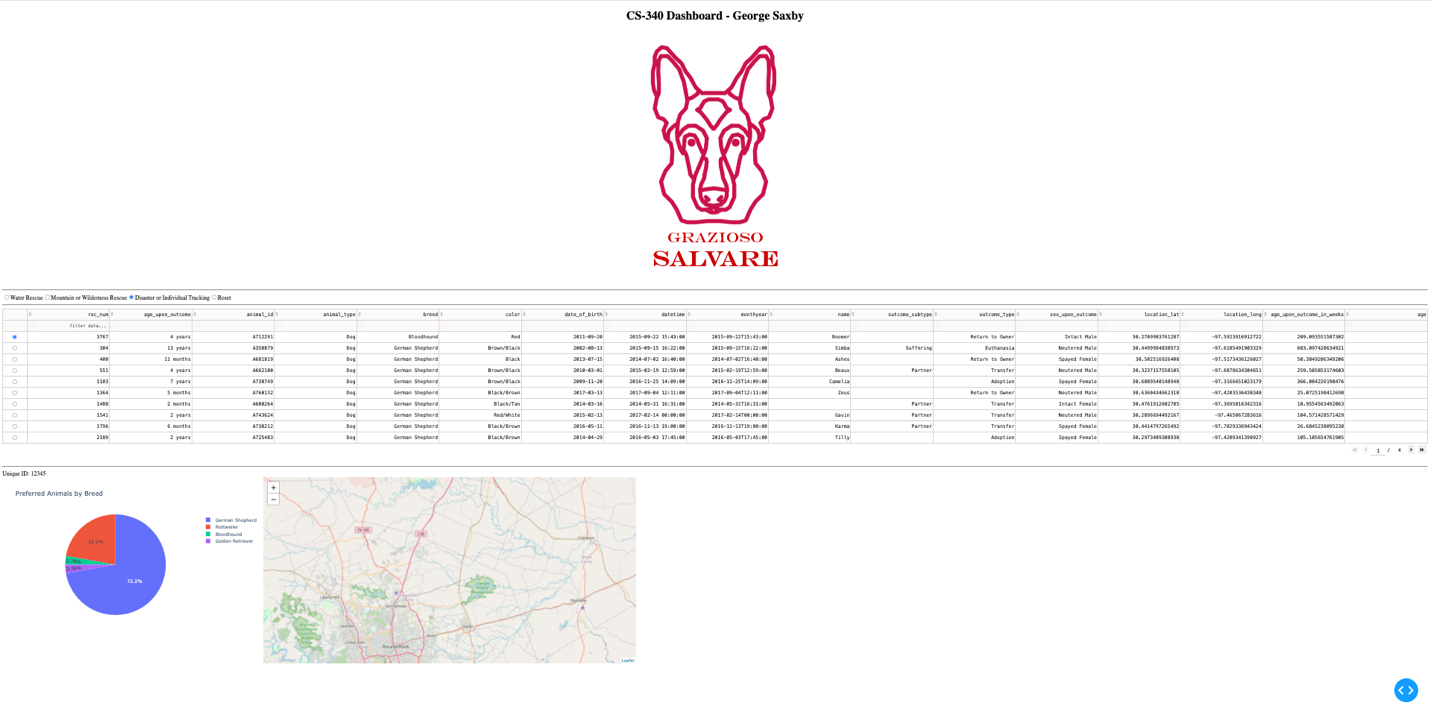
* + Filter applied – Water Rescue



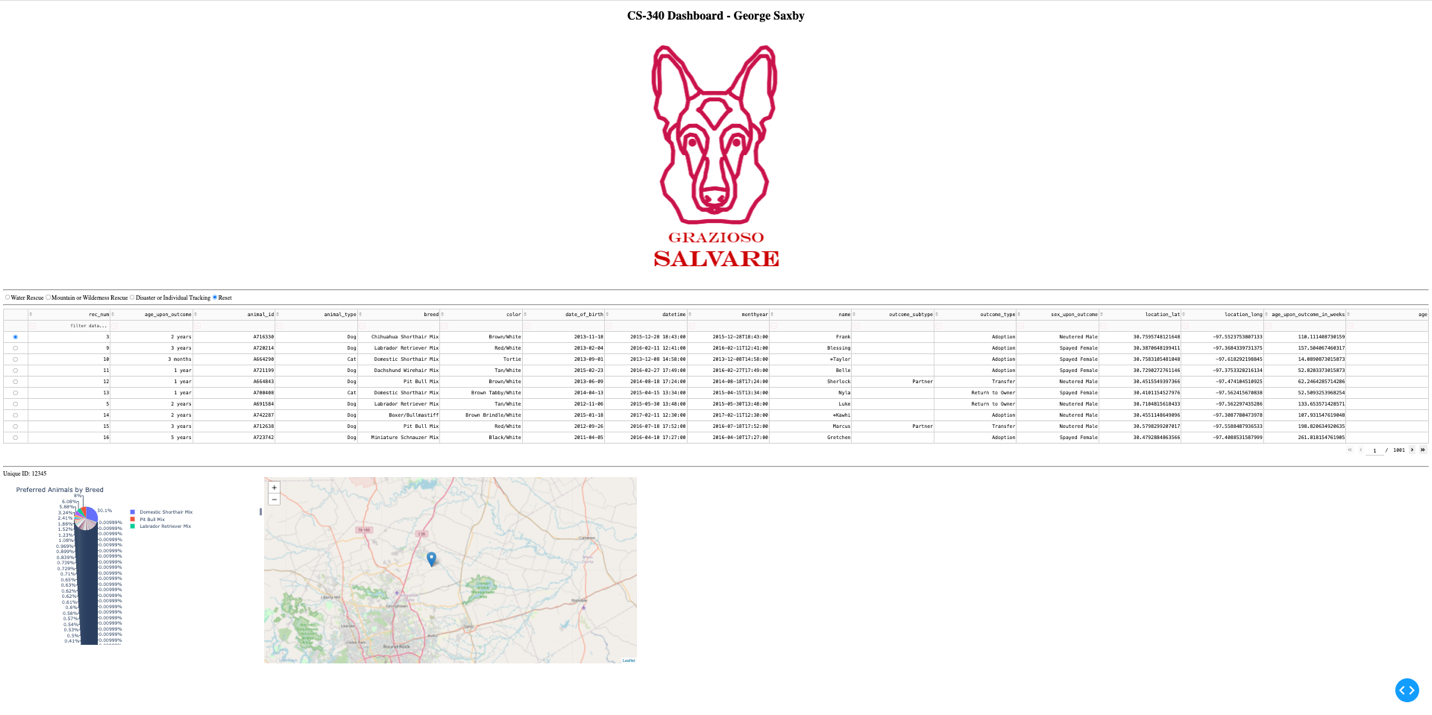
* + Filter applied – Mountain/Wilderness Rescue



* + Filter applied – Disaster/Individual Tracking



* + Reset applied – full dataset



**Tools and Rationale**

* **MongoDB**
  + Serves as the model layer in the MVC structure.
  + Flexible schema, ideal for unstructured JSON-like records such as animal outcomes.
  + Direct integration with Python through PyMongo, enabling seamless CRUD operations.
  + Efficient for handling large datasets with filtering queries.
* **Dash (Plotly Dash Framework)**
  + Provides both the view and controller components of the MVC model.
  + Enables creation of interactive web applications entirely in Python.
  + Includes built-in support for data tables, dropdowns, maps, and charts.
  + Rapid development cycle, making it ideal for prototypes and client deliverables.
* **Python & Jupyter Notebook**
  + Development environment for iterative coding, testing, and debugging.
  + Integration with MongoDB CRUD module from Project One.
  + Supports modular design: CRUD operations in a separate .py file, dashboard logic in .ipynb.

**Steps Taken**

1. Set up MongoDB connection using CRUD module from Project One (AnimalShelter.py) with hard-coded AAC user credentials.
2. Retrieved dataset using a “retrieve all” query for the initial unfiltered data table.
3. Developed data table with pagination and sorting features in Dash.
4. Implemented filter controls (radio items/dropdowns) to query MongoDB by rescue type.
5. Connected filters to data table via callback functions to dynamically update content.
6. Created geolocation chart to visualize animal outcome coordinates.
7. Created second chart (bar chart of breed counts) to add analytical depth.
8. Branded dashboard with Grazioso Salvare logo and unique identifier.
9. Tested and validated functionality, capturing screenshots and/or a screencast for documentation.

## **Challenges and Solutions**

 **Challenge: Making charts respond dynamically to filters**

* Initially, the charts (geolocation and bar chart) were static. The callbacks didn’t properly connect the filter options to update the charts. This created mismatched data between the table and charts.
* **Solution:** I used Dash callback functions to link the filter inputs to MongoDB queries. Each time a filter changed, the query refreshed, and the charts updated in sync with the data table.

 **Challenge: Updating multiple widgets from a single filter**

* Having one filter control multiple outputs (table, map, and bar chart) caused callback conflicts in Dash.
* **Solution:** I combined outputs into a single callback and used dash.callback\_context to ensure the updates worked together, keeping all widgets aligned with the selected filter.