**Grazioso Salvare Animal Outcomes Dashboard**

**Overview**

This project is a fully functional interactive dashboard designed for Grazioso Salvare to streamline their search and selection process for rescue dogs suited to specific rescue missions. Built using Dash and MongoDB, the dashboard offers filtering functionality, dynamic data tables, geolocation mapping, and graphical insights into breed statistics.

**Required Functionality**

The dashboard meets all of the specifications set forth in the UI/UX design document:

* **Interactive Filters**: Users can choose from the following rescue types:
  + Water Rescue
  + Mountain or Wilderness Rescue
  + Disaster or Individual Tracking
  + Reset (View all records)
* **Data Table**: Dynamically updates based on rescue type filter. Features include pagination, filtering, sorting, and single row selection.
* **Geolocation Map**: Displays the selected animal's location using latitude and longitude.
* **Bar Chart**: Displays the top 10 dog breeds based on count in the filtered dataset.
* **Branding**: Includes the Grazioso Salvare logo and a unique identifier: "SNHU CS-340 Dashboard - Cody Adams".

**Screenshot Evidence**

The following screenshots are included as proof of functionality:

1. **Dashboard Default State (Reset)**

A screenshot of a computer

AI-generated content may be incorrect.

A map of the united states

AI-generated content may be incorrect.

1. **Water Rescue Filter Applied**

A white sheet with black text and numbers

AI-generated content may be incorrect.

A map with a blue rectangle

AI-generated content may be incorrect.

1. **Mountain or Wilderness Rescue Filter Applied**

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

1. **Disaster or Individual Tracking Filter Applied**

A screenshot of a map

AI-generated content may be incorrect.

A map with a blue pin on it

AI-generated content may be incorrect.

**Tools Used**

|  |  |
| --- | --- |
| **Tool** | **Purpose** |
| **MongoDB** | Used as the backend database for storing and querying the Austin Animal Center Outcomes dataset. |
| **PyMongo** | Enables the Python script to interact with MongoDB, handling all CRUD operations. |
| **Dash** | Provides the framework to build the web-based dashboard with interactive components. |
| **Dash Leaflet** | Used for embedding geolocation maps into the dashboard. |
| **Plotly Express** | Used to generate the bar chart visualization for top dog breeds. |
| **Jupyter Notebook + JupyterDash** | Simplifies development and debugging of the Dash app within an interactive environment. |

**Rationale for Tool Selection**

* **MongoDB** was chosen for its flexible document model and ease of integration with Python via PyMongo. Its ability to handle complex queries with filters (e.g., by breed, age, and sex) made it ideal for the rescue-type filters.
* **Dash** was selected for its minimal boilerplate and integration of HTML, CSS, and JavaScript through Python. It simplifies the process of creating full-featured web dashboards.

**Resources & References**

* Dash Documentation: https://dash.plotly.com/
* PyMongo Documentation: https://pymongo.readthedocs.io/en/stable/
* Plotly Express: https://plotly.com/python/plotly-express/
* Dash Leaflet: https://dash-leaflet.herokuapp.com/

**Steps Taken**

1. Reviewed the UI/UX specification document.
2. Imported and cleaned the AAC dataset using a MongoDB Atlas instance and PyMongo.
3. Developed CRUD functionality in a separate Python module.
4. Built the dashboard layout and added all visual components using Dash.
5. Implemented filtering logic and callback functions.
6. Added styling for responsive layout and branding.
7. Validated all functionality and took screenshots of each required state.

**Challenges & Solutions**

* **Issue**: Logo not displaying.
  + **Solution**: Verified that the logo file was named correctly and placed in the assets/ folder, which Dash loads automatically.
* **Issue**: Bar chart was cut off in responsive view.
  + **Solution**: Restructured the layout using display: flex and adjusted component sizing to ensure the bar chart remained visible and centered.
* **Issue**: Callback errors due to ID mismatches.
  + **Solution**: Double-checked all component IDs and matched them with callback outputs and inputs.
* **Issue:** Error loading application after one time use.
  + **Solution:** Clear stale outputs, and adjust the port the application runs in.

**Author**: Cody Adams  
**Course**: SNHU CS-340  
**Client**: Grazioso Salvare