**Grazioso Salvare Project Two Dashboard**

Author: Dustin Davis

Course: CS-340 Client/Server Development

**Project Overview**

This project delivers an interactive web-based dashboard for Grazioso Salvare, a company that specializes in training rescue dogs. The dashboard connects to the Animal Adoption Center (AAC) database using MongoDB and displays information about adoptable animals. The system allows users to filter data, visualize animal locations on a map, and explore trends in top breeds through a bar graph.

The dashboard meets all requested functionality by allowing user to:

* Filter dogs by rescue mission type (Water Rescue, Mountain or Wilderness, or Disaster or Individual Tracking)
* Reset filters to display all available records
* Select animals from an interactive data table and view their location on a map
* Export filtered data to a CSV file
* View a dynamically updated graph showing the most common breeds in the current dataset

Screenshots have been included at the bottom of this document to demonstrate each major feature, including filtering, map interactivity, and CSV export.

**Tools and Technologies**

1. **Python** - Used as the primary programming language for data handling and dashboard logic.
2. **MongoDB** - Chosen as the model component because of its flexible schema and ability to handle large datasets with complex, nested fields. MongoDB’s JSON-like documents are ideal for storing and querying animal information such as breeds, ages, and locations. The pymongo library provides direct integration with Python for CRUD operations, enabling efficient data retrieval and filtering.
3. **Dash Framework** - Dash provides both the view and controller components of the application. It combines the power of Flask (backend), Plotly (graphing), and React (frontend) into a single, easy to deploy package. Dash allows for rapid development of interactive web dashboards with clean syntax and automatic linking between components using @app.callback.
4. **Plotly Express** - Used to create the horizontal bar chart displaying top dog breeds in the current table view. Plotly allows interactive hover effects, automatic scaling, and labeling.
5. **Dash Leaflet** - Integrated for rendering a dynamic map showing animal locations. It uses OpenStreetMap tiles and supports real time updates when a new animal row is selected.
6. **Pandas and NumPy** - Utilized for data processing, transformations, and generating summaries before visualizing or exporting data.
7. **JupyterDash** - Used to run the Dash web application directly inside a Jupyter Notebook, making testing and demonstration easier within the Codio environment.

**Rationale for Tool Selection**

**MongoDB:**

MongoDB was chosen for the model layer because it supports flexible document based storage rather than fixed schemas. This is ideal for animal records where different entries may contain slightly different fields. The integration with Python via the pymongo library made it simple to perform CRUD operations within the same application. MongoDB’s ability to quickly filter and query documents using JSON style syntax directly supports the dashboard’s dynamic filtering needs.

**Dash Framework:**

Dash provides an elegant way to combine a Python backend with a web-based user interface. It handles user interactions (controllers) automatically through callback decorators that respond to changes in dropdowns, radio buttons, and tables. Dash components (such as DataTable, Graph, and Map) simplify front-end design and allow all logic to remain in Python without needing to write HTML, CSS, or JavaScript manually.

**Steps Taken to Complete the Project**

1. **Configured MongoDB Connection** - Set up the AnimalShelter class in the CRUD Python module to connect to the database using authentication credentials.
2. **Retrieved Data** - Queried all animal records from MongoDB and converted them into a Pandas DataFrame.
3. **Built Dashboard Layout** - Created the application structure with a centered logo, mission filter buttons, reset/export controls, a data table, a map, and a graph.
4. **Implemented Filtering Logic** - Used helper functions *(\_query\_for* and *\_regex\_breeds*) to dynamically query MongoDB based on the selected rescue type.
5. **Created Interactive Components** - Added callbacks to update the table, map, and graph in real time whenever filters or selections changed.
6. **Enhanced User Interface** - Applied card-style layouts, zebra striped tables, blue row highlights, and modern button designs for usability.
7. **Added CSV Export Feature** - Implemented functionality to export the current filtered dataset to a downloadable CSV file.
8. **Tested and Documented** - Verified all interactions (filtering, resetting, map updates, CSV export) worked as intended and captured screenshots for documentation.

**Challenges and Solutions**

**Challenge 1: Dash callback synchronization**

Early versions produced “ID not found in layout” errors when callbacks were created before their associated elements were defined.

**Solution:** Moved all components definitions into the main layout and enabled *suppress\_callback\_exceptions=True* as needed for dynamically generated elements.

**Challenge 2: Data filtering precision**

Regex-based breed searches occasionally returned partial matches or missed mixed-breed entries.

**Solution:** Added the *$options: “I”* flag in the MongoDB query to ensure case-insensitive matching and properly escaped parentheses in breed names.

**Challenge 3: User interface consistency**

Buttons for reset and export initially displayed mismatched styles and selection circles.

**Solution:** Converted them into standard Dash *html.Button* components with shared styling to match the pill shaped radio buttons.

**Challenge 4: Map and graph disappearing**

When a filter produced an empty dataset, the graph and map elements vanished.

**Solution:** Added conditional checks and default fallbacks to handle empty results gracefully.

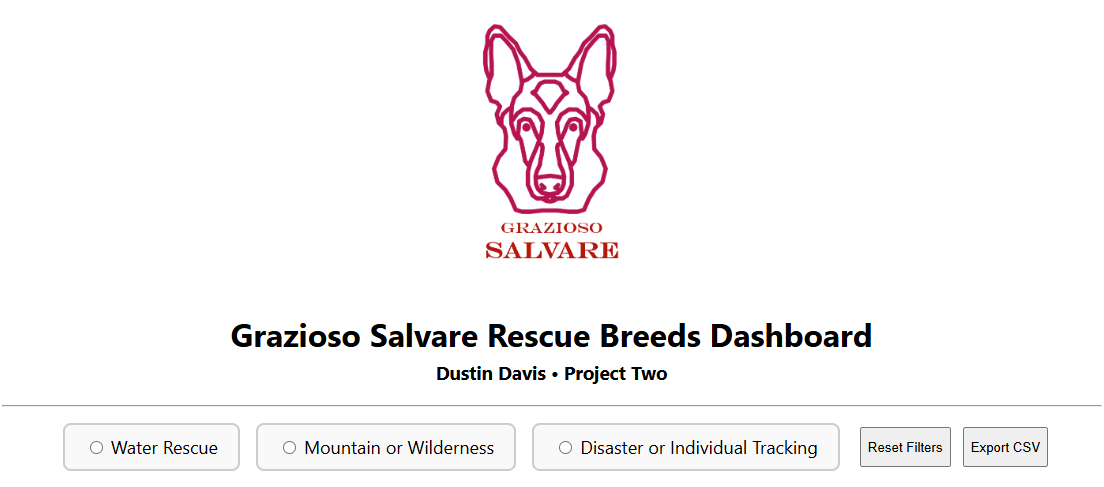
**Resources and References**

* **MongoDB Documentation:** <https://www.mongodb.com/docs/>
* **Dash Framework:** <https://dash.plotly.com/>
* **Plotly Express:** <https://plotly.com/python/plotly-express/>
* **Dash Leaflet:** <https://dash-leaflet.herokuapp.com/>
* **Python Pandas Library:** <https://pandas.pydata.org/docs/>
* **Course-provided starter code and MongoDB datasets**

**Proof of Functionality**

**Screenshots showing:**

1. **Dashboard Overview** - Logo, title, and button layout.



1. **Map and DataTable Integration** - Selecting a row updates the map marker.

NO SELECTION WATER RESCUE SELECTED

A screenshot of a map

AI-generated content may be incorrect. A screenshot of a map

AI-generated content may be incorrect.

1. **Mission Filter Example** - Display after choosing “Mountain or Wilderness.

A screenshot of a map

AI-generated content may be incorrect.

1. **Top Breeds Graph** - Horizontal bar chart showing breed counts.

A screenshot of a graph

AI-generated content may be incorrect.

1. **CSV Export** - Confirmation of downloaded filtered dataset.

A screenshot of a map

AI-generated content may be incorrect.