CS 340 Project Two - Grazioso Salvare Dashboard

**Project Overview**

This project is a client-facing dashboard built for Grazioso Salvare, an organization that identifies and trains rescue dogs. The dashboard connects to a MongoDB database containing data from the Austin Animal Center and provides interactive tools for filtering, visualizing, and locating dogs that meet specific criteria for rescue training.

**Required Functionality**

The dashboard includes the following functionality:  
- Interactive filtering options (Water Rescue, Mountain/Wilderness Rescue, Disaster/Tracking, Reset)  
- An interactive data table that dynamically updates based on filters  
- A breed distribution pie chart that updates with the selected filter  
- A geolocation map that updates with the selected row from the data table  
- Grazioso Salvare logo branding and a unique identifier on the dashboard

**Tools and Rationale**

The following tools were used in the development of this project:  
- MongoDB: Used as the model component to store and retrieve data. It supports flexible schemas, efficient querying, and integrates well with Python.  
- Python (Dash, Dash Leaflet, Plotly): Dash provides the MVC structure for the web application, with Dash Core Components for interactivity and Dash DataTable for tabular data. Dash Leaflet was used for geolocation mapping, and Plotly Express was used for the pie chart visualization.  
- Jupyter Notebook: Used as the development environment to build and run the dashboard within Apporto.

**Steps Taken**

1. Connected to the MongoDB database using a custom CRUD module.  
2. Created an unfiltered data table displaying all records from the database.  
3. Developed MongoDB queries for each of the required filters.  
4. Implemented Dash radio buttons for filter selection.  
5. Integrated callbacks to update the data table, pie chart, and map based on filter and row selection.  
6. Added Grazioso Salvare’s logo and student identifier to the dashboard layout.  
7. Tested all functionalities.

**Challenges and Solutions**

- Challenge: Handling MongoDB’s ObjectID in query results caused errors in the DataTable.  
 -Solution: Dropped the `\_id` column from the DataFrame before populating the table.  
  
- Challenge: Ensuring that the geolocation map updated correctly with row selections.  
 -Solution: Used derived\_virtual\_data and derived\_virtual\_selected\_rows to synchronize the map with table selections.  
  
- Challenge: Including the logo when the file was missing or not found.  
 -Solution: Implemented a try/except block to display a placeholder message if the image could not be loaded.

**Reproduction Instructions**

1. Install required dependencies (Dash, Dash Leaflet, Plotly, Pandas, Numpy, Pymongo).  
2. Ensure MongoDB is running with the Austin Animal Center Outcomes dataset loaded.  
3. Place the CRUD Python module (animal\_shelter.py) in the same directory as the dashboard file.  
4. Update username and password in the code if necessary.  
5. Run the ProjectTwoDashboard.ipynb file in Jupyter Notebook.  
6. The dashboard will launch, and you can interact with the filters, data table, charts, and map.

**Screenshots**

The following screenshots demonstrate the functionality of the dashboard:

A screenshot of a computer

AI-generated content may be incorrect.  
1. Starting state (unfiltered dashboard)

A screenshot of a computer

AI-generated content may be incorrect.  
2. Water Rescue filter applied

A screenshot of a computer

AI-generated content may be incorrect.  
3. Mountain/Wilderness Rescue filter applied

A screenshot of a computer

AI-generated content may be incorrect.  
4. Disaster/Tracking filter applied

A screenshot of a computer

AI-generated content may be incorrect.  
5. Reset filter applied (all data visible again)