Project Two README

**Grazioso Salvare Web Application Dashboard**

This project delivers a complete interactive dashboard designed for Grazioso Salvare, an international animal rescue training company. The dashboard integrates MongoDB data from regional animal shelters and allows users to identify dogs suitable for specific rescue operations, such as water rescue or wilderness tracking.

* Radio buttons for rescue type selection
* An interactive data table that dynamically updates
* A geolocation map showing each selected animal’s location
* A dynamic pie chart showing breed distribution for selected results
* Proper branding with the Grazioso Salvare logo and the developer’s name

**Screenshots**

* Starting State of Dashboard

A screenshot of a computer

AI-generated content may be incorrect.

* Water Rescue Filter Applied

A screenshot of a map

AI-generated content may be incorrect.

* Mountain or Wilderness Rescue Filter Applied

A screenshot of a map

AI-generated content may be incorrect.

* Disaster or Individual Tracking Filter Applied

A screenshot of a map

AI-generated content may be incorrect.

* Reset (returns all widgets to their original, unfiltered state) Dashboard

A screenshot of a computer

AI-generated content may be incorrect.

**Tools & Technologies Used**

* MongoDB (Model)
  + MongoDB is used as the backend model because of:
    - Schema-less structure (great for inconsistent real-world animal records)
    - Fast, flexible querying with Python using PyMongo
    - Easy integration into CRUD-based workflows
* Python CRUD Module (animal\_shelter.py)
  + This module serves as the Controller in the MVC pattern and contains all create(), read(), update(), and delete() methods needed to interact with MongoDB. It is imported into the dashboard file for querying data based on user filters.
* Dash & Dash Components (View)
  + Dash (Plotly): For building the web dashboard entirely in Python
  + Dash Leaflet: For the interactive map
  + Plotly Express: For generating a responsive pie chart

**Required Code Files**

Place the following files in the same directory:

1. ProjectTwoDashboard.ipynb – The main dashboard file
2. animal\_shelter.py – The CRUD Python module (used to connect to MongoDB)
3. Grazioso Salvare Logo.png – The company logo image to be displayed in the UI

Ensure MongoDB is running on your assigned Apporto virtual desktop environment.

**How to Run the Project**

This project must be executed in Jupyter Notebook on Apporto (or a local environment with Jupyter and MongoDB running). Follow these steps:

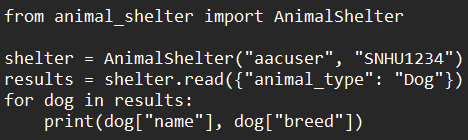
1. Open Apporto Environment
   1. Launch your virtual desktop (Linux or Windows) via SNHU Apporto
   2. Ensure MongoDB is pre-configured and running (typically via mongod on port 33126)
2. Upload Project Files
   1. Upload the following into your working Jupyter directory:
      1. ProjectTwoDashboard.ipynb
      2. animal\_shelter.py
      3. Grazioso Salvare Logo.png
3. Open and Run
   1. Open ProjectTwoDashboard.ipynb in Jupyter Notebook
   2. Run each cell in order by clicking "Run" or pressing Shift + Enter
   3. Click the local link in the output cell to open the interactive dashboard in a new tab
4. Interact
   1. Use the radio buttons to filter rescue types
   2. Select rows in the data table to update the map
   3. See breed breakdown in the pie chart next to the map

**animal\_shelter.py — CRUD Module Details**

This file handles all direct interaction with MongoDB. It includes:

* create(data) – Insert new animal records
* read(query) – Retrieve records based on filters (used by dashboard)
* update(query, updates) – Modify existing documents
* delete(query) – Delete records from the collection

Example usage in a script:



The dashboard code calls shelter.read(query) based on the filter selected in the interface. These queries are pre-defined in the notebook for water, mountain, and disaster rescues.

**Development Process**

Steps Taken:

1. Created and tested CRUD operations via the Python module
2. Connected MongoDB to the notebook and pulled full dataset
3. Implemented data filters for each rescue profile
4. Built interactive widgets (data table, radio buttons, map, pie chart)
5. Embedded the logo and personalized identifier into the UI
6. Verified layout and interactivity with screenshots for submission

**Challenges & Solutions**

* The \_id field from MongoDB caused the data table to crash due to its ObjectID type, so I fixed this issue by dropping the \_id column from the DataFrame before rendering the table.
* The map component would crash when no row was selected or if coordinates were missing, so I fixed this by setting a default selected row and adding logic to safely handle missing or invalid indexes.
* The pie chart originally appeared below the map and wasted horizontal space, so I fixed this layout issue by placing both inside a flex container to display them side by side.
* The Grazioso Salvare logo did not display properly inside the JupyterDash app, so I fixed this by encoding the image in base64 and embedding it using html.Img() with inline styling.
* When restarting the notebook and re-running the dashboard, I kept getting a port conflict error, so I fixed this by restarting the kernel or manually setting a new available port when launching the app.

**Future Enhancements**

While the current dashboard meets the project requirements, there are several opportunities to enhance its functionality and user experience in the future:

* Add more filter options such as filtering by age range, intake condition, or outcome subtype to give users more control over the data view.
* Export filtered results to CSV or PDF for reporting and analysis outside the dashboard.
* Improve mobile responsiveness to make the dashboard usable on tablets and phones.
* Add user authentication so that only authorized users can access or edit sensitive data.
* Implement advanced visualizations such as bar charts showing adoption trends or heatmaps showing outcome locations.
* Integrate live database updates so the dashboard reflects real-time data without manual refresh.
* Package as a standalone web app outside Jupyter using Dash with Flask or Docker for deployment.

These improvements would make the tool more useful for organizations like Grazioso Salvare and more adaptable for other rescue or animal welfare use cases.

**Author**

Abhilash Krishna Raj

Developed for CS-340 Client/Server Development at SNHU