**Project Two**

**Overview**

Grazioso Salvare trains rescue dogs to locate and save humans and animals in crisis. To streamline candidate selection, this dashboard connects to the Austin Animal Center’s database and allows staff to visualize, filter, and identify dogs suitable for specific rescue types (water, mountain, or disaster). The application uses Python, MongoDB, and the Plotly Dash framework to display data interactively through tables, charts, and maps — bringing real-time insight to animal selection.

**Project Functionality**

This dashboard performs the following key functions:  
- Connects to a MongoDB database containing shelter data via a secure CRUD module.  
- Retrieves and displays all animal records in an interactive data table.  
- Filters data dynamically by rescue type (Water, Mountain/Wilderness, Disaster/Individual Tracking, Reset).  
- Visualizes filtered results using a responsive pie chart and a geo-location map.  
- Highlights selected table rows and updates the map marker automatically.  
- Displays the Grazioso Salvare logo and developer identifier on all dashboard views.

**Tools/Framework**

|  |  |  |
| --- | --- | --- |
| Tool / Library | Purpose | Rationale |
| MongoDB | Data storage (Model) | Handles semi-structured JSON-like documents; ideal for animal shelter data. |
| PyMongo | Python driver for MongoDB | Provides simple CRUD operations and direct data access from Python. |
| Pandas | Data manipulation | Converts MongoDB documents into dataframes for Dash components. |
| Plotly Dash | Web framework (View + Controller) | Enables building rich, interactive dashboards directly in Python. |
| Dash Leaflet | Mapping component | Displays animal locations dynamically based on selected records. |
| Matplotlib / Plotly Express | Visualization libraries | Generate charts, such as breed distribution pie charts. |
| JupyterDash | Runs Dash apps in Jupyter Lab | Allows live dashboard testing within the development environment. |

**System Architecture (MVC Pattern)**

|  |  |  |
| --- | --- | --- |
| Layer | Technology | Description |
| Model | MongoDB via AnimalShelter CRUD class | Stores animal records and exposes CRUD operations for retrieval and filtering. |
| View | Dash components (tables, charts, map) | Presents data visually to the user. |
| Controller | Dash callbacks + CRUD module | Manages user interactions and updates dashboard elements dynamically. |

Project Reproduction

A. System Setup

1. Install and start MongoDB: sudo systemctl start mongod

2. Load the Austin Animal Center dataset into the database (DB: aac, Collection: animals).

B. Project Files

Ensure the following files are in the same directory: animal\_shelter.py, ProjectTwoDashboard.ipynb, Grazioso Salvare Logo.png

C. Run the Dashboard

1. Launch Jupyter Lab.  
2. Open ProjectTwoDashboard.ipynb.  
3. Run all cells in order.  
4. The dashboard will appear inline (mode='inline', port 8050).

**Testing and Verification**

Test each filter (Water, Mountain/Wilderness, Disaster/Individual Tracking, Reset) and observe the updates in the data table, chart, and map.

**Screenshots / Evidence of Functionality**

Insert the four required screenshots here:

Figure 1: Dashboard Reset View A screenshot of a computer

AI-generated content may be incorrect.

Figure 2: Water Rescue Filter Active A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 3: Mountain/Wilderness Filter Active A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 4: Disaster/Individual Tracking Filter Active A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Challenges and Resolutions**

|  |  |
| --- | --- |
| Challenge | Resolution |
| Dashboard failed to connect to MongoDB | Adjusted the AnimalShelter constructor call to include all six parameters. |
| DataTable crashed due to ObjectId type | Dropped the '\_id' column before displaying data. |
| Callbacks not updating properly | Verified correct Input/Output component IDs and ensured DataFrame conversion inside callbacks. |
| Map not displaying markers | Confirmed 'location\_lat' and 'location\_long' fields existed and used single-row selection for the map callback. |

**Reflection**

This project demonstrates how full-stack Python development can merge data science and web application design. Using Dash with MongoDB created an agile, modular, and easily maintainable dashboard capable of real-time updates without reloading the page. The Grazioso Salvare dashboard transforms raw shelter data into actionable insights for rescue-dog training operations.

**References**

Austin Animal Center Outcomes (2020). City of Austin Open Data Portal. <https://doi.org/10.26000/025.000001>

Plotly Dash Documentation – <https://dash.plotly.com>

MongoDB Python Driver (pymongo) Documentation – <https://pymongo.readthedocs.io>

Dash Leaflet Documentation – <https://dash-leaflet.herokuapp.com>