**CS-340 Project Two: Grazioso Salvare Dashboard**

**1. Required Functionality**

This project implements an interactive dashboard that visualizes Austin Animal Center data to help Grazioso Salvare identify dogs suitable for different types of rescues.

The dashboard connects to MongoDB through a custom CRUD module and provides:

Interactive radio buttons for filtering by rescue type:

Water Rescue

Mountain/Wilderness Rescue

Disaster Rescue

Individual Tracking

Reset (All)

A Dash DataTable that displays filtered data.

A bar chart showing breed counts that updates dynamically.

A geolocation map displaying the location of animals when a record is selected.

Header section with Grazioso Salvare logo and a unique identifier.

**2. Tools and Rationale**

MongoDB: Used as the database model because it handles flexible, semi-structured data efficiently and supports JSON-like querying.

Dash Framework: Provides the web-based structure for building interactive dashboards in Python.

Plotly Express: Used to create interactive breed count bar charts.

Dash Leaflet: Displays animal coordinates on a map dynamically.

Pandas: Used for reading MongoDB data and converting it into Dash-compatible dataframes.

**3. Steps Taken to Complete the Project**

1. Developed a CRUD module (AnimalShelter) to connect to MongoDB.

2. Queried data from MongoDB and loaded it into a Pandas DataFrame.

3. Created the dashboard layout using Dash and JupyterDash.

4. Added the Grazioso Salvare logo and identifying text.

5. Configured radio button filters for rescue type.

6. Added an interactive Dash DataTable with sorting, pagination, and filtering features.

7. Implemented callback functions to:

Update table data dynamically based on the selected rescue type.

Refresh the breed count bar chart using Plotly Express.

Update the map view when a record is selected.

8. Tested each rescue type and verified chart and map functionality.

**4. Challenges and Solutions**

Repeated Keyword Argument Error: Fixed by removing duplicate style\_data\_conditional parameters.

Indentation Errors: Solved by reformatting with consistent spaces (4 spaces per level).

Port in Use Error: Resolved by specifying a custom port using app.run\_server(port=8061).

Map Not Displaying: Corrected column references for latitude and longitude (columns 13 and 14).

Filter Mismatch: Updated dictionary keys in RESCUE\_BREEDS to exactly match the assignment’s rescue types.

**5. Screenshots**





