**CS 340 README**

**About the Project**

This dashboard was developed for Grazioso Salvare, a company specializing in training dogs for search-and-rescue operations. It connects to a MongoDB database provided by Austin-area animal shelters and allows users to filter, visualize, and explore dogs based on rescue suitability criteria. The dashboard supports operational decision-making by helping identify dogs that meet age, breed, and sex requirements for different types of rescue training.

Technologies Used

 **Backend**: Python, PyMongo, custom CRUD module

 **Frontend**: Dash, Dash Leaflet, Plotly

 **Data**: Austin Animal Center Outcomes dataset

 **Environment**: Jupyter Notebook with JupyterDash

pip install dash dash-leaflet jupyter-dash pandas numpy matplotlib plotly pymongo

**Steps Taken to Complete the Project**

1. **Set up MongoDB** and imported the Austin Animal Center dataset
2. **Built a Python CRUD module** with authentication and query support
3. **Created the dashboard layout** using Dash components
4. **Wrote MongoDB queries** for each rescue type based on client criteria
5. **Implemented callbacks** to update the table, chart, and map dynamically
6. **Tested each filter option** and captured screenshots
7. **Polished UI** with branding elements (logo, creator credit)
8. **Documented the project** in a Word-based README

* **Filter Options**: Radio buttons allow users to filter dogs by rescue type:
  + Water Rescue
  + Mountain/Wilderness Rescue
  + Disaster/Individual Tracking
  + Reset (returns all records)
* **Data Table**: Displays filtered results with sorting, pagination, and row selection
* **Pie Chart**: Visualizes breed distribution based on filtered data
* **Geolocation Map**: Displays the location and name of the selected dog

Each widget dynamically updates based on the selected filter, ensuring a seamless user experience. Screenshots were taken for each filter state and included in the submission to demonstrate full functionality.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Challenges and Solutions**

**1. ObjectID Crash in Data Table**

MongoDB returns \_id as an ObjectID, which caused the Dash table to crash. **Solution**: Dropped the \_id column using df.drop(columns=['\_id'], inplace=True)

**2. NoneType Errors in Callbacks**

Callbacks failed when no row was selected or data was missing. **Solution**: Added safe checks (if not viewData or not index) to prevent iteration over None

**3. Logo File Not Found**

Dashboard crashed when the logo image was missing. **Solution**: Wrapped image loading in a try-except block and added fallback logic

**4. Query Alignment with Client Criteria**

Initial filters were too broad and didn’t match the preferred breeds, age, and sex. **Solution**: Refined MongoDB queries using $in and age ranges from the client’s rescue table