Overview  
This interactive dashboard was developed for Grazioso Salvare; they specialize in training animals for search and rescue teams. The dashboard lets end-users analyze data from the Austin Animal Center to identify dogs suitable for specific rescue missions based on breed, age, and sex.

The dashboard was built using:  
- JupyterDash and Dash for layout and interactivity.  
- MongoDB for data storage.  
- Pandas and Plotly for visualization  
- Dash Leaflet for location mapping

Features  
- Dynamic data table with filtering, sorting, and row selection.  
- Bubble-style radio filter for rescue types including Water Rescue, Mountain or Wilderness Rescue, and Disaster or Individual Tracking.  
- Breed distribution pie chart that updates with search results.  
- Interactive map that pinpoints the selected dog’s location.  
- MongoDB integration via a Python-based CRUD class.  
- Branding elements such as a logo and student attribution.

Setup and Launch Instructions

1. Make sure you have the following packages installed:  
   - dash  
   - dash-leaflet  
   - jupyter-dash  
   - pandas  
   - plotly
2. Run the Jupyter Notebook and open the dashboard file.
3. Make sure the AnimalShelter.py file is in the same directory.
4. Run MongoDB instance.
5. Run all cells in the notebook to launch the dashboard.

Usage Notes  
- Use the radio buttons to filter rescue dogs by mission type.  
- View key information in the data table.  
- View the selected dog’s breed breakdown and map location update automatically.  
- Reset filter to return to the full dataset.

Challenge

A challenge I encountered during development was ensuring that the dashboard could reliably read data from MongoDB hosted in a virtual lab environment (Apporto). Early versions of the CRUD module used default localhost configurations, which led to repeated connection errors and failed queries when I was testing.

Solution

I was able to overcome this by defining the host URL, port, and authentication parameters in the MongoDB connection string within the AnimalShelter.py file. By updating the host to the lab’s remote server and carefully matching the database, I was able obtain a stable connection. This allowed the dashboard to pull live data and dynamically update based on user interactions.