# CS 340 README

***About the Project***

*This is a web-based dashboard created for Grazioso Salvare, a company that trains rescue animals. The dashboard helps users explore data from a MongoDB database containing shelter animal details. It allows filtering, live updates, and mapping features to help identify dogs that might be good candidates for search-and-rescue training. The project was developed using* ***Jupyter Notebook****.*

***Why This Project?***

*Grazioso Salvare needed an easy-to-use tool to sort through shelter animal data and find dogs fit for rescue training. This dashboard makes it simple to filter by rescue type, breed, and location, making the search process more efficient.*

***Getting Started***

*To set up the project on your local machine, follow these steps:*

1. *Install* ***Python*** *if you don’t already have it.*
2. *Install the required libraries.*
3. *Open the* ***Jupyter Notebook*** *environment.*
4. *Run the Dash application from within Jupyter.*

***Installation***

***What You’ll Need***

* *Install the necessary Python packages using:*
* *pip install dash pandas plotly pymongo jupyter\_dash dash\_leaflet*
* *Ensure MongoDB is running and that you have the correct login credentials.*

***How It Works***

*The dashboard includes these main features:*

* ***Filtering Options:*** *Users can pick a rescue category to filter the results.*
* ***Dynamic Table:*** *The data table updates automatically based on selected filters.*
* ***Geolocation Map:*** *Displays the location of shelter animals.*
* ***Pie Chart:*** *Shows the distribution of breeds in the dataset.*

***Running the Dashboard in Jupyter***

1. *Open* ***Jupyter Notebook****.*
2. *Run ProjectTwoDashboard.ipynb.*
3. *The dashboard will open within Jupyter and allow interactive filtering.*

***Example Code***

*@app.callback(Output('datatable-id', 'data'),*

*[Input('filter-options', 'value')])*

*def update\_dashboard(filter\_type):*

*if filter\_type == "Reset":*

*filtered\_df = df # Show all data when reset is selected*

*else:*

*filtered\_df = df[df['breed'].str.contains(filter\_type, case=False, na=False)]*

*return filtered\_df.to\_dict('records')*

***Screenshots***

*Screenshots of the dashboard in action:*

1. *Default view (no filters applied)A screenshot of a computer

   AI-generated content may be incorrect.*
2. *Water Rescue filter appliedA screenshot of a computer

   AI-generated content may be incorrect.*
3. *Mountain Rescue filter appliedA screenshot of a computer

   AI-generated content may be incorrect.*
4. *Disaster Tracking filter appliedA screenshot of a computer

   AI-generated content may be incorrect.*
5. *Reset view (back to original data)A screenshot of a computer

   AI-generated content may be incorrect.*

***Tools Used***

* ***MongoDB:*** *Stores and organizes shelter data efficiently.*
* ***Dash Framework:*** *Powers the user interface and dashboard interactivity.*
* ***Plotly:*** *Creates interactive charts and maps.*
* ***Pandas:*** *Handles data processing and filtering.*
* ***Jupyter Notebook:*** *Used to run and develop the interactive dashboard.*

***Challenges & Fixes***

* ***Slow Data Retrieval:*** *Optimized MongoDB queries to speed things up.*
* ***Map Display Issues:*** *Used Dash Leaflet to correctly render geolocation data.*
* ***Jupyter Integration:*** *Ensured Dash works seamlessly within Jupyter Notebook.*
* ***User Experience Tweaks:*** *Adjusted layout and filtering to make navigation easier.*

***Contact***

***Developer:*** *Yakuba Conde*

*For questions, reach out via email or through the project repository.*