# CS 340 README Template

## About the Project/Project Title

This project is the Grazioso Salvare Animal Dashboard, designed to display data about animals from a MongoDB database. It includes the functionality to filter animals by specific criteria (such as breed or rescue type), visualize the data in a table and pie chart, and show selected animals on a map with location markers.

## Motivation

The motivation behind this project is to help Grazioso Salvare quickly identify suitable animals for different rescue operations (e.g., water rescue, mountain or wilderness rescue, disaster rescue). By combining a visual dashboard (built with Dash) and a flexible database (MongoDB), team members can easily search, filter, and locate animals that meet their exact requirements.

## Getting Started

To get this project up and running you would want to clone/download the project files in you local environment of choice (This project is used with jupyter notebook using the Python programming language). Make sure everything needed is downloaded (ie. Dash, Plotly, dash-leaflet, pandas, PyMongo). Using the CRUD module that is imported into the IPYNB file we can do all the necessary functions required in MongoDB. When trying to connect to the MongoDB database make sure that the correct USER, PASS, HOST, PORT, DB, and COLLECTION for your MongoDB Server.

## Usage

1 Jupyter Environment:

* Open the notebook or .ipynb file containing the dashboard code.
* Run the cells in order.
* Look for a message indicating the server is running (e.g., Running on http://127.0.0.1:8050/).
* Click the link or open it in a browser to view the dashboard.

2 Dash as a Script:

* Alternatively, you can place the code in a .py file, then run python your\_dashboard\_file.py.
* Open the provided URL ( http://127.0.0.1:{PORT}) in your browser.

3 Filtering Data:

* Use the radio buttons on the dashboard to filter animals by rescue type.
* The table updates automatically based on your selection.

4 Viewing Details:

* Select a row in the data table to see additional info in the pie chart and map.

MongoDB was chosen for its flexible, document-based structure that can store varied animal data easily. It integrates smoothly with Python through PyMongo, allowing quick CRUD (Create, Read, Update, Delete) operations and agile schema updates if data fields change in the future.

Dash by Plotly provides a powerful, Python-based framework for building interactive web applications. It automatically handles the “view” (HTML layout) and “controller” (callbacks) with minimal boilerplate. Dash’s integration with Plotly Express and dash-leaflet helps us display charts and maps within the same page.

**Challenges and Solutions**

* **ObjectID Fields**: The MongoDB \_id object caused the data table to crash, so we removed it (df.drop(columns=['\_id'], inplace=True)) before rendering.
* **Filtering Logic**: Determining the correct breed, sex, and age filters for each rescue type required careful attention to detail. We resolved it with clear $and queries in the update\_dashboard callback.
* **Deployment**: Running the app in a cloud or local Jupyter environment involved ensuring correct port forwarding and environment paths. Properly installing jupyter\_dash and referencing the correct path for images (the logo) fixed issues with broken images.

### Screenshots

*Provide screenshots that demonstrate your work.*

*A screenshot of a computer

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.A screenshot of a computer

AI-generated content may be incorrect.*

## Contact

Your name: Mason Williams