# CS 340 README

# Grazioso Salvare Search Dashboard

## Project Overview

The Grazioso Salvare Search Dashboard is a data-driven web application built using Dash, Python's web framework. The purpose of this project is to create an interactive dashboard that provides insights and allows users to explore animal rescue data from an animal shelter.

### Functionality

The dashboard offers the following key functionalities:

- \*\*Filtering Data\*\*: Users can filter animal rescue data based on different rescue types, including Water Rescue, Mountain or Wilderness Rescue, and Disaster or Individual Tracking.

- \*\*Interactive Visualization\*\*: The dashboard displays the distribution of preferred animal breeds using a pie chart, allowing users to visualize the most common breeds.

- \*\*Geolocation Representation\*\*: Users can select a specific data entry from the table, and the corresponding location of the animal is highlighted on the map.

## Tools and Technologies Used

### MongoDB

The MongoDB database was chosen for its NoSQL structure, allowing flexible schema design and efficient storage of unstructured data. The PyMongo library facilitated seamless integration with Python, enabling data retrieval and manipulation.

### Dash Framework

Dash, a productive Python framework for building web applications, was utilized to create the dashboard's frontend. Its reactive components and interactivity capabilities enabled the creation of an engaging and dynamic user interface.

## Project Development Steps

### 1. Data Retrieval

The PyMongo library was employed to establish a connection to the MongoDB database, fetching animal rescue data. The retrieved data was then transformed into a pandas DataFrame for further processing.

### 2. Dashboard Design and Layout

The dashboard layout was designed using HTML and Dash components. It includes filtering options, a data table, charts for breed distribution, and a map showing geolocation data.

### 3. Interactivity Implementation

Dash callbacks were used to enable real-time interactivity. Filtering logic based on rescue types was implemented to update the displayed data table, pie chart, and map based on user selections.

### 4. Deployment

The JupyterDash package was utilized for testing and viewing the dashboard's functionality within a Jupyter Notebook environment. Deployment to a production environment can be achieved using various hosting services.

## Challenges Faced and Solutions

### Challenge 1: Complex Filtering Logic

\*\*Solution\*\*: Conditional logic was applied to filter data accurately based on different rescue types. Careful consideration of the filter conditions ensured the correct retrieval of data from the MongoDB database.

### Challenge 2: Responsive Layout Design

\*\*Solution\*\*: Dash's flexible layout components and styling attributes were employed to create a responsive dashboard layout. Utilizing the `className` property and CSS styling ensured an adaptable layout across various devices and screen sizes.

## Resources and References

- [MongoDB Official Website](https://www.mongodb.com/)

- [Dash Documentation](https://dash.plotly.com/)

- [PyMongo Documentation](https://pymongo.readthedocs.io/en/stable/)

## Screenshots and Deployment

Screen recording showcasing the dashboard's functionality during testing and deployment can be seen below:

