**Project Overview**

The Grazioso Salvare Dashboard is a web application designed to visualize data from an animal shelter database. The project utilizes a MongoDB database as the backend for storing animal data, while the frontend uses Dash (Plotly) to present a user-friendly, interactive dashboard that displays animal details, filters, graphs, and a geographic map of animal locations.

This README document provides instructions for setting up, running, and understanding the project, as well as a description of the tools used, and the challenges encountered during development.

**Required Functionality**

1. Data Display:

- The dashboard presents a table of animals retrieved from a MongoDB database, with the ability to filter data based on the animals’ outcome type (e.g., adoption, transfer, return to owner).

2. Interactive Graphs:

- A pie chart visualizes the distribution of different animal breeds in the shelter, updating dynamically based on the filtered data.

3. Geographic Map:

- A map displays the location of an animal when a specific row is selected from the data table, showing the animal's latitude and longitude along with a tooltip displaying the animal’s name and outcome.

**Proof of Functionality**

Below are screenshots showing the working dashboard:

1. Dashboard Overview:

A screenshot of a computer

Description automatically generated

2. Data Table & Filters:

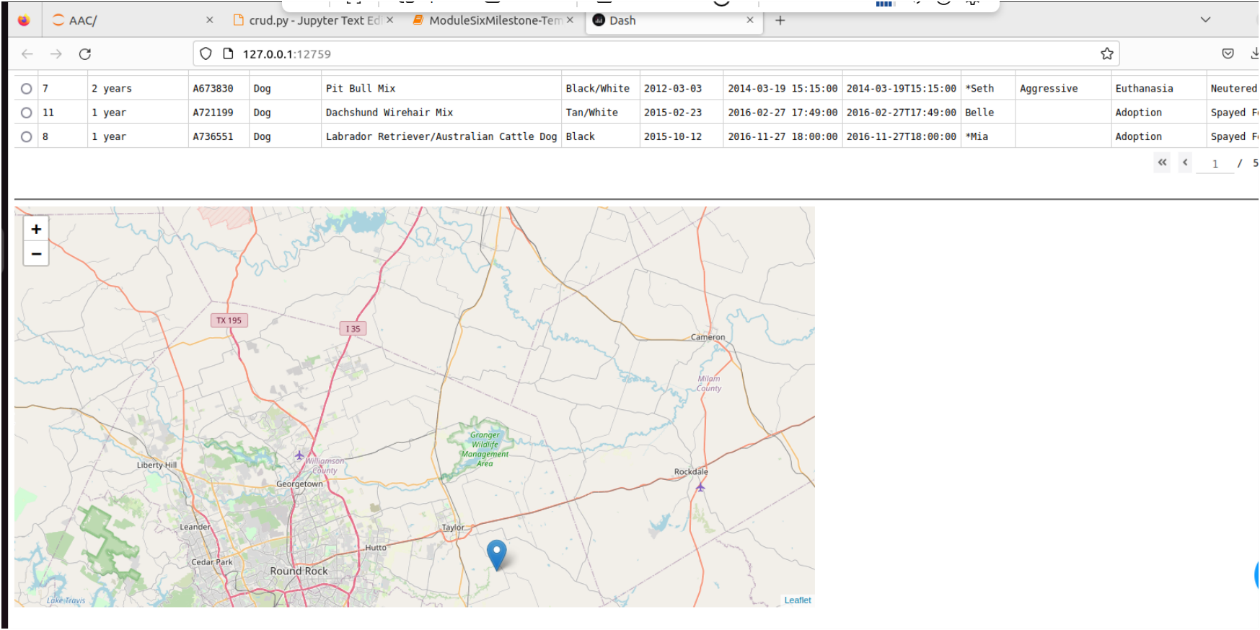
A screenshot of a computer

Description automatically generated3. Interactive Pie Chart:

A graph with a pie chart

Description automatically generated

4. Animal Location Map:



**Tools Used**

MongoDB (Model)

MongoDB is a NoSQL database that excels in handling large datasets with flexible schema structures, making it ideal for unstructured or semi-structured data, like animal shelter records. Its ability to store documents in a JSON-like format allows for easy integration with Python using the `pymongo` library.

Capabilities:

- Scalability: MongoDB’s flexibility allows it to handle large datasets.

- Query Flexibility: MongoDB’s powerful querying language can easily filter, and aggregate data based on complex conditions, which is used in this project to retrieve animal data from the shelter.

- Integration with Python: MongoDB's `pymongo` library allows Python to easily communicate with the database and fetch data.

**Dash Framework (View & Controller)**

Dash, a framework built on Flask, Plotly, and React, is specifically designed for building web applications with interactive data visualizations. It provides a simple interface for building dashboards and is highly compatible with Plotly, which is great for creating the charts in the project.

Capabilities:

- Interactive Components: Dash supports real-time interactivity between user inputs (like dropdowns) and the displayed data (like tables and graphs).

- Ease of Integration: Dash integrates smoothly with Plotly for creating graphs and maps, which is essential for the visual aspects of this project.

- Layout Flexibility: Dash provides a simple, yet powerful way to lay out components and manage user interactions with Python callbacks.

**Additional Libraries**

- Pandas: Used to manipulate and transform data retrieved from MongoDB into tabular form, making it easier to visualize.

- Dash Leaflet: Provides map components to display geographic information, such as the location of animals in the shelter.

- Base64: Encodes images for use in the dashboard (e.g., adding a logo or image).

**Project Setup**

To set up the project locally, follow these steps:

Prerequisites

- Python 3.x

- MongoDB installed or access to a MongoDB instance.

- Libraries: Install dependencies by running the following command:

pip install dash pandas pymongo dash-leaflet jupyter-dash

**Running the Project**

1. Clone the repository:

git clone <repository-url>

2. Set up MongoDB:

Ensure MongoDB is running and accessible with the correct credentials. Modify the `username`, `password`, and `host` values in the code to match your database setup.

3. Run the application:

Start the dashboard by running:

python ProjectTwoDashboard.py

4. Access the dashboard:

Open your browser and navigate to `*http://127.0.0.1:8050/*` to see the dashboard.

**Project Breakdown**

1. Database Connection:

- We connect to MongoDB using the `AnimalShelter` class, which abstracts the MongoDB CRUD operations (create, read, update, delete).

2. Dashboard Layout:

- The dashboard is designed using Dash components (`html`, `dcc`, `dash\_table`, `dl.Map`), allowing for the display of a data table, graphs, and a map.

3. Interactivity:

- Filters are applied using a dropdown, which dynamically updates the table and charts.

- Selecting a row from the data table highlights the corresponding location on the map.

**Challenges Faced**

1. Data Retrieval Issues:

At the start, the dataset was not structured properly, leading to difficulties in querying. This was resolved by ensuring a consistent schema in MongoDB and using `pandas` to properly format and clean the data.

2. Interactive Map:

Ensuring the map updated correctly with each row selection was challenging. We overcame this by carefully using Dash callbacks and ensuring the data passed to the map was in the correct format.

3. Performance:

With 50,000+ records in the MongoDB collection, performance issues were encountered when rendering the full dataset. Filtering and optimizing queries helped to reduce the lag in the dashboard.

**Resources and Software Links**

- MongoDB Documentation: [https://www.mongodb.com/docs/](https://www.mongodb.com/docs/)

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

- Pandas Documentation: [https://pandas.pydata.org/](https://pandas.pydata.org/)

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