**CS 340 Grazioso Salvare Animal Rescue Dashboard**

**README**

## **About the Project/Project Title**

This project implements an interactive dashboard designed for Grazioso Salvare. The dashboard facilitates filtering, visualization, and management of animal rescue data stored in MongoDB. It provides tools for analyzing rescue types, geographic rescue locations, and breed distributions via dynamic filters, charts, and geolocation maps.

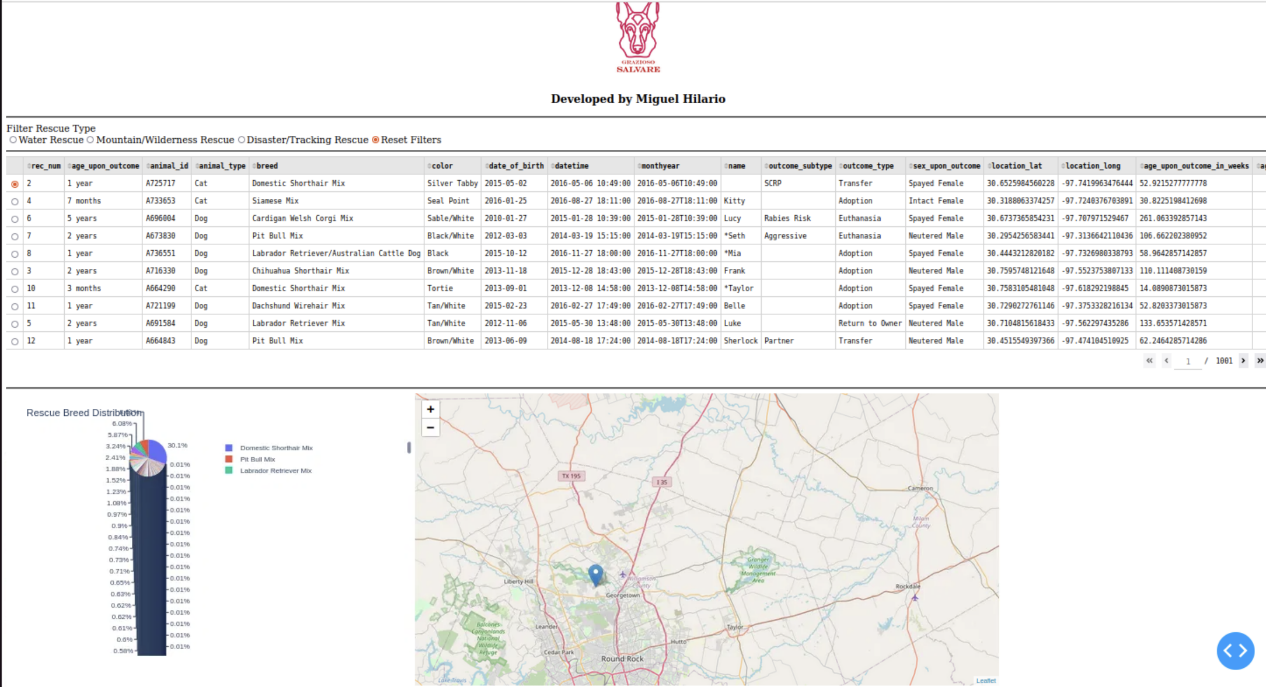
## **Motivation**

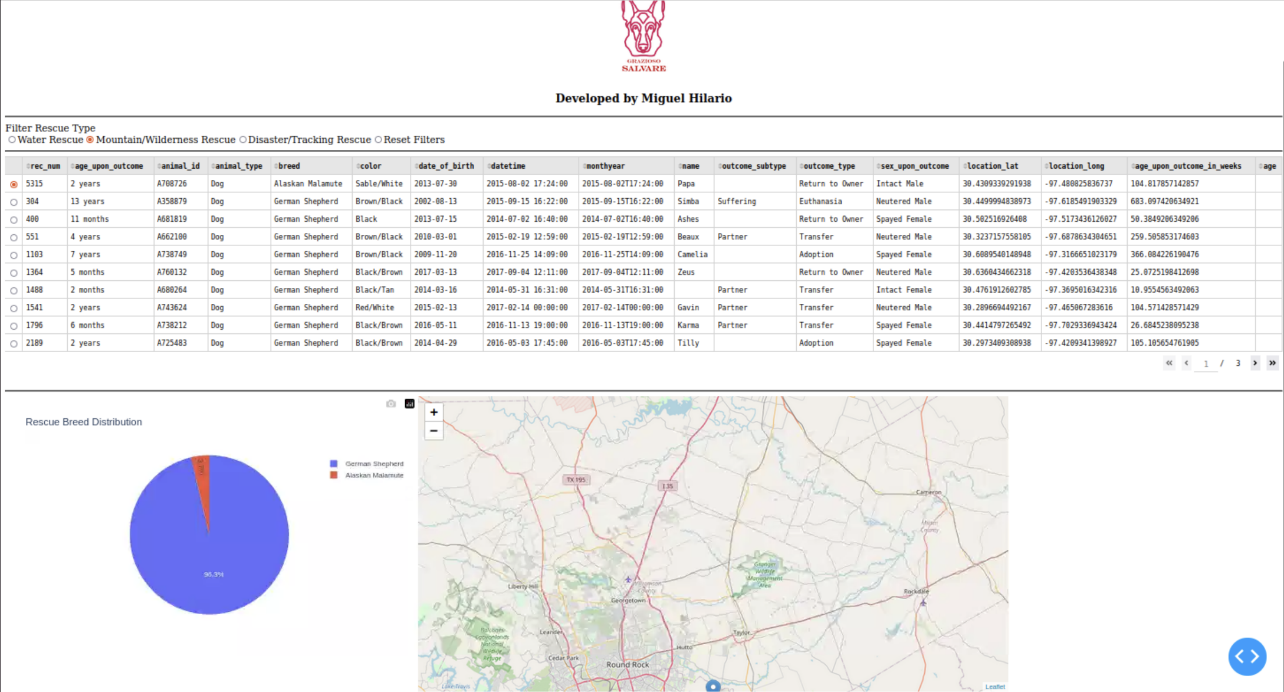
The project was developed to streamline the visualization and analysis of animal rescue data. It demonstrates how Python and MongoDB can work together to create a user-friendly interface for managing data, while also showcasing modern frameworks for web development.

**Getting Started**

To get a local copy of the project up and running, follow these steps:

* Install Python (3.9 or higher is recommended) on your system.
* Set up a MongoDB database and create a user with the following credentials:
  + Username: aacuser
  + Password: kakarot
* Ensure the database (AAC) and collection (animals) exist and grant the user read & write permissions.
* The following are screenshots of the dashboard functionality:





## **Installation**

Tools Used

* MongoDB: Used as the model layer to store and query animal rescue data efficiently.
* Python: Provides the backbone for scripting and dashboard logic.
* Dash: Framework for developing interactive web dashboards in Python.
* Plotly Express: Library used for creating charts, such as breed distribution.
* Dash Leaflet: Used for rendering geolocation data visually.

*To set up the environment, install the following dependencies:*

* pip install dash plotly pandas pymongo dash-leaflet
* Ensure MongoDB is installed and running.

## **Usage**

The Animal Rescue Dashboard supports dynamic data filtering, visualization, and mapping:

* Filters: Users can filter data by rescue types (e.g., Water, Mountain/Wilderness, Disaster/Tracking, and Reset).
* Interactive DataTable: Displays animal data, supports sorting, pagination, and row selection.
* Geolocation Map: Visualizes rescue locations on an interactive map.
* Pie Chart: Provides breed distribution analysis and includes scrollable legend functionality for detailed insights.

**Code Example**

The dashboard integrates CRUD operations with MongoDB to retrieve and display data dynamically:

A screen shot of a computer screen

AI-generated content may be incorrect.

**Steps Taken for Completion**

Initial Setup:

* Installed required Python libraries: Dash, Plotly, Dash Leaflet, Pandas, and PyMongo.
* Configured the MongoDB database by setting up the AAC database and the "animals" collection, along with user authentication credentials.

CRUD Module Development:

* Built a CRUD Python module (mongo\_crud.py) to handle Create, Read, Update, and Delete operations on MongoDB.
* Verified its functionality with MongoDB queries to support the dashboard.

Dashboard Implementation:

* Created components like filters, DataTable, geolocation map, and pie chart using Dash.
* Integrated MongoDB queries to dynamically update dashboard components based on user inputs.

Dynamic Updates:

* Developed callbacks to ensure seamless interactions between filters, DataTable, map, and charts.
* Added functionality to select a free port dynamically during deployment to resolve conflicts.

Testing and Debugging:

* Validated functionality of filters, dynamic updates, interactivity, and chart responsiveness.
* Included error-handling mechanisms for missing or invalid geolocation data.

**Challenges Encountered**

MongoDB Connection Issues:

* Challenge: Difficulties in connecting to the MongoDB database due to authentication errors.
* Solution: Reviewed credentials and permissions to ensure the aacuser account had appropriate readWrite access.

Port Conflicts During Deployment:

* Challenge: Predefined ports were often already in use, leading to server errors.
* Solution: Implemented a dynamic port selection mechanism using Python’s socket library.

Pie Chart Legend Visibility:

* Challenge: The legend in the breed distribution chart displayed too few items, making it unclear to users.
* Solution: Adjusted chart height, enhanced legend positioning, and added a user-friendly note regarding scrolling functionality.

Geolocation Map Errors:

* Challenge: Incomplete or invalid latitude/longitude data in some records caused rendering issues on the map.
* Solution: Implemented validation checks to handle missing or invalid data, providing fallback messages.

### Tests

* Filtering Functionality: Applied rescue type filters and verified updates to DataTable, charts, and maps.
* DataTable Interaction: Tested row selection, pagination, and sorting capabilities.
* Map Integration: Validated rescue location plotting and tooltip/pop-up functionality.
* Legend Scroll Visibility: Verified the breed distribution chart displays multiple items with scroll functionality.

**Contact**

Miguel Hilario