**Yancarlo Guzman**

**October 20, 2024**

**CS -340-10940-M01**

**Grazioso Salvare Rescue Dog Dashboard “Read Me”**

**Overview of the Project**

This project is a web-based dashboard created for the rescue dog training business Grazioso Salvare. The dashboard lets the customer visualize canine data for identifying and classifying possible search-and-rescue candidates by connecting to the Austin Animal Center Outcomes database, which is hosted on MongoDB. The dashboard has the following:

The following interactive filter options are available for rescue types: Reset, Disaster/Individual Tracking, Mountain/Wilderness Rescue, and Water Rescue.

Data table that initially displays unfiltered animal data and dynamically adapts to filtering settings.

Animal locations are visualized using a geolocation chart based on the data.

To show a breakdown of breeds according to the chosen rescue type, use a pie chart.

**Screenshots:**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer screen

Description automatically generated**

**Tools and Technology**

MongoDB

All of the data for Austin Animal Center Outcomes was stored in MongoDB, which served as the application's model component. For this use scenario, MongoDB offers the following benefits:

* Flexibility: Because MongoDB is a NoSQL database, it effectively manages semi-structured and unstructured data.
* Python Integration: The pymongo module provides robust support for MongoDB in Python, allowing for smooth CRUD operations in the Python environment.
* Scalability: When managing enormous animal datasets across several shelters, MongoDB's capability for scaling enables the application to manage large datasets effectively.

Dash Framework

The dashboard's view and controller layers were implemented using Dash. Its ease of use in developing web-based data applications led to its selection:

* Interactivity: Without requiring extensive knowledge of front-end development, Dash makes it simple to construct interactive data visualizations, including responsive charts and filtering tools.
* Integration with Plotly: Dash works nicely with Plotly to create charts and graphs, which were crucial for this project's data visualization.
* Support for Jupyter Notebooks: Dash applications can be executed directly within Jupyter Notebooks, which facilitates dashboard development and testing throughout the project.

Python

The project was coded using python with the following listed libraries:

* **pandas** for data manipulation.
* **pymongo** for MongoDB interaction.
* **dash, dash\_table, dash\_leaflet** for dashboard components.

**Project Setup**

1. Take these actions to replicate this project:

2. Download the project files or clone the repository.

Install the necessary libraries:

*pip install dash pandas plotly dash-leaflet pymongo*

3. Install MongoDB: Make sure the Austin Animal Center Outcomes dataset is in a MongoDB collection and that MongoDB is operational.

4. Set up your database connection: Modify the dashboard script's (ProjectTwoDashboard.ipynb) database credentials (password, host, port, username, etc.).

5. Launch the application: Launch the dashboard after opening the Jupyter Notebook and running the cells. A local web server will provide access to the dashboard.

**Challenges Faced:**

1. Connection to MongoDB

Making sure the right authentication mechanism were used and properly setting the MongoDB connection were two initial challenges. Pymongo was used to fix this, and environment variables were set up appropriately to safely store sensitive credentials.

1. Callback Logic Dash

It was necessary to manage multiple interdependencies between the data table and charts in order to implement the callback logic for data filtering and geolocation map updating. This was fixed by meticulously planning the callback procedures to guarantee that the filters updated every dashboard element simultaneously.

1. Visualization of Data

Data modification was necessary to ensure that the \_id field from MongoDB was eliminated and to format the data appropriately for display in the Dash DataTable. Pandas was used for this, preprocessing the data before it was displayed in the dashboard.

**Resources and links:**

MongoDB: <https://www.mongodb.com/>

Dash Documentation: https://dash.plotly.com/

Plotly: <https://plotly.com/>

pandas: https://pandas.pydata.org/

JupyterDash: https://medium.com/plotly/introducing-jupyterdash-811f1f57c02e