**README Austin Animal Center Outcomes Dashboard**

[**Project Description**](#ProductDescription)[**Required Functionality**](#RequiredFunctionality)[**Tools Used and Rationale**](#ToolsUsedandRationale)[**MongoDB Usage**](#MongoDBUsage)

[**Dash Framework**](#DashFramework)[**Resources and Software Applications**](#ResourcesandSoftwareApplications)[**Project Completion Steps**](#ProjectCompletionSteps)

[**Challenges and Solutions**](#ChallengesandSolutions)[**Screencast**](#Screencast)[**Contact**](#Contact)

**Project Description**

This README file accompanies the Austin Animal Center Outcomes Dashboard, an interactive web application that allows users to explore and filter animal data from the Austin Animal Center. The dashboard provides filtering options based on different rescue types, displays animal data in a DataTable, visualizes the count of each unique breed using a pie chart, and shows the geolocation of selected animals on a map.

**Required Functionality**

The project achieves the following functionalities:

1. Displays Grazioso Salvare's logo, acting as a hyperlink to their home page.
2. Provides interactive filtering options using radio buttons for different rescue types.
3. Presents the animal data in a DataTable, enabling users to edit, filter, and sort the data.
4. Implements a pie chart to visualize the count of each unique breed based on the DataTable's data.
5. Displays a geolocation chart (map) showing the location of a selected animal from the DataTable.

**Tools Used and Rationale**

The project utilizes the following tools for implementation:

1. **Dash:** A Python web framework enabling the creation of interactive web applications using Python.
2. **Dash Leaflet**: A Dash component integrating interactive maps (Leaflet.js) into Dash applications.
3. **Plotly Expres**s: A data visualization library used to create the pie chart with minimal code.
4. **Pandas**: A data manipulation and analysis library in Python, used to handle and transform data from MongoDB for display in the DataTable.
5. **MongoDB:** The model component of the development, chosen for its flexibility, scalability, and efficient handling of JSON-like data when interfacing with Python.

**MongoDB Usage**

MongoDB is used as the model component due to its qualities and capabilities:

1. **Flexible Schema:** MongoDB's document-based nature accommodates storing data with varying attributes, such as animal records with different information.
2. **JSON-like Data Storage**: MongoDB's support for JSON-like documents simplifies storing Python objects directly, easing data handling.
3. **Scalability**: MongoDB handles large datasets and scales horizontally by distributing data across multiple servers.
4. **Fast Queries:** MongoDB's indexing and querying capabilities enable efficient retrieval of data, crucial for real-time applications with large datasets.

**Resources and Software Applications**

The following resources and software applications were used:

1. [Dash Documentation](https://dash.plotly.com/): Official documentation for Dash web framework.
2. [Plotly Express Documentation](https://plotly.com/python/plotly-express/): Official documentation for Plotly Express data visualization library.
3. [Dash Leaflet Documentation](https://dash-leaflet.herokuapp.com/): Documentation for Dash Leaflet, integrating interactive maps into Dash applications.

**Project Completion Steps**

The project was completed through the following steps:

1. **Data Retrieval:** MongoDB was connected using the animal\_shelter.py Python module, fetching animal data into a pandas DataFrame.
2. **Data Processing:** The DataFrame was cleaned by removing the \_id column to avoid DataTable compatibility issues.
3. **Dash Layout:** The dashboard layout was designed using HTML and Dash components, including the logo, filtering options, DataTable, pie chart, and geolocation chart.
4. **Dash Callbacks:** Callbacks were implemented to respond to user interactions and update components accordingly, such as filtering the DataTable and updating the pie chart and map.
5. **Testing and Deployment:** The dashboard was tested for functionality and deployed using the JupyterDash library for demonstration.

**Challenges and Solutions**

The project encountered challenges in filtering logic, callback dependencies, data cleaning, and integrating the interactive map. These were overcome through careful debugging, testing, and consultation with relevant documentation.

**Screencast**

A screencast demonstrating the functionality of the dashboard:

****

Please note that this README focuses on the dashboard's functionality, not the source code details. For technical documentation and source code details, refer to my GitHub and comments.

## Contact

If you have any questions or need assistance with the Python CRUD module, feel free to reach out:

**Name:** Christine Emerson [Email](mailto:christine.emerson@snhu.edu) [LinkedIn](https://www.linkedin.com/in/christine-emerson-b0631b190) [GitHub](https://github.com/Cemerson77)

**Top of Form**