**Project Documentation**

**Purpose**

This project is an interactive dashboard for Grazioso Salvare, designed to help identify and categorize dogs suitable for search-and-rescue training. The dashboard integrates MongoDB for data management and Dash for the web application interface.

**Functionality**

* **Interactive Filtering**

Filters dogs by rescue type: Water Rescue, Mountain/Wilderness Rescue, Disaster/Tracking.

* **Dynamic Data Table**

Displays filtered results in real-time.

* **Charts:**
  + Geolocation Chart: Visualizes rescue data spatially.
  + Secondary Chart: Displays Breed Distribution
* **Branding**

Includes Grazioso Salvare’s logo and creator credit.

***Screen capture will be included here in GitHub. In the case of the Brightspace submission, the screen capture had to be included separately.***

**Tools Used**

* **Python**

Python is an object-oriented programming language, used in this project to manipulate the database with more complex logic than MongoDB alone offers

[Python Documentation](https://www.python.org/)

* **MongoDB**

MongoDB is a NoSQL database designed to handle large datasets with flexible schema, ideal for the semi-structured nature of animal shelter data.

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

* **PyMongo Library**

Provides a Pythonic interface to interact with MongoDB, allowing CRUD operations and aggregation queries.

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

* **Dash Framework**

Dash, developed by Plotly, is a Python-based framework designed for building interactive, data-driven web applications. It Integrates seamlessly with Python, enabling developers to use familiar libraries for data visualization and processing (e.g., pandas, Plotly, NumPy).

* + **Plotly and Pandas:** Enable advanced data visualization and efficient processing, essential for presenting the filtered data in an intuitive way.

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

**Project Recreation**

**Completion Steps:**

**1. Data Preparation**

* Upload cleaned dataset to MongoDB.

**2. Backend Development**

* Use PyMongo to connect the Dash app to the MongoDB database.
* Write queries to filter dogs by:
  + Rescue type (Water, Mountain, Disaster).
  + Breed, age, and training suitability.
* Test the queries in Python to ensure accuracy.

**3. Dashboard Development**

* Set Up Dash App:
  + Create the app layout with the following components:
    - Filtering options (dropdowns or buttons)
    - A data table for displaying filtered results.
    - Geolocation chart using plotly.
    - Secondary chart (e.g., pie chart).
* Configure Dash callbacks to update components dynamically based on user actions.
* Branding**:**
  + Add Grazioso Salvare’s logo and your identifier to the app layout.

**4. Testing**

* Test the app locally to ensure:
  + Filters work correctly.
  + Charts and tables update dynamically.
  + Data queries perform efficiently.
* Debug any issues with rendering or callbacks.

**Troubleshooting:**

The following problems were encountered and overcome in the initial development of the project. Included are the methods used to navigate each issue.

**1. Empty or Missing DataFrames**

* **Challenge:** Initially, the DataFrame was empty or missing expected columns like 'breed', causing KeyError or logic errors in callbacks.
* **Solution:** Debugging was added to check the output of database queries and the structure of the DataFrame. Safeguards like default empty DataFrames and conditional checks were implemented to handle missing data gracefully.

**2. Complex Query Requirements**

* **Challenge**: Radio menu filters required precise MongoDB queries to account for multiple conditions, including breed, age, and gender.
* **Solution**: Queries were refined using $in, $gte, and $lte operators to handle compound filtering criteria. Debugging ensured the queries returned the correct results.

**3. Excessive Debug Output in Jupyter Notebook**

* **Challenge**: Printing large data caused the notebook to exceed the IOPub data rate, leading to interruptions.
* **Solution**: Debug output was reduced to summaries (e.g., head() and info()) or written to external log files for analysis.

**Contact**

Riley Johnson