**Grazioso Salvare Dashboard README**

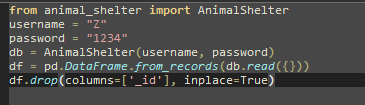
**Project Overview**

The Grazioso Salvare Dashboard is a web-based application designed to assist Grazioso Salvare in identifying and categorizing potential search-and-rescue dogs from regional animal shelter data. This dashboard provides an interactive interface to filter, search, and visualize shelter animal data, streamlining the process of selecting dogs for rescue training programs.

**Functionality**

The dashboard features the following capabilities:

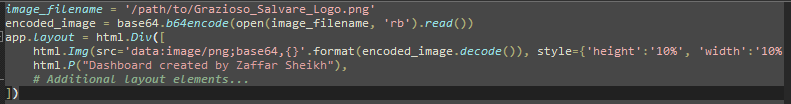
This section of the code connects to MongoDB using a CRUD Python module, fetches data, and prepares it for display in the web application.



* **Import and Instantiate CRUD Module:** Utilizes the **“AnimalShelter”** class from the **“animal\_shelter”** module to handle database operations. This class is instantiated with a username and password to ensure secure connections to MongoDB.
* **Data Fetching:** Employs the **“read({})”** method to retrieve all records from the database, converting them into a pandas DataFrame for easier manipulation and integration with Dash components.
* **ID Column Removal:** Removes MongoDB's **“\_id”** field from the DataFrame since it is incompatible with Dash DataTables and unnecessary for the dashboard's functionality.

**Dashboard Layout and View**

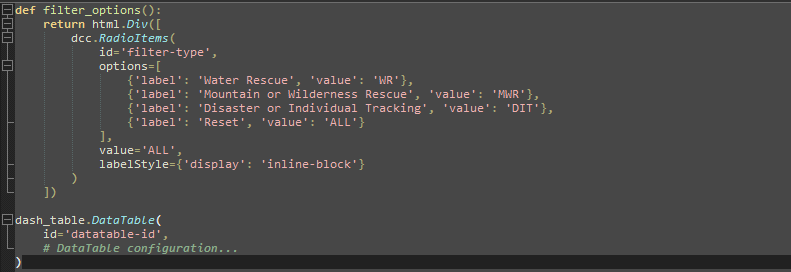
The application's layout is defined here, incorporating visual elements like the company logo and constructing the dashboard's interactive elements.



* **Image Encoding and Display**: The company logo is loaded, encoded in base64 format, and displayed at the top of the dashboard, enhancing brand visibility.
* **Layout Configuration**: The layout includes a title, image, and a personalized identifier, ensuring that the dashboard is not only functional but also visually appealing and branded.

**Interactive Components and Data Table**

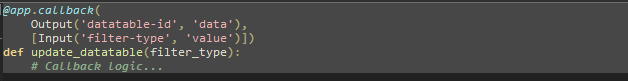
This part sets up interactive filtering options and configures the data table for display.



* **Filter Options**: Radio buttons allow the user to filter data based on specific rescue types, making it easy to find relevant records quickly.
* **Data Table Configuration**: The Dash DataTable is set up to dynamically display data and respond to user interactions such as sorting and filtering, enhancing the interactivity of the dashboard.

**Callback Functions**

Callbacks in Dash connect UI elements to backend logic, allowing for reactive updates based on user interaction.



* **Dynamic Data Updates**: This callback updates the data table based on the selected filter type. It queries the database for new data and updates the table without reloading the page.

**Tools and Technologies**

The following tools and technologies were used to achieve the functionality of this project:

* **MongoDB:** Served as the database to store and manage the animal shelter data. It was chosen for its flexibility with document schemas, scalability, and strong support for Python integration.
* **Python:** The core programming language used for backend development.
* **Dash:** A Python framework for building reactive web applications. Dash was used to create the frontend user interface due to its simplicity and ability to generate HTML components through Python code.
* **Pandas:** Utilized for data manipulation and analysis to convert MongoDB data into a format suitable for the Dash DataTable.
* **Plotly:** Enabled the creation of interactive charts that respond to data table events.

*Relevant links and resources:*

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

Dash Framework: <https://dash.plotly.com/>

Pandas: <https://pandas.pydata.org/>

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

**Steps to Reproduce**

1. Clone the repository from GitHub.
2. Install required Python packages: Dash, Pandas, Plotly, and dash-leaflet (for map visualization).
3. Ensure MongoDB is installed and running and that the ‘*accuser’* along with *‘Z’* accounts are set up with access to the database.
4. Run the Jupyter Notebook containing the Dash app code.

**Challenges and Solutions**

Throughout the development process, several challenges were encountered:

During the development of this dashboard, one of the key challenges was ensuring compatibility between MongoDB data types and Dash DataTables. By converting MongoDB records directly into a pandas DataFrame and handling data type discrepancies (e.g., removing the **\_id** field), we were able to integrate MongoDB seamlessly with Dash.

* **Data Formatting:** Addressed compatibility issues between MongoDB data types and Dash DataTables by converting MongoDB records directly into a pandas DataFrame and modifying incompatible columns.
* **Case-Insensitive Searching:** Enhanced search functionality within the data table to be case-insensitive by applying custom style conditions in the DataTable component.
* **Toggle Columns:** Initially challenging, this feature was implemented successfully by setting the **“hideable”** attribute to **“True”** for all columns in the DataTable setup.

**Conclusion**

This README provides a comprehensive guide to the Grazioso Salvare Dashboard, detailing each component's functionality and the rationale behind technological choices. By documenting the project thoroughly, we ensure that future developers and users can easily understand and maintain the system.

**Here is what you should see when manipulating the database/dataset:**

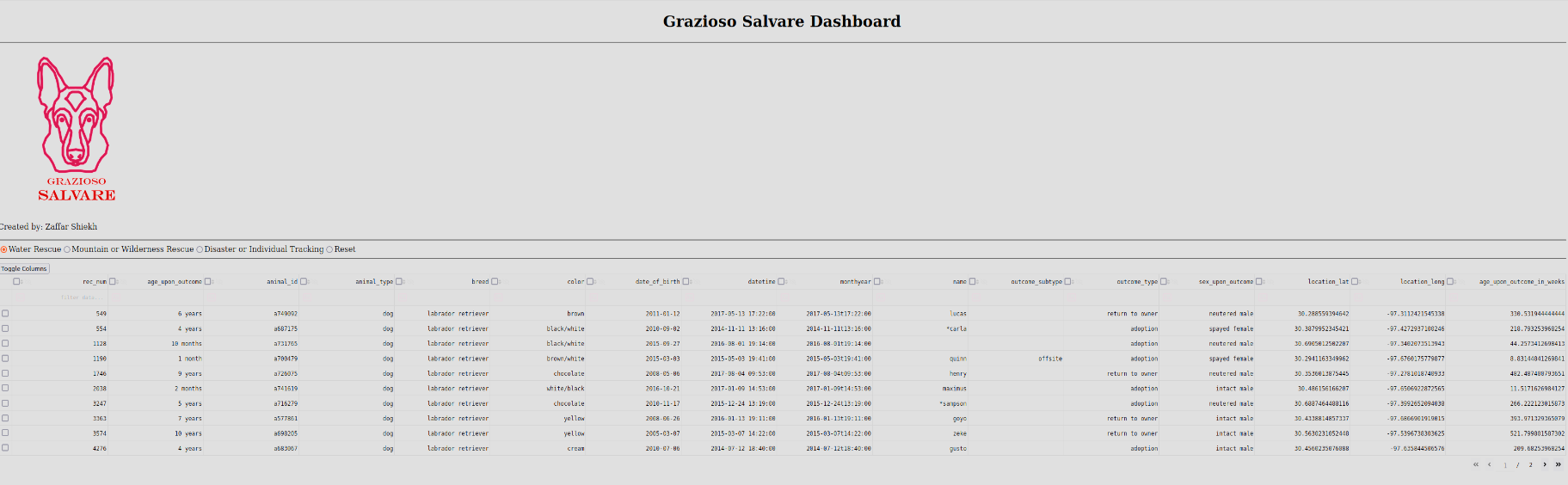
1. When booting up the database



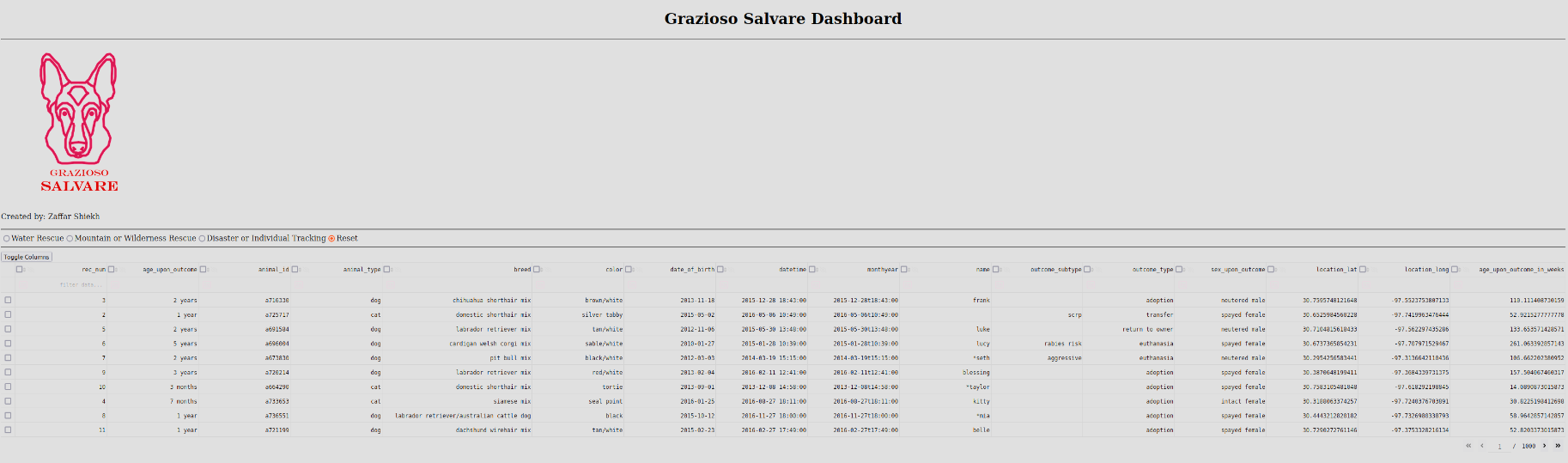
1. Choices such as Mountain, Water or Disaster rescues







1. Resetting the database



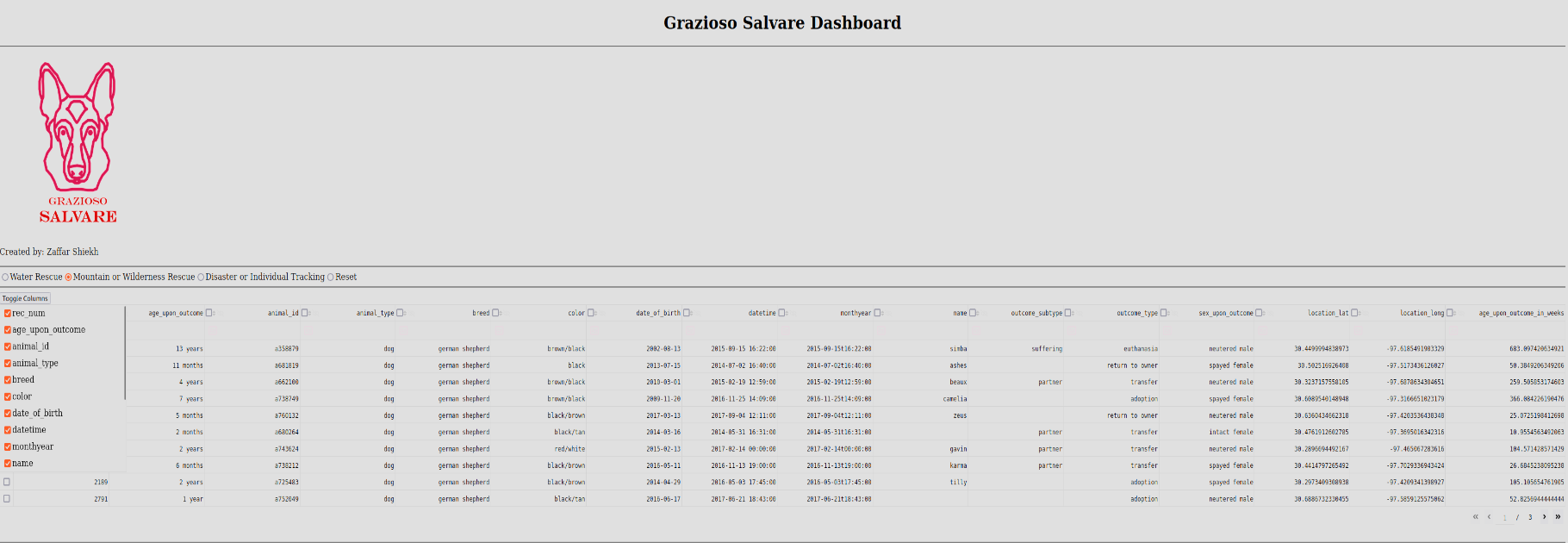
1. Filters like Mountain or Water rescue with a custom filter for in dataset. In this case a dog with all ages that includes 1.



1. Same search option after it has been reset



1. Toggle Menu on the left side. These are to manipulate what portion of the datasets are active for viewing.



1. Same menu with a few of the columns removed.

