**CS-340 Dashboard: Grazioso Salvare Animal Filtering and Visualization**

**1. Introduction**

This project is a MongoDB-powered interactive web dashboard developed for Grazioso Salvare. The dashboard allows users to filter, visualize, and interact with data about animals from various shelters. This enables Grazioso Salvare to identify suitable dogs for training in rescue operations such as **Water Rescue**, **Mountain or Wilderness Rescue**, and **Disaster or Individual Tracking**.

The dashboard includes:

1. Interactive filtering options
2. A dynamic data table that updates based on selected filters
3. A geolocation map and additional visualization that dynamically respond to the table’s data.

**2. Functionality Description**

**Overview of Functionality**

The dashboard includes the following features:

1. **Filtering Options**:
   * Four radio buttons allow users to filter animals by their rescue type:
     + Water Rescue
     + Mountain or Wilderness Rescue
     + Disaster or Individual Tracking
     + Show All (reset state)
2. **Dynamic Data Table**:
   * The data table displays filtered animal records retrieved from the MongoDB database.
   * It includes information such as animal type, breed, age, outcome, location, etc.
   * Pagination allows users to navigate through large data sets.
3. **Geolocation Map**:
   * When a row is selected from the table, the map dynamically updates to reflect the animal's geolocation (latitude and longitude).
4. **Second Visualization** (Optional):
   * This chart can display additional summaries or metrics related to the filtered data set (e.g., average age of animals, breed distribution).

**Screenshots**

The following screenshots demonstrate the dashboard's functionality:

1. **Unfiltered View**: Dashboard displays all animals at startup.
2. **Water Rescue Filter**: Animals suitable for water rescue are displayed.
3. **Mountain or Wilderness Rescue Filter**: Animals suitable for wilderness rescues are shown.
4. **Disaster or Individual Tracking Filter**: Animals suitable for disaster-related rescues are displayed.
5. **Reset State**: All filters are cleared, and the data table returns to its original unfiltered view.

**3. Tools and Rationale**

**MongoDB**

* MongoDB serves as the **model** component in the MVC architecture. Its document-based, schema-less structure makes it ideal for storing and retrieving unstructured animal data.
* The **CRUD Python module** was implemented to perform Create, Read, Update, and Delete operations, enabling seamless database interactions.

**Dash Framework**

* Dash is a Python-based framework for building interactive web applications and dashboards.
* It serves as the **controller** and **view** components, allowing for user interactivity and dynamic updates of the dashboard elements.

**Python**

* Python's ecosystem offers extensive libraries, including:
  + **pymongo**: Facilitates MongoDB interactions.
  + **dash**: For creating the interactive dashboard.
  + **pandas**: For data processing and manipulation.
  + **dash-leaflet**: For geolocation mapping capabilities.

**4. Reproduction Instructions**

Follow these steps to reproduce the project on your local environment:

**Step 1: Install Required Tools**

Ensure Python and the required libraries are installed:

bash

Copy code

pip install dash jupyter\_dash dash-leaflet pymongo pandas

**Step 2: MongoDB Setup**

1. Install MongoDB and start the server.
2. Create a database named AAC and a collection named animals.
3. Import the provided **Austin Animal Center Outcomes** CSV dataset into the animals collection.
   * Use MongoDB Compass or the following command-line tool:

bash

Copy code

mongoimport --db AAC --collection animals --type csv --headerline --file austin\_animal\_outcomes.csv

**Step 3: Prepare the Code**

1. Place the following files in the same directory:
   * app.py (dashboard code)
   * crud.py (CRUD operations for MongoDB)
   * logo.png (Grazioso Salvare logo)
2. Ensure the MongoDB connection credentials (username/password) in crud.py are correctly set:

python

Copy code

username = "aacuser"

password = "finalpassword123"

**Step 4: Run the Dashboard**

Execute the following command in your terminal:

bash

Copy code

python app.py

The server will start, and the dashboard can be accessed at:

arduino

Copy code

http://127.0.0.1:8050

**5. Challenges and Solutions**

1. **MongoDB Authentication Errors**:
   * Challenge: The MongoDB authentication failed multiple times while connecting to the aacuser account.
   * Solution: Reset the user credentials and updated the connection URI in the Python code.
2. **Map Display Issues**:
   * Challenge: The geolocation map did not display correctly due to missing coordinates.
   * Solution: Added error handling to provide default latitude/longitude values if data is missing.
3. **Dynamic Updates**:
   * Challenge: Ensuring that the map and table dynamically update based on selected filters.
   * Solution: Implemented Dash callbacks to tie the radio buttons, data table, and map components together.

**6. Resources and References**

* **Dash Framework Documentation**: https://dash.plotly.com/
* **MongoDB Documentation**: <https://www.mongodb.com/docs/>
* **Python**: <https://www.python.org/>
* **Austin Animal Center Outcomes Dataset**:
  + Source: <https://doi.org/10.26000/025.000001>
* **Grazioso Salvare Logo**: Provided in project materials.

**7. Screenshots**

The following screenshots demonstrate the dashboard's functionality:

1. **Unfiltered State** (Default View)
2. **Water Rescue Filter**
3. **Mountain or Wilderness Rescue Filter**
4. **Disaster or Individual Tracking Filter**
5. **Reset State**

A screenshot of a computer

Description automatically generated

A screenshot of a map

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a map

Description automatically generated

A screenshot of a map

Description automatically generated

A screenshot of a map

Description automatically generated