**CS 340 PROJECT TWO README**

**About the Project / Project Title**

This project implements a Python based dashboard application using the Dash framework, givng interactive data visualization for Grazioso Salvare’s animal rescue operations. The dashboard connects to a MongoDB database containing AAC data and allows filtering by rescue type, outcome type, and age. Interactive charts and a map display provide insights into rescue trends, while an interactive data table shows detailed animal information.

**Motivation**

This module was developed as part of the CS 340 course to demonstrate how to build an interactive, data-driven dashboard using Python, Dash, and MongoDB. The focus was on integrating a backend database with a front end interface for real time filtering and visualization.

**Getting Started**

1. Make sure MongoDB is running in your **Apporto** environment and that the aacuser account is set up with access to the AAC database.
2. Ensure the dataset is imported into the MongoDB animals collection.
3. Save the dashboard .ipynb file in your working directory.
4. Launch the Jupyter Notebook and run all cells to start the dashboard server.
5. Open the provided local URL in your browser to view the dashboard.

**Installation**

* **Python 3.9** (Apporto default)
* **MongoDB** (provided in Apporto)
* **PyMongo**: For interacting with MongoDB from Python
* **Dash**: For building the interactive web-based dashboard
* **Pandas**: For data manipulation
* **Jupyter Notebook**: For running and testing the dashboard interactively

**Usage**

The dashboard allows users to:

* Filter by Rescue Type (Water Rescue, Mountain/Wilderness Rescue, Disaster/Individual Tracking, Reset)
* Filter by Outcome Type
* Filter by Age (weeks)
* View interactive charts for breed distribution
* View map with rescue locations
* Explore detailed animal records in a sortable, filterable data table

**Code Example** (filter callback structure)

@app.callback(

Output('datatable-id', 'data'),

[Input('rescue-type-dropdown', 'value'),

Input('outcome-type-dropdown', 'value'),

Input('age-slider', 'value')]

)

def update\_dashboard(rescue\_type, outcome\_type, age\_range):

query = {}

# Build MongoDB query here based on filters

results = list(animal\_shelter.read(query))

return results

**Tests**

The dashboard was tested by applying each filter option and verifying the table, charts, and map updated correctly:

1. Starting state – All data displayed with no filters applied.
2. Water Rescue filter applied – Only water rescues displayed in table, chart, and map.
3. Mountain/Wilderness Rescue filter applied – Only mountain rescues displayed.
4. Disaster/Individual Tracking filter applied – Only disaster rescues displayed.
5. Reset – All filters cleared and full dataset restored.

**A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a map

AI-generated content may be incorrect.Screenshots**

**A screenshot of a computer

AI-generated content may be incorrect.**

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

* Screenshot 1-2: Dashboard starting state (all widgets, table, chart, and map visible)
* Screenshot 3: Water Rescue filter applied
* Screenshot 4: Mountain/Wilderness Rescue filter applied
* Screenshot 5: Disaster/Individual Tracking filter applied
* Screenshot 6: Reset applied (returns to starting state)

**Driver Used**

This dashboard uses the official MongoDB Python driver, PyMongo, for database interaction.

**Reasons for choice:**

* Well-documented and officially supported
* Allows direct access to MongoDB methods using Python syntax
* Works seamlessly with Pandas for data manipulation
* Supports full CRUD operations

**Framework Used**

The dashboard uses the Dash framework by Plotly for the web-based interface.

**Why Dash was chosen:**

* Allows creation of interactive, data-driven web apps using Python only
* Integrates easily with Pandas and Plotly for charts and data handling
* Supports callbacks for real-time UI updates without manual page reloads

**Steps Taken to Complete the Project**

1. Connected to MongoDB using PyMongo.
2. Queried AAC database for animal rescue data.
3. Preprocessed data using Pandas for visualization.
4. Designed dashboard layout in Dash with dropdowns, sliders, tables, charts, and a map.
5. Implemented callbacks for interactive filtering.
6. Tested all filters and reset functionality.
7. Captured screenshots for README documentation.

**Challenges and Solutions**

* **Challenge:** Ensuring the data table, chart, and map updated simultaneously.  
  **Solution:** Implemented a single callback function to handle all updates at once.
* **Challenge:** Filtering correctly with multiple criteria.  
  **Solution:** Built a dynamic MongoDB query dictionary that only included filters with selected values.

**Contact**  
Joshua Koshy