**About the Project/Project Title**

This project was created for *Grazioso Salvare*, a company that identifies and trains dogs for rescue work. The main objective was to build a Python-based dashboard that connects to a MongoDB database and allows users to interactively explore data from the Austin Animal Center. The dashboard displays information on adoptable dogs, lets users filter animals by specific rescue categories, and shows real-time updates through dynamic charts and maps.

**Getting Started**

Using Python, Dash, and MongoDB, this project combines both back-end and front-end development. It uses the CRUD module from Project One to read data from the database, and Dash components to visualize that data. The final dashboard allows users to filter dogs based on their suitability for Water Rescue, Mountain/Wilderness Rescue, or Disaster/Individual Tracking. It also provides an interactive pie chart to display breed distribution and a map to visualize each dog’s location. This demonstrates how data from a live database can be transformed into useful insights through visualization.

**Installation**

To run the project, the MongoDB server must be active with the Austin Animal Center dataset loaded into the aac database under the animals collection. The aacuser account with password Password123 is used to connect. The Python files needed are ProjectTwoDashboard.ipynb, CRUD\_Python\_Module.py, and an image file named Grazioso Salvare Logo.png. After installing the required packages, Dash, Dash Leaflet, Plotly, PyMongo, Pandas, and JupyterDash, the notebook can be opened and executed. The app runs directly inside Jupyter Notebook in inline mode.

The main tools used in this project include Python for programming, MongoDB for data storage, PyMongo for database connectivity, Pandas for data handling, Dash and JupyterDash for building the interactive interface, Dash Leaflet for mapping, and Plotly Express for the pie chart visualization. Together, these tools make it possible to create a responsive dashboard that connects the database to a graphical user interface.

**Usage**

The dashboard allows users to filter animal records through simple radio buttons. Choosing a category such as “Water Rescue,” “Mountain/Wilderness,” or “Disaster/Individual Tracking” updates the entire dashboard and shows only the dogs that meet the criteria for that type of rescue. A pie chart updates to display the breed distribution for that selection, and clicking a row in the table highlights the corresponding dog’s location on the map. Selecting “Reset” reloads the full dataset. This makes the dashboard easy to use and provides clear visual insight into the data.

**Challenges and Issues**

During development, a few issues were encountered and resolved. The most common error, KeyError: '\_id', occurred when the \_id field was missing from the dataset. This was fixed by adding the parameter errors='ignore' when dropping the column. Indentation and syntax errors in the layout were corrected by keeping all components inside a single html.Div() container.

**Testing**

Each feature of the dashboard was tested to ensure it worked correctly. All four filter categories update the data table, pie chart, and map as expected. Selecting a row on the table highlights that dog’s location in Austin on the map. Resetting the view reloads all animals from the database. The application meets all the functionality and usability goals defined in the project requirements.

**Motivation**

Through this project, I learned how to connect a back-end database to a front-end interface and how to manage data dynamically in real time. I also gained experience debugging errors, managing files and paths, and creating interactive data visualizations. Overall, this project gave me a strong understanding of how full-stack development works in Python and how data-driven applications can be designed to solve real-world problems.

Contact: Charity Deel

A screenshot of a computer

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

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

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

AI-generated content may be incorrect.