***Author***

***Dennis Selfinger***

* ***Role: Lead Developer***
* ***Company: Global Rain***
* ***Project: Grazioso Salvare Dashboard for Rescue-Animal Training***
* ***Contact: dself1976@gmail.com***
* ***Contributions: Designed and developed the client-server interface and dashboard using MongoDB for backend data management and Python for CRUD operations. Spearheaded the project’s coding and testing phases to ensure functionality according to client specifications.***

**Grazioso Salvare Rescue Dog Dashboard**

**Project Overview**

This interactive dashboard application was developed for Grazioso Salvare, an international rescue-animal training company. The application interfaces with a MongoDB database containing animal shelter data from five shelters around Austin, Texas, allowing users to efficiently identify and categorize dogs suitable for different types of rescue training.

Purpose

Grazioso Salvare trains dogs for specialized rescue operations:

* Water Rescue
* Mountain or Wilderness Rescue
* Disaster or Individual Tracking

This dashboard enables rapid filtering of shelter data based on specific breed, age, and sex requirements tailored to each rescue training category.

Technologies Used

* Python 3.x
* MongoDB
* Jupyter Dash
* Dash Leaflet for geolocation visualization
* Plotly Express for charts
* Pandas for data manipulation

Setup and Installation

Prerequisites

* Python 3.x installed
* MongoDB installed and running
* Access to the Austin Animal Center Outcomes dataset

Installation Instructions

1. **Install required Python packages:**

pip install jupyter-dash dash dash-leaflet pandas plotly

1. Clone the repository:

git clone [https://github.com/ DennisjSelfingger/grazioso-salvare-dashboard.git](https://github.com/%20DennisjSelfingger/grazioso-salvare-dashboard.git)

cd grazioso-salvare-dashboard

1. **Ensure these files are in your working directory:** 
   * ProjectTwoDashboard.ipynb
   * animal\_shelter.py (CRUD Python module)
   * grazzios.png (Grazioso Salvare logo)

Database Setup

1. **Import the dataset into MongoDB:**

***mongoimport --db AAC --collection animals --file aac\_shelter\_outcomes.csv --type csv --headerline***

1. **Create a MongoDB user with read access:**

*use AAC*

*db.createUser({*

*user: "create credential",*

*pwd: "create unique password ",*

*roles: [{ role: "read", db: "AAC" }]*

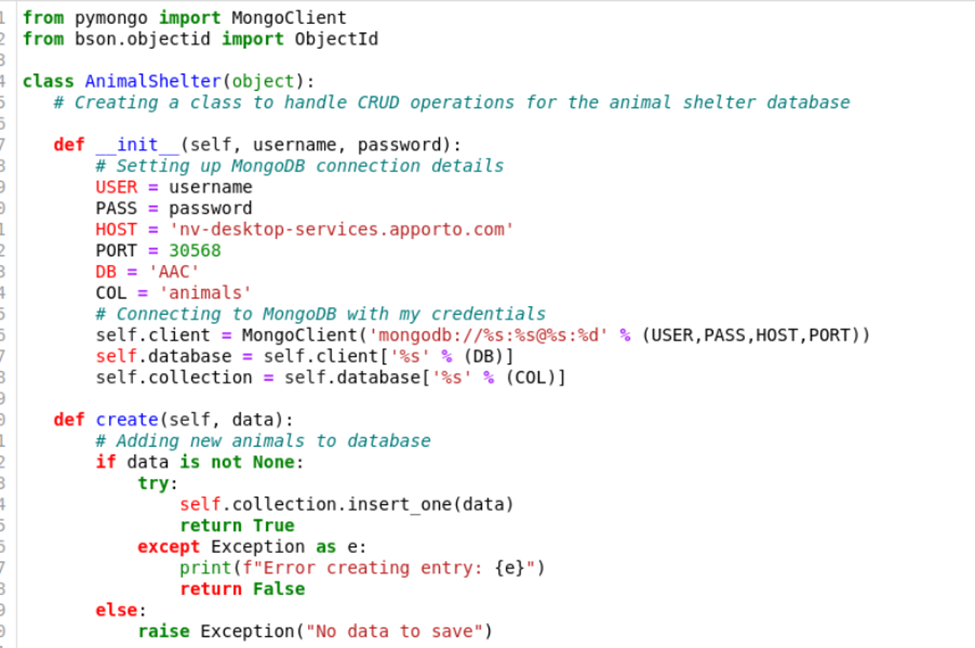
*})*

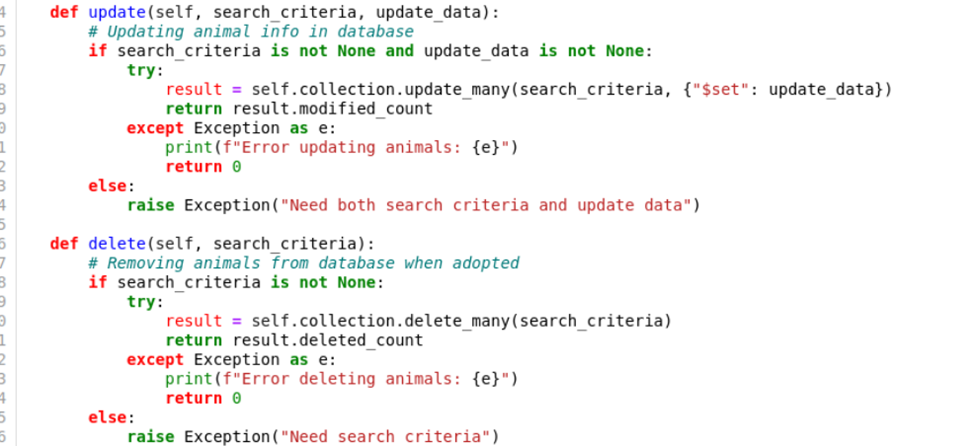
**Running the Application**

1. Launch Jupyter Notebook:
2. Open ProjectTwoDashboard.ipynb and run all cells
3. Access the dashboard at <http://127.0.0.1:8050/> in your browser, (this should be your local address, you can change this if need be)

**IMPLEMENTATION**

**CRUD:**

****

****

**TEST CRUD FILE**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Dashboard Features**

**Interactive Filtering**

The dashboard provides radio button options to filter dogs based on their suitability for different rescue types:

1. **Water Rescue**:
   * Breed: Labrador Retriever Mix, Chesapeake Bay Retriever, Newfoundland
   * Sex: Intact Female
   * Age: 26 to 156 weeks
2. **Mountain or Wilderness Rescue**:
   * Breed: German Shepherd, Alaskan Malamute, Old English Sheepdog, Siberian Husky, Rottweiler
   * Sex: Intact Male
   * Age: 26 to 156 weeks
3. **Disaster or Individual Tracking**:
   * Breed: Doberman Pinscher, German Shepherd, Golden Retriever, Bloodhound, Rottweiler
   * Sex: Intact Male
   * Age: 20 to 300 weeks
4. **Reset**: Displays complete dataset

**Data Table**

The interactive data table displays animal records based on the selected filter:

* Pagination controls for navigating through multiple pages of results
* Sortable columns for easy data organization
* Selection capability for viewing detailed information about specific animals

**Visualizations**

1. **Breed Distribution Pie Chart**:
   * Dynamically updates to show the distribution of dog breeds in the filtered results
   * Provides visual insight into the proportion of each breed available for training
2. **Geolocation Map**:
   * Shows the location where the selected animal was found
   * Displays a tooltip with the breed information when hovering over the marker
   * Provides a popup with the animal's name when clicking on the marker

**Evidence of Functionality**

**Screenshots of GUI**

**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 dashboard

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.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Troubleshooting and Common Issues**

**In the github for this project look for the full code from animal\_shelter.py**

**A close up of a text

AI-generated content may be incorrect.**

**This wil help set up project….I suggest to do this for:**

1. **Modularity**

* **It separates the database operations code from your dashboard code**
* **Makes the code more organized and easier to maintain**
* **Allows you to import and reuse these functions in different parts of your project**

1. **Code Organization**

* **Since this is a class with CRUD operations, having it in a separate file follows Python's best practices and this is very important working in and with teams.**
* **Makes it easier to update database operations without touching the dashboard code**
* **Keeps your Jupyter notebook cleaner and focused on the dashboard implementation**

1. **Importing**

* **This code needs to be in a .py file so you can import it into your Jupyter notebook using:**

**python**

***from animal\_shelter import AnimalShelter***

1. **Best Practices**

* **Keeping database connection details in a separate file is good for security**
* **Makes it easier to change database settings without modifying the main application**
* **Follows the principle of separation of concerns in software development**

**It's like having a specialized tool in a separate drawer - you can grab it whenever you need it, but it doesn't clutter up your main workspace. This is really helpful and for those new to or learning programming(like myself) this will help a lot.**

**Common Errors**

* **Authentication Failures**: Ensure that user credentials are correctly set in your MongoDB instance.
* **Document Not Found**: Check the accuracy of your query parameters if no results are returned.

**Handling Errors**

It is recommended to include exception handling in your script to manage operational exceptions:

try:

*shelter.update({'name': 'Buddy'}, {'$set': {'age\_years': 3}})*

*except Exception as e:*

*print(f"An error occurred: {e}")*

Future Enhancements

Future updates could include more sophisticated query capabilities, integration with other types of databases, or enhanced security features for data access. Also the possibility of integrating with other shelters across the country and possibly international if at all ever needed or wanted by the clients.

**Technology Choices**

MongoDB

- Chosen for its flexible document model that accommodates varying animal data

- Natural integration with Python through PyMongo

- Efficient querying capabilities for complex rescue dog criteria

- Scalability for growing shelter data

Dash Framework

- Provides integrated MVC architecture for web applications

- Enables reactive updates between components without page refreshes

- Seamless integration with Plotly for interactive visualizations

- Built on Flask, offering production-ready performance



1. **File Management and Integration**
   * Challenge: Ensuring proper file organization and accessibility between different components
   * Solution: Improved development workflow by:
     + Learning proper file management in Jupyter Notebook environment
     + Understanding the importance of consistent file paths and locations
     + Implementing proper import structures for module integration
     + Developing better organizational practices for project files

A significant challenge in this project was becoming as proficient as I could in the little amount of time I had with the Dash framework. While having previous experience with Matplotlib and Pandas provided some helpful background for handling data visualization, there was still a noticeable learning curve. The most interesting adjustment was adapting to using Python for creating a graphical user interface, which was quite different from the traditional approach of using HTML, JavaScript, and CSS. Though this transition required some time to get used to, I can see how this could someday become more popular than traditional css, sass, html and such., it ultimately proved to be a valuable learning experience. The process of understanding how to implement web-like functionality through Python instead of conventional web development tools was challenging at first, but became more intuitive as the project progressed. This experience demonstrated how Python could effectively handle both the data processing and user interface aspects of the application, providing a more streamlined development approach.