# CS 340 README for Grazioso Salvare’s Custom Dashboard

**Project Overview**

The Grazioso Salvare dashboard project aims to provide a comprehensive interface for managing animal shelter data. It includes features for viewing, filtering, and editing animal records stored in a MongoDB database. Below are the functionalities implemented:

* Dashboard Interface: Interactive dashboard displaying animal shelter data, including a data table, pie chart visualization, and geolocation mapping.
* Data Filtering: Radio buttons to filter data based on rescue types: Water Rescue, Mountain or Wilderness Rescue, and Disaster Rescue.
* CRUD Operations: Implemented Create, Read, Update, and Delete operations for managing animal records.

**Tools Used**

The following tools were utilized to achieve the functionality of the Grazioso Salvare dashboard:

* **Dash**: A Python framework for building web applications, providing the view and controller structure for the web application.
* **Jupyter** **Dash**: An extension that enables running Dash applications within Jupyter notebooks.
* **Dash** **Leaflet**: A Dash component library for creating interactive maps.
* **Plotly** **Express**: Used for generating pie chart visualizations.
* **Pandas**: Utilized for data manipulation and handling.
* **Matplotlib**: Used for additional data visualization, though not directly involved in the dashboard interface.

**MongoDB Usage**

MongoDB was selected as the model component of the development due to its specific qualities and capabilities for interfacing with Python:

* **Document-Oriented**: MongoDB's document-based storage model aligns well with Python's dictionary data structure, making it intuitive for data manipulation.
* **Flexible** **Schema**: Allows for dynamic schema changes, facilitating easy updates and modifications to the data model.
* **Rich** **Query** **Language**: MongoDB's query language provides powerful features for filtering and querying data, which is essential for the dashboard's functionality.

**Dash Framework**

Dash framework was chosen for building the Grazioso Salvare dashboard due to its simplicity and flexibility:

* **Pythonic** **Syntax**: Dash enables building web applications using pure Python, making development straightforward and accessible.
* **Component-Based Structure**: Dash follows a component-based architecture, allowing for modular development and easy customization of individual components.
* **Integration with Plotly**: Dash seamlessly integrates with Plotly to create interactive visualizations and enhance the dashboard's data presentation capabilities.

**Project Completion Steps**

The project was completed following these steps:

1. **Data** **Retrieval**: Animal data was retrieved from MongoDB using the provided CRUD Python module.
2. **Dashboard** **Setup**: Jupyter Dash was utilized to set up the Dash application within a Jupyter notebook.
3. **Data** **Visualization**: Plotly Express and Dash Leaflet were employed to create interactive charts and maps.
4. **Interaction** **Handling**: Callback functions, such as filtering data and updating visualizations, were implemented to handle user interactions.
5. **Testing** **and** **Deployment**: The dashboard was tested for functionality and deployed within the Jupyter environment.

A screenshot of a computer

Description automatically generated

**If you select one of the common search filters, the pie chart and datatable update to show for that selection.**

**Water Rescue:**

A screenshot of a computer screen

Description automatically generated

**Mountain or Wilderness Rescue:**

A screenshot of a computer

Description automatically generated

**Disaster Rescue or Individual Tracking:A screenshot of a computer

Description automatically generated**

**If you select an animal using the ‘dots’ on the left hand side of the datatable, you will see the map update to its current geolocation.**

**A screenshot of a computer screen

Description automatically generated**

**A screenshot of a computer screen

Description automatically generated**

**Project Creation**

## Motivation

*This project was crafted to assess my proficiency in database management and data manipulation techniques.*

## Getting Started

1. *Begin by accessing Mongo and importing the CSV file named aac\_shelter\_outcome.csv*
2. *Subsequently, it’s essential to establish both a simple and a complex index to navigate through the stored data effectively.*
3. *To authorize users, creating both an Admin account and an aacuser account is recommended for securely accessing the database.*
4. *Finally, ensure Python is accessible or installed, and execute the program from a notebook environment.*

## Installation

* *An up-to-date version of Python is required to execute both the .py and .ipynb files.*
* *MongoDB – to access the database*

## Usage

*Use this space to show useful examples of how your project works and how it can be used. Be sure to include examples of your code, tests, and screenshots.*

### Code Example

*This Python method create is designed to handle the "C" (Create) operation in CRUD (Create, Read, Update, Delete). It inserts the provided data, which should be in dictionary format, into the MongoDB collection named animals. If the data parameter is empty or None, it raises an exception indicating that there's nothing to save.*

*A computer screen shot of a computer code

Description automatically generated with medium confidence*

*This Python method read implements the "R" (Read) operation in CRUD. It retrieves data from the MongoDB collection named animals based on the provided searchData query, excluding the \_id field from the result. If no query is provided, it returns all documents in the collection while still excluding the \_id field.*

*A close-up of a computer code

Description automatically generated*

*This Python method update implements the "U" (Update) operation in CRUD. It updates documents in the MongoDB collection named animals based on the provided searchData query with the values specified in updateData. If updateData is provided, it executes the update operation and returns the raw result. If updateData is None, it returns an empty dictionary.*

**A close-up of a computer code

Description automatically generated**

*This Python method delete implements the "D" (Delete) operation in CRUD. It deletes documents from the MongoDB collection named animals based on the provided deleteData query. If deleteData is provided, it executes the delete operation and returns the raw result. If deleteData is None, it returns an empty dictionary.*

*A close-up of a computer code

Description automatically generated*

### Tests

*I evaluated this code by employing an invalid statement: print(animals.create(0:0)), which constitutes an invalid argument, attempting to create an unsupported data type.*

A screenshot of a computer

Description automatically generated

*To confirm the addition of your animal, you can execute a search using query = animals.read({"name": "NAME"}).*

### Screenshots

*Import AAC Animal Database***A screen shot of a computer

Description automatically generated**

*User AuthenticationA computer screen shot of a program code

Description automatically generated*

*Create*

A screenshot of a computer

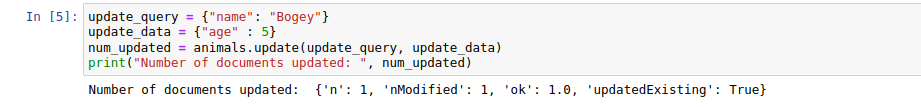
Description automatically generated

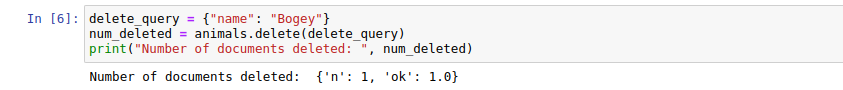
*Search for pet “Bogey”*

A screenshot of a message

Description automatically generated

*Update information for “Bogey”*

**

*Delete “Bogey”  
*

**Challenges Faced**

For individuals endeavoring to replicate the project, a significant challenge arose in filtering the data from the Austin Animal Shelter. Grazioso Salvare had specific criteria regarding dog breeds, yet the data in the database proved to be less than pristine. For instance, while Grazioso Salvare sought information on the Chesapeake Bay Retriever breed, it was recorded as "Chesa Bay Retr" in the raw dataset. Consequently, implementing pattern matching became necessary to generate an accurate list of animals. Regular Expressions were utilized to match the pattern. However, it's noteworthy that Pymongo doesn't inherently support regular expressions, and thus, a different formatting approach is needed to be compatible with the Pymongo API.

## Contact

Your name: Brad Follett