# CS 340 README Template

Juan Rodriguez

*Use this template to complete your README file. When completing the template, keep the headings as they are so that your document has a clear organization. Remove the italicized prompt text after you have completed each section for a polished final document.*

## About the Project/Project Title

This project is a Python module that implements Create, Read, Update, and Delete (CRUD) operations for an animal shelter database. The module connects to a MongoDB database to manage records of animals in the shelter.

## Motivation

The motivation behind this project is to provide a simple and efficient way to manage animal records in a shelter. The CRUD operations allow for easy insertion, retrieval, updating, and deletion of animal data, facilitating better management of the shelter's database.

## Getting Started

1. Clone the repository:
2. Install the required Python libraries
3. Ensure MongoDB is running and accessible

**Python Libraries**

1. pymongo: For interacting with MongoDB.
2. pandas: For data manipulation and analysis.
3. dash, dash-leaflet, plotly.express: For building the interactive web dashboard.

## Installation

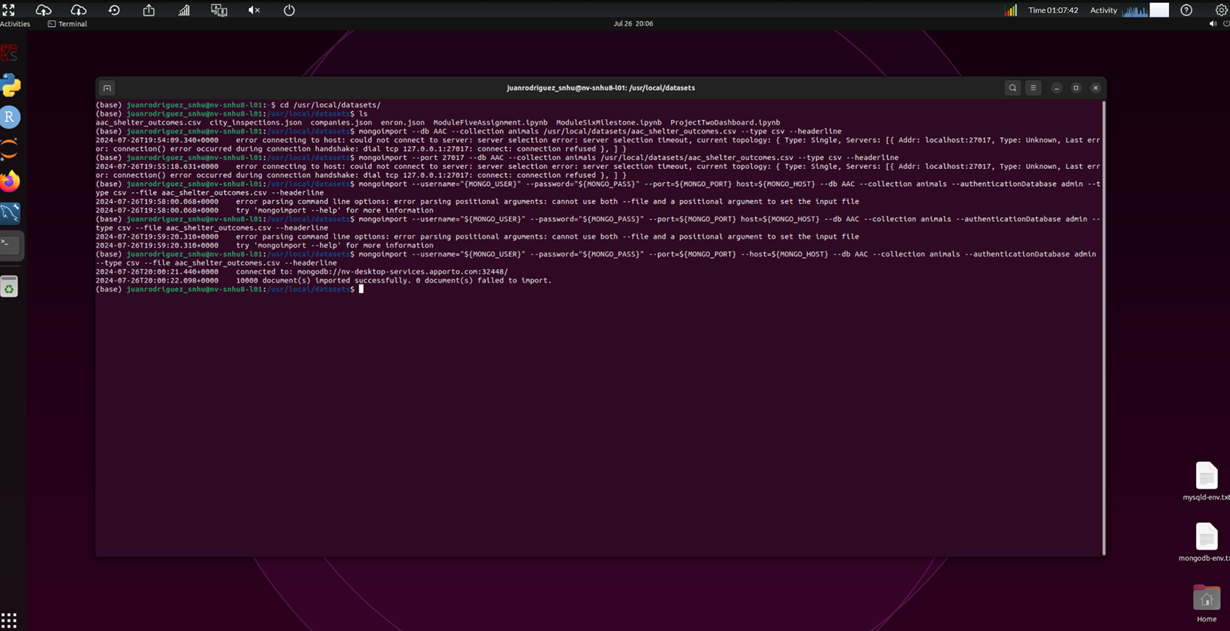
1. **Python 3.x**: Ensure you have Python installed. You can download it from [python.org](https://www.python.org/).
2. **pymongo**: This library is used to interact with MongoDB
3. **MongoDB**: You need a running instance of MongoDB. You can download it from [mongodb.com](https://www.mongodb.com/).

#### Database and User Authentication Setup

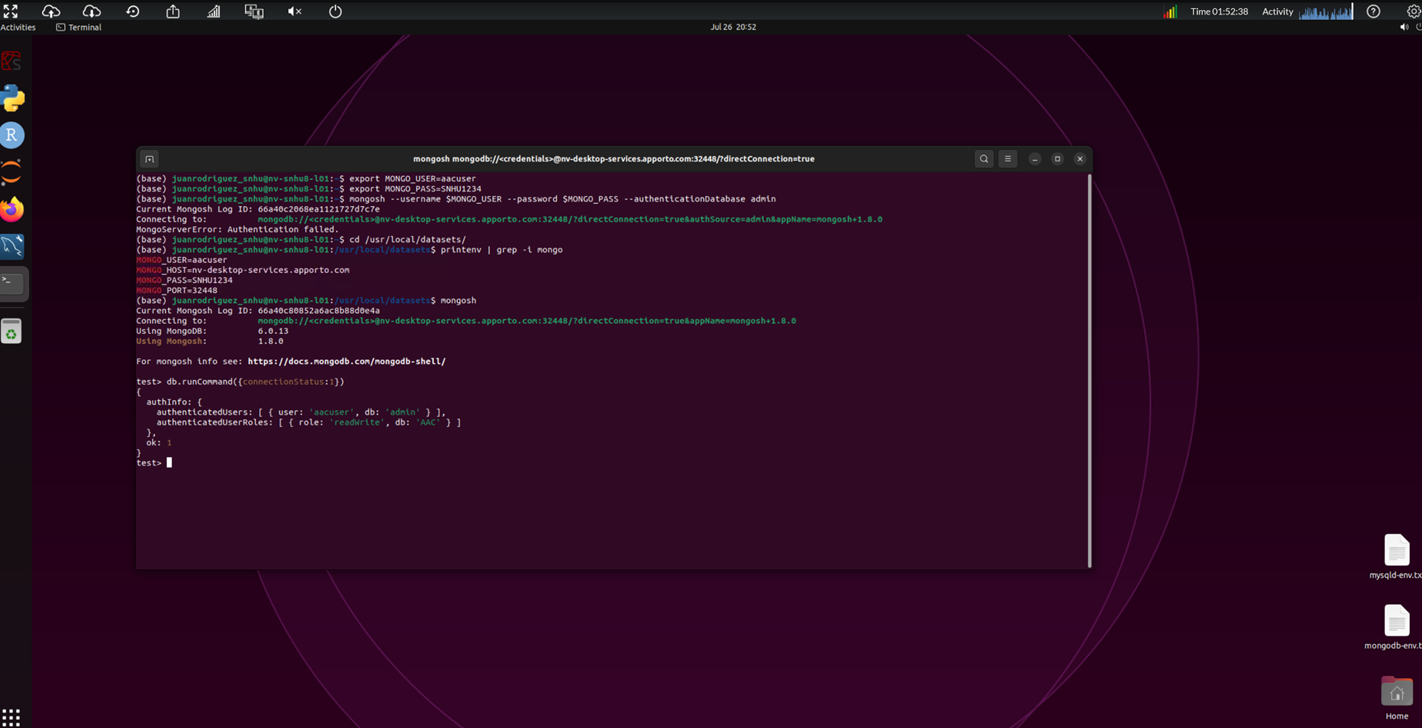
1. **Importing the Dataset**:
   1. Download the Austin Animal Center (AAC) Outcomes data set and place it in the /usr/local/datasets/ directory. The file name should be aac\_shelter\_outcomes.csv.
2. Start your MongoDB server.
3. Create a new database called AAC and a collection called animals.
4. Create a user with the necessary roles to access this database.
5. Import the CSV data into the animals collection.

**Screenshot**

**Import files**

**

## Create user account



## Usage

### Code Example

**Create Operation**: The create method in the AnimalShelter class is designed to insert a new document into the animals collection. It accepts a dictionary containing the animal's details and uses insert\_one to add the document to the collection.

from crud import AnimalShelter

**# Instantiate the AnimalShelter class**

shelter = AnimalShelter()

**# Define some test data for the create method**

test\_data = {

"age\_upon\_outcome": "5 years",

"animal\_id": "A123456",

"animal\_type": "Dog",

"breed": "Labrador Retriever",

"color": "Black",

"date\_of\_birth": "2016-01-01",

"datetime": "2021-06-15 12:00:00",

"monthyear": "2021-06-15T12:00:00",

"name": "Buddy",

"outcome\_subtype": "Neutered",

"outcome\_type": "Adoption",

"sex\_upon\_outcome": "Neutered Male",

"location\_lat": 30.6944,

"location\_long": -96.3344,

"age\_upon\_outcome\_in\_weeks": 260

}

**# Test the create method**

create\_result = shelter.create(test\_data)

print(f"Create result: {create\_result}")

**Read Operation:** The read method retrieves documents from the animals collection based on a search query. If no search query is provided, it retrieves all documents. The results are returned as a list of dictionaries.

**# Define search criteria for the read method**

search\_criteria = {"animal\_id": "A123456"}

**# Test the read method**

read\_result = shelter.read(search\_criteria)

print(f"Read result: {read\_result}")

### Tests

**# Instantiate the AnimalShelter class**

shelter = AnimalShelter()

**# Define test data**

test\_data = {

"age\_upon\_outcome": "2 years",

"animal\_id": "A654321",

"animal\_type": "Cat",

"breed": "Siamese",

"color": "Cream",

"date\_of\_birth": "2020-01-01",

"datetime": "2022-01-15 14:00:00",

"monthyear": "2022-01-15T14:00:00",

"name": "Whiskers",

"outcome\_subtype": "Adopted",

"outcome\_type": "Adoption",

"sex\_upon\_outcome": "Spayed Female",

"location\_lat": 34.0522,

"location\_long": -118.2437,

"age\_upon\_outcome\_in\_weeks": 104

}

**# Test the create method**

create\_result = shelter.create(test\_data)

print(f"Create result: {create\_result}")

**Update Operation:** The update method updates existing documents in the animals collection based on a search query and the update data provided. It returns the number of documents modified.

**# Define search criteria for the update method**

search\_criteria\_update = {"animal\_id": "A123456"}

**# Test update method**

update\_data = {"color": "Golden"}

**# Test the update method**

update\_result = shelter.update(search\_data, update\_data)

print(f"Update result: {update\_result}")

**Delete Operation:** The delete method removes documents from the animals collection based on a search query. It returns the number of documents deleted.

**# Define search criteria for the delete method**

search\_criteria\_delete = {"animal\_id": "A123456"}

**# Test the Delete method**

delete\_result = shelter.delete(search\_criteria\_delete)

print(f"Delete result: {delete\_result}")

### Screenshots

**Test script:**

**Create:**

A screenshot of a computer

Description automatically generated

**Read:**

A screenshot of a computer

Description automatically generated

**Update:**

**A screenshot of a computer

Description automatically generated**

**Delete:**

**A screenshot of a chat

Description automatically generated**

## Usage

**Interacting with the Dashboard**:

* **Filtering Data**: Use the radio buttons to filter the animal data based on rescue type.
* **Viewing Data**: The data table displays the filtered animal data.
* **Visualizations**: The dashboard includes a pie chart that shows the distribution of animal breeds and a map that displays the location of selected animals.

**Initial Dashboard View**: All data retrieved.

A screenshot of a computer

Description automatically generated

**Filtered Data**:

Water Rescue filter:

A screenshot of a computer

Description automatically generated

Mountain or Wilderness Rescue filter:

A screenshot of a computer screen

Description automatically generated

Disaster or Individual Tracking filter:

A screenshot of a computer

Description automatically generated

Reset filter:

A screenshot of a computer

Description automatically generated

## Tools Used and Rationale

## MongoDB

## MongoDB was used as the database for this project because of its flexibility, scalability, and ease of integration with Python. It is a NoSQL database that stores data in JSON-like documents, making it highly adaptable to a variety of data types and structures. MongoDB's document-oriented approach is particularly well-suited for managing complex, hierarchical data like the animal records used in this project.

## Key Qualities of MongoDB:

## Flexibility: MongoDB allows for schema-less data storage, meaning that the structure of the data can evolve over time without requiring changes to the database schema.

## Scalability: It supports horizontal scaling, allowing the database to handle large volumes of data by distributing it across multiple servers.

## Ease of Integration: MongoDB has excellent support for Python through the pymongo library, which provides a simple and powerful interface for performing CRUD operations.

## Rich Query Language: MongoDB's query language allows for complex filtering, aggregation, and sorting of data, making it ideal for applications that require dynamic data manipulation.

## Dash Framework

## Dash was used to build the web dashboard for this project because it allows for the rapid development of interactive web applications using Python. Dash is a Python framework for building analytical web applications, and it integrates seamlessly with Plotly for data visualization.

**Key Features of Dash:**

* **User-Friendly**: Dash abstracts away much of the complexity of web development, allowing developers to create sophisticated web applications without requiring extensive knowledge of HTML, CSS, or JavaScript.
* **Integration with Plotly**: Dash supports Plotly, a powerful graphing library, enabling the creation of interactive and responsive charts and graphs.

### Steps Taken to Complete the Project

1. **Project Setup**:
   * Installed the necessary Python libraries (pymongo, dash, dash-leaflet, plotly, pandas).
   * Set up a MongoDB instance and created the AAC database with the animals collection.
   * Imported the Austin Animal Center Outcomes dataset into MongoDB.
2. **CRUD Operations Development**:
   * Developed the AnimalShelter class to handle Create, Read, Update, and Delete operations using pymongo.
   * Implemented methods for each CRUD operation, ensuring they interacted correctly with the MongoDB database.
3. **Dashboard Development**:
   * Built the basic layout of the dashboard using Dash components.
   * Integrated interactive elements such as radio buttons for filtering the data.
   * Created visualizations using Plotly, including a pie chart and a geolocation map.
   * Ensured that the data table and charts dynamically updated based on user input.
4. **Testing and Debugging**:
   * Thoroughly tested each CRUD operation to ensure correct functionality.
   * Tested the dashboard to confirm that data was filtered and visualized correctly based on user selections.
   * Debugged issues related to data filtering, chart rendering, and MongoDB queries.
5. **Finalization**:
   * Fine-tuned the dashboard layout for better user experience.
   * Added the Grazioso Salvare logo and ensured all visual elements were correctly styled.
   * Documented the project in the README file, including detailed setup instructions and code examples.

### Challenges Encountered and Solutions

1. **Data Filtering Challenges**:
   * **Issue**: Ensuring that the data was correctly filtered based on the user's selection in the radio buttons.
   * **Solution**: Implemented and tested the filtering logic within the update\_dashboard callback, using print statements to verify the accuracy of filtered results.
2. **Chart Updates Not Reflecting Filtered Data**:
   * **Issue**: The pie chart initially displayed all breeds, regardless of the selected filter.
   * **Solution**: Debugged the update\_graphs callback by inspecting the filtered DataFrame and ensured that the data passed to the pie chart was correctly grouped and counted.

## Roadmap/Features

## Future enhancements may include:

## Additional filters based on other animal attributes (e.g., age, color).

## Export functionality to allow users to download filtered data as a CSV or Excel file.

## Advanced analytics and visualization options (e.g., time-series analysis).

## Contact

Your name: Juan Rodriguez