# CS 340 README Template

## About the Project/Project Title

*This application provides users with access to the Austin Animal Center (AAC) animal database, enabling them to create, read, update, and delete (CRUD) operations. It interacts with MongoDB using the Pymongo driver to manage animal records. The user would search using a filter for animals with* Grazioso Salvare*-requester of this project-to create a dashboard. Additionally, this project allows a user to benefit from geolocation mapping, user interaction, and charts to locate and utilize animals for search and rescue.*

## Motivation

*CRUD operations are essential when working with databases but structuring queries and managing errors can be time-consuming. To streamline this process, these operations have been abstracted into simply get and set methods, reducing the need to manually format queries or handle errors individually.*

## Getting Started

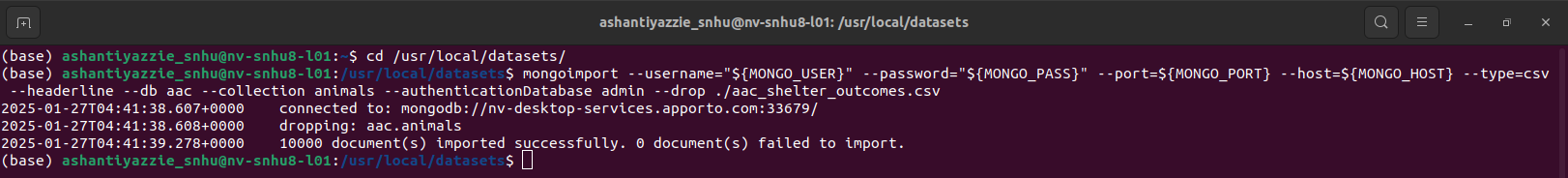
*To set up this program, follow these steps:*

1. *Insert CSV file* aac\_shelter\_outcomes.csv *using MongoDB import tool.*
2. *Open MongoDB and import the* aac\_shelter\_outcomes.csv *file.*
3. *Create both a simple and a complex index to efficiently parse the stored data.*
4. *Set up user authentication by creating an Admin account and an aacuser account for database access.*
5. *Ensure Python is installed and run the program from a notebook environment.*

## Installation

*To use this abstraction the following tools must be installed:*

* *Python 3.6*
* *Pymongo 4.2*
* *MongoDB 4.2*
* *Plotly*
* *Pandas*
* *Imported Mongo database*
* *User account, with assigned permissions*



A screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generated

## 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.*

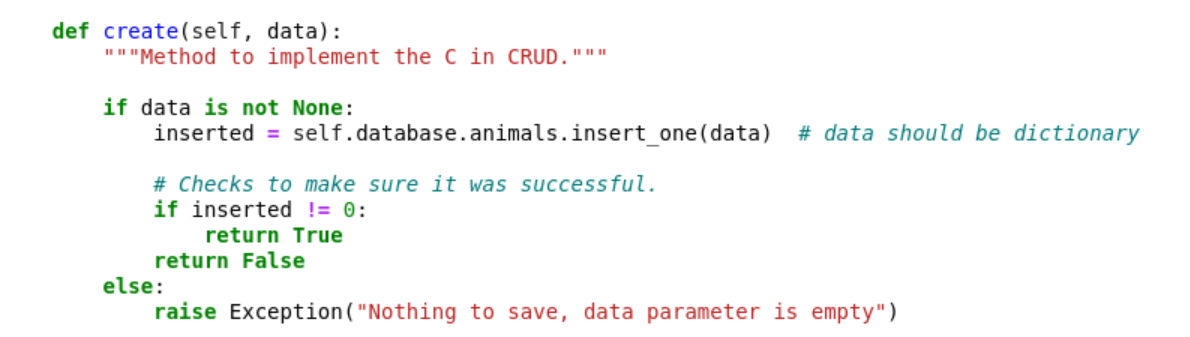
### Code Example

*This code enables users to test, add, and modify animal records in a shelter database. To test functionality, users must first start MongoDB and load the required Python files. They can then add an animal, with the program returning a boolean value upon success or an error message if the addition fails.*

A Create method designed to add a document to a specified MongoDB database and collection.

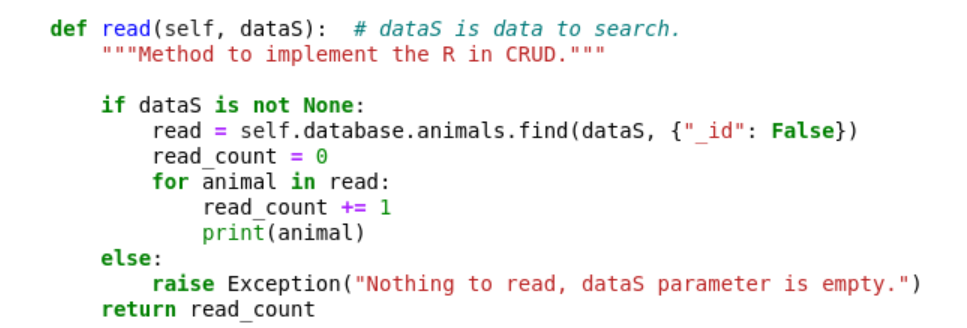
The constructor is responsible for initializing the MongoDB server and establishing a connection to the database.

* **Input:** The function accepts a set of key/value pairs in a format compatible with MongoDB's insert API.
* **Return:** Returns True if the insertion is successful; otherwise, returns False.



A Read method that retrieves document(s) from a specified MongoDB database and collection.

* **Input:** The function takes key/value pairs as arguments to be used with MongoDB's find API.
* **Return:** If successful, it returns a cursor containing the query results; otherwise, it returns the corresponding MongoDB error message.



An Update method that searches for and modifies document(s) within a specified MongoDB database and collection.

* **Input:** The function accepts key/value pairs for querying documents using MongoDB’s find API. The final argument should be a set of key/value pairs formatted for MongoDB’s update operation.
* **Return:** If successful, it returns the result in JSON format; otherwise, it returns the corresponding MongoDB error message.

A close-up of a computer code

AI-generated content may be incorrect.



A Delete method that locates and removes document(s) from a specified MongoDB database and collection.

* **Input:** The function takes key/value pairs as arguments to query documents using MongoDB’s find API.
* **Return:** If successful, it returns the result in JSON format; otherwise, it returns the corresponding MongoDB error message.

A close up of text

AI-generated content may be incorrect.



## Usage

This application includes three primary features. The first allows users to filter rescue dog data using radio buttons based on breed, age, and sex preferences specified by Grazioso Salvare. Selecting a radio button triggers a database query and updates the displayed data frame accordingly. A "Reset" button restores the table to its original, unfiltered state.

The second feature is an interactive map with dynamic updates. Initially, the map displays a marker at the location of the first entry in the data frame. When a user selects one or multiple rows (up to five), the map updates to reflect the chosen locations by repositioning or adding markers.

The third feature is a dynamic pie chart that visualizes breed distribution within the currently displayed data frame (not the entire database). The chart adjusts based on the filtered dataset, providing a visual breakdown of the breeds shown.

Additionally, this application incorporates four database functions. Two functions handle user input—one for adding new records (create) and another for searching existing data (read). The remaining two functions execute these operations. The input functions prompt users for specific details: in the read function, a key and value are required to perform a search, while in the create function, predefined keys are provided, and users input the corresponding values. The create function inserts a Python dictionary into MongoDB, returning True if successful and False otherwise. The read function retrieves matching records or returns False if no results are found.

**All (Reset):**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Water Rescue:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A map with a blue circle and a black background

AI-generated content may be incorrect.**

**Mountain Rescue:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Disaster Rescue:**

**A screenshot of a computer

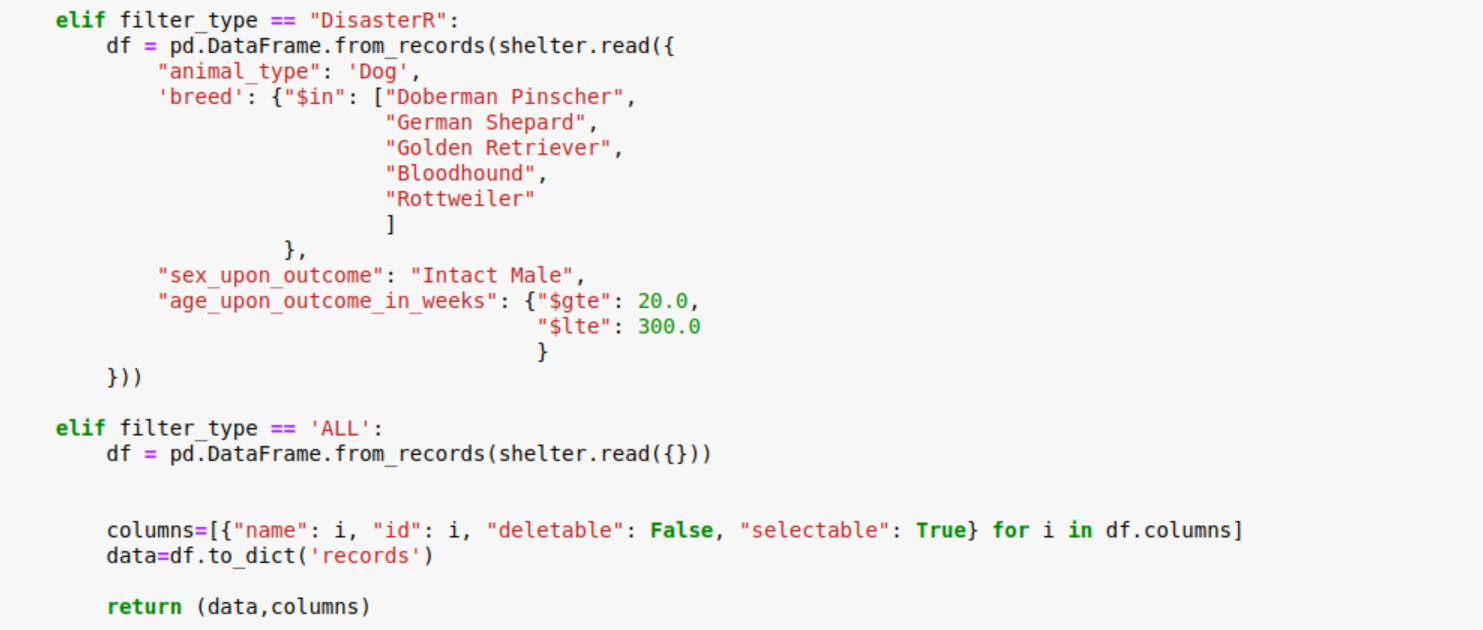
AI-generated content may be incorrect.**

**Dashboard Code Samples:**

*Complex Query*

*A computer screen shot of a program

AI-generated content may be incorrect.*

**

*Pie Chart:*

*A screenshot of a computer code

AI-generated content may be incorrect.*

*A computer screen shot of a code

AI-generated content may be incorrect.*

*Map Markers:*

*A computer screen shot of a computer code

AI-generated content may be incorrect.*

*A screenshot of a computer code

AI-generated content may be incorrect.*

### Tests

The create and read functions each include two unit tests—one to verify a failure scenario and another to confirm a successful execution. The create function tests whether the returned value is True or False, depending on the outcome. Additional tests are implemented to assess various aspects of each CRUD operation. Below is an example of a test case:

def test\_data():

test\_shelter = AnimalShelter()

assert test\_shelter.create(sample\_data) is True

## Roadmap/Features (Optional)

Currently, the project allows up to five items to be selected and displayed on the map. A future update will expand this limit, enabling users to select any number of items, with markers dynamically populating accordingly. Additionally, the map feature will be enhanced to automatically clear markers when the filter is changed, ensuring the displayed data remains accurate.

Another planned improvement is making the pie chart more interactive. Users will have the ability to modify the chart’s data categorization using a dropdown menu. This would allow them to switch between different attributes, such as age or breed, while maintaining full functionality.

## Contact

Your name: Ashanti Yazzie, SNHU-340 Client/Server Development