# CS 340 README Template

## About the Project/Project Title

This Python module is part of a software solution for **Grazioso Salvare**, an international rescue-animal training company. The purpose of this module is to manage data from regional animal shelters and support the identification and categorization of dogs suitable for search-and-rescue training. This is the backend component of a full-stack application that will streamline Grazioso Salvare's process of finding suitable canine candidates.

## Motivation

This project was developed to identify and categorize dogs suitable for search-and-rescue training. Grazioso Salvare’s mission to train these animals for lifesaving efforts drives the purpose of this software, aiming to enhance the efficiency and precision of the selection process.

By integrating with animal shelters’ data, the project seeks to improve the overall effectiveness of Grazioso Salvare’s training programs, ensuring that the right dogs are chosen for the right rescue roles, thus maximizing their potential impact during emergencies.

Additionally, this project is envisioned to grow as a collaborative effort within the animal rescue and training community, so it has been made open source.

## Getting Started

Please refer to the following guide for information on how to use the Python module:

## Installation

To use this module, ensure you have the following installed:

1. Python 3.1 (or better): The programming language used to develop this module
2. MongoDB: Used as the database for storing and managing animal records
3. PyMongo (pip install pymongo): Enables database connectivity and operations
4. BSON: For handling MongoDB-specific ObjectId values

**Dashboard**

This interactive dashboard allows users to filter animal shelter data by rescue types (Water, Mountain, and Disaster) and reset. The results are then displayed in a dynamic table. The dashboard also features a pie chart displaying the distribution of breeds and a map showing the location of selected animals from the dynamic table and their name. Built with Dash and leveraging a MongoDB database, this application enables users to engage with animal shelter data in a meaningful and interactive way.

**Dash** **Screenshots**

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Usage

### Code Example

1. Create Operation Example:

# Initialize AnimalShelter()

crud = AnimalShelter()

Method: create(data)

shelter.create({"animal\_id": "A999902", "animal\_type": "Dog", "breed": "Schnauzer", "color": "Black"})

1. Read Operation Example:

# Initialize AnimalShelter (if you haven’t already)

crud = AnimalShelter()

Method: read(query)

results = crud.read(dog)

# Iterate through the results and print each document

for result in results:

print(result)

1. Update Operation Example:

Method: update(query, update\_data, many=False)

shelter.update({"animal\_id": "A999902"}, {"$set": {"age": 7}})

1. Delete Operation Example:

Method: delete(query, many=False)

shelter.delete({"animal\_id": "A999902"})

### Tests

The CRUD functionality can be tested using the provided Test.ipynb notebook. Ensure the MongoDB instance is running and pre-loaded with the necessary shelter data. The testing notebook allows:

1. Adding new animal profiles.
2. Querying for dogs based on suitability.
3. Updating profiles for specific breeds or rescue types.
4. Deleting outdated records.

How to run the tests with code examples:

# Import the AnimalShelter class

from CRUD import AnimalShelter

# Initialize AnimalShelter()

Crud = AnimalShelter()

# Test the data

dog = {“animal\_id”: “A999902”, “animal\_type”: “Dog”, “breed”: “Schnauzer”, “color”: “Black”}

# Create the record

crud.create(dog)

# Query the collection for our animal

results = crud.read(dog)

Iterate through the results and print the doc

For result in results:

Print(result)

# Test the Update operation

print("\nTesting Update Operation")

update\_query = {"animal\_id": "A999902"}

update\_data = {"$set": {"age": 7}}

modified\_count = crud.update(update\_query, update\_data)

print(f"Number of Documents Updated: {modified\_count}")

# Verify the Update operation

updated\_result = list(crud.read(update\_query))

print(f"Updated Document: {updated\_result}")

# Test the Delete operation

print("\nTesting Delete Operation")

delete\_query = {"animal\_id": "A999902"}

deleted\_count = crud.delete(delete\_query)

print(f"Number of Documents Deleted: {deleted\_count}")

# Verify the Delete operation

deleted\_result = list(crud.read(delete\_query))

print(f"Deleted Document Result: {deleted\_result}")

### Screenshots

Import Data:

A screenshot of a computer screen

Description automatically generated

Python module screenshots:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer code

Description automatically generated

A screenshot of a computer code

Description automatically generated

Testing screenshots:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

*A screenshot of a computer code

Description automatically generated*



A screenshot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

## Contact

Your name: Mitchell Fontaine

Email: Mitchell.fontaine@snhu.edu