# CS 340 README Template

*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 provides a simple, reusable Python module (crud.py) that implements Create and Read operations against a MongoDB collection of animal shelter outcomes. It defines an AnimalShelter class that connects to the AAC database and animals collection, enabling easy insertion and querying of shelter data

## Motivation

Working with live data stores is a critical skill for backend development. This module lets me practice authenticating to MongoDB, writing clean object­-oriented wrappers around its driver, and exposing a simple interface that other scripts or notebooks can import without duplicating boilerplate. By encapsulating Create, Read, Update, and Delete logic in one class, I ensure maintainable, consistent data‑access patterns across any project that uses the Austin Animal Center outcomes data.

## Getting Started

To get a local copy up and running, follow these simple example steps.

1. You can download the repository
2. Make sure you have accesss to the MongoDB and import the csv file aac\_shelter\_outvome.csv
3. Complete a simple and complex to parse the data in the document
4. Now you will need the authorize a user and create Admin account and a aacuser.
5. Install python required packages

## Installation

*List the tools you need to use the software and how to install them.*

1. Current version of Python to run .py and ipynb
2. MongoDB

Install Dependencies

pip install pymongo

## 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

### In just a few lines, you can exercise all four CRUD operations:

### Import & Instantiate

### Bring in the AnimalShelter class and create an instance so you’re connected to the AAC database and animals collection.

### Create - Call create() with a dictionary representing a new animal record. The method returns the newly inserted document’s \_id, confirming it was saved.

### Read - Use read() with a filter (for example, the same animal\_id) to retrieve a list of matching documents. You’ll get back a Python list of dictionaries containing the stored data.

### Update - Invoke update() with the same filter plus a second dictionary of fields to change. It returns an integer count of how many documents were modified.

### Delete - Finally, call delete() with your filter to remove the record. This method returns the number of documents removed, proving the deletion succeeded.

### Tests

To RUN TEST we can:

1. Import the Class
2. Start by importing your custom AnimalShelter class:

from crud import AnimalShelter

1. Connect to your MongoDB database by creating an instance:

shelter = AnimalShelter()

1. Define a Test Document
2. Create a Python dictionary for your test animal, for example:

animal = {

"animal\_id" : "ABCE1234",

"animal\_type" : "Wolf",

"breed" : "Arctic Wolf",

"color" : "White",

"name" : "Zya"

}

1. Test Create
2. Insert the document and capture its returned \_id:

new\_id = shelter.create(animal)

1. If successful, new\_id will be the new ObjectId.
2. Test Read
3. Retrieve the inserted document by filtering on animal\_id:

results = shelter.read({"animal\_id": "TST100"})

for doc in results:

print(doc)

1. This confirms the record exists as stored.
2. Test Update
3. Modify one or more fields and capture the count of modified documents:

updated\_count = shelter.update({"animal\_id": "ABCE1234"}, {"color": "Grey"})

A non‑zero updated\_count proves the update succeeded.

1. Test Delete
2. Remove the test document and capture how many were deleted:

deleted\_count = shelter.delete({"animal\_id": "ABCE1234"})

A non‑zero deleted\_count shows the delete worked.

Verify Deletion

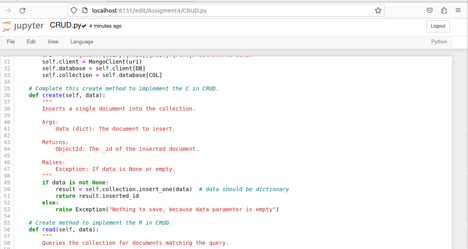
13. Optionally run another read to ensure no documents remain:

14. print(shelter.read({"animal\_id": "ABCE1234"})

### Screenshots

*Provide screenshots that demonstrate your work.*

### create() with a dictionary representing a new animal record. The method returns the newly inserted document’s \_id, confirming it was saved.



1. *Read - Use read() with a filter (for example, the same animal\_id) to retrieve a list of matching documents. You’ll get back a Python list of dictionaries containing the stored data.*

*A screenshot of a computer code

AI-generated content may be incorrect.*

1. *Update() with the same filter plus a second dictionary of fields to change. It returns an integer count of how many documents were modified*

*A screenshot of a computer code

AI-generated content may be incorrect.*

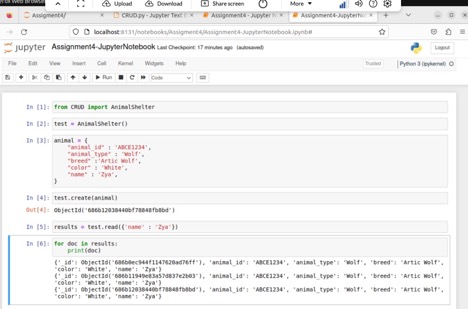
1. *Delete() method for deleting call delete() with your filter to remove the record. This method returns the number of documents removed, proving the deletion succeeded*

*A screenshot of a computer code

AI-generated content may be incorrect.*

***Test Screenshots:***

1. Make a test = animalShelter()
2. Make a naimal dictionary and
3. Test.create(animal) itll create a id
4. And then read to verify
5. Run for loop to display the newly developed dictionary



*Update: To update youll use test.update and it will print how many files were updated*

*A screenshot of a computer

AI-generated content may be incorrect.*

*Verify test worked – Test.read and now color is grey.*

*A screenshot of a computer

AI-generated content may be incorrect.*

*Invalid test:*

*A screenshot of a computer

AI-generated content may be incorrect.*

*Test Delete:*

*Test. Delete will delete and print out how many was deleted*

*A screenshot of a phone

AI-generated content may be incorrect.*

*Next you can use the radio buttons to select The dashboard includes a set of intuitive radio buttons that allow the user to filter animals by:*

*Water Rescue*

*Mountain or Wilderness Rescue*

*Disaster or Individual Tracking*

*Reset (all data)*

*At the top of the dashboard:*

*The Grazioso Salvare logo is displayed with a clickable link to snhu.edu.*

*A personal identifier ("Dashboard created by Joseph Dumke") is prominently shown, as requested by the client.*

*A screenshot of a computer

AI-generated content may be incorrect.*

*A screenshot of a computer

AI-generated content may be incorrect.*

*When selecting between the different radio button each will show a percentage graph based off selection. Itll also show a map of location of animal, breed, and name.*

*A screenshot of a computer

AI-generated content may be incorrect.*

## Contact

Your name: Joseph Dumke