# CS 340 README Austin Animal Shelter

## About the Project/Austin Animal Shelter

*MongoDB CRUD Python Module – Create & Read Operations  
This project is a Python module designed to enable Create and Read functionality with a MongoDB database using the Austin Animal Center (AAC) Outcomes dataset. It allows for inserting new animal records and retrieving existing records based on search criteria, updating entries, as well as deleting entries.*

## Motivation

*The motivation behind this project was to build a modular and reusable Python interface to interact with MongoDB using object-oriented programming. The goal was to understand how to work with databases in Python, practice good coding practices, and prepare for future project integration, such as building a dashboard in Project Two.*

## Getting Started

*This is an example of how you may give instructions on setting up your project locally: “To get a local copy up and running, follow these simple example steps.”*

## Installation

*To get a local copy up and running inside the Apporto virtual environment:*

1. *Log in to your Apporto instance.*
2. *Import the AAC Outcomes dataset using the following command in the terminal:*

*mongoimport --host <your-host> --port <your-port> -u aacuser -p 'SNHU1234' --authenticationDatabase admin --db AAC --collection animals --type csv --headerline --file /usr/local/datasets/aac\_shelter\_outcomes.csv*

1. *Use the Jupyter Notebook interface to create a .py module and a testing notebook in the same directory.*
2. *Follow the installation steps below to ensure tools are ready.*

***Tools Used:***

* ***MongoDB*** *(hosted in Apporto): Used to store AAC animal records*
* ***PyMongo****: Python driver used to interact with MongoDB*
* ***Jupyter Notebook****: Used for interactive testing and development*
* ***Apporto Environment****: Provides hosted access to all tools without local setup*

## Usage

*To use the module, follow these steps in Jupyter:*

1. *Upload animal\_shelter.py into the same directory as your notebook.*
2. *In your notebook:*

*A screenshot of a computer

AI-generated content may be incorrect.*

*Animal\_shelter.py*

*A screenshot of a computer code

AI-generated content may be incorrect.*

*Update and delete*

*A screenshot of a computer

AI-generated content may be incorrect.*

*Import .py file into notebook*

*A screenshot of a computer

AI-generated content may be incorrect.*

*Run file, should get the output of insert successful*

*A screenshot of a computer code

AI-generated content may be incorrect.*

*Testing update and delete output successfully.*

### Code Example

*This code is responsible for the CREATE, READ, UPDATE and DELETE part of CRUD. It allows us to create a new animal object to insert and view within the MongoDb database AAC. Read what is there in the database, update the entry within the database to something else, and also delete it from the database. The code below shows how all of this is possible. The code below is also what you would import within your notebook to run the tests.*

*from pymongo import MongoClient*

*from pymongo.errors import PyMongoError*

*class AnimalShelter:*

*def \_\_init\_\_(self):*

*USER = 'aacuser'*

*PASS = 'Test1234'*

*HOST = 'nv-desktop-services.apporto.com'*

*PORT = 31930 # <-- Adjust based on your environment*

*DB = 'AAC'*

*COL = 'animals'*

*self.client = MongoClient(f'mongodb://{USER}:{PASS}@{HOST}:{PORT}/?authSource=admin')*

*self.database = self.client[DB]*

*self.collection = self.database[COL]*

*def create(self, data):*

*if data:*

*try:*

*result = self.collection.insert\_one(data)*

*return result.acknowledged*

*except:*

*return False*

*else:*

*raise ValueError("No data provided.")*

*def read(self, query):*

*try:*

*return list(self.collection.find(query))*

*except:*

*return []*

*def update\_documents(self, query, new\_values):*

*try:*

*result = self.collection.update\_many(query, {'$set': new\_values})*

*return result.modified\_count*

*except Exception as e:*

*print(f"Update error: {e}")*

*return 0*

*def delete\_documents(self, query):*

*try:*

*result = self.collection.delete\_many(query)*

*return result.deleted\_count*

*except Exception as e:*

*print(f"Delete error: {e}")*

*return 0*

### Tests

*Tests were run by inserting a sample animal record and then querying the collection to confirm that the record was added. The entry is then updated, and then deleted.*

*You can type the tests within Jupyter notebook and run to see if the test passed. You can also open a terminal, navigate to your DB within the terminal and check to see if the animal was added, updated, and deleted. Here is a screenshot of what that looks like:*

*A screenshot of a computer screen

AI-generated content may be incorrect.*

*A screenshot of a computer code

AI-generated content may be incorrect.*

## Contact

Your name: Cody Adams

SNHU Student -CS 340

Cody.adams2@snhu.edu