# CS 340 README

About the Project/Project Title

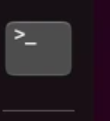
This project grants users access to a database of animals in an animal shelter. The application implements the CRUD methodology for manipulating the data.

## Motivation

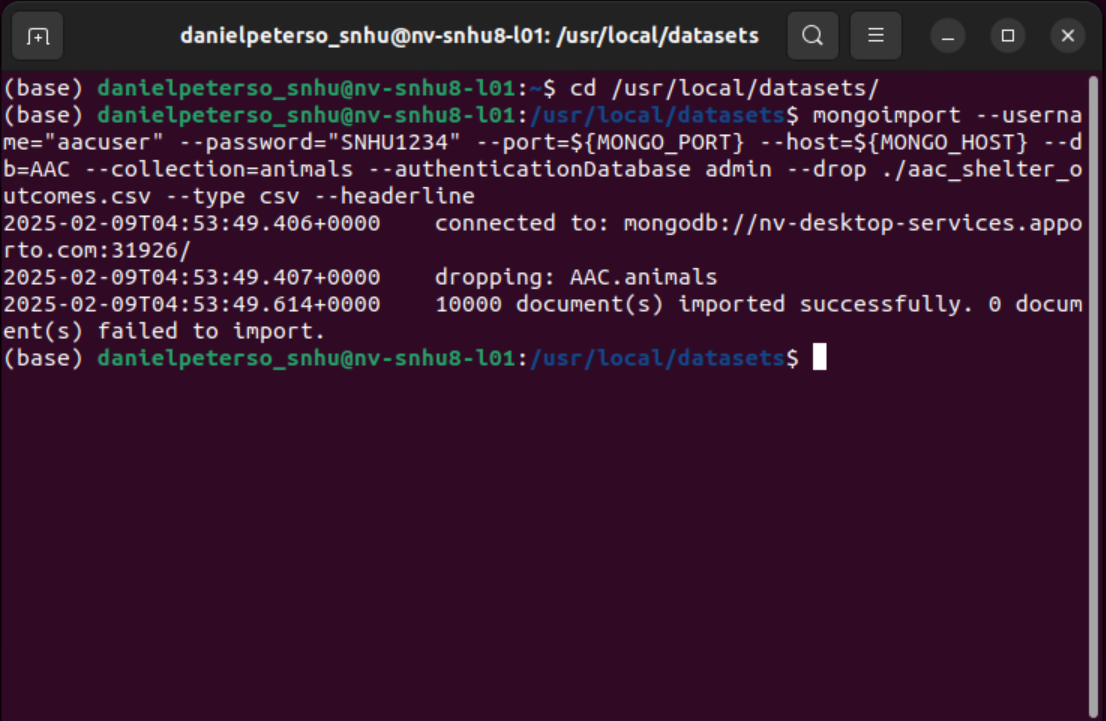
This project aims to increase my skills in working with Mongo databases utilizing the Python programming language. The secondary purpose is to familiarize myself with the implementation of the CRUD methodology. (Create, Read, Update, Delete).

## Getting Started

1. Open a terminal.



1. Import aac\_shelter\_outcome.csv into the mongo shell.



1. Create an index parsed with the data from the .csv file.
2. Create an Admin account with full access to the database. Utilize a password to ensure security and user authentication.
3. Create a user account with only read access to the database. Utilize a password to ensure security and user authentication.
4. Install Python and run the program.

## Installation

A version of Python that has access to both .py and .ipynb file types. I utilized Jupyter Notebook to run Python version 3.9

MongoDB to access the database.

## Usage

### Code Example

### The code will allow the user to test the add and read functions of the animals database. It contains the actual application code within an animal\_shelter.py file and the preset test parameters in a CRUD.py file. To utilize the database the user must open a terminal and run the Mongo shell. It must then load the appropriate.py files. The user can use animals.create(STRING) command to add animals to the database. If added successfully the terminal will return the BOOLEAN TRUE. If unsuccessful the terminal will return an error message. The user can also use the animals.read(STRING) command to search the database for the parameter provided. The terminal will return all data on every animal that fits the search parameter. In my example, I search for animals named Rocket since my test animal is Rocket Racoon from Guardians of the Galaxy. The update method allows the user to update the data of an animal. In my example, I use the command test.update(STRING) to update Rocket’s outcome type from sheltered to adopted. The delete method allows a user to delete an animal from the database. My testing script uses the test.delete(STRING) command to delete Rocket from the database as he is a test animal and doesn’t exist. The delete method can be used to delete specific animals by animal ID since they are unique to each animal. It can also be used to delete all animals that fit a parameter. For instance, if there were multiple Rockets the command would delete all Rockets. For this reason, I suggest using animal ID when deleting it. This will ensure that only the correct animal gets deleted. If you leave the key blank, the delete function will also delete all animals. This is a good way to reset your database if things get out of hand. Delete all animals and then repeat step 2 to establish a fresh database.

The final product is a dashboard that filters the animal database for rescue dogs best equipped for certain scenarios. The scenarios are Water Rescue, Mountain or Wilderness Rescue, Disaster or Individual Tracking. The reset button allows the user to see the entire database. The dashboard includes a map that can pan to show the location of the shelter that has the animal. The dashboard also includes a pie chart showing the percentage of each breed within the search criteria. The instructions were to include the dash table, the map, and a second chart of my choosing. I chose to include the pie chart because I like to see percentages.

### Tests

The code establishes data that it attaches to the animal variable. It then uses the command test.create(animal) to add the test data to the database. If it returns the Boolean TRUE it was successful at adding the data. It then tests the read function by using the test.read({‘name’: “Rocket”}) command to read the data that was added. The test script then uses the test.delete({‘name’: “Rocket”}) command to delete Rocket from the database. I added this command to the test to ensure multiple tests wouldn’t bloat the database with unnecessary data. Finally, I use the test.read({‘name’: “Rocket”}) command to verify that the delete command functioned properly. If the delete command functions properly the read command will no longer return data on Rocket.

### Screenshots

### File structure within Jupyter Notebook

### 

### *Screenshots of the test in Jupyter Notebook*

*A screenshot of a computer program

AI-generated content may be incorrect.*

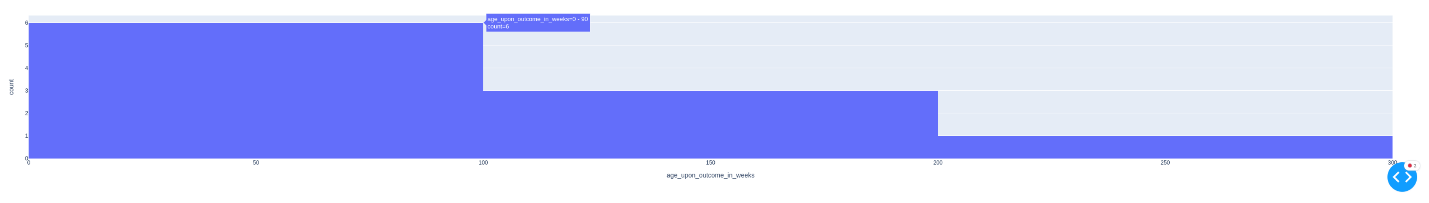
### *Screenshot of the .py file that establishes the CRUD methodology.*

### 

### *Screenshot of the .py file containing the test script.*

Screenshots of the Dashboard from the final project.

A screenshot of a map

AI-generated content may be incorrect.

Roadblocks:

I had issues getting the Dashboard to load. I spent a bit of time trying to fix this before reaching out to the instructor. My error was that I was pulling data as part of the read function. I was returning the data as one single object. When I changed it to return the data as a list everything worked. I had a few call-back errors show up when running the dashboard and instead of leaving it be I decided to try and troubleshoot them. I somehow ruined the read function, I think it was a combination of my toddler sitting at my desk and pretending to type on the keyboard and me trying to fix the keystrokes.

## Contact

Dan Peterson