CS 340 README Template

CRUD Python module

This CRUD Python module is the middle-ware between a MongoDB database and the Python web application. After the constructor, this module implements the Create, Read, Update and Delete functions. Its purpose is to query or alter database with an insert (Create), find (Read), Update or Delete command.

Motivation and requirements

Grazio Salvare has hired Global Rain, a software engineering company, to create a full stack application that can work with data from animal shelters to categorize dogs available for search-and-rescue training. The application will have a back-end database, a client-facing dashboard and an interface between them.

The included interactive Python notebook file was created using Jupyter Notebook.

MongoDB was selected as the back-end database. Python has uses a native PyMongo package as the driver for MongoDB, and since the middleware class and front-end dashboard were designed in Python, MongoDB was a natural choice.

The Dash framework is "the original low-code framework for rapidly building dash apps in Python." Coupled with the Python data analysis packages pandas and graphing library Plotly Express, the Dash framework can be used turn raw data into visually appealing information.

The dashboard has the following requirements:

1. The Grazioso Salvare logo. The company has requested that this logo include a URL anchor tag to the client's home page: www.snhu.edu.
2. A unique identifier (text or image) containing your name. Grazioso Salvare would like to credit you as the creator of the dashboard.
3. Interactive filter options (buttons, drop-downs) to filter the Austin Animal Center Outcomes

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

data set by:

- Water Rescue
- Mountain or Wilderness Rescue
- Disaster Rescue or Individual Tracking
- Reset (returns all widgets to their original, unfiltered state)

4. A data table which dynamically responds to the filtering options
5. A geolocation chart and a second chart of your choice (such as a pie chart) that dynamically respond to the filtering options

Getting Started

To set up a local MongoDB instance, use the mongoimport command to import the CSV file using admin credentials and host/port information:

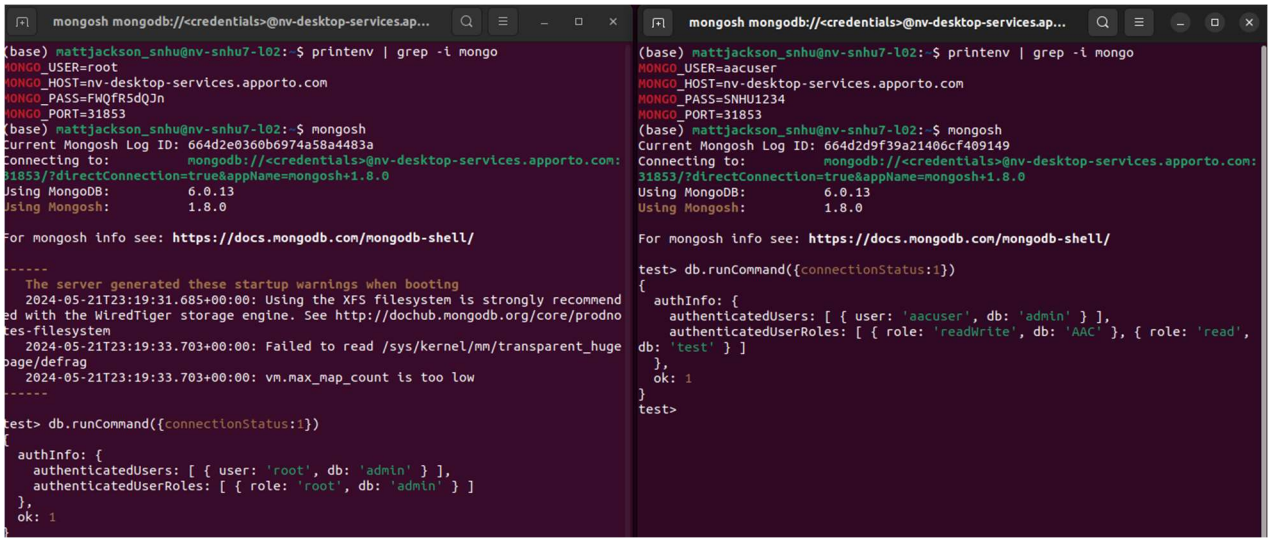
```
(base) mattjackson_snhu@nv-snhu7-l01:/usr/local/datasets$ mongoimport --username="${MONGO_USER}" \
    --password="${MONGO_PASS}" --port=${MONGO_PORT} \
    --host=${MONGO_HOST} --db AAC --collection animals \
    --type csv --headerline --ignoreBlanks \
    --authenticationDatabase admin --drop ./aac_shelter_outcomes.csv
2024-05-21T15:38:36.364+0000    connected to: mongodb://nv-desktop-services.apporto.com:31853/
2024-05-21T15:38:36.365+0000    dropping: AAC.animals
2024-05-21T15:38:37.551+0000    10000 document(s) imported successfully. 0 document(s) failed to import.
```

Then create a local user in the admin database:

```
admin> db.createUser({ user:"aacuser",pwd:passwordPrompt(), roles:
[{ role: "readWrite", db:"AAC"} , { role:"read", db:"test"}] } )
Enter password
********{ ok: 1 }
```

Set your local variables to the new user and password and login:

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.



Installation

MongoDB can be installed on your machine by following the instructions on the MongoDB website:

https://www.mongodb.com/docs/manual/installation/

I used Jupyter notebook (https://jupyter.org/) to build and run the class file and tests, but any python environment with the requisite class and test files can be used.

Usage

To insert a record, query the database, update a record (or records), or delete a record (or records) instantiate an AnimalShelter object and use the .create, .read, .update and .delete methods, respectively. The .create method takes a json dictionary and returns True if the create method was successful, else it returns False. The read .method also takes a json dictionary and returns a list of matching results. If there is an error or no match, it returns an empty list. The .update method takes a filter to match and a value to update. It also takes an optional Boolean value (default is True). When the Boolean value is set to False, the update_many method is invoked, otherwise the update_one is invoked. The .delete method takes a json list and an optional Boolean value. If the Boolean value is set

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

to False, the delete_many method is invoked, otherwise the delete_one method is invoked. By default, this Boolean variable is set to True.

Code Example

```
from CRUD import AnimalShelter


 animalShelter = AnimalShelter()
 data = {“breed”:”Alaskan Husky Mix”,”name”:”Joe”}
 animalShelter.create(data)


 query = {“breed”:”Alaskan Husky Mix”,”name”:”Joe”}
 animalShelter.read(query)


fltr = {"breed":"Alaskan Malamute Mix
update = {"$set": {"breed":"Alaskan Husky Mix"}}
animalShelter.update(fltr,update)


# invokes delete_many, will delete all the remaining matches
animalShelter.delete({"breed":"Alaskan Malamute Mix"},False)
```

Tests

```
data = {
  'age_upon_outcome': '1 year',
  'age_upon_outcome_in_weeks': 52.9560515873016,
  'animal_id': '1111111',
```

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

```
 'animal_type': 'Dog',

 'breed': 'Alaskan Husky Mix',

 'color': 'Black/White',

 'date_of_birth': '2016-12-28',

 'datetime': '2018-01-02 16:37:00',

 'location_lat': 30.4790154956102,

 'location_long': -97.2867023977915,

 'monthyear': '2018-01-02T16:37:00',

 'name': 'Joe',

 'outcome_type': 'Adoption',

 'rec_num': 1851,

 'sex_upon_outcome': 'Neutered Male'}


animalShelter.create(data)


query = {'breed': 'Alaskan Husky Mix', 'name': 'Joe'}

animalShelter.read(query)


fltr = {"breed":"Alaskan Malamute Mix"}                    # only one matching breed, will return 1

update = {"$set": {"breed":"Alaskan Husky Mix"}}

animalShelter.update(fltr,update)


fltr = {"breed":"Alaskan Husky Mix"}                    # multiple matching records, will return >1

update = {"$set": {"breed":"Alaskan Malamute Mix"}}
```

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

animalShelter.update(fltr,update,False)

animalShelter.delete({"breed":"Alaskan Malamute Mix"}) # delete_one is True by default

animalShelter.delete({"breed":"Alaskan Malamute Mix"},False) # invokes delete_many, will delete all the remaining matches

Test Screenshots

In [1]:
```
from CRUD import AnimalShelter
from pprint import pprint
from os import system

# Poor practice to hardcode credentials, done here for simplicity of test
# In production, prompt user for username and password:
#
# from getpass import getpass
# USER = input("Enter username: ")
# PASS = getpass("Enter password: ")

USER = 'aacuser'
PASS = 'SNHU1234'

animalShelter = AnimalShelter(USER,PASS)  # instantiate Animal Shelter object

# initialize clean database before tests
system('mongoimport --username="${MONGO_USER}" \
             --password="${MONGO_PASS}" --port=${MONGO_PORT} \
             --host=${MONGO_HOST} --db AAC --collection animals \
             --authenticationDatabase admin --drop \
             --headerline --type=CSV \
              /usr/local/datasets/aac_shelter_outcomes.csv')
```

```
2024-05-31T17:26:22.187+0000    connected to: mongodb://nv-desktop-services.apporto.com:31853/
2024-05-31T17:26:22.187+0000    dropping: AAC.animals
2024-05-31T17:26:22.388+0000    10000 document(s) imported successfully. 0 document(s) failed to import.
```

Out[1]: 0

In [2]:
```
data = {
  'age_upon_outcome': '1 year',
  'age_upon_outcome_in_weeks': 52.9560515873016,
  'animal_id': '1111111',
  'animal_type': 'Dog',
  'breed': 'Alaskan Husky Mix',
  'color': 'Black/White',
  'date_of_birth': '2016-12-28',
  'datetime': '2018-01-02 16:37:00',
  'location_lat': 30.4790154956102,
  'location_long': -97.2867023977915,
  'monthyear': '2018-01-02T16:37:00',
  'name': 'Joe',
  'outcome_type': 'Adoption',
  'rec_num': 1851,
  'sex_upon_outcome': 'Neutered Male'}
```

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

```
In [3]: animalShelter.create(data)
Out[3]: True

In [4]: animalShelter.create(data) # duplicate animal_id
        Create Error: duplicate animal_id
Out[4]: False

In [5]: animalShelter.create("Invalid entry")  # not formatted properly
        Create Error: string indices must be integers
Out[5]: False

In [6]: query = {"breed":"Alaskan Husky Mix","name":"Max"}  # valid query
        animalShelter.read(query)
Out[6]: [{'_id': ObjectId('665a083ef430865cd4474929'),
          'rec_num': 1851,
          'age_upon_outcome': '1 year',
          'animal_id': 'A764373',
          'animal_type': 'Dog',
          'breed': 'Alaskan Husky Mix',
          'color': 'Black/White',
          'date_of_birth': '2016-12-28',
          'datetime': '2018-01-02 16:37:00',
          'monthyear': '2018-01-02T16:37:00',
          'name': 'Max',
          'outcome_subtype': '',
          'outcome_type': 'Adoption',
          'sex_upon_outcome': 'Neutered Male',
          'location_lat': 30.4790154956102,
          'location_long': -97.2867023977915,
          'age_upon_outcome_in_weeks': 52.9560515873016}]

In [7]: query = {"breed":"No Breed"}                # valid query, no match
        animalShelter.read(query)
Out[7]: []

In [8]: query = "Invalid query"                     # invalid query
        animalShelter.read(query)
        Read Error: filter must be an instance of dict, bson.son.SON, or any other type that inherits from collections.Mappi
        ng
Out[8]: []

In [10]: # UPDATE

In [11]: fltr = {"breed":"Alaskan Malamute Mix"}                              # only one matching breed, will return 1
         update = {"$set": {"breed":"Alaskan Husky Mix"}}
         animalShelter.update(fltr,update)
Out[11]: 1

In [12]: fltr = {"breed":"Alaskan Husky Mix"}                         # multiple matching records, will return >1
         update = {"$set": {"breed":"Alaskan Malamute Mix"}}
         animalShelter.update(fltr,update,False)
Out[12]: 7

In [13]: fltr = {"breed":"Alaskan Malamute Mix"}                      # multiple matching breeds, will only update first 1
         update = {"$set": {"breed":"Alaskan Husky Mix"}}
         animalShelter.update(fltr, update)
Out[13]: 1

In [14]: fltr = "invalid filter"
         update = {"$set": {"breed":"Alaskan Husky Mix"}}
         animalShelter.update(fltr, update)  # error will return 0
         Update Error: filter must be an instance of dict, bson.son.SON, or any other type that inherits from collections.Map
         ping
Out[14]: 0
```

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

```
In [15]: animalShelter.update(None, None)                          # raise exception
---------------------------------------------------------------------------
Exception                                 Traceback (most recent call last)
Input In [15], in <cell line: 1>()
----> 1 animalShelter.update(None, None)

File ~/Desktop/CRUD.py:71, in AnimalShelter.update(self, fltr, update, one)
     69         return 0
     70 else:
---> 71     raise Exception("Nothing to update, because filter or update parameter is empty")

Exception: Nothing to update, because filter or update parameter is empty

In [16]: # DELETE

In [17]: animalShelter.delete({"breed":"Alaskan Malamute Mix"})   # delete_one is True by default
Out[17]: 1

In [18]: animalShelter.delete({"breed":"Alaskan Malamute Mix"},False)  # invokes delete_many, will delete all the remaining ma
Out[18]: 5
```

Deployment

The final deployed web application loads with a query of the full database displaying up to 10 results per page with each row being selectable; the Grazioso Salvare logo which is hyperlinked to SNHU.edu; four sorting options based on the client requirements; a map showing the location of the selected animal with breed and name displayed upon hover and click of the dropped pin. Upon selecting one of the sorting options, a pie chart populates next to the map showing the breakdown of available breeds. Reset clears the pie chart and returns the table to its initial state.

Demonstration

Demonstration.mp4

Challenges

The project was completed with separate milestones to build the database, the API to for easier functionality and the web application dashboard. The biggest challenges I faced were in building the dashboard. Identifying which part of the stack to target when errors or unexpected behavior occurred was hard. Not returning the right search results had me looking at my CRUD file when I should have been looking at the syntax passed to it from the dashboard. Passing multiple return objects from the callback function took me quite a while to figure out, but defining the callback outputs as a list was the answer that worked. It was also tricky to figure out how to hide the pie chart upon load and reset; I had

to change the display style of the figure from none to block and return the style as a parameter from the callback.

Contact

matt.jackson @ snhu.edu

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.