# Katie Stapleton

CS 340 -Mod 4

Readme Template

05/26/2021

# MongoDB README

## About the Project/Project Animal Rescue

The project is known as “project animal rescue”. The international rescue-animal training company Grazioso Salvare requested custom software for users to access and operate with the database. The project is to create and maintain a client-server database with an interface that employees can use to operate the system.

## Motivation

The software is being developed to support proper animal treatment internationally. Our contribution is providing the services and software Grazioso Salvare to continue their services internally and with local animal shelters.

## Getting Started

To get started, make sure you have Python 3.+ and PyMongo installed in your system. If you are directly assigned to the project, we will provide you with a username and password to access the MongoDB. Otherwise, you need to create your MongoDB account and database. If desired, sample data is available to upload into the server. Tests are performed by executing the Python scripts inside a Jupyter Notebook.

If you choose to make your own version of the software, make sure to include an authentication function to both connect and authorize access into correct MongoDB database. The “init” module currently handles the authorization. Any user roles need to be setup beforehand in the MongoDB server. When establishing the connection, you will need the username, password, host/port, and the database that stores the authentication. All this information can be retrieved from the MongoDB server. *This information is vital to a proper connection*. Authorization issues were the largest hiccup when first developing the MVP. An admin-level account was created to overcome the issues while in early development.

The coding is written in PyMongo which is mainly using Python to generate modules compatible with MongoDB. Python is used to create the script to send between the user and server. In simple terms, Python does all the work of sending commands to the MongoDB instead of retyping the entire command in a MongoDB terminal. This is how the functions are designed, as shown in the basic examples below.

Testing are generated in Python and executed using a Jupyter Notebook. First, create a Jupyter Notebook to host the tests. For each test, call in the python script and class that contains the coding to test. Supply the data needed to perform the test. *Data needs to be defined in dictionary formatting, abiding to MongoDB standard syntax.* It is recommended to create a function specifically for testing. Within the testing function, call upon the class and name of the function. Specific a response for both successful and failed executions/results. Jupyter is confused at first. Remember, this process needs to be repeated for every new test. It is also recommended to create a new section for each function. This helps by separating the issues apart instead of combining the feedback. It also the function to execute without raised exceptions from other functions halting the execution.Finally, Make sure to save/create checkpoints to prevent losing progress.

## Installation

The project requires the installation and usage of Python 3.+, MongoDB, PyMongo, and Jupyter Notebook. Please see the official sites for Python, MongoDB, and Jupyter to setup each tool.

For a brief overview: The server is hosted through MongoDB; , the server can function locally, but still requires access to a MongoDB server. The software scripts are written in Python 3.+ and PyMongo. The tests are generated in Python and executed using the Jupyter Notebook.

## Usage

We have provided some hopefully-useful examples to demonstration how the project operates. We have including a coding sample(s), test execution, and screenshots. All scripting are written in Python.

### Code Example

All software scripting are written (and saved) in Python 3.+.

# inserts document into specified database/collection

def AddDoc(self, userData):

"""Insert one document into collection"""

if userData is not None:

self.database.animals.insert(userData) # data should be dictionary

#result = 'Document added to the database'

result = True

else:

#raise Exception("Parameter is empty. No data saved")

result = False

return

return result return result

def ReadDoc(self, search): # data should be dictionary

"""Read/query collection. Requires standard key:pair syntex used for find()."""

if search is not None:

result = self.database.animals.find(search)

return result

else:

# raise Exception("Data query failed. Please check the search parameters.")

result = 'Data query failed. Please check the search parameters. Data may not be located within the system.'

return result

### Tests

Testing are generated in Python and executed using a Jupyter Notebook. Below are the basic examples for testing the functions.

from aac\_animals\_CRUD\_updated import AnimalShelterCRUD

class TestAnimalShelterCRUD:

# class that stores the modules

run = AnimalShelterCRUD()

userData = {

"age\_upon\_outcome" : "3 years",

"animal\_id" : "TEST001",

"animal\_type" : "Test breed",

"breed" : "Test Cat Mix",

"color" : "Test Color Black",

"date\_of\_birth" : "2014-04-10",

"datetime" : "2017-04-11 09:00:00",

"monthyear" : "2017-04-11T09:00:00",

"name" : "Complex name",

"outcome\_subtype" : "SCRP",

"outcome\_type" : "Transfer",

"sex\_upon\_outcome" : "Neutered Male",

"location\_lat" : 35.625439455,

"location\_long" : -96.3435972102188,

"age\_upon\_outcome\_in\_weeks" : 156.767857142857

}

search = {"breed" : "Beagle"}

# test create/upload function

addData = run.AddDoc(userData)

print(addData)

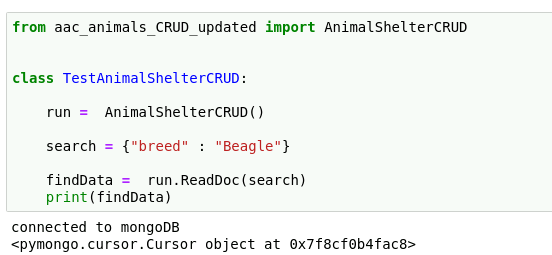
# Test read/find() function

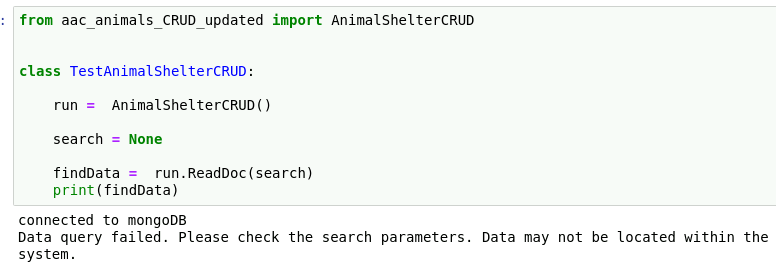
findData = run.ReadDoc(search)

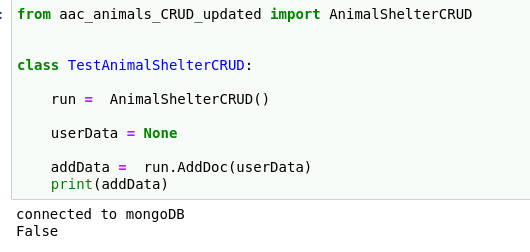
print(findData)

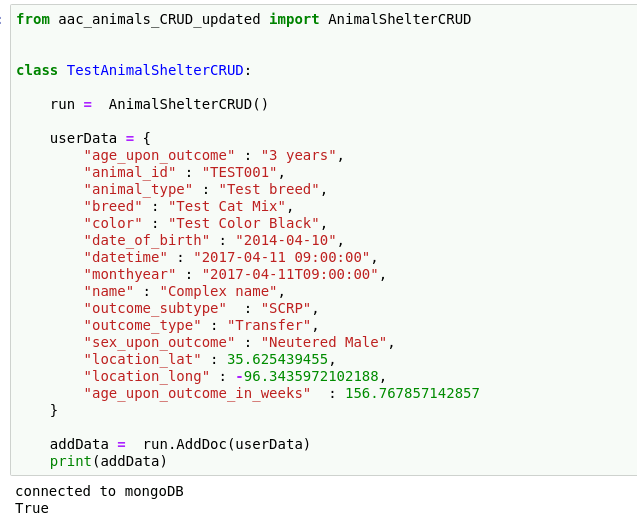
### Screenshots

The Python Script for functions is located above, within the Usage-Code section. This section displays the testing functions and outcomes of the test. To research syntax errors, enable the terminal function in Jupyter. When errors occur, Jupyter will provide a line-by-line overview of the execution. For checking logic errors, it is recommend to have the function return something to indicate a success or failure; the expected outcome and actual outcome can then be better compared.









## Roadmap/Features (Optional)

Here is the list of our major issues. For a detailed listing of all issues, please see the “issues” section of the repo. (Please alerts us of any issues your notice!)

Feel free to submit ideas for features to improve or advanced the Animal Shelter software for Grazioso Salvare!

## Contact

Your name: Katie Stapleton