# CS 340 README AAC Milestone 4

## About the Project

*The AACDatabaseLayer is a Python class that provides functionality to interact with a MongoDB database for a future web-based app. It facilitates Create (C) and Read (R) operations using the PyMongo library from CRUD. This module aims to simplify the database interaction process and allows developers to easily store and retrieve data from a MongoDB database.*

## Motivation

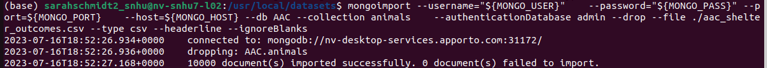
*The motivation behind creating the AACDatabaseLayer Python module was to streamline the process of database interaction for our upcoming project where we will focus on web-based applications. MongoDB is a popular NoSQL database, and by providing a simple and efficient class, developers can focus more on building the application's features rather than handling database operations.*

## Getting Started

*Make sure you have access to your database in the Mongo shell. Next, you will want to make sure that your computer has pymongo installed. If it doesn’t you can install Python on their original website* [*https://www.python.org/*](https://www.python.org/)*.*

*Next, you need to have a running MongoDB server running. If you don’t have MongoDB installed, you can also use* [*https://mongodb.com*](https://mongodb.com) *to install it on your device.*

*You’ll want to make sure that you have authentication set up for your MongoDB server.*

1. *Create a database in Mongodb and call it AAC.*
2. *In your terminal's admin section, you need to give this database a read/write privileges and set up authentication for the user to use in the database.*
3. *Next you will import the AAC base.* 
   1. *Example of importing in Mongo*
      1. *Make sure you are in the Linux terminal, and not mongosh*
      2. *cd /usr/local/datasets*

## Installation

*This Module requires the following tools:*

* *Python (version 3.9)*
* *PyMongo library*
* *Jupyter Notebook*
* *Southern New Hampshire class CS340 - Apporto*

## Usage

*Below you will see examples of screenshots of sample code, tests, and how they ran to demonstrate how to use the AAC database module for Create (C) and Read (R) operations.*

### Code Example

*from pymongo import MongoClient*

*from bson.objectid import ObjectId*

*# Class definition, accessing & connecting to mongo database for a web-based app.*

*class AACDatabaseLayer(object):*

*# Create Constructor*

*def \_\_init\_\_(self, HOST, PORT, username, password) :*

*self.host = HOST*

*self.port = PORT*

*self.username = username*

*self.password = password*

*# Store all credentials in a single URI*

*self.uri = 'mongodb://' + username + ':' + password + '@' + HOST + ':' + PORT*

*print ('uri: ' + self.uri)*

*# Connect to DB*

*def connect (self, logging=False) :*

*self.connection = MongoClient(self.uri)*

*if logging:*

*#show the db*

*print(self.connection.list\_database\_names())*

*# Need to set current MONGO DB for query options (mongodb use command)*

*# Parameters specifiy the db to use as a string:*

*def setDatabase(self, database) :*

*self.db = self.connection[database]*

*# Complete Create method to implement the C in CRUD*

*def create(self, collection, data) :*

*if data is not None:*

*# Data should be in dictionary*

*insert\_dictionary = self.db[collection].insert\_one(data)*

*if insert\_dictionary != 0:*

*return True*

*else:*

*return False*

*else:*

*raise Exception("Nothing to save, because data parameter is empty.")*

*# Implement the read capability method to implement the R in CRUD*

*# Like mongodb find/findOne()*

*# Specify collection as a string and specify the filter as a .py dictionary*

*def read(self, collection, filter) :*

*#Get collection*

*c = self.db[collection]*

*#find function*

*return c.find()*import sys, os, pprint

sys.path.append(os.path.expanduser('~/Desktop'))

from MyClassFile import AACDatabaseLayer

# Environment Variables

mPort = os.environ["MONGO\_PORT"]

mUser = os.environ["MONGO\_USER"]

mPass = os.environ["MONGO\_PASS"]

mHost = os.environ["MONGO\_HOST"]

# Verify

print(mPort, mUser, mPass, mHost)

# Set up connection

myConnection = AACDatabaseLayer(mHost, mPort, mUser, mPass)

# Connect to mongodb

myConnection.connect(logging=True)

### Tests

*Read (R) in CRUD Test*

*-----------------------------------------------------------------*

*myConnection.setDatabase('AAC')*

*# Test reading*

*cursor = myConnection.read('animals', {})*

*print(cursor)*

*for index, doc in enumerate(cursor):*

*print('\nDoc ' + str(index) + ':')*

*for prop in doc:*

*pprint.pprint(str(prop) + ": " + str(doc[prop]))*

*-----------------------------------------------------------------*

Create (C) in CRUD Test

*-----------------------------------------------------------------*

# Test create method

createData = {

"rec\_num": 1,

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

"animal\_id": "A746896",

"animal\_type": "Dog",

"breed":"Labrador",

"color":"Cream",

"date\_of\_birth": "10-11-2015" ,

"datetime": "2017-04-11 09:00:02" ,

"monthyear": "017-04-11T09:00:50" ,

"outcome\_subtype": "SCRP" ,

"outcome\_type": "Transfer" ,

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

"location\_lat": "30.5066578739495" ,

"location\_long": "30.5066578739698" ,

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

}

test = myConnection.create('animals', createData)

print("Successfully created new data:", test)

### Screenshots

A screenshot of a computer program

Description automatically generated

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Contact

Sarah Schmidt