# CS 340 README Template

## About the Project/Project Title

In this project we will be developing a web application that connects a client-side user interface (such as a dashboard) to a database. Creating a database and establishing successful CRUD routines in Python for MongoDB

## Motivation

The motivation behind this project is to create, read, update, and delete (CRUD) to understand and benefit from any database system. Creating a database of document collections and performing basic reading and querying to retrieve specific documents.

## Getting Started

To get started with using MongoDB:

1. Log into MongoDB
2. Set up a MongoDB environment by creating a directory
3. Run the Mongoshell
4. Create a new database and collection

When logging in to MongoDB use the username and password that you created, also the port number will be specific to your session.

A screenshot of a computer

Description automatically generated with medium confidence

A screen shot of a computer code

Description automatically generated with low confidence

The Python Code also implements the C and R in Crud. C will create data and a document and return true if successful. To read data you will be able to convert objects to a list and return the list of documents searched for.

## Installation

A list of tools needed to use the software:

1. **Jupyter Notebook:** For new users, it is recommended to install Anaconda. Anaconda conveniently installs Python, the Jupyter Notebook, and other commonly used packages. Here is a link to getting started [Installing the classic Jupyter Notebook interface — Jupyter Documentation 4.1.1 alpha documentation](https://docs.jupyter.org/en/latest/install/notebook-classic.html)
2. **Python:** Visit the official python website to get started with download Python. [Download Python | Python.org](https://www.python.org/downloads/)
3. **MongoDB:** Visit the official MongoDB website to get started with downloading for use. [MongoDB Atlas: Cloud Document Database | MongoDB](https://www.mongodb.com/cloud/atlas/lp/try4?utm_source=google&utm_campaign=search_gs_pl_evergreen_atlas_general-phrase_prosp-brand_gic-null_ww-multi_ps-all_desktop_eng_lead&utm_term=mongodb&utm_medium=cpc_paid_search&utm_ad=p&utm_ad_campaign_id=11295578158&adgroup=116363205048&cq_cmp=11295578158&gad=1&gclid=CjwKCAjwscGjBhAXEiwAswQqNAilxVNFIesinmTwgZ0rlWPkUtbUw3-__jlwrHSqlndvWIlicNE7mxoCSO0QAvD_BwE)

## Usage

### Code Example

### # -\*- coding: utf-8 -\*-

### """

### Spyder Editor

### This is a temporary script file.

### """

### from pymongo import MongoClient

### from bson.objectid import ObjectId

### class AnimalShelter(object):

### """ CRUD operations for Animal collection in MongoDB """

### def \_\_init\_\_(self, USER, PASS):

### # Initializing the MongoClient. This helps to

### # access the MongoDB databases and collections.

### # This is hard-wired to use the aac database, the

### # animals collection, and the aac user.

### # Definitions of the connection string variables are

### # unique to the individual Apporto environment.

### #

### # You must edit the connection variables below to reflect

### # your own instance of MongoDB!

### #

### # Connection Variables

### #

### #USER = 'aacuser'

### #PASS = 'SNHU1234'

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

### PORT = 31135

### DB = 'AAC'

### COL = 'animals'

### #

### # Initialize Connection

### #

### self.client = MongoClient('mongodb://%s:%s@%s:%d' % (USER,PASS,HOST,PORT))

### self.database = self.client['%s' % (DB)]

### self.collection = self.database['%s' % (COL)]

### print ("Connection Successful")

### # Complete this create method to implement the C in CRUD.

### def create(self, data):

### if data is not None:

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

### return True

### else:

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

### # Create method to implement the R in CRUD.

### def read(self, data):

### cursor = self.collection.find(data)#finds data

### result = list(cursor) #converts the cursor object to a list

### return result #returns the list of documents

### # Create method to implement the U in CRUD

### def update(self, query, data):

### if query and data: #checking if query and data

### result = self.collection.update\_many(query, {"$set": data})#update documents that match the query providing the correct date

### return result.modified\_count #returns the modified documents amount

### else:

### raise Exception("Invalid query or data")

### # Create method to implement the D in CRUD

### def delete(self, query):

### if query:

### result = self.collection.delete\_many(query) #deletes documents the match the query

### return result.deleted\_count #returns the amount of deleted documents

### else:

### raise Exception("Invalid query")

### Tests

**TESTING CREATE AND READING DOCUMENTS**

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

**UPDATING AND DELETING DOCUMENTS**

**A screenshot of a computer

Description automatically generated with medium confidence**

*A screenshot of a computer

Description automatically generated*

**EXECUTION OF WIDGETS, LOGO, AND WIDGETS**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**SUMMARIZATION**

This code sets up a dashboard using the Dash framework in Jupyter Notebook. The purpose of the dashboard is to display and interact with data from an animal shelter database. The code imports the necessary modules and libraries for building the dashboard and creates an instance of the AnimalShelter class using the provided username, password, and database connection details. It also retrieves data from the animal shelter database and stores it in a pandas DataFrame called df. The code configures the layout of the dashboard using HTML and Dash components, including an interactive filter, a DataTable to display the data, and placeholders for a graph and a map. The components I struggled with was getting my widgets and pie chart to update. I will continue to research the issue I have been having and update when I have fixed all the bugs.

## Contact

Your name: Eric Bryant