# CS 340 README

*Use this template to complete your README file. When completing the template, keep the headings as they are so that your document has a clear organization. Remove the italicized prompt text after you have completed each section for a polished final document.*

## About the Project/Project Title

The title of the project is Project One: CRUD Operations in Python. The project is to develop a portable Python module that enables the Create, Read, Update, and Delete functionalities to work towards data connection for a website application that connects a client-side user interface.

## Motivation

This project exists so it can make things convenient by connecting the user interface component to the database component to allow querying the Mongo database. The python script runs on the mongo server and carries out the CRUD operations.

## Getting Started

1. Initializing MondoDB:

Start by entering the following commands:

/usr/local/bin/mongod\_ctl start noauth

Mongoimport -port YOUR\_PORT -db YOUR\_DATABASE -collection YOUR\_COLLECTION ./filename

mongo

show dbs

use YOUR\_DATABASE

Once the database has been initialized you can then begin to create indexes, collections, or users.

1. Create Users

In the mongo shell switch to the admin db with the use command. You can then create an admin account with the db.createUser() command. This method will require that you create a name, password, and select a role for the user. With the admin account you can then create a user account called “aacuser” for the aac db.

I created an admin user called “adminlauren” that can read and write to all databases. I also created “aacuser” with the readWrite role which used pymongo to access the aac database.

Text

Description automatically generated

Text

Description automatically generated

1. Import the Austin Animal Center Outcome data set into the aac database:

Text

Description automatically generated

1. Lastly using the new user created, make changes using pymongo to the database with CRUD Operations.

I created a class called AnimalShelter that holds the CRUD operations and then a test script to call upon the functions that run those operations.

*Challenges:*

The one challenge I did face was continuously getting an error that stated OperationFailure: Authentication failed.

After several troubleshooting steps I ended up figuring out that my user I created was created for the aac database under the aac database and so I then created the same user under the admin database for the aac database and then was able to successfully authenticate.

## Installation

The tools used to complete this project were Jupyter notebook, Python (Pymongo), Plotly, Dash, Pandas, and MongoDB.

**Jupyter Notebook’s** installation guide can be found here: [Installing Jupyter](https://jupyter.org/install)

**Python’s** installation guide can be found here: [Python's Beginners Guide](https://wiki.python.org/moin/BeginnersGuide/Download)

**Plotly** is used to generate the charts needed for this project. It is used with Python and is imported into your Python module. A getting started guide can be found here: [Plotly: Getting Started](https://plotly.com/python/getting-started/)

**Dash** is a framework used to build the web application used in this project. You can use the following guide to import and install dash: [Dash Information](https://pypi.org/project/dash/) or [Dash for Python Documentation](https://dash.plotly.com/installation)

**Pandas** is a tool for Python as well that is used to create data frames. Information on importing and installing it can be found here: [Pandas Documentation](https://pandas.pydata.org/docs/getting_started/install.html)

**MongoDB** is a cross platform database program and documentation on installation and more information can be found here: [MongoDB Manual](https://www.mongodb.com/docs/manual/installation/)

## Usage

The project current has 3 main functions. The first one is a data table that offers interactive filter buttons to filter the Austin Animal Center Outcomes data set by “Water Rescue”, “Mount or Wilderness Rescue”, “Disaster Rescue or Individual Tracking”, or Unfiltered. The second function is a pie chart which dynamically responds to the filtering options. The third function is a geolocation chart (map) that also responds to the filtering options and displays the selected dog’s information.

The project can be used to query the MongoDB by creating data and reading data in with MongoDB.

To **create** data in the database I use the following code in the animal\_shelter.py file:

# Create method to implement the C in CRUD.

def create(self, data):

if data is not None:

if data:

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

return True

else:

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

return False

Then call upon this method in my script file as shown here:

# call the create method

creation = a.create(data)

print(creation)

To **read** data in the database use this section of code from my animal\_shelter.py file:

# Read method to implement the R in CRUD.

def read(self, search):

# Checks to see if the data is null or empty and returns exception in either case

if search is not None:

if search:

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

return searchResult

else:

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

Then call upon this method in my script to execute the read operation:

# call the read method

readone = a.read(search)

print(readone)

To **update** data in the database use this section of code from my animal\_shelter.py file:

# The method to implement the U in CRUD.

def update(self, data, newData):

if data:

results = self.database.animals.find\_one(data, newData)

if results is not None:

changeData = self.database.animals.update\_one(data, {'$set' : newData})

print("Successfully Updated to: ")

return newData

else:

changeData = self.database.animals.insert\_one(newData)

print("Successfully Added: ")

return changeData

else:

results = self.database.animals.find\_one({}, {"\_id": False, "name":1, "type":1})

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

Then call upon this method in my script file as shown here:

# call the update method

val = a.update(search, newVals)

print(val)

To **delete** data in the database use this section of code from my animal\_shelter.py file:

# The method to implement the D in CRUD.

def delete(self, data):

if data is not None:

print "deletion successful"

return self.database.animals.delete\_one(data)

else:

raise Exception("Nothing to delete, because remove parameter is empty")

Then call upon this method in my script file as shown here:

# call the delete method

success = a.delete(data\_1)

print(success)

print(success.deleted\_count, "deletion successful")

**Printing my Unique Identifier**

A picture containing graphical user interface

Description automatically generated

**Unfiltered Dashboard:**

A picture containing chart

Description automatically generated

**Water Rescue Dashboard:**

**A picture containing graphical user interface

Description automatically generated**

**Mountain Rescue Dashboard:**

**A picture containing chart

Description automatically generated**

**Disaster Rescue/Individual Tracking Dashboard:**

**A picture containing chart

Description automatically generated**

### Code Examples from Dashboard

Text, timeline

Description automatically generated

Text

Description automatically generated

Additionally, there are screen shot provided below that show the code I wrote in python for both the AnimalShelter Class and the test script.

### Tests

The test was executed in Jupyter notebook by importing the python class created above and creating, reading, updating, and deleting sample data.

### Screenshots

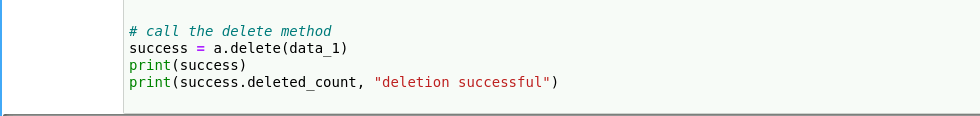
### Text Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated



**Screenshots of program output tests**

Table

Description automatically generated

Table

Description automatically generated

## Contact

Your name: Lauren Lindhurst