# CS 340 README

# Jason Gregg

## About the Project/Project Title

This project it used to create a Python module that enables the ability to create, read, update, and delete entries in a database. We are using a front end, Dash, to display the information from the database and filtering data.

## Motivation

This project will be used as a portable Python module that enables CRUD functionality that connects a client-side user interface to a database.

## Getting Started

To get started open your command prompt, linux, and navigate to where MongoDB is stored. Example cd usr/local/bin. After navigating to folder use the command line to start database. mongod\_ctl start for starting with authentication or mongod\_ctl start-noauth without authentication. Import your data set into MongoDB as a CSV file.

Import MongoClient by using the following:

*from pymongo import MongoClient*

Import ObjectID by usingthe following:

*from bson.objectid import ObjectID*

Create your class within Python that takes an object as argument and initialized your class. Create methods to create and read the data. An example of how to do this will be included within the usage section.

## Installation

Python will be needed and can be downloaded from the Python website.

MongoDB is needed and installation instructions can be found at mongodb.com

A code editor or Jupyter Notebook will be needed. The ability to import frameworks such as Dash, Plotly, Numpy, Pandas, etc, needs to be known.

## Usage

Pymongo was chosen for its ease of use and scalability instead of writing within the command prompt. With this code, you can use Pymongo to create, read, update, and delete items from your MongoDB database. Dash uses a model, view, controller architecture or MVC. The model, which interacts with the database, is accessed using the controller, code, and is seen through the view, UI. Dash uses @app.callback for defining controller methods and uses app.layout for structuring views. The following is a coding example used in this project to call different breeds of animals based on the users needs.

Text

Description automatically generated

### Code Example

The following are some coding examples:

*Function used to create an entry*

*def create(self, data):*

*if data is not None:*

*self.database.animals.insert(data)*

*return True*

*else:*

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

Function to read entries

def read\_all(self):

*cursor = self.database.animals.find()*

*return cursor*

Function to update entries

*def update(self, data, change):*

*if data is not None:*

*return self.database.animals.update(data,{"$set":change})*

*else:*

*print("Nothing to update because data parameter is empty")*

Function to delete entries

*def delete(self, data):*

*if data is not None:*

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

*else:*

*print('Nothing to delete, data parameter is empty")*

### Tests

The example includes an exception that will be triggered if there is no data to save. For testing purposes, self.client can use localhost to bypass needing a password. Self.client with username and password must be commented out. Also, in the following example, it can be coded when you call your class to print whether the data has been added or not:

*# import class*

*from animal\_shelter import AnimalShelter*

*# now need to create the object from the class*

*shelter = AnimalShelter()*

*data = {"age\_upon\_outcome":"1.7 years", "animal\_type":"Cat"}*

*if shelter.create(data):*

*print("animal added")*

*else:*

*print("data not added")*

After creating the front end, check to make sure the database populates and that each button has the ability to filter the data.

### Screenshots

Accessing database linux commands.

Text

Description automatically generated In the following screenshot, you can see how to import MongoClient by using “from pymongo import MongoClient”. Also import Binary JSON(BSON) by using “from bson.objectid import ObjectId” which provides additional data types and ordered fields.

You can also see user authentication in the next two screenshots by using line 11, self.client = MongoClient with your local host and port number that takes username and password parameter. The following image shows how to call that with AnimalShelter(username, password).

*Graphical user interface, text, application, email

Description automatically generated*

*Graphical user interface, text, application, email

Description automatically generated*

The last few images show how to create methods to create, read, update, and delete items from the MongoDb database and how to call those functions to be used.

## Contact

A picture containing table

Description automatically generated

Text

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated

Some example UI screenshots of how the program should.

A picture containing logo

Description automatically generated

A picture containing chart

Description automatically generated

The radio buttons can be used to filter data

Graphical user interface, application

Description automatically generated Chart

Description automatically generated A picture containing chart

Description automatically generated

Your name: Jason Gregg