# CS 340 README

## About the Grazioso Salvare Project

The Grazioso Salvare project is to create a Python module that connects a user interface to a database that helps store the outcomes of the Austin Animal Center.

## Motivation

The client started the project to create a record of the animals going through the shelter and allow users access to create, read, update, and delete entries as needed. To connect a user interface to a database with these options of data access, we needed to create a Python module to create this connection to allow these operations to happen.

## Getting Started

To be able to utilize this Python module, a database using MongoDB must be set up with a root admin user account and a user account that has access to the desired database for authentication. Once that is established, the Python module and the .ipynb files can be downloaded (mentioned below). The python driver of choice is PyMongo due to it being considered by MongoDB as the standard Python driver database operations such as all the CRUD operations that we will be needing. To start, the HOST, PORT, DB, and COL variables will need to be changed to their respective values for where the database will be hosted and what it is named as well as the collection that will be accessed. The create and read functions were designed to either create or read documents for/from the database using the given data value when either function is called. This passed value must be a dictionary type to ensure it is formatted properly for MongoDB to use and complete the relevant action. When in use the create function can insert documents into the selected collection with the provided data in the dictionary and the read function is able to find and list all documents with fields that match the key-value pair given. To use this functionality with a more user-friendly interaction, the python Dash framework is used to make the creation of an interactive dashboard (frontend) easier to work with our Python module (middleware) which will end up connecting to and interacting with our database (backend) to complete our desired operations.

## Installation

* Python IDE of your choice: This is needed to easily create/edit the CRUD functionality.
* Jupyter Notebook: This will allow you to run .ipynb files for testing code.
  + Install: [Jupyter Notebook](https://jupyter.org/install#jupyter-notebook)
* Sample Python and .ipynb files: <no link for download yet>
* Dash: To create an interactive webpage using dashboard widgets with Plotly
  + Install: [Dash](https://dash.plotly.com/installation)

## Importing CSV

A computer screen shot of a program

Description automatically generated

**User Account**

* **Create User Account**



* **Verify Account Creation**

A screenshot of a computer program

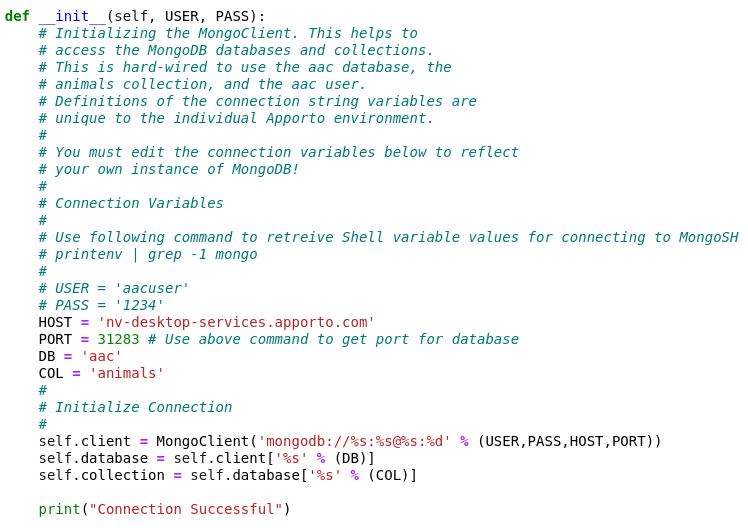
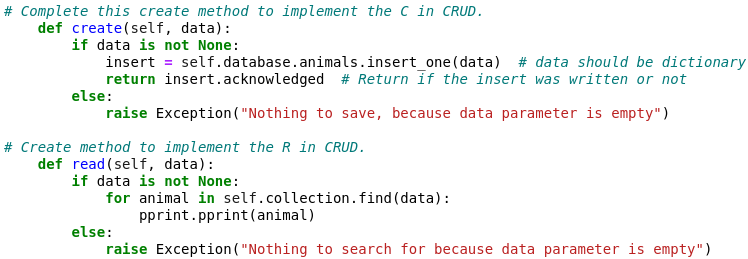
Description automatically generated

## Usage

### Code Example

Shown below (left) is the init function that is called when creating an AnimalShelter object. In this function we have hard-coded the HOST, PORT, DB, and COL values. These values hold where the host is, the port it’s on, the database to be accessed, and the collection to further access respectively. After that we intialize the connection with the database using the previous variables as well as the ones passed during the object’s creation which are the username and password to use for connecting to the database. If this connection is successful, a message is displayed in the console.

In the class are 2 functions (right), the create and read function. When called, the create function will use the database variable created in the \_\_init\_\_ function and insert a document into the data base using the data passed in the function call. Lastly a value of true or false is returned if the insertion was successful. In the read function, a for loop loops through the database looking for any document that has a matching field as the one passed when the function is called. For every document found to match the given argument, it is printed to the console.



**Tests**

* **Create & Read Functions**

To utilize the module above and run it with the database, you can use the testScript.ipynb file which imports the AnimalShelter class from animalShelter. The current setup which is shown in the following image shows an AnimalShelter object being created called CRUD with the login parameters for the database being passed as well. Following that the create function is called which creates a new animal document in the database which returns true if successful, then to test the read function we call the read function with a query for an animal with a breed of type “German Shepherd” and will return a list of all animals that are of breed “German Shepherd”.

A screenshot of a computer program

Description automatically generated

* **Update Function**

In the following image, we show how you can test the functionality of the update function and give an example of how it can be used. As in the last example, you must import the class containing our CRUD functionality as well as pprint to print things in a nicer format. To start, you must have the document you would like to update in the database, so in this instance I create a variable called create and using the create method mentioned above I create a document with an *animal\_id* of “TEST1”, and the *animal\_type* as “TEST1” as well. I then call the pprint on the create variable to print whether the creation completed or not. After creating the document, we can now update it using the update function. I first call the update function and provide it with 2 dictionary arguments, the first is the field query for what documents we want to update, and the second is the field we want to set and to what value. As shown, once the function is run it will tell you how many documents were updated based on the values provided which in this instance, is 1.

A screenshot of a computer code

Description automatically generated

* **Delete Function**

In the following image, we show how you can test the functionality of the delete function and give an example of how it can be used. Like the previous examples, you must import the class containing our CRUD functionality as well as pprint to print things in a nicer format. To start, you must have the document you would like to remove in the database, so in this instance I create a variable called create and using the create method mentioned above I create a document like before with an *animal\_id* of “TEST1”, and the *animal\_type* as “TEST1” as well. I then call the pprint on the create variable to print whether the creation completed or not. After creating the document, we can then delete it and any others by using the delete function. All you need to do is call the delete function on the collection where the document you would like to delete is, then pass a dictionary argument that holds the field data to query for and delete. Once called any documents that have matching fields will be deleted and a message will appear telling you how many documents were deleted.

A screenshot of a computer program

Description automatically generated

## Dashboard Functionality

In the following 4 images the CRUD functionality of the Python module is used in conjunction with the Dash framework to show the user their selected filtering option (query) in a more user-friendly way utilizing 3 different dashboard widgets. The selected widgets for this dashboard were the data table , pie chart, and a map. Each of these works together to show the selected data in different ways based on the selection made in the filtering options dropdown. When “ALL” is selected in the dropdown, the data from the database is shown non-filtered (Image 1). The user will see what percentage of a certain breed makes up the whole database. However, for the pie chart, if a breed’s overall percentage is too small, it will not show in the pie chart due to the number of total breeds. Using the dropdown menu, you can select 3 other options to filter out the data based on the type of rescue you are wanting to filter by. The three other options are as follows: “Water Rescue” (Image 2), “Mountain or Wilderness Rescue” (Image 3), and “Disaster Rescue or Individual Tracking” (Image 4). As you select each dropdown option, you will notice all 3 dashboard widgets will update based on what filter is selected and what dogs match that filter.

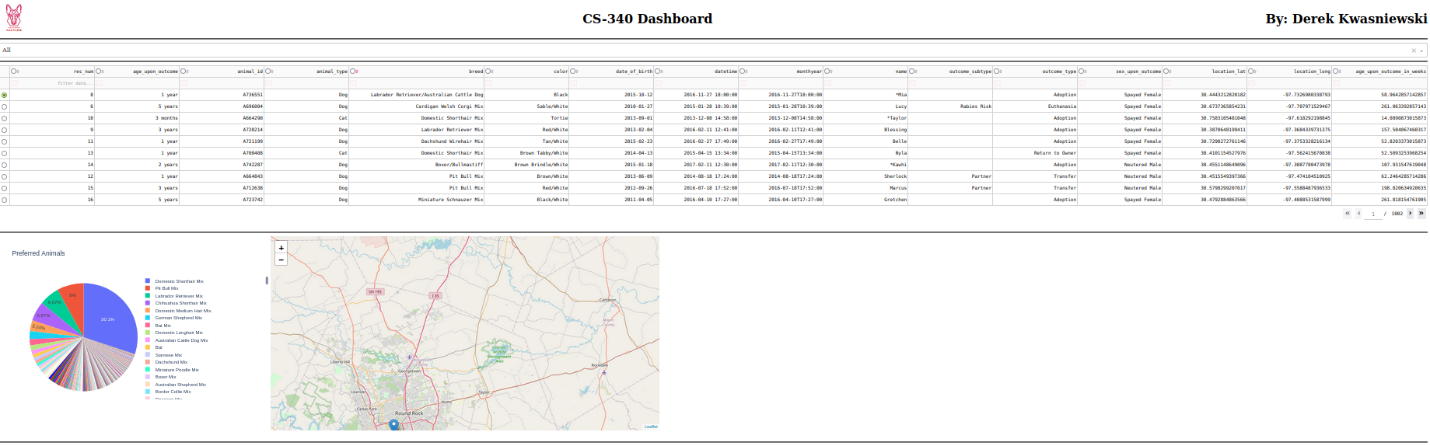
Image 1

Image 2

A screenshot of a computer screen

Description automatically generated

Image 3

A screenshot of a computer

Description automatically generated

Image 4

A screenshot of a computer

Description automatically generated

## Roadmap/Features

Further things to be addressed in the future will be updating the format of the site so that the dashboard widgets are more dynamic in their sizing so that they are not a set size. This will also include some functionality of a dynamic number of animals shown in the data table list so that on bigger screens more rows of animals can be shown at a time, and less if the screen or window is smaller.

## Contact

Your name: Derek Kwasniewski