# CS 340 - *AnimalShelter* README FILE

## About the Project/Project Title

*The AnimalShelter project provides the developer with the opportunity to easily connect to the ‘AAC’ database in MongoDB and seamlessly perform Create, Read, Update, and Delete (CRUD) operations without concerns about the connectivity process and the internal implementation of the methods in Python. This project serves as the middleware layer between the MongoDB database, where the documents for the ‘animals’ collection are stored, and the main application’s driver.*

## Motivation

*The purpose of this project is to save valuable development time by connecting your app’s front end to the database and ensuring that the end user utilizes proper authentication and authorization with the minimum permissions necessary to query the database safely and effectively.*

*A screenshot of a computer

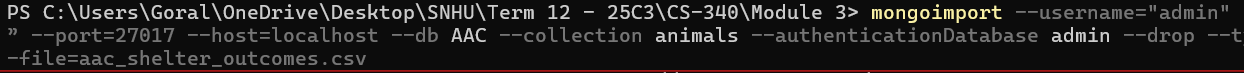
AI-generated content may be incorrect.*

*The project is still in its development phase. Therefore, additional methods for convenient use and effective queries will be implemented later. This code is offered as open source; thus, distribution, pull requests, and contributions are welcome. Despite the examples in this document being intended to be used with the ‘AAC’ database within the localhost environment, the setup can be easily modified to fit any similar database locally, or by deploying it into the Atlas environment by MongoDB.*

## Getting Started

*To get a local copy up and running, follow these simple example steps.*

*Firstly, you will need to set up your database locally. Assuming that you already have the MongoDB infrastructure on your machine installed, you can simply open your command prompt and run the setup database code and pass the attached CSV file from the* [*GitHub  repo:*](https://github.com/Dgoralkin/CS-340-Client-Server-Development/tree/79f708e86a0da74f734dc9c86c1d7c8ada20a7fb/Project%201/Project_1_files)

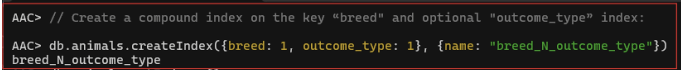
**

*This will create your own local database and load some cleaned data into it. Next, by creating a database user, we can grant him the appropriate level of permissions to the database by running:*

**

*The last line of code will create user access to the ‘AAC’ database, which will later authenticate the user and allow them to perform read, write, update, and delete operations within the database. This step is crucial as it prevents the risk of deleting the database or damaging its performance.*

*To achieve optimal performance, we should consider setting some indexes in the database to hone our queries on the appropriate data. Just run the following code, which will create a compound index in the database.*

**

*As we will see in the next output examples, our Create, Read, Update, and Delete functions will receive their commands from the readWrite user and search the database fields according to the indexed key values. This will ensure smooth operation and fast response times. The C, R, U, and D functions are constructed with efficiency and security in mind; these functions are expected to receive a BSON object file containing the desired key-value pairs to search from the collection. However, if an empty dictionary is passed to these functions, the exception handler will activate and prevent the crash of normal function operations for the Create and Read functions.*

A screen shot of a computer

AI-generated content may be incorrect.

## Installation

*List the tools you need to use the software and how to install them.*

*Are you still following? We are almost halfway through it already. Next, to set up the connection to the database from your Python code, feel free to download the distribution file from our* [*GitHub repo*](https://github.com/Dgoralkin/CS-340-Client-Server-Development/tree/79f708e86a0da74f734dc9c86c1d7c8ada20a7fb/Project%201/Project_1_files) *[](https://github.com/Dgoralkin/CS-340-Client-Server-Development/tree/79f708e86a0da74f734dc9c86c1d7c8ada20a7fb/Project%201) and follow the upcoming instructions to import the package into your development environment.*

*First, navigate to your Python project directory and run the following command in your command prompt to install the module in your project using the Package Installer for Python (PIP):*

**

*After successfully installing, you can expect to see the following message:*

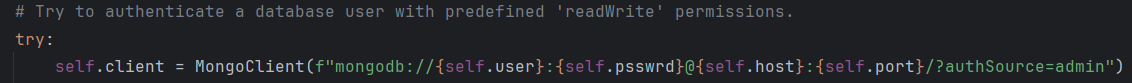
*A black background with white text

AI-generated content may be incorrect.*

*Now that your copy of the AnimalShelter class is ready to use, navigate to your Py IDE and import the new package into your code by including this import statement in the header, and the package will be ready for instantiation:*

**

*However, we must remember that in addition to our AnimalShelter package, which already supports and includes the MongoClient drivers from the pymongo library, a widely used NoSQL database library ideal for managing semi-structured data like BSON documents. By utilizing connection variables such as username, password, hostname, port number, and database and collection names, this library ensures we connect securely to the correct database:*

**

*Additionally, we will need to import the ObjectId from the bson library and the datetime function from the datetime package, which will aid in reading and converting the BSON format, as well as manipulating date and time objects. The OS library will assist in masking sensitive username and password variables from exposure (these variables should be defined in your system variables environment section).*

*A screen shot of a computer

AI-generated content may be incorrect.*

## Usage

*We can demonstrate the project’s functionalities and operations in several ways. One method is to instantiate the CRUD class in Python’s \_\_main\_\_ method by passing our user credentials to the AnimalShelter constructor and expecting to retrieve the class instance. Then, after the user has been successfully authorized and a connection has been established, we can run our Read, Create, Update, and Delete queries. Upon successful connection, we should expect to see the following prompt that confirms that ‘aacuser’ with readWrite permissions is connected to the ‘AAC’ database.*

**

### Code Example – C,R,U,D functions:

*A screen shot of a computer code

AI-generated content may be incorrect.*

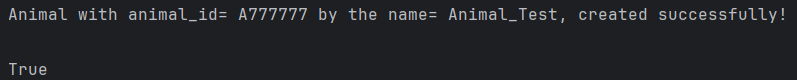
*The* ***Create*** *function is expecting to get a predefined dictionary that contains the key and values passed, such as:*  
new\_animal = {  
 '\_id': ObjectId('682d5fbc2a579703eb609fc4'),  
 'breed': 'Domestic Shorthair Mix',  
 'animal\_id': 'A777777',  
 'date\_of\_birth': '4/10/2014',  
 'datetime': datetime.now().strftime('%#m/%#d/%Y %#H:%M'),  
 'location\_lat': 49.809566429306486,  
 'name': 'Animal\_Test',  
 'outcome\_type': 'Transfer',  
}

*Now, we can call the Create function on our ‘aacuser’ instance and pass it the new animal object:*

*A computer screen shot of a black screen

AI-generated content may be incorrect.*

*The output of such a call will result in an output prompt and will return a Boolean value indicating whether the request was satisfied or not:*

**

*Not surprisingly, we can validate that the animal was indeed added to the database by querying it in the database directly:*

A screenshot of a computer program

AI-generated content may be incorrect.

*Similarly, the* ***Read*** *function will search the database fields based on the indexes created by the B-tree and will output a sample prompt in case at least one animal is found, returning a list of dictionaries for all animals found in the collection. For example, we can define the animal parameters that we would like to retrieve from the dataset as follows:  
# Define animal parameters to search from the 'animals' collection.  
find\_animal = {  
 'breed': 'Domestic Shorthair Mix',  
 'outcome\_type': 'Transfer'  
}*

*We will pass this argument to the Read function:*

A screen shot of a computer

AI-generated content may be incorrect.

*Which is defined as follows:*

A black screen with white text

AI-generated content may be incorrect.

*The output that we should expect to see in the case of an animal with 'Domestic Shorthair Mix' breed, and ‘outcome\_type’ as 'Transfer' was found would be:*

A black screen with white text

AI-generated content may be incorrect.

*The prompt displays the first document found on the screen and informs us of how many documents were found in total. The return from the function is a list of all documents found in a dictionary format.*

*The* ***Update*** *function will work in a similar way, with a little tweak that allows the user to choose between two update options by setting the optional argument to ‘update\_many=True’. Ignoring the optional argument will result in an update of the first found document in the collection, while passing the ‘update\_many=True’ argument to the function will update all found documents in the collection. The update function is defined as follows:*

*A screen shot of a computer

AI-generated content may be incorrect.*

*It is expected to get a data argument in the form of a predefined dictionary that will hold the filter parameters, e.g., which documents we would like to update as 'filter\_fields', and the 'update\_fields' dictionary that will hold the keys and values of the new entries. Therefore, we would first need to define the parameters that would be passed to the function as follows:*

*A screenshot of a computer program

AI-generated content may be incorrect.*

*And the call request for the function will look like:*

**

*As we can see, our Update function can hold one or several filter fields, as well as one or many update values. For example, this definition will update an animal record that contains a 'Domestic Shorthair Mix' breed, and ‘outcome\_type’ as 'Transfer', with breed=’Test Domestic Shorthair Mix', otcome\_type=’Test Transfer’, etc. As we can see and expect, the update function will return an integer value of the number of documents that were updated in the collection.*

*Lastly, our package’s* ***Delete*** *function is able to delete one or all of the documents in the collection controlled by the optional ‘update\_many’ argument and filter the documents in the collection by the predefined dictionary, such as delete\_animal = {  
 'filter\_fields': { 'name': 'Test Animal Name', 'outcome\_type': 'Transfer' }  
 }. In this example, we defined two filter values, where the function will accept one, many, or an empty {}. The definition of the function looks as follows:*

**

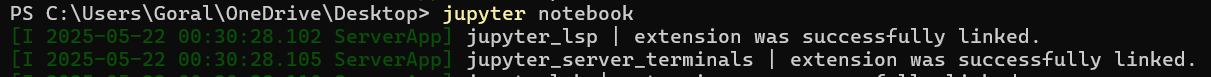
*The function also contains an internal message prompt mechanism that outputs the number of found and deleted documents from the collection, as well as an integer return value of the deleted documents.*

**

### Tests

*As mentioned above, the functions are prone to errors and have missing arguments. To evaluate the function’s performance, we will run our code from the Jupyter Notebook environment.*

*For example, we can run our server by calling Jupyter Notebook from our terminal:*

**

*A screenshot of a computer

AI-generated content may be incorrect.*

*After importing the required libraries, we can test the functionality of the* ***Create*** *function when passing it an empty dictionary, which would trigger an exception and return False to indicate that the request was denied:*

*A screenshot of a computer error

AI-generated content may be incorrect.*

*Similarly, we can observe the execution of the* ***Raed*** *function when calling it without passing a valid argument:*

*A screenshot of a computer code

AI-generated content may be incorrect.*

*As expected in these examples and more, the error handler will throw an error, and the function will return an empty list. Such scenarios and more are handled easily by the functions, and greater functionality is expected to emerge in the near future.*

*Next, we will test the functionality of the* ***Update*** *function in our Jupyter Notebook.*

*As mentioned above, we will first define our filter and update parameters in the dictionary format and pass them on to the execution of the animals\_user.update(update\_animal) function:*

*update\_animal = {*

*'filter\_fields': { 'breed': 'Domestic Shorthair Mix', 'outcome\_type': 'Transfer' },*

*'update\_fields': { 'breed': 'Test Domestic Shorthair Mix',*

*'outcome\_type': 'Test Transfer', 'name': 'Test Animal Name' }*

*}*

*A screenshot of a computer

AI-generated content may be incorrect.*

*As we can see, the result of this call without the optional parameter, the function found the first document with the defined parameters and updated it in the collection. Similarly, the ‘update\_many=True’ option resulted in an update of 1383 documents from the collection that comply with the defined filter requirements. \*Note: defining and passing the value of 'filter\_fields' key as an empty dict {} will result in an update of all documents in the collection to the defined 'update\_fields' values.*

*Testing the* ***Delete*** *function will be done in a similar manner to the Update function.  
We will first define the filter fields that we would like to delete as a single or multiple values specified in the ‘filter\_fields’ dictionary. For example, the following request will search for documents where the key ‘name’ is equal to the value ‘Test Animal Name’*

*A screenshot of a computer

AI-generated content may be incorrect.*

*After passing the parameters stored in ‘delete\_animal’ dict to our function, we can again omit or specify the optional ‘delete\_many=True’ argument, which will delete the first, or all found records in the collection, as shown in the following output, and will return the number of deleted documents:*

*A screenshot of a computer

AI-generated content may be incorrect.*

*Similar to the Update function, the Delete function will delete without confirmation all collection records if defining and passing the value of 'filter\_fields' key as an empty dict {}.*

## Roadmap/Features (Optional)

*As mentioned before, we are diligently working to enhance our library and its functionality. Therefore, we are developing the library to support the Update function and the Delete function, with the latter intended for use by the admin user.*

## Contact

Daniel Gorelkin  
[Daniel.Gorelkin@SNHU.edu](mailto:Daniel.Gorelkin@SNHU.edu)  
<https://github.com/Dgoralkin>