# CS 340 README Template

## About the Project/Project Title

This project is a CRUD Python Module that takes in the Austin Animal Shelter database of animals to return queries. Through Python and MongoDB, users can both create new database records and read current ones, update existing records to include new data, as well as delete existing records. Users can use the dash framework interface to interact with the database by using the radio items to organize the data into different categories. Depending on whether a dog is needed for water rescue, wilderness or mountain rescue, or disaster rescue and individual tracking, users can filter the results to see only dogs with matching specifications, as well as where they are located.

## Motivation

Our team was hired by a rescue-animal training company named Grazioso Salvare. Since it is a global company, they need a program that will be able to interact with their database and organize the data in a way that is beneficial to them. The python code will be able to keep up with their large amount of data in MongoDB and allow users to look through and use the database to help the animals find an appropriate match.

## Getting Started

To get started, you must first download the AAC database into MongoDB, as well as have an account that is allowed to access the database. You must assign a username and password to a specific user and assign only certain roles to their account. Once the user has been authenticated and authorized to access the data, they can use the python module to create, read, update, and delete records. For this project, RStudio and Jupyter Notebook were used to write and test the code. The CRUD module was coded into RStudio, and then uploaded into Jupyter Notebook where I also created tests to import the module and check its functionality. These tools were used to help ensure reusability and that all code works as it is expected to. You can use RStudio, or another Python IDE, to connect to MongoDB in order to access and manipulate the data. The dash framework was used to filter the results to match only the type of rescue dog needed, as well as show a pie chart and geolocation map of the types of dogs for each rescue operation. To recreate this project, you will need the following tools, as well as follow the same steps, provided in this document.

## Installation

You will need MongoDB. To use MongoDB to connect to the database, you must enter your authorized username and password, as well as the mongosh command. You can then use the python script in a python IDE to enter in the connection variables, such as username, password, host, and port number. Once these variables are coded, the Python shell can connect to the database in MongoDB, and it is ready to be used.

To download the MongoDB terminal, you may use this link: https://www.mongodb.com/docs/v4.4/mongo/

For this project, I am currently using the 6.0.13 version of MongoDB and 1.8.0 version of mongosh.

To download RStudio, you may use this link: https://posit.co/download/rstudio-desktop/

For this project, I am currently using the 4.2.2 version of RStudio.

To install Jupter Notebook, you may use the following link: <https://jupyter.org/install>

To use the dash framework for Python, you can enter the following command in your terminal:

pip install dash

## Usage

### Code Example

So far, this project allows users to create records to add to the AAC database, as well as read records that have already been added.

### Tests

*In the python module, there are create, read, update, and delete methods.*

*try:*

*result = self.collection.insert\_one(data) # data should be dict*

*return True if result.inserted\_id else False*

*except Exception as e:*

*print(f"Nothing to save, because data parameter is empty {e} ")*

*return False*

*This is the create method. It includes a try block that assigns results to the data that is being added to the animals collection. If the data was inserted as a record correctly, than the method returns true. If there was an error or an exception was raised, then the method returns false and prints and error message*

*If query is not None:*

*Result = self.database.animals.find(query)  
 for item in result:*

*Print(item)*

*Else:*

*Raise exception*

*This is the read method. It says that if there is a query that the user enters and it exists, then it is set to Result. For every item in result, it needs to be printed. If the query is not found in the database, an exception is raised. To test this, try using the read function to find a name key value pair. If the pair is found in the database, then the data is printed. Otherwise, an error message is sent.*

*def update(self, query, updateData):*

*try:*

*result = self.database.animals.update\_many(query, {'$set': updateData})*

*return result.modified\_count*

*except Exception as e:*

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

*return 0*

*This is the update method. It states that the result is assigned to the data that should be updated. The parameters include the existing query that is already in the database, and the updated data that should replace the old key-value pair. This can be applied to one or many queries. The method returns the number of results that were updates if successful, or returns 0 updated results and an error message if the parameters were invalid.*

*def delete(self, query):*

*try:*

*result = self.database.animals.delete\_many(query)*

*return result.deleted\_count*

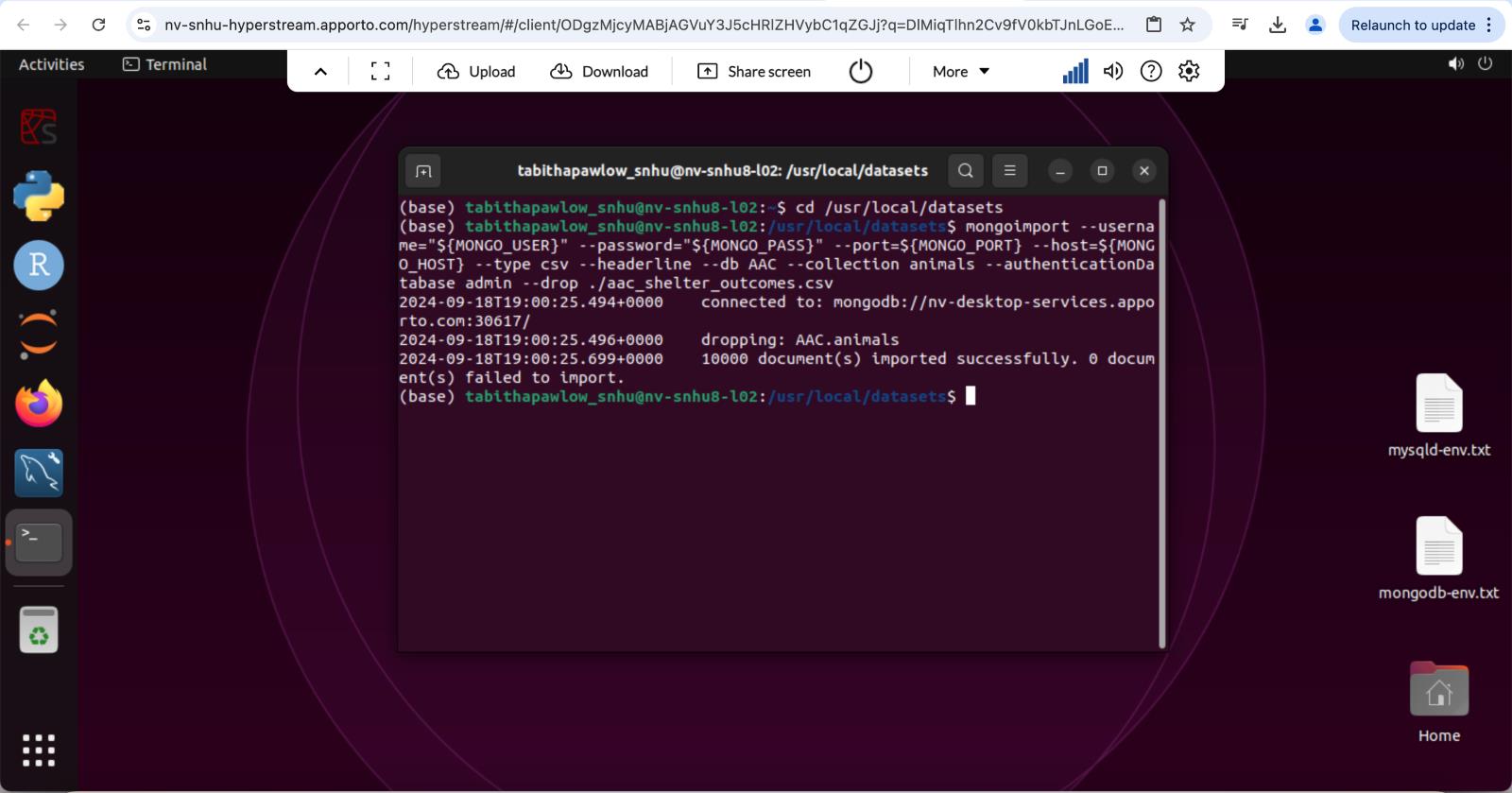
*except Exception as e:*

*print("Nothing to delete, because data parameter is invalid")*

*return 0*

*This is the delete method. It assigns result to the existing query that needs to be deleted. Users pass in this query object as the parameter and the method returns the number of records that matched and were successfully deleted. If the method was unable to delete the record, then 0 is returned and an error message is printed*

### Screenshots



*A screenshot of a computer

Description automatically generated*

A screenshot of a computer

Description automatically generated

*A screenshot of a computer

Description automatically generated*

## A screenshot of a computer Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Contact

Tabitha Pawlowski

Tabitha.pawlowski@snhu.edu