# CS 340 README – Ishmael Kwayisi

## About the Project/Project Title

“Animal Heroes” is an open-source full stack software project focused on storing information about pets within a database. Specifically, the software needs to provide accurate and efficient queries to satisfy the needs of the international rescue-animal training company, Grazioso Salvare. Grazioso Salvare is specifically looking for dogs who meet the requirements specific attributes that will confirm which ones are good candidates for training.

## Motivation

“Animal Heroes” was created with the intention of providing custom software to clients with database that can interact with client-side code. In doing so, the software will account for great usability, security, and reliability for clients to deliver software that can ease the process of categorizing dogs by their level of readiness for training.

## Getting Started

To start working with “Animal Heroes” you will need to login to your Apporto account. Next, launch the Terminal and use your credentials to gain authorization to the database and code. Afterward, open Jupyter Notebook in the case you are looking to make test cases. Lastly, run the file in Terminal through the command, “python3 driver”.

## Installation

* Apporto – A virtual lab that allows you to collaborate with Python and MongoDB. Installation is not necessary (only the creation of an account is required) since it is a desktop application. Also, Terminal and Jupyter Notebook are packaged with the virtual lab as well.
* MongoDB – A database utilized within this lab. However, it does not require installation. Though, to access it, enter: “mongosh” within the Terminal.

## CRUD Python Module Purpose

This software follows a CRUD (Create, Read, Update and Delete) framework to ensure the persistence and integrity of data entries within the database. In addition, the CRUD framework allows for fluent understanding of how data is maintained within the server. For instance, without a “Create” method, there is no viable way for a Read, Update, and Delete method to work because there is not an existing method to insert data to begin with.

## Using The CRUD Python Module

## The Python driver for Mongo was used for the intention of focusing on higher-level operations without having to know the details of the database with which they are collaborating. The purpose of utilizing this is to ensure accessibility to individuals of distinct levels who are looking to modify or make suggestions related to the Python driver for Mongo.

When it comes to the attributes of each CRUD function, each utilize the “self” variable to refer to the instance of the class being used. However, Create and Update functions require the use of a data variable to store/change information about a new/old entry in the database respectively. While, Read, Update, and Delete all use a filter\_query variable that enables the use of a query to specify what needs to be addressed within the database without applying it to every entry. As for the functionality of each CRUD operation:

* C (Create) – Allows you to enter new data entries and storing them into your database
* R (Read) – Retrieves data from the database and displays the information to the user
* U (Update) – Enables the ability to adjust pre-existing data from the database
* D (Delete) – Removes data existing entries the database

## Usage

### Code Example

#Creating a data entry to store in the MongoDB database

*def create(self, data):*

*if data is None:*

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

*if not isinstance(data, dict):*

*raise TypeError("Data must be in a dictionary") #Check that the input is a dictionary to avoid runtime errors*

*try:*

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

*return result.inserted\_id*

*except Exception as e:*

*raise Exception(f"Failed to insert data: {e}")*

### Tests

Testing code is done within .ipynb files through the Jupyter Notebook software. To run tests within this environment, you will need to import the python file(s) and write code in python with test cases that satisfy your goal. Utilizing a print statement at the end will indicate whether your test was successful. Then, to run the test you will need to debug any errors within the code and hit the run button in Jupyter Notebook.

Example Test Case for Create in CRUD:

sample\_data = {

    "name": "Spyro",

    "type": "Cat",

    "age": 5,

    "adopted": False

}

inserted\_id = shelter.create(sample\_data)

print("Inserted ID: ", inserted\_id)

**Required Functionality For Using Project**

To use this software, first you must run the code file “ProjectTwoDashboard.ipynb”, then click on the IP address that displayed at the end of the program. After clicking on the IP address, you will be prompted to enter a username and password before accessing the data table and its filtering options. Once this is complete, the data table will display for the client. The client has the option of selecting the radio item options (Water Rescue, Mountain Rescue, Disaster Rescue and Reset) to meet their desired objectives. Also, by selecting a radio item option, the pie chart associated with the rescue type will dynamically change to accurately represent the breeds for said rescue type. In addition, the user will need to select a radio item in the first column of the data table in respects to a row to be gain access to the geolocation of the animal being examined. Lastly, the user can use the previous and advance buttons to change the information they are viewing on the data table; along with having the ability to perform a search query within the second row of the data table inside the cell corresponding with its column.

**Tools Used**

* MongoDB – MongoDB was employed during the development of this project for an efficiently effective development process due to its qualities. One of the key qualities MongoDB possesses that makes it desirable when working with Python is the PyMongo library. This library allows for easy access and interaction to the MongoDB database from a Python driver file. PyMongo also allows for a user-friendly way to develop CRUD methods, queries and indexes. Another key aspect of MongoDB that made it beneficial for developing this project was that its seamless interactions between the Python code and the schema-less design structure of the data stored.
* Dash Framework – The Dash Framework works with Python code to develop and handle data for web applications. This framework is mostly utilized to handle the view and controller portions of the MVC design philosophy. How this works is that the Dash Framework contains html functionality to develop a visually appealing dashboard layout, which displays data from the model. While for the controller, the Dash Framework utilizes a function known as “callbacks” to handle user input to which trigger events in accordance with the input requests. The Dash Framework is beneficial for this project because it supports a strong structure and maintainability for the project. Along with having flexibility with reusing code for other aspects of the project.
* Other resources and software applications
  + MongoDB official website for references: [MongoDB: The World’s Leading Modern Database | MongoDB](https://www.mongodb.com/)
  + Plotly official website for references in developing pie chart: [Pie charts in Python](https://plotly.com/python/pie-charts/)
  + Mastering MongoDB .6x textbook for Python reference: [Mastering MongoDB 6.x : Expert Techniques to Run High-volume and Fault-tolerant Database Solutions Using MongoDB 6.x - EBSCO](https://research.ebsco.com/c/cyb354/ebook-viewer/epub/6rs46l7ncj/section/top?auth-callid=f50b620d-57ef-4d30-986f-93d7dbd07e1d)
  + W3 Schools for references in HTML syntax: [HTML Tutorial](https://www.w3schools.com/html/)
  + Pandas official website for references in querying: [pandas documentation — pandas 2.3.0 documentation](https://pandas.pydata.org/docs/)
  + Dash official website for references in querying: [Dash Documentation & User Guide | Plotly](https://dash.plotly.com/)

**Project Reproduction**

For reproducing this project, here is a series of steps to follow:

1. Import the Austing Animals Center dataset into MongoDB through the process of importing a csv file with the MongoDB import tool.
2. Next, create an account with a username and password to authenticate the database. This will be done in the terminal application.
3. Afterwards, use Jupyter Notebook to create a Python driver to create the CRUD functionality for interactions with the database.
4. After this, create a README document to record the various aspects of the project, such as its functionality, features, why it was made, etc. Continue to update this document whenever changes have been applied to your software.
5. Once this is done, being to make the view for the client. This view should include the title of the project, the author’s name, a unique identifier and prompt for the user to login with a username and password.
6. Along with this, you would need to develop a data table that represents the information in your data, with UI elements such as radio items or buttons for filtering the data. You also need to develop a geolocation chart and a graph of some kind (preferably a pie chart) to represent the location of the animal and provide representation of information that could be analyzed.
7. Lastly, finalize all the files and documents used and store them in a zipped file for submission to the client.

**Challenges**

During the development of this project, there were a few challenges that I faced that needed to be addressed for the completion of this project:

* Connection Refused Error - For the development for Assignment 5-2 and an extra feature for the final submission of Project Two, I worked with setting up user authentication to verify the user before they gain access information from the database. However, when I tried to run the program for both assignments at their respective periods, I ran into connection errors preventing access to the database despite entering the correct credentials I had previously established. This error took a few hours to solve for both assignments, but what I did to solve this was to run “mongosh” in my terminal to make sure it was working before I did anything else. And I found that for both assignments, “mongosh” was struggling to connect to the MongoDB database. This entitled me to restart Apporto twice, which did end up solving my problem for Project Two. Unfortunately, that didn’t work for Assignment 5-2, to which I decided to backtrack and go through the process of re-importing the database, but this time with my username and password. Afterwards, I ran the program again and it was able to display the information because at that point, the information was stored correctly into the database.
* Understanding Python and HTML syntax – Understanding Python and HTML syntax was a bit of challenge for me during the development of this project because I’ve barely touched Python prior to this project and the last time I had used HTML was several years ago. However, knowing that I needed to use the combination of both for a successful implementation of CRUD functionality and client-side UI, I decided to study and practice the syntax. For both, I utilized various resources for reference and learning. And when it came to working on assignments, I would use those opportunities to practice the syntax. Now by the end of this project, I can confirm that I am more comfortable using both programming languages for software development.
* Implementing the Pie Chart – Implementing the Pie Chart was important for the purpose of analyzing the breed types for the appropriate rescue situations. However, I ran into an issue where the chart wasn’t showing after the initial test of the “update\_graphs” function. After, reviewing my code, I found that I had been using a variable outside the function to retrieve the information for breeds. To solve this, I adjusted the placement of variables and utilized a new local variable that would retrieve the information I needed for retrieval.

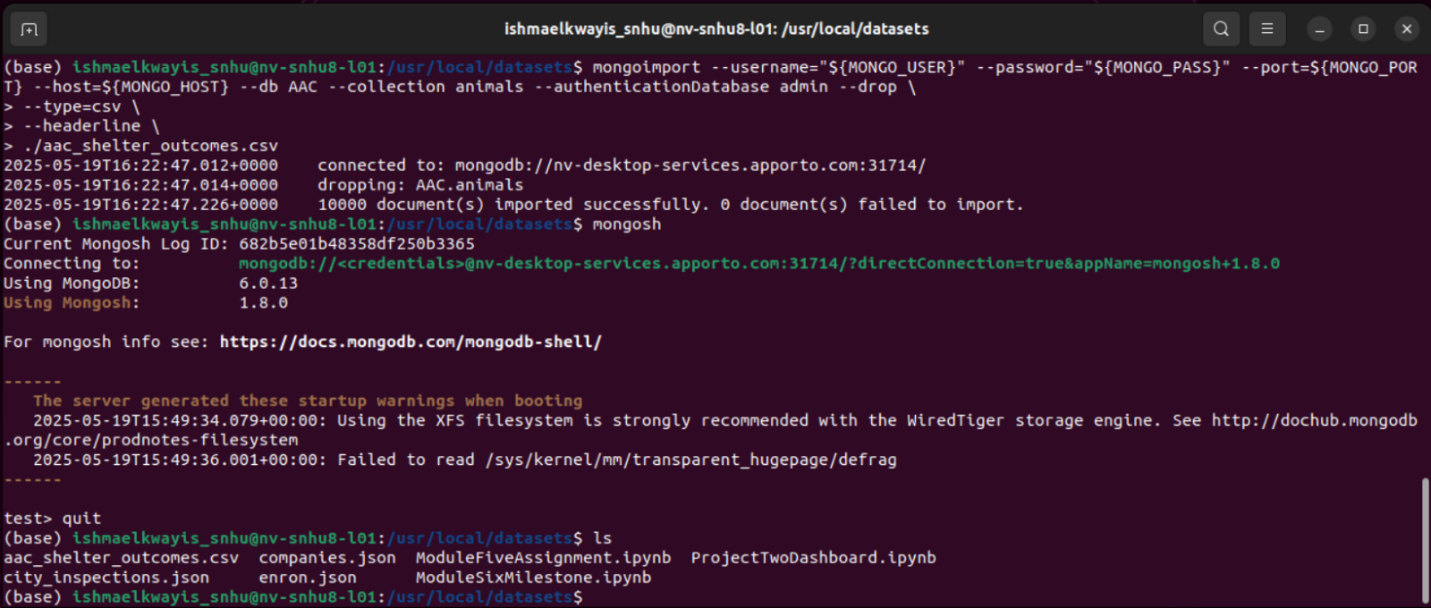
### Screenshots

***Example of the “create” method for the C in CRUD***

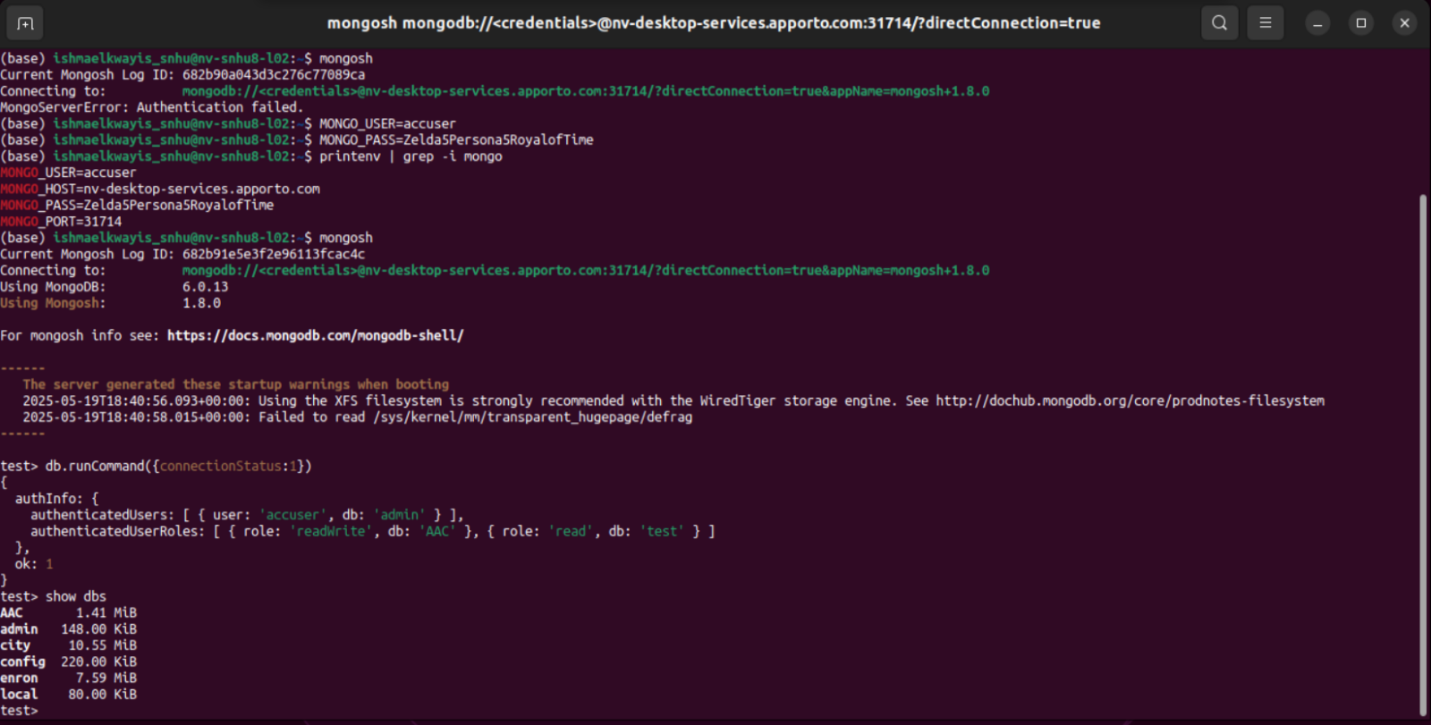
*A screenshot of a computer

AI-generated content may be incorrect.*

***MongoDB Import Command and Execution***

**

***User Authentication Execution***

**

***CRUD Funcionality\_1 (Create)***

*A screenshot of a computer

AI-generated content may be incorrect.*

***CRUD Funcionality\_2 (Read, Update, Delete)***

A screenshot of a computer

AI-generated content may be incorrect.

## *Starting State 1*

## *Picture*

## 

## 

## 

## 

## 

## 

## 

## 

## *Starting State 2*

## *PicturePicture*

## 

## 

## 

## *Water Rescue*

## *PicturePicture*

## 

## 

## *Mountain or Wilderness*

## *PicturePicture*

## 

## 

## *Disaster Tracking*

## *PicturePicture*

## 

## 

## *Reset*

## *Picture*

## Picture

## Contact

Your name: Ishmael Kwayisi