# CS 340 README

## About the Project/Project Title

*Global Rain has been contracted to create a software application for, Grazioso Salvare, an international rescue-animal training company. Using the software, they will be able to identify dogs suitable for search-and-rescue training. The software will allow them to create, read, update, and delete any of the dogs in the database. As the lead developer, I will develop all the CRUD functions and test cases to ensure they work as expected.*

## Motivation

*This project is designed to test our ability to create an application that allows users* *to create, update, read, and delete from a NoSQL database. The underlying software will be generic enough so that the functions of CRUD can be reused in other applications.*

## Getting Started

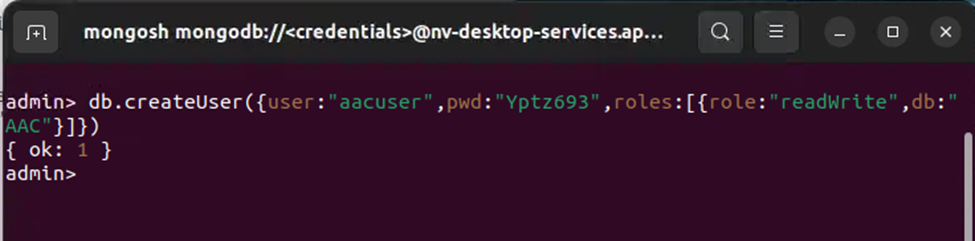
1. *First you will need an* *Austin Animal Center database. If you do not have one follow the steps in the* [*4-1 Milestone: Create and Read in Python*](https://learn.snhu.edu/d2l/le/content/1698594/viewContent/35438191/View) *rubric to obtain one.*

*Your import should look like the following:*

A computer screen with white and blue text

Description automatically generated

1. **Create a new user account called “aacuser” for the database AAC in the mongo shell. Refer to steps 6–7 of the MongoDB Manual Enable Access Control tutorial for help with this task. This is necessary to restrict access to only the AAC database.**

****

1. *Place the AnimalShelterDriver.py in a folder*
2. *Start Jupyter and create a notebook in the same folder you placed the AnimalShelterDriver.py file in.*
3. *You can see example usage of the functions by looking at the main.py or the Jupyter.ipynd file. In those files, there are examples of.*
   1. *database\_read() Allows you to search for a record with exception handling*
   2. *database\_create(): Allows you to insert a record with exception handling*

## Installation

*This application was developed with the following software packages and the minimum version recommended.*

1. *Linux OS (Ubuntu Ver. 8.32-4) – This should be installed by a knowledgeable Systems Administrator*
   * *Security is one of Linux's strongest features and it is open source making it free to use. It also has several applications and programming languages that make it ideal for a developer. Additionally, Linux is very stable, requires little disk space, and is very powerful operating system.*
2. *Spyder (Ver 5.1.5) – Installation:* [*https://docs.spyder-ide.org/current/installation.html*](https://docs.spyder-ide.org/current/installation.html)
   * *It lets you run Python code interactively and is like a traditional development environment. Including a dedicated code editor with keyword color coding and debugging tools.*
3. *Jupyter (Ver 6.4.8)* *– Installation:* [*https://jupyter.org/install*](https://jupyter.org/install)

* *Has an Interactive user interface. Allows for Live code execution and integration with several programming languages. It also allows for visualization of data and results.*

1. *Python (Ver. 3.9.12)* - *Installation:* [*https://docs.python-guide.org/starting/install3/linux/*](https://docs.python-guide.org/starting/install3/linux/)
   * *Python is commonly used for developing websites, and software, automating tasks, analyzing data, and visualizing it. Additionally, there is a wide range of add-on packages that expand the functionality of Python.*
2. *Mongodb (Ver 6.0.13)- Installation:* [*https://www.mongodb.com/docs/manual/installation/*](https://www.mongodb.com/docs/manual/installation/)

* *MongoDB makes it easy for developers to store structured or unstructured data. It has a scalable architecture that allows developers to create scalable applications with evolving data schemas.*

*Note: You may need System Administrator privileges to install the software*

## Usage

*To use the example code, first, open the Jupyter\_mod4.ipynb notebook. Then run the main cell, from this point, you can use the database\_creade() and database\_read() functions in individual cells.*

1. *database\_creade(): Will return a True if the operation was successful or False if it failed.*
2. *database\_read(): Will return a list of objects matching the criteria or an empty set if none were found.*

### Code Example

*In the main.py and Jupyter\_mod4.ipynb files, two functions show how to use the create and read functions.*

***Reading from a database:***

A computer screen shot of text

Description automatically generated

**Creating a document in the database:**

*A screenshot of a computer code

Description automatically generated*

**Updating a document in the database**

**A screen shot of a computer code

Description automatically generated**

**Creating a document in the database**

A computer screen shot of a computer code

Description automatically generated

**Tests**

### *Creating a database entry:*

* Create an entry for the database called animal (shown below)
* Followed by the **database\_create(animal)** function
* Execute the cell

1. If everything was done right, you should see a message stating that an entry was created.

### *A screenshot of a computer code Description automatically generated*

1. If an invalid parameter is passed, an exception will be thrown, and an error message will be displayed.

A white text box with black text

Description automatically generated

### *Reading from the database:*

* Enter into a cell **database\_read({“name”:”Sunshine”})**
* If you used a different name for the creation example, replace “Sunshine” with that name.
* Execute the cell

1. If everything was done right, the record(s) will be displayed.

### A screenshot of a computer code Description automatically generated

1. If an invalid parameter is passed, an exception will be thrown and an error message will be displayed

A white text box with black text

Description automatically generated

1. ***Updating a record(s) in the database***

### *If you have not created a new entry from the “Creating a database entry:” section, do so now.*

1. Enter into a cell:
   * database\_create(animal) # Valid entry
   * database\_read({"name":"Sunshine"}) # Valid query
   * database\_update({"animal\_id": "A9675231"},{"age\_upon\_outcome": "7 years"})
   * database\_read({"name":"Sunshine"}) # Valid query
2. If everything was done right, the record(s) will be displayed.

**A screenshot of a computer code

Description automatically generated**

1. If an invalid parameter is passed, an exception will be thrown, and an error message will be displayed



A black and white text

Description automatically generated

1. ***Deleting a record(s) in the database***

### *If you have not created a new entry from the “Creating a database entry:” section, do so now.*

1. Enter into a cell:

* **database\_read({"name":"Sunshine"}) # Valid query**
* **database\_delete({"name":"Sunshine"}) # Valid query**
* **database\_read({"name":"Sunshine"}) # Valid query**

1. If everything was done right, the record(s) will be displayed.

**A group of text on a white background

Description automatically generated**

**A close-up of a computer code

Description automatically generated**

1. If an invalid parameter is passed, an exception will be thrown, and an error message will be displayed

****

****

### 

### Dashboard Overview

*A screenshot of a computer

Description automatically generated*

## Features

**Tabs**: The Tab keys allow for the filter of only Cats or Dogs to populate the data table

**Data Columns:** The top row of each column can be selected and the table will be sorted on that field.   
 The row underneath can screen the table further by entering in the term to match.

**Data Row:** Each of the data rows can be selected one a time and, in the Geolocation map a marker will   
 be placed at the coordinates for that animal.

**Paginate:** The number of pages in the data table is located in the lower right corner. It allows you to   
 move forward or backwards in the table. You can also enter in a desired page number and   
 “jump” to that page.

**Type of Service Animals**: By selecting the type of service animal the data table can be filtered more.  
 the options are:  
 a) Water Rescue  
 b) Mountain/Wilderness Rescue  
 c) Disaster Rescue and Individual Tracking  
 d) Reset – which puts the data table in the default state

**Chart View**: The default chart is a Histogram, that can be switched to a Pie chart by selecting the radio   
 button next to “Pie Chart”. It can then be switched back to the Histogram by selecting the   
 radio button. The charts are a representation of the data in the Data Table. So, if you   
 use any of the filter options, the graphs will reflect that change.

## Example Usage

|  |
| --- |
| **Water Rescue** |
| A screenshot of a computer  Description automatically generated |
|  |
|  |
| **Mountain/Wilderness Rescue** |
| A screenshot of a computer  Description automatically generated |
|  |
|  |
| **Disaster Rescue and Individual Tracking** |
| A screenshot of a computer  Description automatically generated |
|  |
| **Reset** |
| A screenshot of a computer  Description automatically generated |

**Explain the steps that were taken to complete the project**

This application was developed using the Model-View-Controller (MVC) architectural pattern. Using the MVC, developers can focus on specific aspects of the application by dividing the application into three interconnected components - Model, View, and Controller.

The first step was to import the data stored in a JSON file into the Mongo database. The next step was to verify that the data was in the Mongo database by using the Mongo shell and performing read, update, create, and delete operations.

The next step was to create a Python driver that connects to the underlying database. The interface is generic enough so that it can be used in other applications. With this driver, you can create, read, update, and delete database items. Together they form the CRUD driver.

The final step was to create a web interface that presents the data in an easy-to-access form. The webpage presents the data in a data table that also has a histogram or a Pie that represents the data in a graphic.

**Identify any challenges that were encountered and explain how those challenges were overcome.**

I did not experience any real challenges, two aspects that I did spend more time on were replacing the mechanism that controlled the tab functions. The starter code was based on the number of “clicks” one button had over the other. This leads to many problems, for instance, if you were to click the “Cats” button 12 times, you would then have to click “Dogs” 13 times for it to register. I changed this so that it reads the name of the button pressed to determine which button was selected.

The other was getting the “Buttons” and “Radio buttons” to work together. The issue was that it used the same callback function. The solution was pretty simple since I was already using names to determine which “button” was pressed I just expanded this to check if the named radio button was selected in the same call-back function.

## Roadmap/Features

## Contact

David S. Harmor