# CS 340 README CRUD Module 7

# 

## Project Description and Rationale



**Grazioso Salvare Database Dashboard**

**Website Design by Christopher Clark**

This project is designed based on the requirements documents provided by the client, Gazioso Salvare. The intent of this project is to provide a searchable database of shelter animals to facilitate the client’s ability to provide animals best suited to the needs of their customers. There is an Animal Shelter database that can be searched intuitively using a web-based dashboard. Other features provided include a geolocation chart for selected items and a pie chart showing the breakdown for animal breeds based on search criteria provided by the client.

The idea is to simplify the identification and location of suitable animals that can be trained based on certain selection criteria. Grazioso Salvare provides its customers with animals that are suited to rescue operations training and this project demonstration will streamline the selection process.

## Requirements

The requirements provided by the client included the ability to perform CRUD (create, read, update, and delete) operations on a database provided by a third party. We were required to provide a web-based frontend for user operations. An interactive data table with intuitive filtering options and data charts with geo location were necessary. The mongoDB database system was employed to manage database operations and a Python module with CRUD capability is provided to facilitate the use of widgets designed to search based on user requirements. The dashboard design is accomplished with Python code and provides for database filtering options to include:

• Water Rescue

• Mountain or Wilderness Rescue

• Disaster or Individual Tracking

• Reset (returns all widgets to their original, unfiltered state)

These filtering options also generate a geolocation map for selected animals as well as a pie chart for data related to the four filtering options.

Some of the tools used to complete this module include the PyMongo driver which utilizes standard operability for MongoDB operations with Python. The API used is intuitive and works well with database controls, documents, and dictionaries. This allows for much easier CRUP operations as required in the project.

The Dash framework was used to facilitate the user front end. Dash is a Python framework used to create web-based applications. Dash precludes requirements for advanced knowledge of HTML, CSS, and JavaScript. Python code can be used to create the application and the outcome provides intuitive and creative web-based applications.

Initially database connection was established to verify the information provided. CRUD operations were created to test the Python module. Once the functionality of this module was verified, more features were added. The dashboard design was implemented to include a searchable data table. User selectable filtering options were added. A geo location chart was added, and these features were verified as operational. Finally, the client logo with an embedded link to their webpage was added. The table was labeled, and dropdown menu options were added for filtering searches. A pie chart was also added that displays animal information based on the selected search filtering options.

**Usage**

Below are screenshots of module operations with a dropdown menu:

Water Rescue widget selected:

Graphical user interface, text, application, Word

Description automatically generated

Graphical user interface, application, Excel

Description automatically generated

Graphical user interface, application

Description automatically generated

## Getting Started

To get going with the project simply login to and start your mongo database. Make sure it is running.

* Install/save the crudbackup.py module.
* Save the file path to the py module
* Import the system and append the path to your file location for the crud.py file.
* Import the main class from crud
* Define your object with the class name, username and password
* Create your dictionary
* Import the MongoClient from pymongo
* Define your client with your port name, username, password
* Create the dictionary within the database
* Query the addition
* Print the output
* To update, add the modified dictionary entry
* Run the update function
* Run your query again and print the output to verify update
* To delete an entry, update your query
* Run the delete function and print the output of the query to verify deletion

That should get you the information you need to demonstrate CRUD functions.

## Installation

Make sure to have a terminal to access the mongo database. Install Python if not already on your system. Save the crudbackup.py file and notate the file path. Utilize the crudbackup.py module to run with your Python code for database operations. This project includes Python code to perform CRUD operations on a database that include widgets for filtering options and output of a data table, geolocation chart, and a pie chart based on selection criteria.

Utilize the Python code in the IPYNB file included to run the user dashboard. This code provides for a web-based application designed as a dashboard with filtering options. These options allow for CRUD operations over the database provided by a third party. This code was built using the Jupyter notebook, but can be employed by using your favorite Python IDE and running from your console.

### Code Provided

The crudbackup.py module provided is designed to facilitate CRUD operations on a mongoDB system. There are four methods included:

* The create method will query the database and cerate new documents based on user input if those documents do not already exist.
* The read method will query the database and return the results based on user input for requested data.
* The update method will query the database and update documents based on user input of the document is found.
* The delete method will query the database and delete or remove data based on user input if the data exists.

All methods will prompt the user to enter data if user input is empty. The update and delete methods will output an error message if the operation is unsuccessful.

The IPYNB file provided has Python code used to test these operations. This file will also provide code to generate the project requirements for an interactive dashboard with selectable filtering options based on the client requirements listed above. This dashboard will generate filtered results based on user input and each result will accompany a geolocation chart and pie chart related to the search results.

## Roadmap/Features (Optional)

A future system console that prompts for username and password with a GUI designed to have selectable operations such as radio buttons or other menu items, could be helpful. This could eliminate the need for updating scripts and allow for a better user interface and experience. This would be a user added feature but is not contained in this project currently.

**Issues Encountered**

Some of the issues encountered during this project included database connection errors, data table formatting, geo chart location data updating, and pie chart generation. The database connection errors mainly revolve around being certain you have used the correct port address. This project is set up as a locally hosted system and the dataset port information is critical.

Data table formatting was resolved with the addition of style options in the table definitions. The methods used to create the table were updated with code to allow style options that eliminated formatting issues. The geo chart updating issues we resolved with proper code to identify where the coordinated for the selected search terms were located in the database.

The pie chart issues were resolved with proper use of the Dash framework as found on the Plotly support pages.

## Contact

Your name:

Christopher Clark

[Christopher.clark7@snhu.edu](mailto:Christopher.clark7@snhu.edu)

**References**

*Introduction: Dash for python documentation*. Plotly. (n.d.). Retrieved August 14, 2022, from

<https://dash.plotly.com/introduction>

*MongoDB python connection*. MongoDB. (n.d.). Retrieved August 14, 2022, from

<https://www.mongodb.com/languages/python#:~:text=PyMongo%2C%20the%20standard%20M>

ongoDB%20driver,%2C%20iterate%2C%20and%20print%20them.