Augustus Mendy

Southern New Hampshire University

CS-340-10011-M01

Professor Gebre-Amlark

April 20, 2025

## About the Project/

I work as a lead developer for Global Rain, a software engineering firm that focuses on creating and designing unique software. The goal of the project, Grazioso Salvare, a global rescue animal training organization with five adoption facilities in Texas is to develop an interactive and fully functional web-based dashboard for a training firm that specializes in animal rescue. Model, View, Controller is the pattern that will be used to construct this project.

* **Model:** This is an example of the data or database utilized in the development backend.
* **View:** This layer serves as the user's visual representation. (User Interface)
* **Controller:** The communication between the user and the requested data is managed by this layer**.**

MongoDB is utilized for the model layer database in this project. The controller layer between MongoDB and the user will be Python using Pymongo, while the view layer uses Dash Plotly and Dash Leaflet.

## Motivation

Search-and-rescue training candidates are identified by Grazioso Salvare. He has made a deal with a nonprofit organization that runs five animal shelters in the Austin, Texas, area. He is looking for a piece of software that can use the data that is currently available from the animal shelters and group them into various search and rescue groups according to a number of criteria.

**Tools Used:**

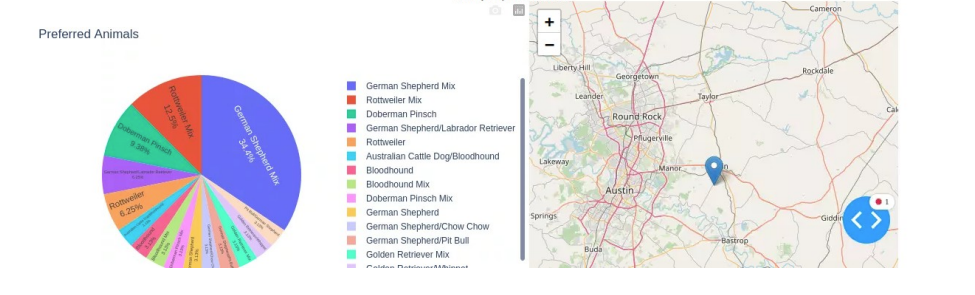
* **MongoDB-** One of the most often used databases among developers, it offers scalable applications and the ability to store both structured and unstructured data. Along with complete technical assistance and documentation, it supports a number of programming languages. Because MongoDB supports a Python driver, it can fully generate an MVC pattern application with Python as the programming language.
* **Dash Framework –** The view and controller structure of the web application are provided by this framework. This framework for creating data visualization interfaces is open-source. Additionally, it supports Python, which facilitates effective development between the controller and view.
* **Python-**-High-level programming languages like Python are simple to read, understand, and write. Because it is open source and contains a wide range of standard libraries, it is the perfect language for developers. It has sufficient capability to support the development of apps and is compatible with both Dash and MongoDB.
* **Python CRUD Library -** You will need to utilize the Pymongo driver to develop a CRUD module for your MongoDB. This will be utilized in conjunction with the controller to manage client-server communication.

**Installation/Pre-Conditions**

To use this software the following tools must be installed:

* **Python 3.6**
* **Pymongo 4.2**
* **MongoDB 4.2**
* **Plotly Dash**
* **Dash Leaflet**
* **Database admin user account with read/write permissions**

****

****

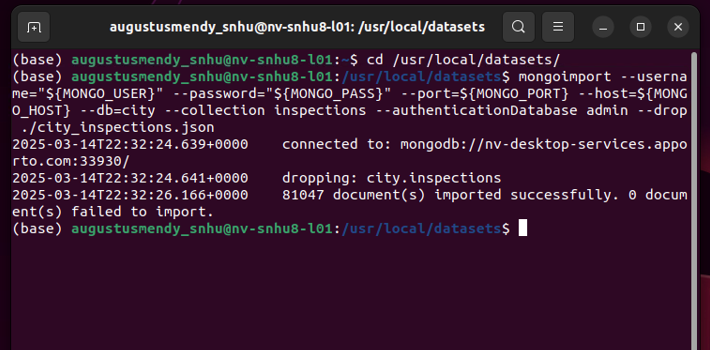
**Functionality:**

This application's functionality makes the user's experience pleasant and easy to use. The primary dashboard displays the dashboard name and developer along with the Graziosos Icon, which, upon running the program, redirects back to the SNHU page as required. The user is provided with a row of buttons. Based on Mr. Graziosos's desired outcomes, the user can filter results by interacting with four buttons. The five animal shelters that provided the data are the source of the data table that is displayed beneath these buttons. A pie chart showing the percentage of each breed used in the filters is shown below. A geomap with a marker indicating the animal's present location chosen from the data database is displayed next to it.

**Start from here…**

**Project Creation:**

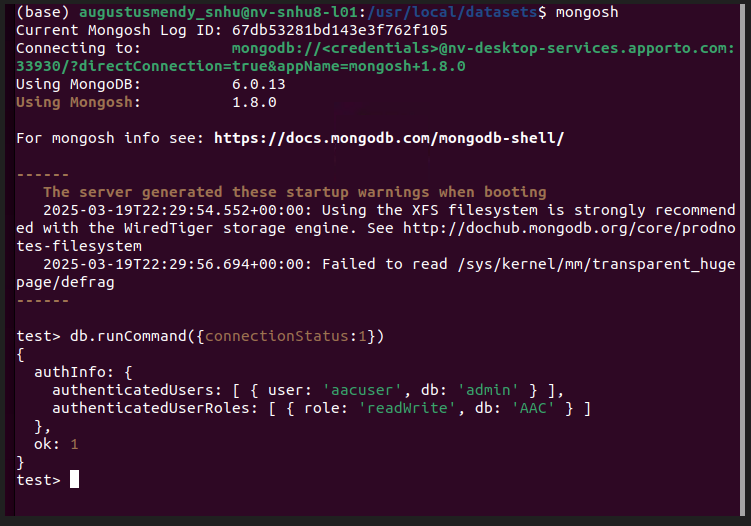
Setting up user authentication and database access is a prerequisite for using the CRUD Python module. You can do this by entering mongoimport into your terminal's command line. This is where you enter your <options> <connection-string> <file> to set up your settings, the connection method, and the file containing the information you want to add to the database.

Once mongoimport is successful, you receive confirmation, as seen in the sample below. providing you with information about your connection status, imported documents, and whether the import was successful or not.

## You are now able to access your database. User access control can be set up here by adding users with various levels of power.

## Example of database user authentication and "aacuser" collection:





## After gaining access to your database and assigning users access control, you may proceed to developing the Python module that will serve as a controller between the database and the client. You can do this by utilizing the Pymongo-using Python driver. Pymongo contains Python methods for MongoDB CRUD activities and is a collection of tools for working with MongoDB. In particular, you must import Pymongo's MongoClient. Connecting to MongoDB is done via MongoClient. The AnimalShelter class accepts username and password as inputs and is initialized with your connection string, database, and collection.

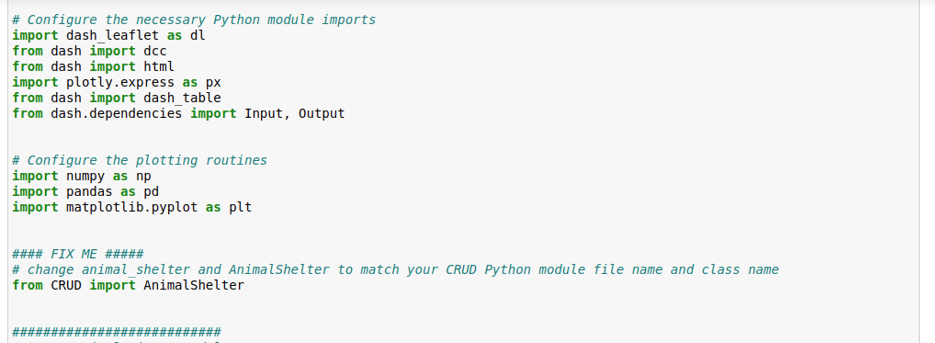
**Steps to Connect:**

* Import MongoClient from Pymongo and import ObjectId from bson.objectid.
* In order to initialize the connection using a connection string, create a MongoClient to the MongoDB instance.
* Use attribute style access on the MongoClient instance to declare your database and collection to be accessible.

## An example of an AnimalShelter class that defines MongoClient for CRUD operations.

## 

By generating an instance of the class and providing your username and password as parameters to access MongoDB with the appropriate database credentials, you can instantiate the AnimalShelter class after defining MongoClient within the class. Now that your database is configured and you know how to connect to it, you can begin implementing the dashboard.A number of libraries must be imported in addition to your CRUD module in order to finish the dashboard.



The interface in this example was constructed using the Plotly Dash and Dash Leaflet frameworks. To improve programming capabilities, these can be combined with a number of open-source libraries.

You can access the data frame using your CRUD module now that the appropriate libraries have been imported. After that, you want to read the database into memory and save it as a variable in Panda's Data Frame Structure. To avoid issues from deserialization techniques, we must remove this field since MongoDB automatically allocates an object id to each document element.

This is accomplished by dropping the column.

A computer code with text

AI-generated content may be incorrect.

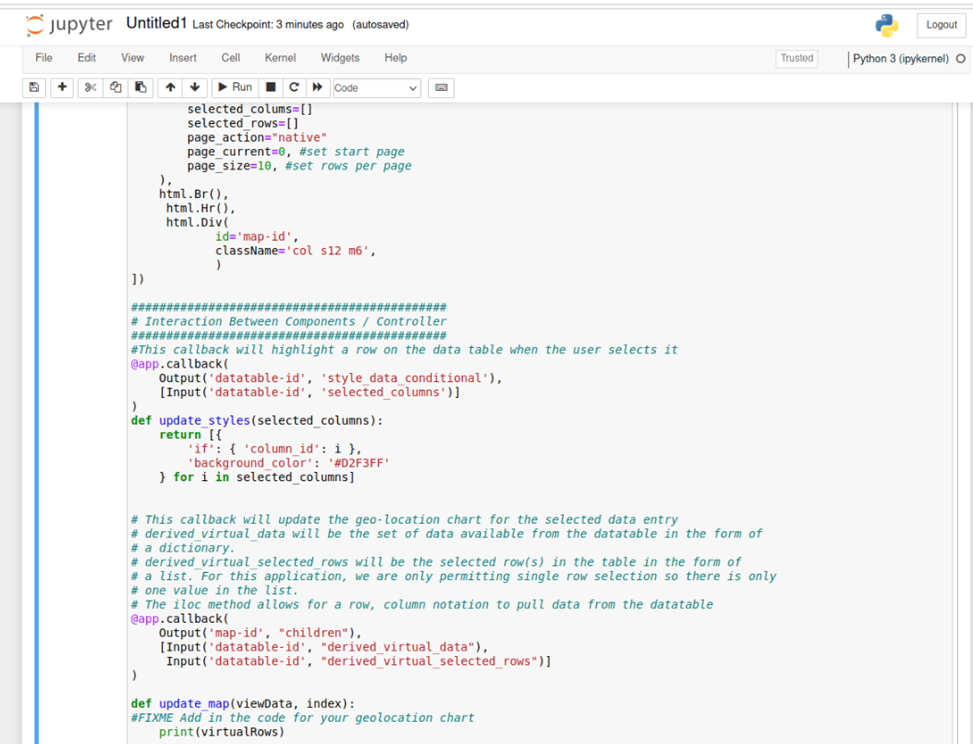
## You may now create the dashboard. The dashboard must be initialized using app.layout. The application is divided into two sections. The callbacks and the arrangement. You can customize your dashboard's appearance and features in the layout, and you can interact with the data using the callback.

**Add layout example:**

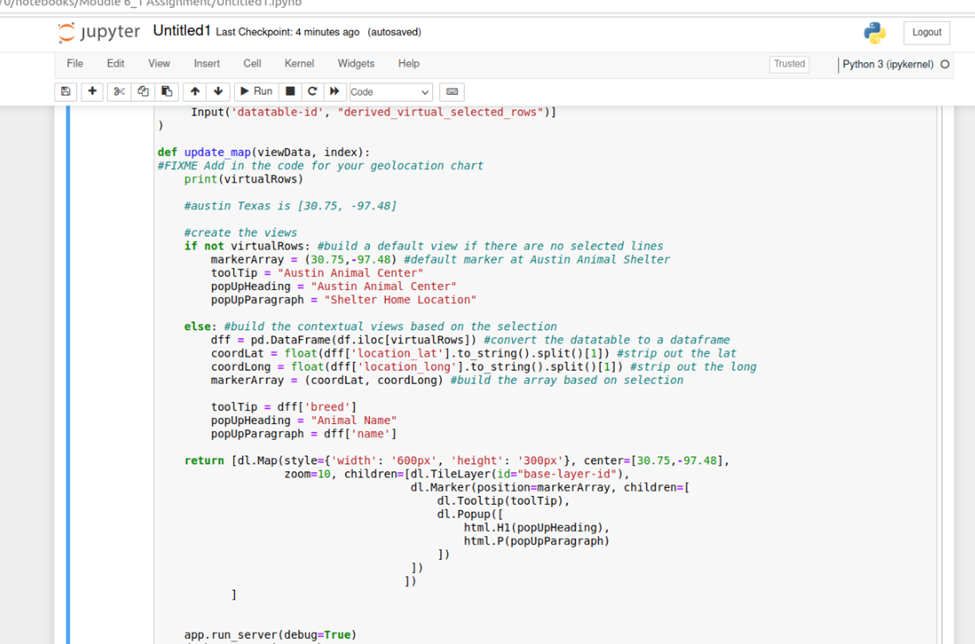
A screenshot of a computer

AI-generated content may be incorrect.

## Add Dash Table Example:



**“Add Callback to display pie chart when Filter buttons change”:**

****

## To execute the application, use the app.run\_server() line. This will cause a link to appear, which you may click to access your dashboard.



**Challenges:**

While creating this application, I faced a number of difficulties. My biggest difficulty was comprehending the application's layout. The dash documentation is based on HTML and CSS principles, which are interrelated. It takes some time to become familiar with the syntax and all of the various layout elements. I watched a number of lessons on HTML, CSS, and Dash Plotly, which helped me get beyond the obstacle. The way these all interacted and improved the layout's design became evident after that. The button callbacks presented me with yet another difficulty.

I had to figure out how to compare which button was pressed. Rereading the callback and button documentation helped me get beyond this, and it was there that I found the ctx.trigered\_id. I was able to simplify the reasoning required to ascertain the appropriate callback output as a result. I had trouble getting the right search terms as well. The suggested specifications given by Grazioso differ slightly from several of the breed names. This caused certain searches, such Chesapeake Bay Retriever, to yield no results. This resulted from the abbreviations of some of the names. Additionally, there were breeds that combined two dog breeds, like the Australian Cattle Dog and Labrador Retriever. Because of this, I was able to efficiently look for the right breeds needed by using the standard library for regular expressions.

## Contact

Your name: Augustus Mendy