5-2 MILESTONE

5-2 Milestone Four Enhancement Three Databases

CS-499-12473-M01 Computer Science Capstone 2024 C-5 (Sept-Oct)

Michael Zietz

Southern New Hampshire University

Oct 6, 2024

* Briefly describe the artifact. What is it? When was it created?

This artifact was developed for a previous course, CS-340 Client/Server Development, at SNHU. The client, Grazioso Salvare, needed to interact with their existing data from animal shelters. This artifact intended to identify and categorize animals which have been entered into the system based on their criteria. These entries possess various properties: \_id, rec\_num, age\_upon\_outcome, animal\_id, animal\_type, breed, color, date\_of\_birth, datetime, monthyear, name, outcome\_subtype, outcome\_type, sex\_upon\_outcome, location\_lat, location\_long, and age\_upon\_outcome\_in\_weeks.

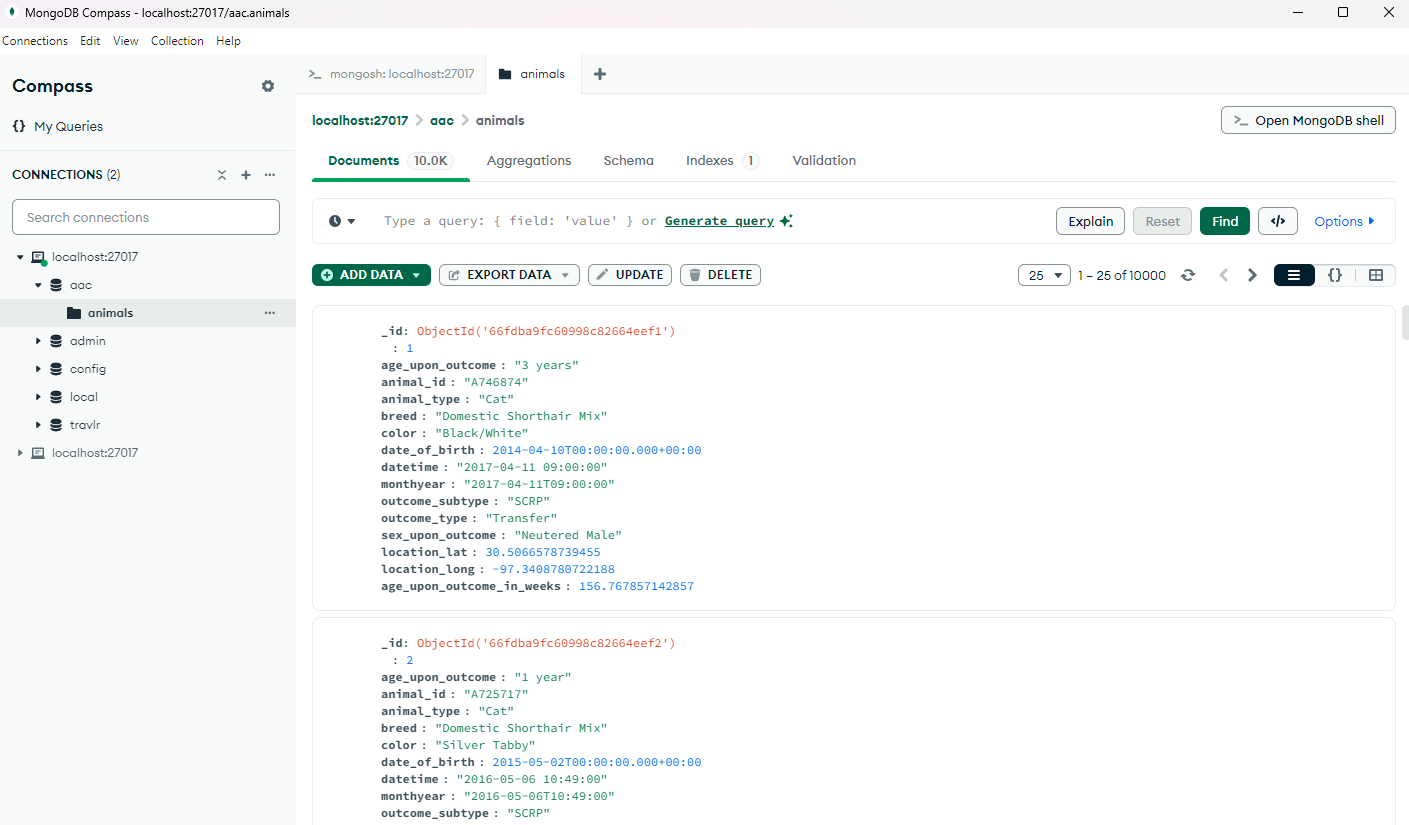
Grazioso Salvare needed to use this software to identify animal profiles to train for various tasks, and they have requested this project be open source and accessible on GitHub to aid other, similar organizations. Rather than manually sorting through each database, which can take tedious and time-consuming effort, this software will reduce this human waste by requiring simple inputs to retrieve data from databases quickly. It will be able to create new entries, read/search for existing ones, update entries, and delete them as well.

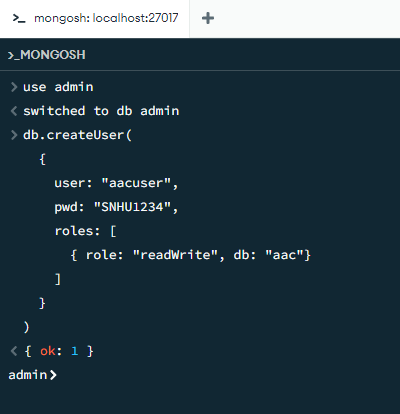
I sought to enhance this application’s current GUI, as its pie chart often results in unappealing visuals that clutter the screen, migrate the project, as it was initially created for a virtual Linux-based lab environment, named Apporto, and recreate it in a Windows OS environment. This required me to install and coordinate compatible versions of Python and MongoDB. For this application, the PyMongo library is need to be utilized in order to successfully connect Dash’s capabilities with MongoDB. The recreated application must include authentication before allowing access to CRUD features as only specified individuals should be granted permissions to modify stored data.

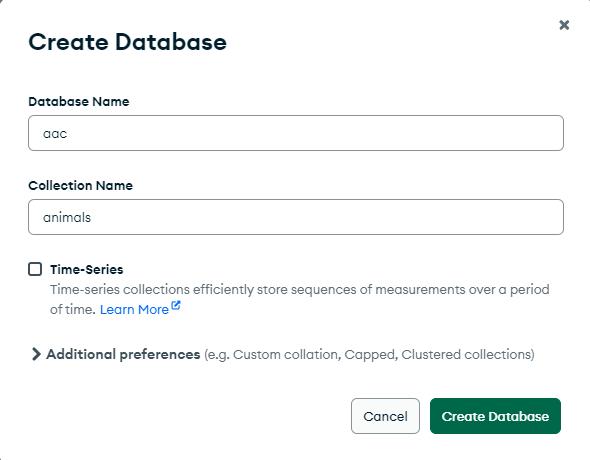
* Justify the inclusion of the artifact in your ePortfolio. Why did you select this item? What specific components of the artifact showcase your skills and abilities in software development? How was the artifact improved?

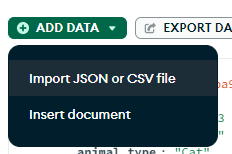
I have selected this artifact to enhance because the successful recreation of this application initially designed for Linux OS demonstrates my understanding of the Python language as well as Dash, skills deploying MongoDB and CRUD functionality, and ability to reproduce and recreate scripts inside Jupyter Notebook.

The first step in this enhancement was to import the previous database’s data into a new collection stored locally on my Windows environment. I used Mongo Compass’ import method in their application to do this as well as create a user necessary for reading and writing to the database. I intentionally had to recreate the database using the same naming convention decided before for the database, collection, username, and password to be compatible with the python file’s requirements. For the database, this meant correcting datatype errors that occurred after the import to ensure values like coordinates remained and were stored as integers. After the database was recreated, I then created a new user with read and write permissions to only the “aac” database from the Mongosh terminal. (Create a User, 2024)



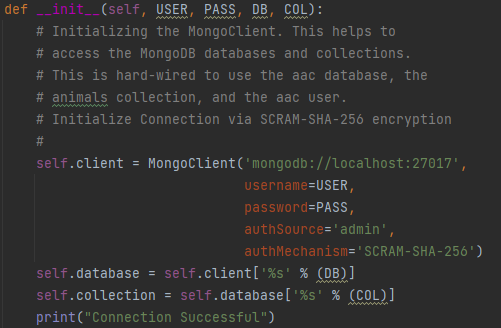


(Origin: Mongo Compass)



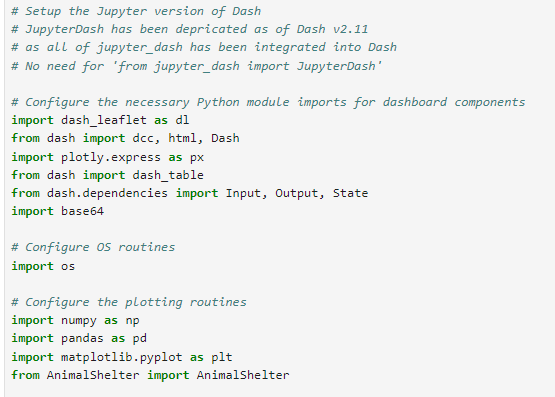
Now that the user was recreated, I needed to establish the connection between Jupyter Notebook and MongoDB. In the artifact’s original state, this was implemented inside the CRUD python file, named AnimalShelter.py, and initialized the host for Apporto: “HOST = 'nv-desktop-services.apporto.com' PORT = 32582.” MongoDB’s default path and port, once installed, to access databases is mongodb://localhost:27017, but this lacked any identification or authorization and used the admin account to perform read and write actions. To create a higher level of security integrity in the enhanced artifact, I implemented SCRAM-SHA-256 encryption that uses the provided username and password and checks the admin’s users database before authorization is allowed.

(Origin: AnimalShelter.py)

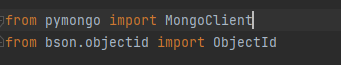


The next step was to install the required libraries and libraries to utilize the frameworks that the Linux based project used. Of these, the most important ones needed were jupyter notebook, dash, dash\_leaflet, pymongo, plotly, numpy, pandas, and matplotlib. This was done by accessing Window’s built-in command prompt terminal and issuing the following prompt for each of the required libraries: “py -m pip install dash”

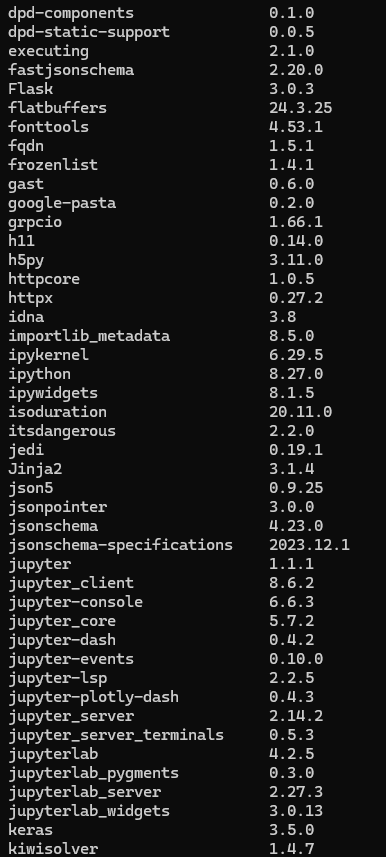
(Origin: ProjectTwoDashboard.ipynb)

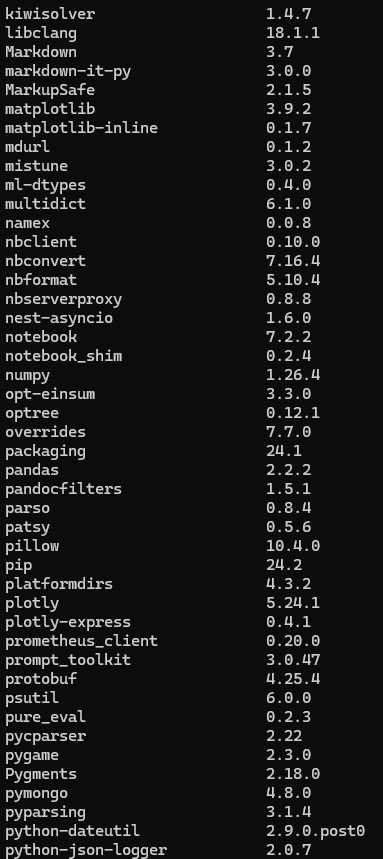
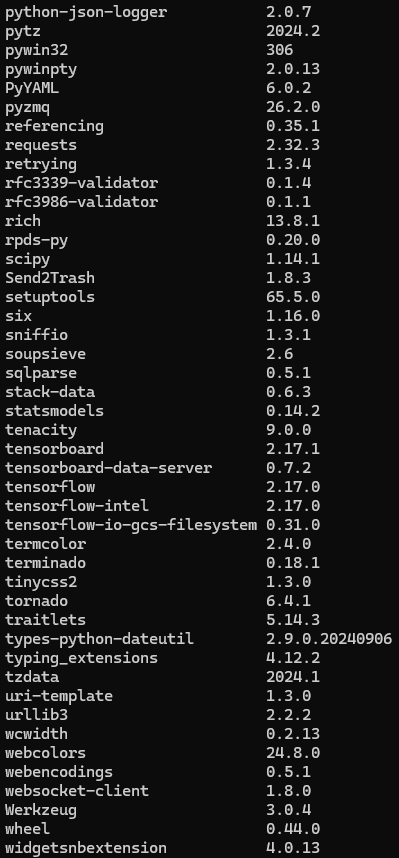


(Origin: AnimalShelter.py)

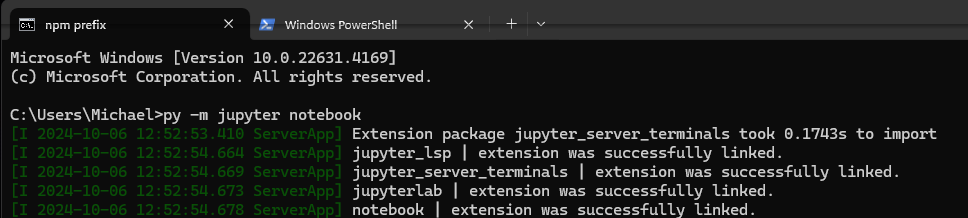


Below is a list of all libraries and their current used versions (needed and not needed) in my Windows OS environment.

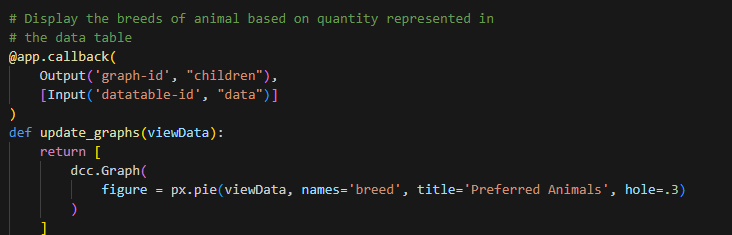
 

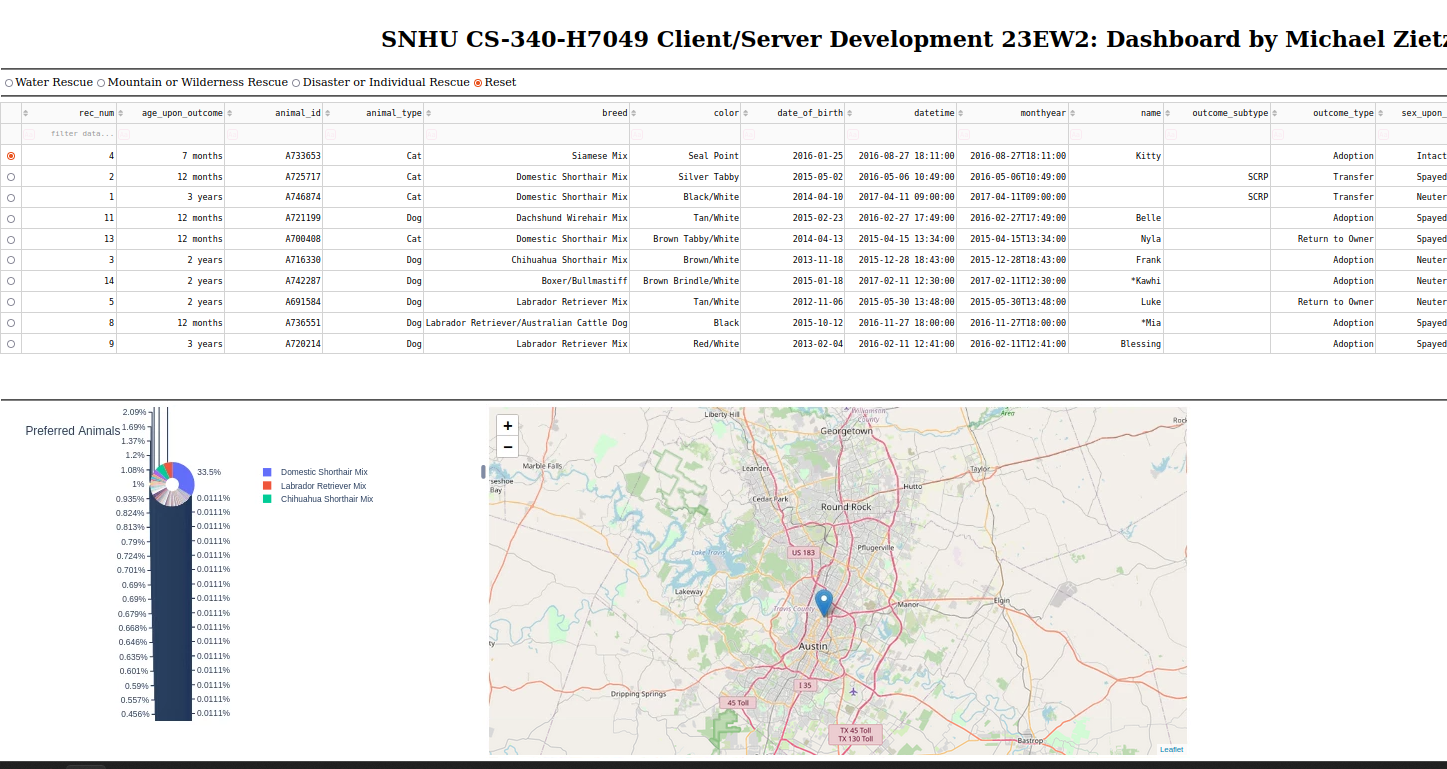
To access Jupyter Notebook once installed, I simply opened a command prompt window and ran the python command to open the web browser platform in Google Chrome.



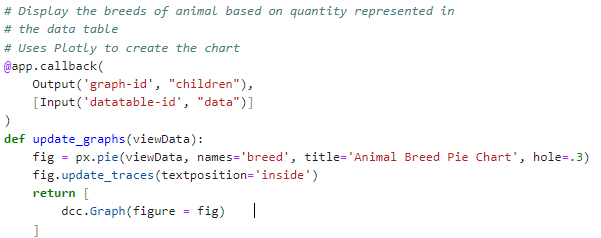
The artifact’s original state possessed a pie chart that displayed analytics on the various animal breeds returned by the filtering options, but the result often created unappealing and confusing charts that bordered on unreadable. To improve this, I implemented additional styling that moved the lines identifying each section and moved the text inside the “pie piece”, and when the cursor hovers over a slice, the breed is displayed. The result was a much cleaner and readable pie chart that users could effective utilize.

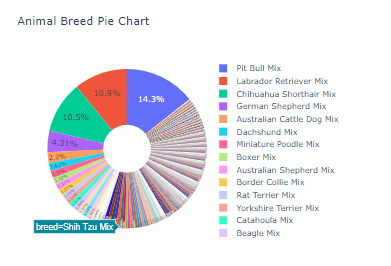
(Origin: ProjectTwoDashboard.ipynb) - **before**





Origin: ProjectTwoDashboard.ipynb) - **after**





The final improvement to the artifact was creating additional filtering options that functioned in Dash’s framework. I created the more general filtering options for just populating the site with cat and dog results.

(Origin: ProjectTwoDashboard.ipynb)



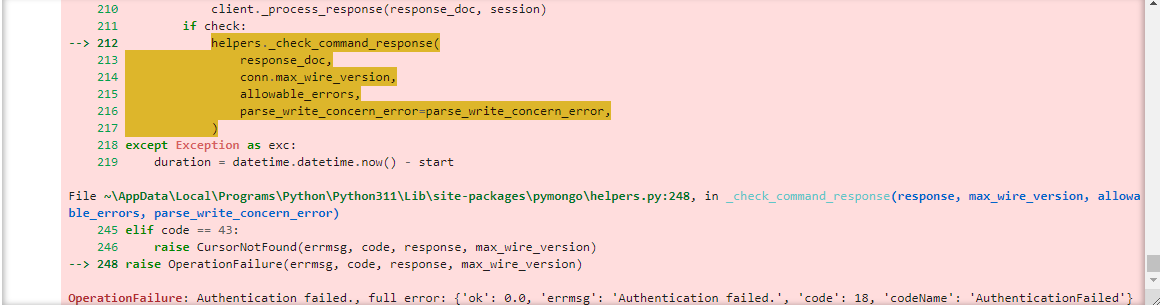
* Did you meet the course outcomes you planned to meet with this enhancement in Module One? Do you have any updates to your outcome-coverage plans?

As laid out in the artifact’s original code review, I have successfully implemented the desired enhancements for my ePortfolio. I had intended to migrate this database from a Linux base environment called Apporto, improve the security through encrypting the authenticated user session for accessing the database, and enhancing the displayed graphics to make them more interactable and readable for users. All three of these intended goals have been shown to be implemented. At this time, I have no planned improvements to this artifact though I’m sure additional features utilizing more complex SQL concepts could be easily implemented in the Jupyter Notebook environment.

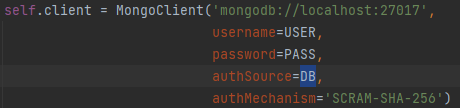
* Reflect on the process of enhancing and modifying the artifact. What did you learn as you were creating it and improving it? What challenges did you face?

As I was developing this enhancement, I was unsure how to properly handle the authenticated database access that the Jupyter Notebook python file would need to initiate through the AnimalShelter.py CRUD file. In the previous artifact’s enhancement, I utilized MongoDB Compass’s capabilities to create a different database on my Windows OS but had used several other frameworks in JavaScript to function. I was able, early on, to use MongoDB Compass’s default port, localhost: 27017, once the library pymongo was installed, but this lacked any user authentication and used the default admin account which has more access and authority than needed for this instance. After researching pymongo and the MongoClient() function’s documentation, I discovered that SHA-256 encryption could be specified as the authentication mechanism. (mongo\_client – Tools for connecting to MongoDB - PyMongo 4.10.1 documentation, 2024) However, when I first attempted this, I thought the authSource should be from the desired database, in this case “aac,” but after receiving the following error message, I knew I had made a mistake.

(Origin: ProjectTwoDashboard.ipynb) - **failed**

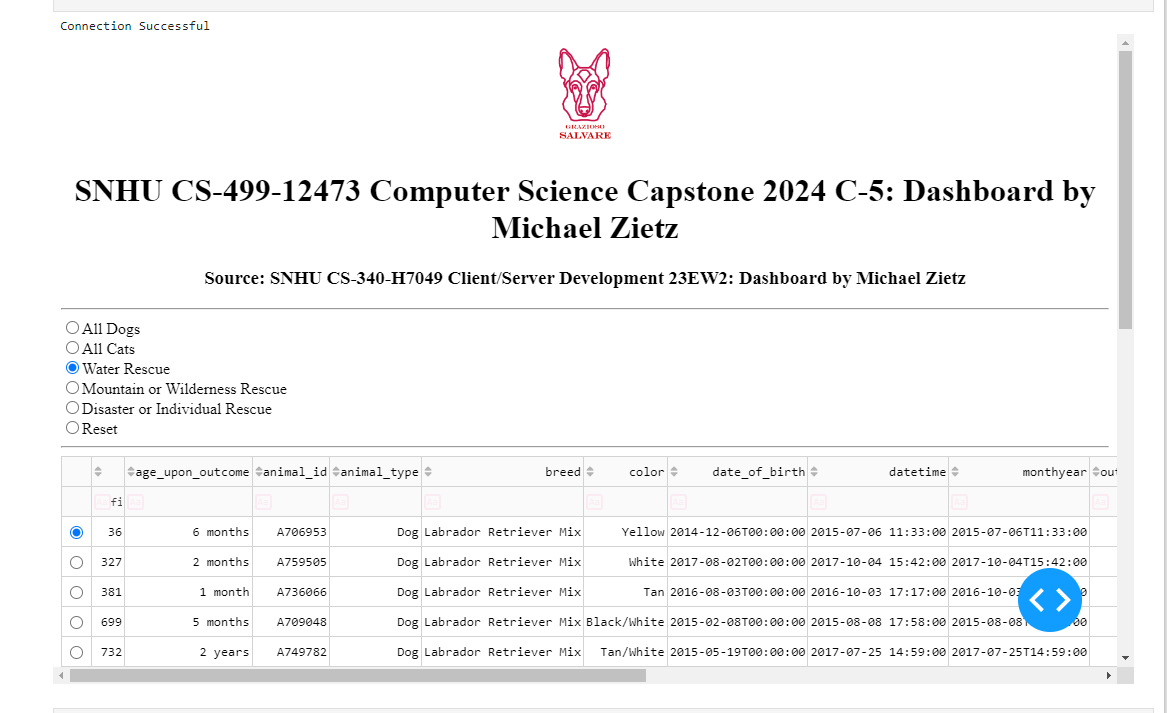


(Origin: AnimalShelter.py)

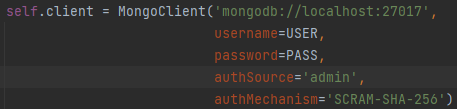


Deriving that I should instead be searching the submitted database for proof of authorization, I updated the code base to instead use the admin database. Now when the application is run, the connection is authenticated, producing no errors, and allows read and write access.

(Origin: ProjectTwoDashboard.ipynb) - **success**

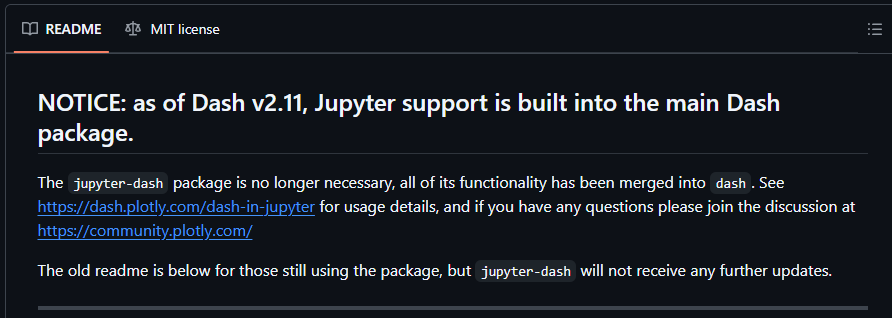


(Origin: AnimalShelter.py)

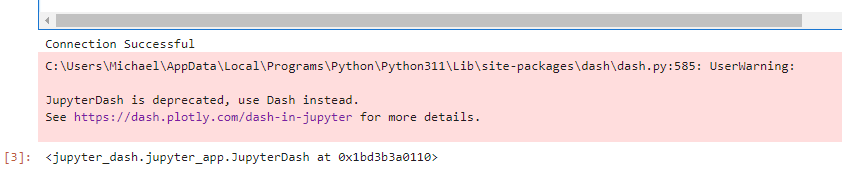


The last issue I came across when migrating this artifact into a Windows environment was that the current code base imported the JupyterDash library from jupyter\_dash, but as of Dash v2.11, this use had been deprecated. The current versions of the Dash library instead integrate the previous functionality within itself. The fix for this issue was easy enough though, and instead of importing and initializing the app using JupyterDash(), I had to import Dash from dash, and replace the deprecated function as seen in the screenshots below. (plotly, 2022)

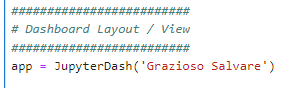
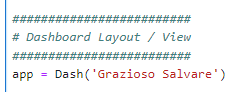
(Screenshot source: https://github.com/plotly/jupyter-dash)



(Origin: ProjectTwoDashboard.ipynb) - **depricated**



(before) (after)

Resources

*Create a User*. (2024). Mongodb.com. https://www.mongodb.com/docs/v6.2/tutorial/create-users/

*mongo\_client – Tools for connecting to MongoDB - PyMongo 4.10.1 documentation*. (2024). Readthedocs.io. https://pymongo.readthedocs.io/en/4.10.1/api/pymongo/mongo\_client.html#pymongo.mongo\_client.MongoClient

‌plotly. (2022, April). *GitHub - plotly/jupyter-dash: OBSOLETE - Dash v2.11+ has Jupyter support built in!* GitHub. https://github.com/plotly/jupyter-dash

‌

‌