

**CS – 499 Capstone: Milestone Four**

**Enhancement Three: Databases**

Milton Francisco

Southern New Hampshire University

March 31, 2025

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

This artifact is a data dashboard that was created in CS-340 during my Computer Science degree. Originally, it was located and ran within an Apporto virtual machine (VM). Additionally, the required dependencies and software were included in the VM, so the inner workings of the program were abstracted away from students. It used a provided instance of MongoDB, Python middleware to interact with the database, and Jupyter Notebook, with Plotly Dash, to render a friendly user interface to interact with the database.

**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?**

This item was selected because it allowed me to demonstrate proficiency in both software design and database enhancements. After the class ended, the program was unable to run as it depended on the specific Apporto virtual environment. The first phase of enhancements allowed the program to run independently. The database was migrated to a cloud-hosted EC2 server, making it more accessible, and the python middleware was refactored to account for this change.

This enhancement involved refactoring the application to phase out the deprecated Jupyter Dash and to solely function with Python and the newer Plotly Dash library. This allows the dashboard to provide a more dynamic experience, and additional functionality could be incorporated more easily. Additional queries were implemented, more informative features in the pie chart, and more concrete database indexing. Moreover, unnecessary database information was hidden to allow users to view the most important information by removal of columns like location latitude and longitude and datetimes. Importantly, this information is still able to be toggled and viewed, but the removal declutters the interface. Finally, during my development of enhancement one, I also noticed the dashboard required potentially complicated instructions to run the application. Ultimately, I decided to implement Docker with the application to support an easier setup for users. The

instructions are included in the public Docker image description and the URL will be included in the setup instructions.

**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?**

The course outcomes I planned were fulfilled with this enhancement. My goal was to demonstrate proficiency in the following outcomes:

- The 1<sup>st</sup> outcome, *(Employ strategies for building collaborative environments that enable diverse audiences to support organizational decision making in the field of computer science).*
- The 4<sup>th</sup> outcome, *(Demonstrate an ability to use well-founded and innovative techniques, skills, and tools in computing practices for the purpose of implementing computer solutions that deliver value and accomplish industry-specific goals (software engineering/design/database)).*
- The 5<sup>th</sup> outcome, *(Develop a security mindset that anticipates adversarial exploits in software architecture and designs to expose potential vulnerabilities, mitigate design flaws, and ensure privacy and enhanced security of data and resources).*

Overall, the application demonstrates these outcomes by providing database information to a less technical audience. The inner workings of the application are not required to be known, and additional functionality has been implemented through further queries and more information. This allows the application to be more accessible to less technical groups, the refactoring skills show the ability to use well-founded techniques, and security is ensured with the secure database queries.

**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?**

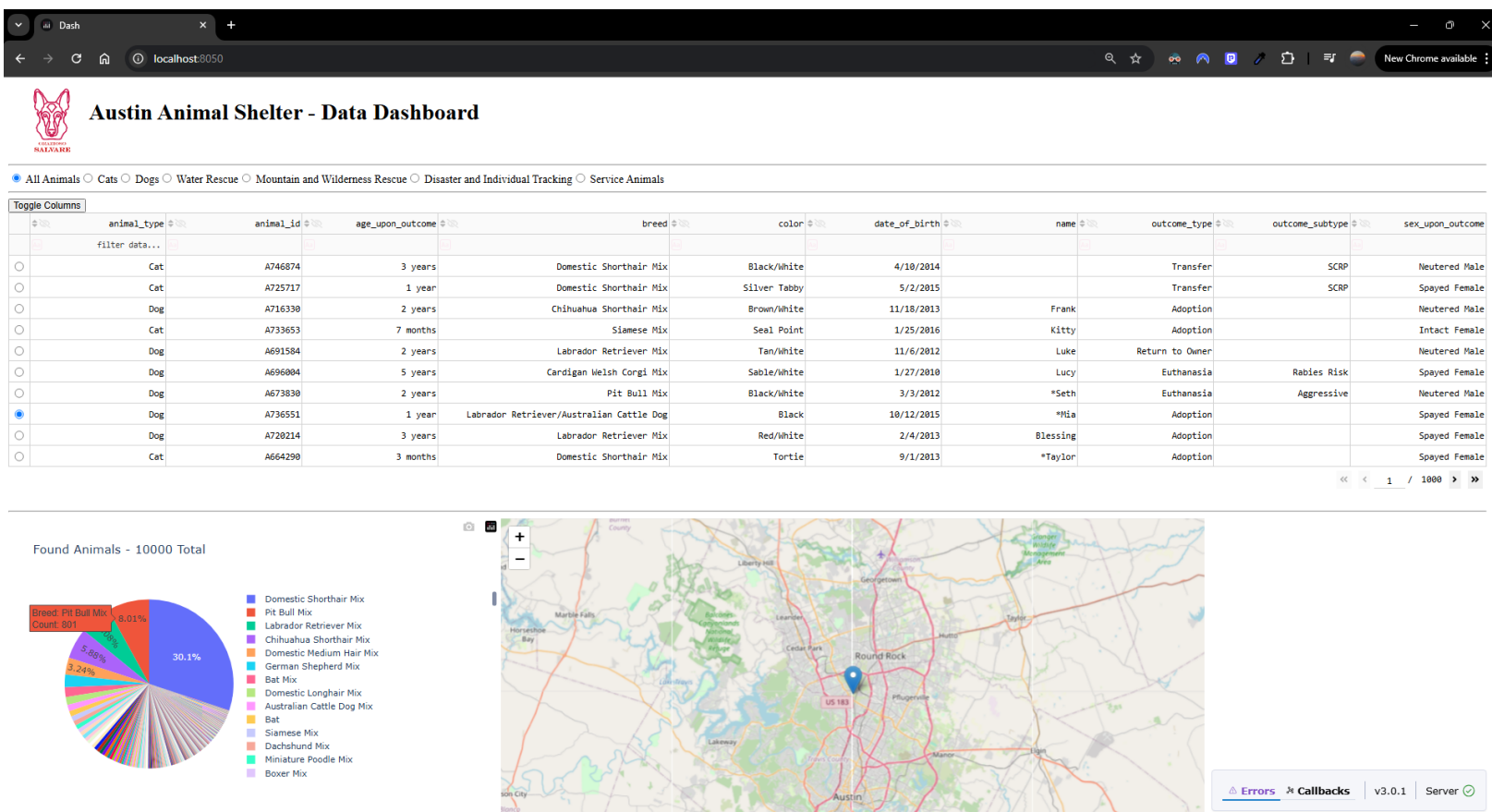
I learned a lot in the refactoring as it introduced unanticipated issues. For example, after changing the map callback to use named indices, i.e. `location_lat` or `location_long`, this resulted in errors when initially populating the map. This was due to the map callback attempting to find the indices prior to the data frame being loaded. To account for this, I implemented a default map location of Austin, TX in case the data frame is not yet loaded. The animals are located around Austin, TX so this seemed like an ideal solution.

Another issue I encountered was in the pie chart. I attempted changing it to a bar chart and histogram, but the database entries caused the charts to not look as good as the pie chart. Ultimately, I decided to maintain the pie chart, but include further information on each popup marker. This involved reading the Plotly Dash documentation thoroughly to determine what data I could glean for my uses. I decided to include a line for the number of each breed. Previously, the chart only included a percentage and the breed's name, which caused some confusion as the amount was not shown. Additionally, the overall count of breeds found was included in the chart title. Finally, the chart can be further modified to compare breeds by double-clicking individual breeds from the legend.

Finally, implementing Docker with this project seemed like an ideal solution to resolve the necessary dependencies. While this handled the necessary dependencies in a relatively simple manner, it introduced more complexity when developing. It required extensive research to determine how the application could work within a container environment. A large issue was the necessary host and ports. The container required the host to allow all, but the application is found on `localhost:8050`. The issue lays in the fact that Dash automatically outputs a console statement for the address used in the file, in this case `0.0.0.0`. To account for this, I implemented a separate console statement for the correct address. Additionally, the Docker image (<https://hub.docker.com/r/miltfrancisco/cs499-capstone>) provides further instructions for how to run this application, in an effort to combat the automatic Dash logging.

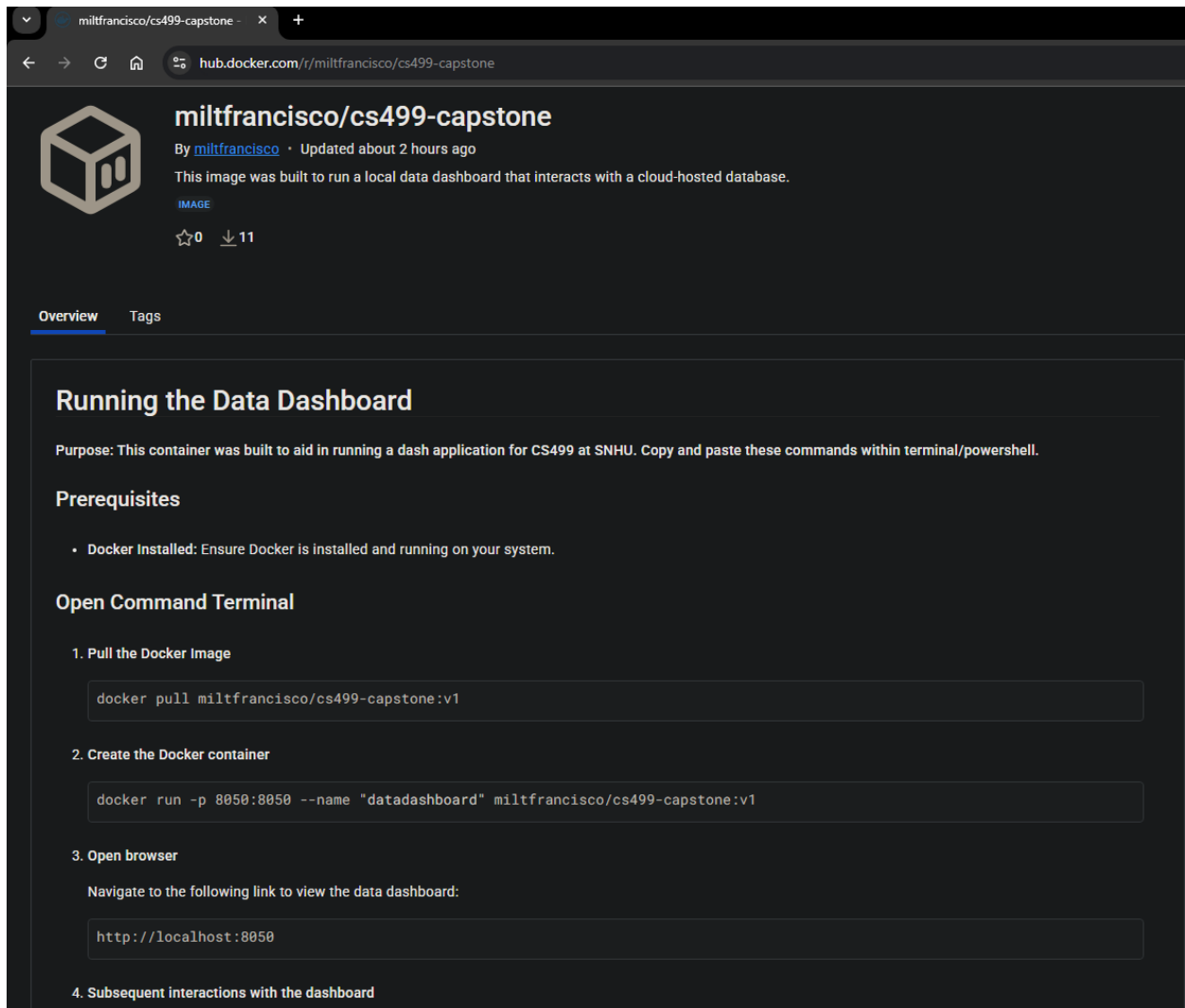
# Enhancement Pictures

## Running the Data Dashboard:



(Note: Additional queries are shown, working options to toggle database columns, and additional pie chart information.)

## Docker Image Instructions:



The screenshot shows the Docker Hub page for the image `miltfrancisco/cs499-capstone`. The page includes the Docker logo, the image name, the creator `miltfrancisco`, and a description: "This image was built to run a local data dashboard that interacts with a cloud-hosted database." It also shows 0 stars and 11 downloads. Below the image details, there are tabs for "Overview" and "Tags". The "Overview" tab is selected, showing a section titled "Running the Data Dashboard". This section contains a purpose statement, prerequisites, and a step-by-step guide to run the dashboard using Docker.

**miltfrancisco/cs499-capstone**  
By [miltfrancisco](#) · Updated about 2 hours ago  
This image was built to run a local data dashboard that interacts with a cloud-hosted database.  
[IMAGE](#)  
☆0 ↓ 11

**Overview** Tags

### Running the Data Dashboard

Purpose: This container was built to aid in running a dash application for CS499 at SNHU. Copy and paste these commands within terminal/powershell.

#### Prerequisites

- **Docker Installed:** Ensure Docker is installed and running on your system.

#### Open Command Terminal

1. Pull the Docker Image  

```
docker pull miltfrancisco/cs499-capstone:v1
```
2. Create the Docker container  

```
docker run -p 8050:8050 --name "datadashboard" miltfrancisco/cs499-capstone:v1
```
3. Open browser  
Navigate to the following link to view the data dashboard:  

```
http://localhost:8050
```
4. Subsequent interactions with the dashboard

(The application can be run using Python 3.12 and installing the requirements.txt, but the Docker image handles this for most users)