Gregoria Ramirez

CS 499 Capstone Artifact Narrative

Software Engineering and Design

Module 3

July 15, 2025

DR Troy Hawk

For my CS 499 Capstone, I chose to use my Animal Shelter project from the CS 340 Client-Server Development course that I originally created in February 2025. The artifact was a web dashboard built in Jupyter Notebook using Python and Dash that connected to a MongoDB database which pulled data from an uploaded CSV file. It displayed a table of animal shelter outcomes through a searchable record list that included filters and charts. The project used CRUD operations to query the database for different types of analysis. Some searchable patterns included filtering by shelter type such as transfer or adoption.

I chose this project because it brings together everything I have learned by working with data, building the interface, and connecting it all through Python. It combines backend work using MongoDB, data handling with pandas, and a user-friendly interface built with Dash. The original version worked, but it lacked structure and security, which made it a great candidate for enhancement. I reorganized the project into a more modular Model-View-Controller (MVC) format, replaced hardcoded credentials with environment variables using python-dotenv, and added logging to help with debugging. Specifically, I split the logic into separate files: model.py handles all database operations usingpymongo, controller.py manages Dash callbacks and user input validation, and app.py serves as the main entry point for launching the application. This MVC separation helped improve scalability, debugging, and future maintenance.

On top of that, I included new filters for breed and color and improved the dashboard visuals to make them more clear, polished, and responsive. I refactored thefiltering logic using vectorized operations in pandas within controller.py, which allowed faster lookups and cleaner code. This improved performance, especially when handling large datasets, and eliminated duplicate filter logic across components.

In Module One, I originally planned to meet two outcomes, which were:

1. **Outcome 3 (Software Design and Engineering):** I met this by applying modularity, following MVC principles, and using environment variables to secure sensitive information like the MongoDB URI.
2. **Outcome 4 (Algorithms and Data Structures):** I improved the filtering logic and added input validation to strengthen how the app handles different rescue types and attributes like breed and color.

As I continued working on the project, I realized I had missed an important outcome related to the database enhancements I made. Because those changes were important, I later added **Outcome 5 (Databases)** to my plan. I moved away from using Mongo shell and instead used MongoDB Compass to visually manage the database and collections. I added error handling in model.py and made sure all queries use safe, parameterized logic through pymongo methods such as find() with filters and insert\_one() for controlled document creation. I also ensured the database connection is properly closed using context managers and added logs to track database actions. In addition, controller.py validates all incoming inputs before passing them into queries, reducing the risk of injection or malformed requests. I updated my Module One template after receiving feedback from my instructor.

Working on the enhancements helped me look at my code differently and focus more on structure and reusability. I split the code into separate files, kept the logic organized, and followed best practices like using a .env file for sensitive info. One issue I ran into was reorganizing the file structure. I had to fix my import paths across files like app.py, controller.py, and model.py, and test each callback to make sure everything was still working. I also had some trouble setting up the virtual environment. I created a virtual environment using python -m venv venv to keep my project’s dependencies separate from other projects and to make setup easier. I then activated the environment in VS Code and installed packages like dash, pandas, plotly, pymongo, and python-dotenv. Using venv helped avoid version conflicts and made it easier to manage a clean setup for testing and future deployment.