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 ten thousand records 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, applying principles such as separation of concerns and the Single Responsibility Principle to make the code easier to maintain.

I replaced hardcoded credentials with environment variables using python-dotenv to improve security and added logging to assist with debugging and troubleshooting. Specifically, I split the logic into separate files: model.py handles all database operations using pymongo, 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 improved scalability, made debugging more efficient, and prepared the application for future feature expansions.

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 achieved this by applying modular design principles, following the MVC architecture, securing sensitive information with environment variables, and restructuring the application for better maintainability, performance, and scalability.
2. **Outcome 4 (Techniques, Skills, and Tools in Computing Practices):** I achieved this by using well-founded and innovative techniques such as MVC architecture, vectorized pandas’ operations for performance optimization, and environment-based configuration for security. These improvements demonstrate my ability to implement computing solutions using professional tools and practices that deliver value and meet industry standards.

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 to follow the MVC pattern and kept the logic organized for easier maintenance. I applied secure coding practices, such as using an .env file to store sensitive information, and ensured my imports were modular and well-structured.

During the enhancement process, I learned the importance of planning architecture changes before implementation. Updating the file structure required adjusting all import paths in app.py, controller.py, and model.py, and retesting every component to ensure proper function. Another key learning moment came when setting up the virtual environment using python -m venv venv. This allowed me to isolate dependencies, avoid version conflicts, and ensure that other developers could easily replicate the environment. I activated the environment in VS Code and installed required packages, including Dash, pandas, Plotly, pymongo, and python-dotenv.

Overall, this enhancement strengthened the artifact by aligning it with professional software engineering standards. The application now has clear architecture, secure credential management, improved filtering logic, and an interface that performs efficiently under larger workloads. The experience reinforced the value of clean architecture, reusability, and forward-thinking design choices in delivering software solutions that are both functional and maintainable.