Category Three Narrative

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

This artifact is a course catalog application adapted from a project I completed in CS-300: Data Structures & Algorithms course last fall. It also blends elements of a full stack application I created during a course in CS-340: Client-Server Architecture. The original project from CS-300 was a simple C++ console app that only gave users the ability to search for a course or print courses in alphanumeric order. This new version has a fully functional web interface that allows users to search for courses, explore the catalog, and even register and deregister from courses.

Justify the inclusion of the artifact in your ePortfolio.

This artifact serves as my enhancement for categories one and three. As I've already completed my narrative for category one, I will focus on this artifact's inclusion as it relates to category three. The original program stored the courses in a binary search tree, but I chose to implement this new solution using MongoDB and an HTML interface. I went through the process of loading the dataset into MongoDB and then writing a Python module that would handle calls to the database. This module is then invoked in the main application code in order to display the course catalog and retrieve course information. Additionally, I created a second database for students that allows unique student users to create accounts in order to register or deregister for classes by keeping an array of student IDs associated with each course in the catalog. This artifact demonstrates my ability to effectively use MongoDB to create solutions with multiple databases, all while providing a simple, intuitive interface for users.

Did you meet the course objectives you planned to meet with this enhancement in Module One?

This originally was meant to accomplish the following objective:

Employ strategies for building collaborative environments that enable diverse audiences to support organizational decision-making in the field of computer science.

This has been accomplished. This catalog allows students with no computer science knowledge to explore their course options and register with ease. Though the backend logic may not make sense to a non-computer science student, he/she is still completely capable of using this application. Additionally, though I did not plan for it in module one, I think this artifact also accomplishes the objective:

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.

I wasn't originally planning on having students input credentials to login to the registration platform. However, this inclusion takes the application to a more functional level and also provides security. Without student names and IDs, a malicious user could go into the system and just register the same user for a course many times. Additionally, the use of student credentials to register ensures that only students who have registered for a course are able to deregister for that course. Furthermore, the code features a high volume of input validation before passing arguments to database calls. Limiting the length of input and checking it for suspicious characters prevents most potential injection attacks.

Reflect on the process of enhancing and modifying the artifact.

When starting on this artifact, I was just attempting to transfer most of the functionality from the CS-300 C++ project into a Python/MongoDB format. I knew that I wanted to add some additional functionality, but I was not totally sure of what that might be. I came to develop the application as it is now over the course of much trial and error. I struggled quite a bit to integrate the student database into the project structure. It took a lot of debugging to ensure that CRUD operations were working for both the ‘students’ database and the ‘courses’ database. Furthermore, ensuring that the interface provided meaningful information to users when input was incorrect or invalid presented its own challenges. There is a lot of complicated logic in the application because there are many different cases that could occur, and strong development tries to anticipate unusual behavior. The next step was working with Plotly Express to create the degree audit function. This required the styling of the pie charts and the proper logic for communicating between the Majors.py file for the class definitions of each major and the Dash dataframe for selected rows to be included in the audit. The two different pie charts provide different information to users and once again enable diverse audiences to comprehend raw data in a way they wouldn’t be able to without visual aids. Working on the loading, querying, and manipulation of data required for this application to run was a great learning experience, and it improved my grasp of how the components of a full stack application can work together to create a finished product with rich and varied functionality.