**Artifact Enhancement Essays for Capstone Project – Christian Foster**

**Artifact 1: ABCU Course Planner (C++)**

**Category: Software Design and Engineering & Algorithms and Data Structures**

This artifact represents a console-based course planning system built in C++ that loads, stores, and organizes course information using a binary search tree. Originally developed in CS 300, this project was a key moment in my understanding of data structures and real-world applications of algorithms. The core functionality includes reading from a file, inserting nodes in a tree, and allowing the user to search and display course information.

The enhancements I implemented focused on cleaning up the code structure, correcting flawed logic in prerequisite validation, improving memory management, and modernizing file parsing techniques. For example, I replaced the original validatePrerequisites() logic with a correct recursive traversal of the tree and replaced manual string parsing with stringstream for safer input handling. I also replaced the use of dynamic memory allocation with stack allocation to avoid memory leaks and made user prompts clearer and input validation stronger. These changes significantly improved the maintainability and robustness of the software.

This artifact showcases my growth in both software design and algorithm development. I demonstrated the ability to use advanced data structures, like binary search trees, and applied solid software engineering practices, including clean code, separation of concerns, and user-centered design.

**Artifact 2: GameService Singleton (Java)**

**Category: Software Design and Engineering**

This artifact is a Java-based singleton class that manages game instances in a game application. The GameService class maintains a list of games and ensures no duplicates are added, while also managing unique IDs for players and teams. Initially written for an object-oriented programming course, this artifact reflects my understanding of design patterns and encapsulation.

My enhancements addressed issues like redundant logic in addGame(), inconsistent use of static fields, and lack of modularity. I simplified game lookup by reusing the getGame(name) method and replaced manual indexing with enhanced for-loops. Static state variables were encapsulated properly, and I adopted Java conventions such as the use of final and more meaningful comments. These changes improved the structure and maintainability of the code and aligned it more closely with Java best practices.

This artifact demonstrates my ability to apply design principles in real-world code, showing growth in clean, scalable software engineering. It highlights my ability to refactor code with maintainability in mind and to communicate clearly through thoughtful documentation and structure.

**Artifact 3: Animal Shelter MongoDB Script (Python)**

**Category: Databases**

This Python script connects to a MongoDB database, retrieves animal shelter data, and loads it into a pandas DataFrame. It was originally a simple script meant to demonstrate basic data retrieval. I chose this artifact to reflect my skills in handling databases and performing data operations in Python.

Enhancements included modularizing the code by separating the database connection logic into a function, improving security by encoding the password using quote\_plus(), and using the logging module instead of print() statements. I also added exception handling to gracefully manage connection errors and implemented a guard clause to prevent the program from failing if the database fetch fails.

This artifact shows my progress in writing secure, maintainable code for database access. It reflects my understanding of good software practices in scripting, especially around data security, error handling, and clean modular design. These enhancements meet the capstone outcomes related to database integration and secure coding practices.

**Conclusion**

Each of these artifacts has been enhanced to demonstrate my growth in key areas of computer science: software design, algorithm efficiency, and database development. Together, they show my ability to not only build functional code but also to refine and elevate it to professional standards.