Jordan Mitchell

5/28/2025

Professor Maciosek

# Milestone Three: Narrative

This artifact is a C++ console based Course Planner application originally developed in CS-300 Algorithms and Data Structures. It allows users to load course information from a CSV file into a Binary Search Tree (BST), search for specific courses, and view their prerequisites. The project demonstrates tree traversal, structured data handling, and user interaction through logic driven by a menu. The original version was created during the CS-300 course and functioned as a practical implementation of BST concepts. It was a great way to practice using and working with a basic data structure. The artifact provided an opportunity to apply algorithmic thinking to real world use cases like data lookup and sorting.

I selected this artifact for my portfolio because it strongly demonstrates my competency with algorithms and data structures. Using a Binary Search Tree to hold and look up course data is a basic example of how data structures work in practice. The changes I made to the original artifact show how I've grown and better understand how to design and apply algorithms. I added a sorting feature that uses a lambda function with std::sort to organize courses based on how many prerequisites they have. To do this, I had to walk through the tree, collect the data, and then sort it using the standard library. This helped me practice working with both custom data structures and built-in tools in C++. It also made the program more flexible and user-friendly. I made sure to improve the code organization and added input validation to ensure more robust user interaction. These changes are reflections of real-world practices, like defensive programming, modularity, and clean user feedback. This means the updated version of the

project now shows my stronger skills in expanding existing code and keeping C++ programs clean and easy to manage.

At this point I believe I've met the intended outcomes outlined in Module One. The enhancements align with the course outcomes related to designing and evaluating computing solutions using algorithmic principles. The addition of a sorting mechanism layered on top of BST traversal, combined with enhanced user input handling, shows how I used algorithmic thinking and kept the user experience in mind. I also made the program more reliable by stopping repeated data loads and checking that the user entered valid input, which helps it run more smoothly and be easier to use. My original plan is still the same because the changes I made clearly support my skills in data structures, writing control flow, and working with efficient algorithms. I think the project now does a better job of showing that I'm ready to work with real world software tasks.

Enhancing this artifact taught me how to add new features to an existing data structure without compromising its functionality. When I added the sorting feature, I had to focus on two things. First, walking through the tree while keeping it intact, and then changing the data format so I could sort it. Doing both helped me understand how to work with tree data and still use tools like std::sort. I got better at connecting custom structures with built-in library functions. This was a useful skill to practice, and it made the code more flexible overall. One of the challenges I encountered was restructuring the code to make the Node structure accessible outside the class, so I had to make sure the changes didn't mess with how the rest of the program worked while still allowing the new feature to function properly. Another challenge was implementing user input validation in a clean, repeatable way without cluttering the main logic. These technical challenges helped me refine my modular approach to enhancements. Overall, I learned how to

update older code in a careful way while still keeping it easy to read and use. I also took time to improve the existing documentation, which was already fairly clear. I worked to explain the purpose of each part of the code more effectively and added comments in places where they were missing. To the best of my ability, I made sure the comments would help someone new to the code understand how it works and why certain decisions were made.