For part 2 of the enhancement plan I added the Binary Sort Tree code from CS 300. The binary search tree was added to make efficient searching, insertion, updating, and deletion. This is added onto the existing infrastructure from the last enhancement of CRUD functionality. It also helps to demonstrate advanced data structure manipulation and algorithmic implementation in C++. The justification for why this was added as a part of the capstone project was to demonstrate my ability to design and implement complex data structures and algorithms in a structured, object-oriented way. Improvements from the original version include, replacing the simple crud storage model with a BST structure to efficiently organize and retrieve data, implementing a more robust exception handling, and organizing the program into modular files to better fit professional coding practices. These updates strengthened both the functionality and readability of the project, aligning it more closely with a more finished product.

I believe this meets the course outcomes for this enhancement based on first demonstrating the ability to implement the BST algorithm into my existing code base. Next, I applied modularization, encapsulation, and error handling to reflect good software design principles. Finally I analyzed the limitations of the original crud only code and enhanced it with algorithmic efficiency and data abstraction. During the process of creating this part of the enhancement, having to break down the parts for handling the crud operations while maintaining the node handling in the BST. One challenge I faced was the delete operation, which had some errors with handling node deletion cases based on children. Overall this enhancement was a good test of my abilities to add on to an existing code base and connect algorithmic design with practical data management.