Karina Jacuinde

CS 499 Computer Science Capstone

Instructor Joseph Martinez

November 22nd, 2025

Milestone 3 Narrative

For this enhancement, I continued improving the Course Catalog program that I originally created a few months ago for CS300. That project was written in C++ and used a Binary Search Tree to store courses and look up course information. For this milestone, I focused on the Algorithms and Data Structures category by enhancing the Python version of that same project. I picked this artifact because it gave me the chance to show what I've learned in terms of recursion, searching, and choosing the most appropriate data structures. I also felt like this project already had a clear structure from CS300, so it was a good base to enhance without being overwhelming.

The biggest change I made was creating a recursive function called get_all_prereqs() to gather every prerequisite for a course, not just the direct ones. I used a Python set to keep track of which courses had already been checked. This was simple but important because it prevented duplicates and potential loops. Once I finished that, I split the results into two groups: the direct prerequisites and the indirect ones. This made the output a lot cleaner and more helpful for the user. I first tried to do this without two groups, and for each course that had a prerequisite, sometimes a long list was printed, I realized this could be confusing to the user, that is when I decided to go back, make two groups, and make sure the first one clearly stated direct prerequisites, followed by indirect prerequisites.

```
Enter course number including prefix: CSCI400

CSCI400
Large Software Development
Prerequisites:
CSCI301 - Advanced Programming in C++
CSCI350 - Operating Systems

Indirect prerequisites:
CSCI101 - Introduction to Programming in C++
CSCI100 - Introduction to Computer Science




CSCI300 - Introduction to Algorithms
CSCI200 - Data Structures
CSCI101 - Introduction to Programming in C++

MATH201 - Discrete Mathematics
```

Once I finished this portion and was happy with the results, I had the most mind-blowing moment in this Capstone so far. As I was giving my code a third time look over, to ensure my BTS and dictionary were the most efficient choices for data structures, I looked at all the menu options in main.py, and I realized that I never even called my BinarySearchTree class, or any of the functions in it, I did not use it in the menu at all to do anything. I am strictly referencing the dictionary, and simply calling Python's built in sorted() function to print the alphanumeric list.

```
31
32      elif choice == "2":
33          if not data_loaded:
34              print("Please load data before continuing (Option 1)")
35              continue
36          print("\nCourse List:")
37          for number in sorted(courses_by_number.keys()):
38              course = courses_by_number[number]
39              print(f"{course.number}: {course.title}")
40
```

I had to pause for a moment and think. BTS made the most sense for my CS300 final project, because it was written in C++ (in the previous enhancement when I translated the same code to Python, I simply translated the tree as well and kept it), here, this version is in Python, which has simple to use, and efficient built-in methods, especially with the small data set for this project, we don't need a tree to sort, I'm not even calling it, I automatically just used the dictionary and Python's built in methods. I then went back to my load_data function in data.py, commented out any tree related code, then I went to models.py and commented out 55 lines of code, Nodes class and BTS class, and my in_order function that wasn't started, where I planned to print the list in order for this category (I had this method in the original C++ version). I then tested my project again and it works as intended without the tree being created.

Now I am facing a dilemma and would love some input. Do I keep my tree commented out? Or eliminate it all together? For now, I kept the BTS code, just commented out. I am stuck between keeping the tree, to show that I can implement one, I mean, this is the Data Structures and Algorithms category, or should I stick to what I feel is the best and most simple solution with cleaner code, and eliminate the tree? I am open to feedback on this.

For the course outcomes, I feel like with this enhancement, I met the ones I intended. I applied algorithmic principles and used recursion to solve a real problem. I also evaluated my design choices and noticed where a data structure wasn't serving a purpose. This week has helped me understand that picking the right data structure is just as important as writing the code for it. Overall, this enhancement taught me a lot, it forced me to think about why I chose a certain structure or algorithm, instead of just using them because they were the best choice in the original assignment. The biggest challenge right now for me is deciding to let go of, or keep the Binary Search Tree, because it feels like an impressive piece of the original project. Simplifying

my program and making sure every part serves a purpose has been a great lesson for me in this module, and that has made the program cleaner, easier to read, and more efficient.