James (JJ) Hatzibiros

CS 499 Module Four Journal

**1. Briefly describe the artifact. What is it? When was it created?**  
 This artifact is a Jupyter notebook I built in Spring 2024 for my CS-340 class. It connects to a MongoDB “companies” collection, runs several queries to filter and count data, and shows the results in tables. At first, it simply pulled records into Python and used loops to count or group the data.

**2. Justify the inclusion of the artifact in your ePortfolio. Why did you select this item? What specific components showcase your skills in algorithms and data structure? How was the artifact improved?**  
 I chose this notebook because it highlights both database work and core algorithm skills. This primarily demonstrates the Data Structures and Algorithms Enhancements part of my ePortfolio. Originally, the code asked MongoDB for all the data and then let Python loop through every record to count offices by state. In this milestone, I moved that work into MongoDB’s built-in aggregation pipeline stages: $match, $unwind, $group, and $sort, so the database handles the operations inside the server instead of in Python. I also added timing code before and after each query to prove the new method runs much faster. I also built a Binary Search Tree (BST) in Python, inserted my aggregation results into it, and wrote an in-order traversal to sort states by name. I then benchmarked the BST lookup against Python’s built-in sorted() to show I can implement and measure my own data structure. These steps demonstrate real work in both algorithms (the BST) and data structures (database indexes and tree nodes).

**3. Did you meet the course outcomes you planned to meet with this enhancement in Module Two? Do you have any updates to your outcome-coverage plans?**  
 This milestone addresses several of our course outcomes. It meets Outcome 3 (“Design and evaluate computing solutions…using algorithmic principles”) by moving data processing into MongoDB’s aggregation pipeline, using stages like $match, $unwind, $group, and $sort, and by building my own Binary Search Tree in Python to sort results. Also, it fulfills Outcome 4 (“Demonstrate an ability to use well-founded and innovative techniques…”) through adding timing benchmarks and database indexes to measure and improve performance. Lastly, it supports Outcome 2 (“Design, develop, and deliver professional-quality…communications”) because the notebook includes clear Markdown explanations and labeled code cells that make the enhancement easy to understand.

Course Outcomes:

1. Employ strategies for building collaborative environments that enable diverse audiences to support organizational decision-making in the field of computer science.
2. Design, develop, and deliver professional-quality oral, written, and visual communications that are coherent, technically sound, and appropriately adapted to specific audiences and contexts.
3. Design and evaluate computing solutions that solve a given problem using algorithmic principles and computer science practices and standards appropriate to its solution while managing the trade-offs involved in design choices.
4. Demonstrate an ability to use well-founded and innovative techniques, skills, and tools in computing practices for the purpose of implementing computer solutions that deliver value and accomplish industry-specific goals.
5. 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.

**4. Reflect on the process of enhancing and modifying the artifact. What did you learn as you were creating it and improving it? What challenges did you face?**  
 When I first timed the queries, I saw that the simple filters still took noticeable time, teaching me to measure before guessing where the slow spots are. Creating indexes was easy, but I learned that an index only helps if you query exactly on that field, otherwise MongoDB still scans all documents. Building the BST taught me how to write a tree data structure from scratch and how in-order traversal yields a sorted list. Benchmarking the BST against Python’s built-in sort showed me the trade-offs between custom code and optimized library functions.