Course: CYBR-330
Group Name: The Sleep Deprived
Date: 12-13-2025

1.) Introduction

The purpose of this project was to analyze and improve upon the efficiency of a Library Management System (LMS) by applying principles learned in the CYBR-330 (Algorithms and Data Structures) course. The benchmark project that was improved upon relied on linear search techniques to locate books. This can lead to performance issues as the library gradually increases. In order to address this problem, we designed an optimized LMS that benchmarks and compares multiple search algorithms against the initial implementation. The project focuses on understanding how different algorithmic choices can impact performance and scalability.

2.) Benchmark of Base System

The benchmark system was based on an open-source project created by a GitHub user that was practicing their coding skills. This project they created is a command-line based LMS written in Python. This original implementation stored book titles in a single Python list and relies on linear searching to perform operation such as borrowing and returning books. The characteristics of the benchmark system focuses on a single-file menu design for user navigation, performs searches using linear methods, has no separation between data, logic or algorithms and has no performance measurement or benchmarking built in.

The algorithm behavior in this benchmark in locating a book requires scanning through the list of books sequentially. Doing so, the time complexity in the search operations follows O(n) time and the worst case of this is in O(n^2) when searching and removing books. While the system they built functions ass intended, it does not benefit well in scalability and does not demonstrate optimized algorithmic behavior.
(Import screenshot here – or somewhere around here)

3.) Optimized Project Implementation

For our optimized implementation for the LMS, we transformed the original concept into a system that has been completely refactored and expanded. This was done to support multiple search algorithms, to create a more modular design and has built in benchmarking to demonstrate the measured time for each search type implemented (linear, binary, and hash). The key enhancements that have been implemented is an object-oriented design, external datasets loaded from a CSV file, and an automated performance timing and comparison of each type of search.

The linear search is used for a direct comparison to the original system. The binary search applies to sorted datasets with O(log(n)) complexity. The hash map lookup

uses book titles as keys with average O(1) lookup time.
(Import screenshot here – or somewhere around here)

4.) Performance Comparison

The optimized system was tested small and large datasets to evaluate performance scales as the number of records increases. In the original benchmark of the LMS it utilizes a Python list using an implicit linear search algorithm. There is no built-in benchmarking system, and the data size is relatively small and hardcoded.

While the optimized version of the original project includes a list, ordered list and hash map search methods. While the search algorithms themselves are based on linear, bilinear and hash searching methods. The time complexity is demonstrated, and the dataset size ranges from small to large via a CSV and not hardcoded. Benchmarking is also built into this optimized project.

Our observed results include that linear search performance degrades significantly over time as the dataset increases in size. Binary search, however, consistently outperforms linear search on sorted data. Hash map lookup is by far the fastest and most consistent in performance.
(Screenshots of Linear, Bilinear and Hash-map go here to show difference.)

5.) Reflections: Challenges

One of the main challenges was refactoring the original concept of the project by the original creator. That being the user focused interactive menu-driven system to a not as interactive benchmarking-focused implementation. The original LMS required more of the user input to control the program which made it an unsuitable benchmarking tool for strict performance testing between linear, bilinear and hash map lookups. We had to remove the user selection almost entirely and rewrite the code to perform a more predetermined search operation. However, the optimized system still needed to be able to perform the same operations as the original to ensure fairness in the benchmarking tests ran to show a good comparison between the original application and the optimized version.

6.) **Conclusion**

This optimized project successfully demonstrates the impact of algorithm selection on system performance. By benchmarking an original linear search based LMS (with changes in the code structure, but still follows original benchmark) implementation

against an optimized implementation using binary search and hash maps. We show clear and measurable improvements in efficiency and scalability as well. The optimized LMS provides a practical and real world example of algorithms and data-structure concepts being applied into a academic project to fulfill the objectives of the final project for this course.