Name: Steven Valdivieso Lemus

Class: CS-499-13167-M01

Date: March 30th, 2025

4-2 Milestone Two: Enhancement Two: Algorithms and Data Structure

Artifact:

The artifact is a Contact Management System originally created for the CS320 course, which I developed to manage contact information using CRUD operations. Initially, the system employed an ArrayList to store contact data, which worked but was not the most efficient for operations like searching, updating, and deleting contacts. The enhancement involves replacing the ArrayList with a HashMap, which will significantly improve the performance of these operations, reducing the time complexity from O(n) to O(1). The new version of the system will also include optimizations in the update and search algorithms to ensure that only the fields that need updating are modified.

Justification for Inclusion:

This artifact highlights my ability to apply algorithmic principles and data structure optimizations to a real-world software application. The improvement from an ArrayList to a HashMap directly addresses performance bottlenecks in the original system, aligning with key course outcomes. By enhancing both the data structure and the efficiency of the algorithms, the artifact demonstrates the impact of thoughtful design choices in real-world systems.

Data Structure Optimization: Replacing the ArrayList with a HashMap demonstrates my understanding of selecting the right data structure based on the performance requirements of the system.

Algorithmic Efficiency: Refining the update and search functions minimizes redundant operations, illustrating my ability to design efficient algorithms.

Scalability: The use of a HashMap enables the system to scale efficiently, especially as the number of contacts grows.

Specific Components Showcasing Skills:

Data Structure Optimization: The shift from an ArrayList to a HashMap will allow the system to handle CRUD operations in constant time (O(1)) instead of linear time (O(n)). This enhancement directly improves the performance of the system, especially when dealing with large amounts of data.

Algorithmic Efficiency: The update and search functions have been refined so that only the necessary fields are updated, and the search is performed using the contact ID as the key in the HashMap. This ensures that retrieval and modification operations are as efficient as possible.

Course Outcome Coverage:

I successfully met the course outcomes related to software design, particularly:

Course Outcome 3: I designed and evaluated a computing solution by optimizing the contact management system’s data structure and algorithms. This improvement illustrates my ability to apply algorithmic principles to enhance the efficiency of a real-world application while managing the trade-offs involved in design choices.

Course Outcome 4: I applied well-founded techniques and tools, such as selecting appropriate data structures and refining algorithms, to improve the system’s efficiency. This aligns with industry goals of delivering scalable, high-performance software solutions.

Reflection on the Enhancement Process:

The enhancement process provided valuable learning opportunities:

Optimization Of Performance: I gained a deeper understanding of the impact of selecting the right data structure for the task at hand. Replacing the ArrayList with a HashMap directly improved the system's time complexity for search, update, and delete operations, allowing for constant-time performance rather than linear-time performance.

Efficiency Of Algorithms: I learned how refining algorithms to work on only the necessary data can significantly enhance performance by minimizing redundant operations.

Challenges: One of the challenges I faced during this enhancement was ensuring backward compatibility with the existing code. Since the original system used an ArrayList and relied on linear searches, I had to carefully plan the transition to a HashMap without disrupting existing functionality. This required modifying the contact management logic while ensuring the updated methods still worked seamlessly with the rest of the application. I also had to test the new data structure thoroughly to avoid introducing new bugs, especially in edge cases where a contact might not exist.