Dennis Ward

01/29/2025

CS499

# Enhancement Two Narrative

The artifact selected for Enhancement Two is a mobile application designed for simple inventory management, originally developed in CS360. The initial implementation utilized local database storage and contained legacy code that required refactoring. While the project demonstrated effective use of Android development principles, it was designed as a single-user application for tracking inventory. Over time, the need for a scalable and team-oriented system became apparent. To transition the application toward production-level functionality, improvements focused on refining data structure management, modularity, and performance optimizations. Among the most pressing concerns was search performance, as the original implementation relied on inefficient list-based searching, causing slowdowns as the dataset grew. Enhancement Two addresses this issue by optimizing the search functionality in SearchViewActivity, ensuring faster and more scalable search operations.

To improve search efficiency, this enhancement introduced a dual HashMap approach, replacing the previous linear search method. The new implementation stores item data in two HashMaps: itemIdMap, which maps unique item IDs for constant-time (O(1)) retrieval, and itemNameMap, which stores lowercase item names for case-insensitive lookups. The previous search approach required iterating over the entire dataset (O(n)), leading to significant performance concerns for large inventories. The new structure ensures that ID-based searches are instantaneous, while name-based searches remain optimized but still dependent on iteration (O(n

* m)). These changes significantly improve the application's responsiveness, making it more suitable for managing larger datasets without performance degradation.

The algorithmic improvements within this enhancement focus on refining the filterItems() function, ensuring that search queries execute efficiently while maintaining real-time responsiveness. The ID-based search now operates in O(1) time, as the function directly retrieves an item from itemIdMap without requiring iteration. This is a substantial improvement over the previous list-based search, where even an exact ID match required scanning the entire dataset. For name-based searching, the function iterates over itemNameMap, performing substring comparisons against each stored key. While this method remains O(n * m) in complexity, it is still an improvement over the previous full list iteration approach.

Despite these improvements, name-based searching is still an area that could benefit from further optimization. To reduce search complexity, two potential solutions were considered. The first is implementing a Trie (Prefix Tree) structure, which would enable prefix-based lookups in O(m) time, significantly improving performance for partial name searches. A Trie organizes words in a character-based hierarchy, allowing for rapid query matching without requiring a full dataset scan. The second option involves utilizing SQLite Full-Text Search (FTS5), which would allow indexed searches within a database-backed text-search system. By leveraging an FTS5 virtual table, search operations could execute in near constant time (≈ O(1)), eliminating the need for manual string comparisons. Both approaches present viable paths toward further improving the search functionality in future iterations of the project.

This enhancement aligns with multiple course outcomes. It demonstrates proficiency in algorithmic problem-solving by replacing a list-based search (O(n)) with HashMaps (O(1)) for ID lookups. The transition from a linear search model to a hash-based retrieval system showcases an understanding of algorithmic trade-offs, prioritizing lookup speed at the expense of slightly increased memory usage. Additionally, evaluating alternative search optimizations such as Tries and SQLite FTS5 highlights an ability to assess advanced computing techniques for further scalability. Beyond algorithmic improvements, this enhancement also required clear documentation and structured explanations, ensuring that future developers can understand and maintain the updated search logic. The SearchHashMapsAnalysis document provides a detailed breakdown of time complexity and potential improvements, reinforcing the ability to communicate technical decisions effectively.

The optimizations made in Enhancement Two significantly improve search performance, particularly for ID-based queries, which now execute in O(1) time. While name-based searches remain O(n * m), the system is now more efficient and scalable than before. Future enhancements, such as implementing a Trie-based structure or SQLite FTS5, could further reduce search time while maintaining real-time responsiveness. Overall, this enhancement successfully demonstrates algorithmic problem-solving, efficiency trade-offs, and the application of computing techniques in a real-world mobile development environment.