Dennis Ward

01/29/2025

CS499

Enhancement Two Narrative

Enhancement Two focused on optimizing StockSense's search functionality to improve performance and scalability. Originally developed in CS360, the application relied on list-based searching (O(n)), which caused slowdowns as the dataset grew. This enhancement introduced a dual HashMap approach, significantly improving search speed and system efficiency.

Algorithmic Improvements

To optimize search performance, I implemented two HashMaps:

- **itemIdMap:** Maps unique item IDs for O(1) retrieval.
- **itemNameMap:** Stores lowercase item names for case-insensitive lookups.

Previously, the application required iterating over the entire dataset for every search (O(n)), leading to performance issues. With this optimized structure, ID-based searches now execute in O(1) time, while name-based searches remain O(n * m) but are still more efficient than the original full list iteration.

Enhancements to SearchViewActivity

The filterItems() function was restructured to leverage the new HashMaps, ensuring real-time search responsiveness. This refactor significantly improved query execution times, especially for large datasets.

Further Optimization Considerations

While this enhancement significantly improved efficiency, I explored future optimizations:

- **Trie (Prefix Tree):** Could reduce name search complexity to O(m) by organizing words in a hierarchical structure.
- **SQLite Full-Text Search (FTS5):** Would enable indexed text searches, improving query performance to near O(1).

These approaches provide potential next steps for further improving StockSense's scalability.

Technical Skills Demonstrated

This enhancement showcases several key software engineering principles:

- **Data Structures & Algorithm Optimization:** Transitioned from O(n) list searches to O(1) HashMap lookups.
- **Software Maintainability:** Refactored `SearchViewActivity` for better modularity and performance.
- **Efficient Query Processing:** Restructured filterItems() function for faster real-time searching.
- **Scalability Considerations:** Evaluated Trie and SQLite FTS5 for future enhancements.

Challenges & Learning

Debugging search performance issues was a major challenge, as legacy code dependencies made restructuring difficult. I utilized Postman to test API calls, which streamlined debugging and improved database interaction reliability. Additionally, balancing memory usage (HashMaps) vs. query efficiency reinforced my ability to analyze computational trade-offs.

Course Outcomes Addressed

- **Algorithmic Problem-Solving:** Applied efficient data structures to improve search performance.
- **Software Engineering Best Practices:** Refactored code for maintainability and efficiency.
- **Technical Communication:** Documented search optimizations in SearchHashMapsAnalysis.