Milestone Three Narrative

Christopher Carnell

Southern New Hampshire University

CS-499 Computer Science Capstone

Professor Maryann Krupa

November 20, 2024

Milestone Three Narrative

The artifacts I selected, DatabaseManager.java, DataDisplayActivity.java, and WeightEntry.java—are core components of my WeightWatcher Android application developed during my CS-360 course.

* DatabaseManager.java manages all database operations, including user registration, login, and CRUD operations for weight entries. It interacts with the SQLite database to store and retrieve user data, ensuring data persistence across sessions.
* DataDisplayActivity.java is responsible for displaying the user's weight entries and goal weight. It allows users to add new entries, set or change their goal weight, and track their progress toward fitness goals. This activity retrieves and displays data from the database, providing users with an up-to-date view of their weight history.
* WeightEntry.java is a fundamental data model representing individual weight entries, including the date and weight value. It serves as the backbone for tracking user progress over time, facilitating storage, retrieval, and manipulation of weight data throughout the application.

I selected these artifacts for my ePortfolio because they collectively showcase my ability to optimize algorithms, data structures, and implement efficient programming practices within an application. Enhancing these components demonstrates my proficiency in several key areas:

* Optimizing Database Queries and Data Handling:
  + *DatabaseManager.java*: Implemented projections to retrieve only necessary data, reducing overhead. Continued using parameterized queries to prevent SQL injection and improve security. Introduced a caching mechanism for goal weights to minimize database calls and enhance performance.
* Algorithm Optimization and Efficient Data Structures:
  + *DataDisplayActivity.java*: Implemented efficient sorting algorithms to maintain the order of weight entries. Introduced a method to find the correct insertion point for new entries, reducing computational overhead. Utilized the List interface for flexibility in choosing the most appropriate data structure, optimizing data handling for operations like insertion and retrieval.
* Optimized Data Handling and Use of Modern Java APIs:
  + *WeightEntry.java*: Optimized data handling by storing dates as LocalDate objects using the Java Time API. This improved the efficiency of date operations and comparisons, reduced overhead from parsing date strings multiple times, and minimized parsing errors.
* Exception Handling and Resource Management:
  + Added comprehensive exception handling around database operations in *DatabaseManager.java* and *WeightEntry.java* to enhance robustness. Ensured proper closing of database cursors and connections to prevent memory leaks.
* Performance Improvements:
  + *DataDisplayActivity.java*: Optimized RecyclerView updates by using specific methods like notifyItemInserted() instead of notifyDataSetChanged(). Reduced unnecessary processing by avoiding full list re-sorting after each insertion.
* Code Quality and Maintainability:
  + Improved code readability through detailed documentation, method-level comments, and consistent formatting across all three artifacts. Encapsulated logic within private methods to promote code reuse and clarity.

These enhancements significantly improved the performance, reliability, and user experience of the application, aligning it with industry best practices and demonstrating my proficiency in software optimization and development.

Through these enhancements, I achieved the following course outcomes:

* Course Outcome 3: Demonstrated the ability to design computing solutions using algorithmic principles by optimizing database queries, data handling algorithms, and data structures, thereby improving the application's efficiency and performance.
* Course Outcome 4: Showed proficiency in using innovative techniques, skills, and tools in computing practices by implementing caching mechanisms, utilizing the Java Time API and modern programming practices, and refining data structures for better performance.

Throughout the process of enhancing and modifying the artifacts, I gained valuable insights and deepened my understanding of several key areas in software development. Working on DatabaseManager.java, I developed a deeper appreciation for database optimization techniques. I learned how projections and parameterized queries can significantly improve database performance and security. Implementing caching mechanisms for frequently accessed data demonstrated how caching can greatly enhance application responsiveness by reducing unnecessary database calls.

In enhancing DataDisplayActivity.java, I delved into algorithm efficiency and data structures. I realized how sorting algorithms and data insertion methods directly impact application performance. By implementing custom comparison logic to sort complex data types, I improved the efficiency of data handling. Additionally, I appreciated the flexibility provided by coding against interfaces like List, which allows for more adaptable and maintainable code.

Working on WeightEntry.java allowed me to recognize the benefits of modern APIs. Leveraging the Java Time API over older date handling classes provided advantages such as better immutability, thread-safety, and more intuitive methods. This modernization aligned the code with current industry standards and improved overall code quality.

Across all artifacts, I recognized the critical role of proper exception handling and resource management in preventing application crashes and memory leaks. Implementing comprehensive exception handling ensured that the application could gracefully handle unexpected situations without compromising stability. Moreover, I understood the importance of optimizing RecyclerView updates in DataDisplayActivity.java. By using specific notify methods, I significantly improved UI performance. Learning how RecyclerView handles data changes enabled me to leverage its features for better efficiency and a smoother user experience.

I also learned the importance of robust error handling practices. By throwing exceptions when invalid data is encountered, I promoted early detection of issues and ensured data integrity throughout the application.

During this process, I faced several challenges. One challenge was balancing performance and complexity, particularly in DatabaseManager.java. I needed to ensure that the caching mechanism did not introduce unnecessary complexity or potential synchronization issues. In DataDisplayActivity.java, maintaining data order while preventing UI lag was a key concern. I had to ensure that weight entries remained in the correct order without significant overhead, balancing between sorting after each insertion versus maintaining a sorted list. It was also crucial to ensure that RecyclerView updates did not cause noticeable lag or jank in the UI, which required testing different update methods to find the most efficient approach.

In WeightEntry.java, ensuring compatibility and date format consistency was a significant challenge. I needed to make sure that changes did not break other parts of the application that depended on this class. Maintaining consistent date formats throughout the application, especially during database interactions, was essential to prevent parsing errors and data inconsistencies. Another challenge was avoiding redundant data retrieval. I had to identify where data could be cached versus where fresh data was necessary to maintain accuracy, ensuring optimal performance without sacrificing data integrity.

To overcome these challenges, I adopted a strategy of incremental implementation and testing. I implemented changes incrementally across all artifacts, testing each enhancement thoroughly to ensure it worked as intended without side effects. This approach allowed me to identify and address issues early in the development process. I also conducted extensive research and consulted industry best practices on database optimization, caching strategies, list management, RecyclerView optimization, and modern Java APIs. This research informed my decisions and helped me apply proven techniques to improve the application.

In DataDisplayActivity.java, I employed iterative development. I tested various approaches to insertion and updating, assessing their impact after each modification. Based on observations and performance measurements, I refactored the code to achieve the most efficient solution. Finally, I conducted thorough testing and updated dependent classes to ensure compatibility. After implementing changes in WeightEntry.java, I verified that all functionalities worked correctly. I reviewed and updated other classes that interact with WeightEntry to accommodate the changes, ensuring that the application remained stable and functional.

By reflecting on this process, I recognize how these enhancements have not only improved the application's performance and reliability but also enhanced my skills as a developer.