Milestone Four Narrative

Christopher Carnell

Southern New Hampshire University

CS-499 Computer Science Capstone

Professor Maryann Krupa

November 30, 2024

Milestone Four Narrative

In the third enhancement of my WeightWatcher Android application, I focused on optimizing the database schema and improving input validation to enhance security and user experience. The primary artifacts I worked on are:

1. **DatabaseHelper.java**: This class extends SQLiteOpenHelper and is responsible for creating and managing the application's SQLite database schema. Initially created during my CS-360 course, it defines the tables, columns, and handles database creation and version upgrades.
2. **SmsPermissionActivity.java**: This activity handles SMS permission functionality and collects users' phone numbers for SMS notifications. It was initially developed to prompt users for SMS permissions and store their phone numbers, ensuring they receive timely notifications about their weight tracking progress.
3. **DatabaseManager.java** and **DataDisplayActivity.java**: These classes were also updated to ensure consistent handling of database operations and input validation throughout the application.

I selected these artifacts for inclusion in my ePortfolio because they collectively demonstrate my ability to optimize database structures, implement secure coding practices, and enhance user experience through robust input validation. The specific components that showcase my skills and abilities in software development include:

* **Database Schema Optimization in DatabaseHelper.java**:
  + I added constraints like NOT NULL, UNIQUE, and CHECK to enforce data integrity.
  + Implemented foreign key constraints with ON DELETE CASCADE and ON UPDATE CASCADE actions to maintain referential integrity between tables.
  + Created indexes on frequently queried columns to improve query performance.
* **Input Validation in SmsPermissionActivity.java**:
  + Implemented phone number validation using regular expressions to ensure users enter valid phone numbers for SMS notifications.
  + Enhanced user experience by providing immediate feedback and guidance on correct input formats.
  + Improved error handling to prevent application crashes and inform users appropriately.
* **Exception Handling and Security Enhancements**:
  + Replaced instances of e.printStackTrace() with proper logging using Android's Log class.
  + Caught specific exceptions to handle different error scenarios effectively.
  + Ensured that sensitive information is not exposed in logs or error messages, aligning with secure coding practices.
* **Consistent Handling of Database Operations**:
  + Updated DatabaseManager.java to handle NULL values appropriately, especially for fields like goal\_weight.
  + Modified DataDisplayActivity.java to correctly prompt users to set their goal weight upon first login if it's not already set.

These improvements significantly enhanced the application's performance, security, and user experience. They align with industry best practices and demonstrate my proficiency in advanced software development techniques, particularly in database management, input validation, and secure coding.

Through these enhancements, I achieved the course outcomes I planned to meet:

* **Course Outcome 2**: Demonstrated the ability to design and develop professional-quality software by optimizing the database schema and improving input validation mechanisms. These changes enhanced the application's reliability, performance, and user experience.
* **Course Outcome 3**: Showed the ability to design computing solutions using algorithmic principles by implementing efficient database structures, constraints, and indexes to improve data handling and query performance.
* **Course Outcome 4**: Displayed proficiency in using innovative techniques and tools in computing practices by applying advanced database optimization strategies, implementing robust input validation, and enhancing exception handling mechanisms.
* **Course Outcome 5**: Developed a security mindset that anticipates and addresses potential vulnerabilities by replacing insecure exception handling practices with secure logging, preventing information leakage, and ensuring sensitive data is protected.

These enhancements allowed me to fully meet my outcome-coverage plans for this module, reinforcing my understanding of database optimization, secure coding practices, and their critical roles in software development.

Enhancing and modifying the artifacts for the third enhancement was an insightful journey that deepened my understanding of database optimization, input validation, and secure coding practices. As I worked on optimizing the database schema in DatabaseHelper.java and improving input validation in SmsPermissionActivity.java, I learned the significance of meticulous planning and attention to detail in software development. The process reinforced the importance of aligning application logic with database structures to ensure seamless functionality.

One of the key lessons was recognizing how changes in one part of the application could have unintended consequences elsewhere. For instance, updating the database schema to set the goal\_weight field to NULL by default required corresponding adjustments in the application logic to handle NULL values appropriately. This experience highlighted the interconnectedness of different components within an application and the need for comprehensive testing after making changes.

Implementing robust input validation also emphasized the balance between security and user experience. Crafting an effective validation pattern for phone numbers involved considering various international formats and ensuring that the validation logic was neither too strict nor too lenient. This taught me the value of user-centric design, where security measures must be implemented without compromising usability.

During the enhancement process, I faced the challenge of inadvertently introducing new bugs into the application. A significant issue arose when new users were not being prompted to enter their goal weight upon first login. This problem emerged after modifying the database schema to set goal\_weight to NULL by default, whereas the application logic was checking for a value of -1 to trigger the prompt. Consequently, the condition failed, and users were not prompted as intended.

Overcoming this challenge required thorough code review and strategic use of debug statements. I revisited the DatabaseManager.java and DataDisplayActivity.java classes to trace how the goal\_weight value was being handled throughout the application. By inserting debug statements, I was able to monitor the values at different execution points, which revealed that the application was not correctly interpreting NULL values.

To resolve the issue, I updated the getGoalWeight() method in DatabaseManager.java to return a Double object instead of a primitive double, allowing it to accommodate NULL values. This change ensured that when goal\_weight was NULL, the method would return null, and the application could accurately determine that the user had not set a goal weight. In DataDisplayActivity.java, I modified the condition to check if goalWeight was null or less than or equal to zero, thereby correctly prompting the user when necessary.

This challenge underscored the importance of aligning application logic with database structures and the need for comprehensive testing after making changes. It also highlighted the effectiveness of code reviews and debugging in identifying and resolving issues. By systematically analyzing the code and utilizing debug statements, I was able to pinpoint the root cause of the problem and implement a solution that restored the intended functionality.