Module 7-1: Final Project: Enhancement One: Software Design & Engineering

Stephanie S. McClurkin

Southern New Hampshire University

CS-499: Computer Science Capstone

Professor Federico Bermudez

February 19th, 2025

Briefly describe the artifact. What is it? When was it created?

The artifact I chose is a weight-tracking app that I originally developed in Android Studio during my CS-360 course. This app allows users to log their weight, set goals, and track their progress over time. The original version includes features like daily weight tracking, goal setting, SMS notifications, and database management for storing user data. Its primary purpose is to help users visualize their progress and stay motivated as they work toward their weight goals.

Justify the inclusion of the artifact in your ePortfolio. Why did you select this item? What specific components of the artifact showcase your skills and abilities in software development? How was the artifact improved?

I chose this artifact because it provides an opportunity to showcase the skills I've developed in software design and engineering during my time at SNHU. The original project gave me a strong foundation to build on, and it shows my ability to work with important Android components. For example, implementing a RecyclerView to display data efficiently and using SQLite for database management demonstrates my ability to integrate popular functionality into an app. These components show my understanding of building user-friendly applications that solve real-world problems, all while following core software development principles.

For my enhancement, I chose to convert the code from Java to Kotlin, a more modern language – and the preferred language for Android development. This enhancement reflects my ability to adapt to industry standards and demonstrates flexibility in working with multiple programming languages. The conversion improved the app's maintainability by streamlining the code and taking advantage of Kotlin's features, such as null safety, which helps prevent runtime errors. I also simplified code structures, such as getter and setter methods, to make the application more readable and easier to maintain.

In addition to the language conversion, I updated the app's dependencies, compile SDK, and Gradle configurations to support API level 35 – which provides stronger encryption and better permission handling. This ensures compatibility with the latest Android features and devices, further future-proofing my app. These enhancements demonstrate my ability to modernize legacy code, optimize software for maintainability, and align development with evolving industry standards.

Did you meet the course outcomes you planned to meet with this enhancement in Module One? Do you have any updates to your outcome-coverage plans?

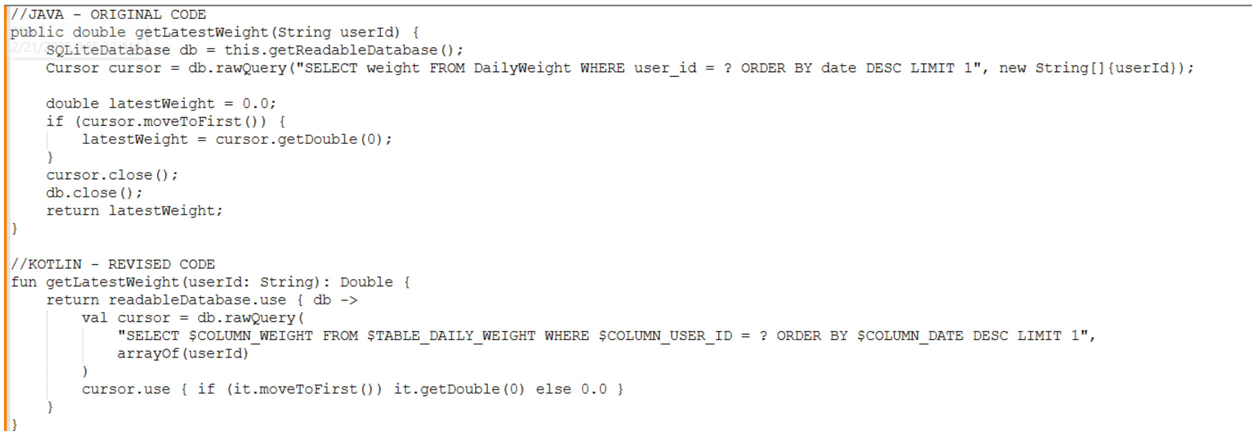
Course Outcome 1: Employ strategies for building collaborative environments that enable diverse audiences to support organizational decision-making in the field of computer science.

Even though I worked on this project independently, I made sure the code is easy to understand and maintain for anyone who might work on it in the future; this is important because it supports building software in a collaborative setting. One way I did this was by improving documentation with clearer comments, especially in DatabaseHelper.kt, where I added comments to clarify where the database handles user weight records and optimized SQLite queries. Adding clear and straight-forward comments makes it easier for future developers to expand or modify the app without running into confusion or potential data issues. This makes sure that future developers can quickly understand its functionality. Well-documented code is important for effective collaboration, making the app easier to maintain and grow.

Next, I also focused on making the code itself easier to read and manage. A good example of this is the getLatestWeight() function. In the original Java version, I had to manually open and close the database and cursor objects, which made the code longer and made errors more likely to occur. With Kotlin, I simplified the code with the use function, which automatically handles resource management. This minimized the lines of code, which in turn reduced the risk of memory leaks and makes the function easier to maintain. These updates show that I can write clean, well-documented code that is easy to understand and build upon. This is an important skill in any collaborative development environment.

Course Outcome 4: Demonstrate an ability to use well-founded and innovative techniques, skills, and tools in computing practices for implementing solutions that deliver value and accomplish industry-specific goals.

The swap from Java to Kotlin wasn't just about switching languages; it was a way to modernize the project and follow industry standards, which directly supports the fourth course outcome. Since Kotlin is the preferred language for Android development, making this change ensures my app stays compatible with future updates and best practices. It also makes the code easier to maintain and expand over time, which is especially important since Google is prioritizing Kotlin in Android libraries (Mehra, n.d.). One of the biggest ways this enhancement aligns with the fourth course outcome is through improving efficiency with Kotlin's modern features. A great example is the getLatestWeight() function. In the original Java version, I had to manually open and close the database and cursor objects, which meant extra lines of code and a higher risk of errors. In Kotlin, I was able to simplify it using the use function, which automatically handles resource management. This means I don't have to worry about manually closing database connections, which makes the function cleaner, easier to read, and safer to use.



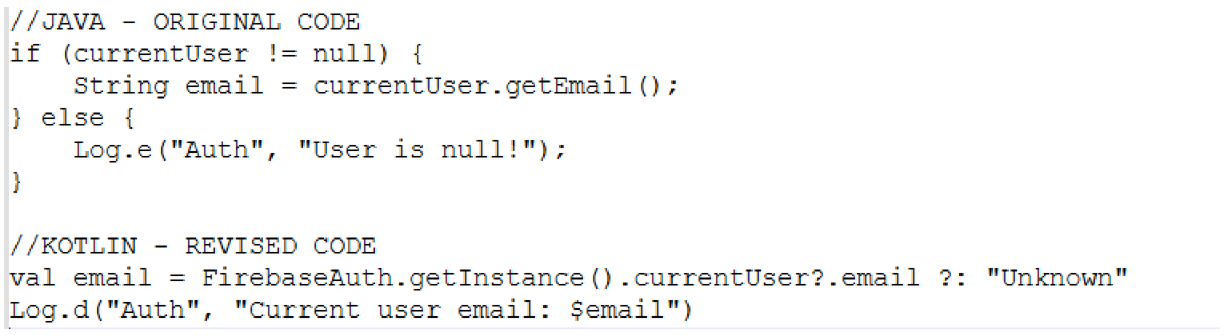
```
//JAVA - ORIGINAL CODE
public double getLatestWeight(String userId) {
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = db.rawQuery("SELECT weight FROM DailyWeight WHERE user_id = ? ORDER BY date DESC LIMIT 1", new String[]{userId});

    double latestWeight = 0.0;
    if (cursor.moveToFirst()) {
        latestWeight = cursor.getDouble(0);
    }
    cursor.close();
    db.close();
    return latestWeight;
}

//KOTLIN - REVISED CODE
fun getLatestWeight(userId: String): Double {
    return readableDatabase.use { db ->
        val cursor = db.rawQuery(
            "SELECT $COLUMN_WEIGHT FROM $TABLE_DAILY_WEIGHT WHERE $COLUMN_USER_ID = ? ORDER BY $COLUMN_DATE DESC LIMIT 1",
            arrayOf(userId)
        )
        cursor.use { if (it.moveToFirst()) it.getDouble(0) else 0.0 }
    }
}
```

The screenshot above shows exactly how Kotlin helped streamline the getLatestWeight() function. Kotlin takes care of managing resources, making the code more efficient and reducing the chances of memory leaks. This supports the fourth course outcome because it really shows how modern programming techniques can lead to a more maintainable piece of software. Writing clean and efficient code makes sure the app runs better and is easier to work with.

Beyond converting the language from Java to Kotlin, I also updated the app's dependencies and compile SDK to API level 35, which ensures that it stays compatible with the latest Android development standards. This is important because it means the app can take advantage of newer features, such as security improvements, and it will have an overall better performance. This also supports the fourth course outcome because it shows I can keep up with industry trends and make sure my app is using the best tools available.

Course Outcome 5: Develop a security mindset that anticipates adversarial exploits in software architecture and designs to expose potential vulnerabilities, mitigate design flaws, and ensure privacy and enhanced security of data and resources.

Security is a big part of software development, and I made several improvements to ensure my app follows best practices and reduces potential vulnerabilities. One of the biggest changes I made was improving how user authentication data is handled to prevent crashes and unexpected behavior, which supports the fifth course outcome. Before converting the app to Kotlin, there was a security risk related to how authentication data was handled in MainActivity. If user authentication data wasn't properly initialized, the app could crash with a NullPointerException, which is both a hassle for users and a security concern. With Kotlin, I was able to take advantage of null safety features to prevent this type of problem before it ever happens. Preventing crashes helps reduce unexpected behaviors that could later be exploited as potential security vulnerabilities.



```
//JAVA - ORIGINAL CODE
if (currentUser != null) {
    String email = currentUser.getEmail();
} else {
    Log.e("Auth", "User is null!");
}

//KOTLIN - REVISED CODE
val email = FirebaseAuth.getInstance().currentUser?.email ?: "Unknown"
Log.d("Auth", "Current user email: $email")
```

The screenshot above shows how I improved the way user authentication data is handled. The original Java version required manual null checks, meaning if I forgot to check for null, the app could crash. Kotlin, however, can check automatically and if there is no authentication data available, it can default to unknown instead of crashing. This change improves both security and app stability by reducing the chances of unexpected failures.

Finally, I updated the app to API level 35, which helps ensure that it is following Google's latest security standards. This update includes stricter permission handling, stronger encryption policies, and better data protection (Singh, 2025). By making these enhancements, I showed that I can develop software with a security mindset while anticipating potential vulnerabilities and designing features that help protect user data, all satisfying the fifth course outcome.

In wrapping up the course outcome section, I believe this enhancement successfully satisfies the three outcomes I've highlighted. By improving documentation and code readability, I've demonstrated strategies that support collaboration, aligning with the first course outcome. The transition to Kotlin, along with updating the app’s dependencies, ensures the project follows modern industry standards, reinforcing course outcome four. Finally, by implementing security-focused coding practices - such as null safety, authentication improvements, and error handling - I’ve improved my security mindset and addressed potential vulnerabilities, meeting the fifth course outcome.

Reflect on the process of enhancing and modifying the artifact. What did you learn as you were creating it and improving it? What challenges did you face?

During this enhancement, I learned how to modernize legacy code and resolve issues in an efficient and systematic way. Converting the app from Java to Kotlin taught me a lot about Kotlin’s key features, like null safety and concise syntax, which made the code cleaner and less error-prone. These updates not only improved the app’s overall quality but also made the code easier to read and maintain.

One of the bigger challenges I faced was updating the app's dependencies and API level to meet the latest Android standards. Upgrading to API level 35 required extra testing to ensure all the app’s features worked properly with the new Android version. I also ran into several build errors, like resource-linking failures and Gradle JDK compatibility issues, which at first felt overwhelming. However, I was able to research potential fixes, try out different solutions, and learn from the process. These moments of troubleshooting really helped me understand Android Studio’s build tools and Gradle configurations in much greater depth.

While I’ve tested the app’s functionality and addressed the issues that came up during the conversion process, I know there’s always room for improvement. I plan to continue testing the app and exploring opportunities to integrate even more advanced Kotlin features where they make sense. Staying open to ongoing evaluation and improvement is something I’ve come to appreciate more through this process.

Enhancement Summary

Enhancement 1: Conversion from Java to Kotlin (Software Design & Engineering)

- Converted the entire codebase from Java to Kotlin to align with modern Android development best practices.
- Improved code maintainability and readability using Kotlin’s null safety, use function, and short syntax.
- Updated dependencies, Gradle configurations, and the compile SDK to API level 35 to ensure compatibility with the latest Android versions.
- Final testing confirmed the conversion was successful, with no errors or issues.

Enhancement 2: Predictive Weight Trend Algorithm (Algorithms & Data Structures)

- Implemented an algorithm to estimate future weight trends based on historical data, providing users with text-based predictions.
- Integrated MPAndroidChart to visually display only recorded weight value in a line graph.
- Added a custom date entry option for weight logs instead of restricting users to the current date.
- Resolved previous graphing issues that prevented accurate trend visualization.
- Removed predicted weight line trend from the graph; shelved for a future update due to visualization issues.
- Final testing confirmed that predictions are now accurate, and all features function correctly.

Enhancement 3: Import/Export Feature (Database & Cloud Integration)

- Integrated Firebase Firestore functionality to allow users to import and export weight data.

- Added Firebase authentication to securely manage user accounts and access control.
- Fixed UI refresh delay issues that previously caused imported data to not display immediately.
- Final testing confirmed that all import/export functionality works as expected, with no major issues remaining.

How to Run the Project

GitHub Repository & Branch Information

The latest version of my project - with the weight prediction algorithm enhancement - is located on the enhancement3 branch of the GitHub repository.

- Repository Name: CS499weighttrackingapp
- Repository Link: https://github.com/smcclurkin0312/CS499weighttrackingapp
- Latest Version (Enhancement 3): enhancement3 branch

The latest version of my project can be ran using the following Git commands:

- git clone https://github.com/smcclurkin0312/CS499weighttrackingapp.git
- cd CS499weighttrackingapp
- git checkout enhancement3

System Requirements

- **Android Studio:** 2024.2.2
- **Kotlin Version:** 2.0.21
- **Gradle Version:** 8.10.2
- **Minimum SDK:** 31 (Android 12)
- **Target SDK:** 35 (Android 14)
- **MPAndroidChart 3.1.0**

- **Firebase Firestore & Authentication**

Android Studio

- Open Android Studio and choose to "Open an Existing Project."
- Open the cloned project folder.
- Sync Gradle by selecting Sync Project with Gradle Files from the top navigation bar in Android Studio.
- Click Run to launch the app on Android Emulator (Pixel 6, API 35).

Testing & Evaluation

To make sure everything worked properly after converting the app from Java to Kotlin, I tested all key features to confirm there were no errors, crashes, or unexpected behavior. First, I built and ran the project in Android Studio (2024.2.2) with the latest Gradle and SDK updates, and everything compiled successfully with no warnings or issues. I then tested the app on the Pixel 6 emulator (API 35) to make sure it launched and functioned correctly.

Once I confirmed the app was running correctly, I went through all of the features to check that nothing broke during the transition to Kotlin. I tested user login, registration, and authentication to make sure user accounts were handled correctly. I also checked weight logging and goal setting, making sure that all calculations were accurate and that data was stored and retrieved correctly. Everything worked as intended with no apparent issues caused by the swap to Kotlin.

Since Kotlin includes built-in null safety features, I focused on testing scenarios where user data might be missing to see how the app would handle it. In the old Java version, a NullPointerException could have caused a crash if authentication data wasn’t set up correctly. In the Kotlin version, I tested these same cases, and the app handled them without breaking or crashing. This helps enforce that the switch to Kotlin improved stability and performance of the app. All tests passed successfully, confirming that the Kotlin conversion improved maintainability without introducing any issues.

Sources

Mehra, D. (n.d.). *Reasons why Google and Android developers incline to Kotlin Android development.* Binmile. https://binmile.com/blog/reasons-why-google-and-android-developers-incline-to-kotlin-android-development/

Singh, T. (2025, January 13). *Development differences between Android API level 35 (Android 14) and API level 36 (Android 15).* LinkedIn. https://www.linkedin.com/pulse/development-differences-between-android-api-level-35-14-singh-2lquc/