Module 7-1: Final Project: Enhancement Two: Algorithms & Data Structures

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 to enhance is my weight-tracking app, originally developed in Android Studio during my CS-360 Mobile Architect & Programming course. The app was created to help users log, track, and visualize their weight progress over time while setting personal weight goals. The original version provided functionality for daily weight tracking, goal setting, and SMS notifications to help users stay motivated and on track.

The original application stored weight entries using SQLite and allowed users to manually enter their weight data, which was displayed in a RecyclerView list. The original app lacked any predictive or analytical functionality beyond storing and displaying recorded data. For my second enhancement, I decided to improve the app by implementing a predictive weight trend algorithm. This feature takes historical weight data and uses it to predict future weight trends, giving users a unique insight into the weight progress.

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 for enhancement because it allowed me to demonstrate my skills in algorithm design and data structures while significantly improving the app's functionality. Implementing a predictive weight trend algorithm helps solve a real-world problem by allowing users to see potential future trends based on their actual weight data. Predictive analytics is widely used in mobile applications, and integrating this feature aligns with industry practices for fitness and health-tracking apps. In addition to implementing the algorithm itself, this enhancement also demonstrates my ability to work with dynamic graphing, UI improvements, and structured data processing.

The enhancement I implemented introduced several noteworthy improvements. First, I developed the weight prediction algorithm to calculate future weight trends based on historical data. The prediction results are displayed in a dropdown menu, allowing users to select a time frame (1-4 weeks) and receive an estimated future weight based on their current recorded data. While working on these enhancements, I also made UI refinements, such as adding a scrolling functionality to the weight history list for smoother navigation when viewing previous entries and dates. Next, I included manual date selection for weight entries, allowing users to input specific weight data for past dates. This helps make sure that the prediction algorithm has the most accurate, user-controlled data to generate more reliable predictions.

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 2: Design, develop, and deliver professional-quality communications that are coherent, technically sound, and appropriately adapted to specific audiences and contexts.

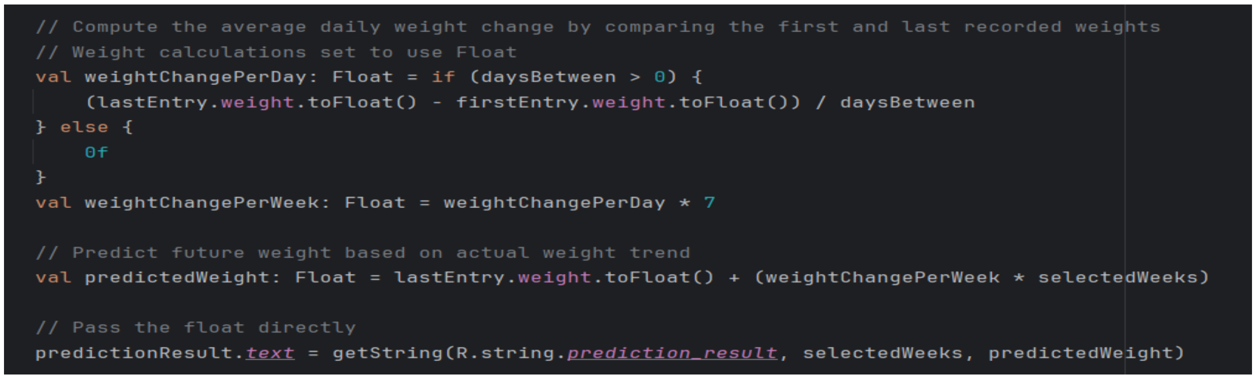
For this enhancement, I wanted to make sure that my implementation was not only functional but also well-documented and easy to follow. One way I did this was by improving code readability with structured comments and documentation, ensuring that future developers (or even my future self) could easily understand how the predictive weight trend feature works. I included clear comments in key areas, such as explaining how the algorithm processes past weight changes to predict future trends, how MPAndroidChart is used to visualize recorded weight trends, and how stored weight entries are retrieved and structured. This demonstrates my ability to write technically sound, professional-quality documentation that is clear and informative.

Beyond ensuring my code, comments, and technical documentation were well-structured, I also focused on making sure the user-facing side of this enhancement was clear and intuitive. Since the prediction results needed to be easy to interpret, I integrated MPAndroidChart to visually display recorded weight trends instead of just listing numbers. The prediction results themselves are provided in the drop-down as text-based calculations. The graph dynamically plots weight data with a clear x-axis for time and y-axis for weight, allowing users to quickly recognize patterns in their weight trends. This enhancement improves the app's usability by making weight trends and predictions easier to understand at a glance, aligning with best practices in professional software development.

Course Outcome 3: Design and evaluate computing solutions that solve a given problem using algorithmic principles and data structures.

The predictive weight trend algorithm is the centerpiece of this enhancement and it demonstrates my ability to design and evaluate computing solutions using algorithmic principles and structured data. This algorithm takes a user's past weight entries and estimates future trends based on their recorded data. The prediction is based on averaging weight changes over time, which helps account for short-term fluctuations and provides a realistic trend line instead of a simple projection. To make sure the algorithm was efficient and scalable, I used sorted lists and dynamic data structures so that new weight entries could be processed quickly, without extra recalculations. This makes sure that calculations are performed only when necessary, which helps to keep the app responsive. The prediction logic focuses on actual progress rather than static assumptions, making the results both accurate and useful for the user. This demonstrates my ability to apply structured data principles to build efficient, real-world computing solutions.

The screenshot below show how MPAndroidChart is used to visually display recorded weight history. The prediction results, however, are presented as text-based calculations in the dropdown menu, allowing users to estimate their future weight based on their recorded data.



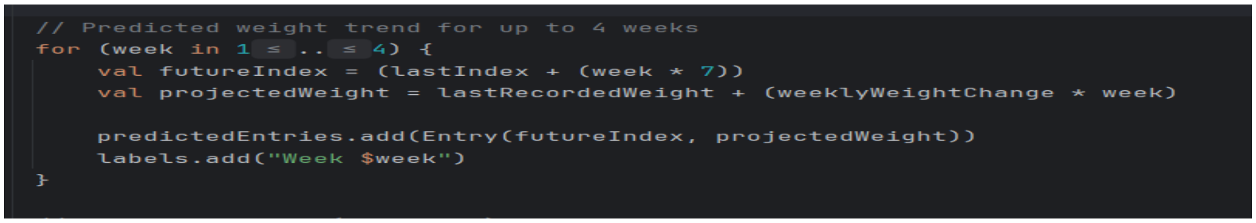
```
// Compute the average daily weight change by comparing the first and last recorded weights
// Weight calculations set to use Float
val weightChangePerDay: Float = if (daysBetween > 0) {
    (lastEntry.weight.toFloat() - firstEntry.weight.toFloat()) / daysBetween
} else {
    0f
}
val weightChangePerWeek: Float = weightChangePerDay * 7

// Predict future weight based on actual weight trend
val predictedWeight: Float = lastEntry.weight.toFloat() + (weightChangePerWeek * selectedWeeks)

// Pass the float directly
predictionResult.text = getString(R.string.prediction_result, selectedWeeks, predictedWeight)
```

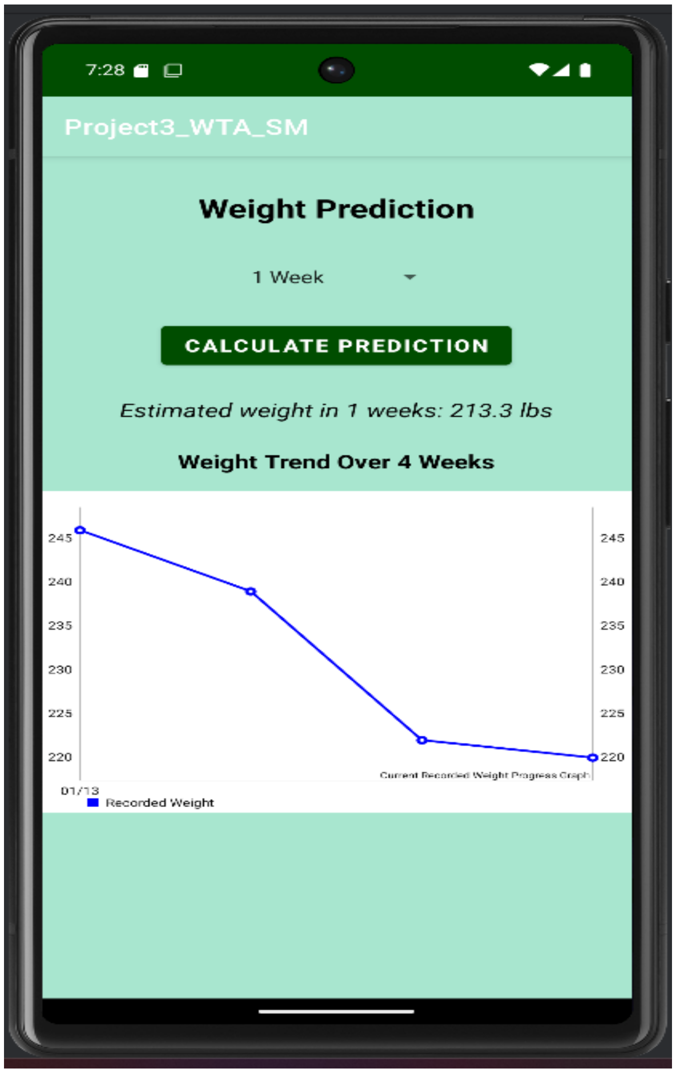
By structuring the algorithm this way, I helped make sure that users receive realistic, data-driven predictions that adjust dynamically as they log new weight entries. Once the algorithm was in place, I checked its accuracy by testing it against both steady and fluctuating weight trends to make sure the predictions made sense and weren't too rigid or too extreme. This testing process demonstrates my ability to analyze and refine algorithmic solutions to ensure they provide reliable results.

I also focused on making sure users could easily interpret their recorded weight trends by integrating MPAndroidChart to visually display their historical weight data. Instead of relying on plain numbers, users can now see a clear, interactive graph that plots both their historical weight data. This helps them quickly recognize patterns and adjust their goals accordingly. The first screenshot below shows how users are provided with a text prediction of their anticipated weight loss. The second screenshot shows the results of this prediction at 1 week, and how MPAndroidChart is used to visually display recorded weight data. This graph-based visualization ensures that users can see trends at a glance, making this feature functional and user-friendly.



```
// Predicted weight trend for up to 4 weeks
for (week in 1 ≤ .. ≤ 4) {
    val futureIndex = (lastIndex + (week * 7))
    val projectedWeight = lastRecordedWeight + (weeklyWeightChange * week)

    predictedEntries.add(Entry(futureIndex, projectedWeight))
    labels.add("Week $week")
}
```



7:28

Project3_WTA_SM

Weight Prediction

1 Week

CALCULATE PREDICTION

Estimated weight in 1 weeks: 213.3 lbs

Weight Trend Over 4 Weeks

Course Outcome 4: Demonstrate an ability to use innovative techniques, skills, and tools in computing practices to implement computing solutions that accomplish industry-specific goals.

One of the biggest improvements in this enhancement was integrating MPAndroidChart to create a dynamic weight trend visualization. Instead of just displaying numbers in a list, this enhancement applies modern data visualization techniques to show recorded weight entries as a visual graph, making it easier for users to track their past progress over time. MPAndroidChart is commonly used in mobile apps for analytics and trend tracking, so adding it to my project directly demonstrates my ability to integrate professional-grade tools that align with industry expectations for mobile applications handling user data.

Another way I demonstrated industry-relevant skills was by adding manual date selection for weight entries. Instead of restricting users to logging their weight for the current date, they can now enter past weights, ensuring that the prediction model is working with the most complete dataset possible. This feature improves the accuracy and reliability of the predictions by allowing the algorithm to analyze a broader, user-controlled data range. It also promotes better data integrity, since users can correct mistakes or fill in missing entries, which is an important part of maintaining clean, structured datasets in professional applications. By implementing these enhancements, I showed that I can use modern computing tools and best practices to improve both functionality and usability in mobile app development.

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?

The process of implementing this enhancement provided useful insights into algorithm design, data handling, and UI integration. I gained a better understanding of weight prediction modeling and how to process historical user data to make informed estimates about future trends. One of the biggest challenges was ensuring that the weight prediction algorithm produced results that were both realistic and useful. The algorithm needed to properly account for fluctuations in weight trends, rather than following a strict, linear trajectory. To fix this, I adjusted the calculations to average weekly changes, creating a more balanced prediction curve instead of an overly rigid trend line that could lead to inaccuracies. Testing is ongoing to ensure the algorithm functions as expected.

I also ran into some struggles when integrating MPAndroidChart while keeping the UI responsive. A lot of time was spent fine-tuning the graph to ensure both the x-axis and y-axis displayed correct information. Originally, I planned to make the graph dynamically update based on the user's selected time frame (1-4 weeks), but I ultimately opted for a static 4-week trend to provide a more stable, informative view. The implementation was mostly a success, but I still want to conduct additional testing to ensure the x-axis labels display the correct dates consistently.

Finally, UI layout and scrolling behavior turned out to be more challenging than expected. I ran into issues where the weight history list was causing overlapping elements and missing UI components. After multiple iterations, I optimized the layout using ConstraintLayout and NestedScrollView, ensuring all elements remained properly positioned and fully accessible. This enhancement significantly improved the app's capabilities and user experience, while also demonstrating my ability to apply algorithmic principles, handle structured data, and integrate visualization tools effectively.

A final project update for this particular question - ultimately, I struggled to include the predicted weight line on the graph. The main issue was finding a way to populate the predicted data when the page loaded. Without accurate weights to reference, the graph ended up displaying meaningless trends, often skewing unrealistically downward. I tried troubleshooting this by checking predicted weights at page load, but ran into authentication issues that I couldn't find a workaround for. After multiple failed attempts, I decided to shelve the predicted weight line for a future update. However, I still believe I have fully met the requirements for the algorithms and data structures category by implementing my predictive weight loss or gain algorithm.

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 the predictive weight trend algorithm worked the way it was supposed to, I ran multiple tests to check accuracy, functionality, and how well it handled different types of user input. The main goal was to make sure the algorithm correctly analyzed past weight entries, made reasonable predictions, and presented those predictions accurately from a dropdown menu. I also tested how the app handled different user interactions to make sure it was stable and responsive.

I focused on functional testing to confirm that the algorithm correctly calculated weight trends based on past data. I tested different timeframes (1–4 weeks) to see if the predictions adjusted based on real recorded weight changes. I also checked that the prediction updated correctly when new weight entries were added or modified, making sure that the app was always working with the most up-to-date information. Then, I ran accuracy tests to make sure the predictions made sense in different scenarios. I tested the algorithm against users with steady weight loss, steady weight gain, and fluctuating weight trends to see how well it handled different patterns. One thing I wanted to check was whether the predictions were reasonable instead of extreme, so it was important to verify how the app handled small fluctuations in weight data. I also made sure the weekly weight change calculation didn't create unrealistic trends, like drastic jumps in projected weight when the actual recorded data had only small variations.

Test Case	**Input**	**Pass or Fail**
Consistent weight loss	User logs weight loss of 1lb per week	Pass – prediction follows a smooth downward trend
Fluctuations in weight loss	User logs weight loss of varying up/down while trending downward overall	Pass – prediction eventually smooths out and follows the general trend
User selects prediction(s) for week(s) 1-4	User selects week(s) 1-4 in the dropdown prediction menu	Pass – prediction accurately displays trend of week(s) 1-4
User has insufficient data (1 weight entry)	User logs a single weight entry and tries to predict weight	Pass – app displays a insufficient data available to predict weight message
User logs a weight input with an incorrect format	User inputs incorrect weight format with non-numeric characters	Pass – invalid input entry is prevented

Beyond making sure the algorithm worked, I also focused on testing how well the data was displayed. MPAndroidChart needed to correctly plot recorded weight entries without any errors or misrepresented information. I checked the x-axis labels to make sure dates displayed properly and confirmed that the graph remained clear and readable. Another key test was seeing if the graph dynamically updated when new weight entries were added - users should be able to see an updated trend line right away without having to restart the app. Finally, I did error handling and stability testing to make sure the app responded properly to edge cases. If a user had fewer than two weight entries, the app displayed an “insufficient data” message instead of trying to generate a prediction. I also tested cases where weight entries were missing or entered incorrectly, confirming that the app handled them properly without crashing.

Ultimately, the enhancement worked as expected. The algorithm produced realistic weight trend estimates, MPAndroidChart displayed the data correctly, and the app remained stable and user-friendly. These tests confirmed that the enhancement improved the app’s functionality while keeping predictions easy to understand and interpret.

Sources

Atukorala, S. (2021, November 18). *How to use MPAndroidChart in Android Studio!* Medium. https://medium.com/@SeanAT19/how-to-use-mpandroidchart-in-android-studio-c01a8150720f

GeeksforGeeks. (2025, January 29). *NestedScrollView in Android with example.* https://www.geeksforgeeks.org/nestedscrollview-in-android-with-example/

Babajide, O., Hissam, T., Anna, P., Anatoliy, G., Astrup, A., Martinez, J. A., Oppert, J. M., & Sørensen, T. I. A. (2020). A machine learning approach to short-term body weight prediction in a dietary intervention program. *Computational Science – ICCS 2020: 20th International Conference, Amsterdam, The Netherlands, June 3–5, 2020, Proceedings, Part IV*, *12140*, 441–455. https://doi.org/10.1007/978-3-030-50423-6_33