Paolo Valenti

Dr. Ogoh

CS-499 Computer Science Capstone

27 September 2025

Milestone Three Narrative – Enhancement Two

As mentioned in the previous narrative, the artifact being used for the algorithm and data structure enhancement is the weight tracker app from CS: 360 Mobile Architecture & Engineering. The purpose of the app was to allow local user accounts to track weight entries for specific dates, after entering a username and password to login. Along with being able to compare a set goal weight to their current weight on a specific entry. The other functions include editing, deleting, and adding multiple weights. With this enhancement though those listed weights are now sorted in charts. The app was created around April of 2024, towards the end of CS:360 course.

The weight tracker app with an algorithms and data structure enhancement is the perfect addition to the ePortfolio. When you consider how UI, interactivity, and simple data displays can keep users engaged, adding charts to a weight tracker app is a notably useful enhancement. I selected this item because continuously working on this mobile project not only fits within my plan for this capstone, but helps me sharpen my skills for a future career in mobile programming. So using this app for the second enhancement is great way to put both my coding knowledge and ability to select common sense improvements into action. The specific components of this artifact that showcase my skills in algorithms and data structures is the way that the existing data is now visually displayed to the user in two distinct ways. On the same screen where the user can add, edit, and remove weights, they can select the view chart button. The view chart button takes them to the chart screen that has the line chart displayed by default. If they select the toggle option it switches to a bar chart. The data displayed on each chart corresponds to the original weight list, a weight that corresponds to each date. There is also a goal line that matches the user's set goal weight, so it's easy to compare progress to the goal weight for each chart. These data structures improve the app drastically, instead of having user's read line by line the charts communicate that same info faster and in a simpler manner. So if someone is on the go they can check their progress at a glance then close the app until they need it next.

The course outcome I planned on meeting for this enhancement was course outcome two, specifically with the focus being on, "visual communications that are coherent, technically sound, and appropriately adapted to specific audiences and contexts." Charts visually communicate bulks of data in a easy to analyze manner, so it's adapts well to the purpose of the

app: add current weight, compare progress so far/compare to goal weight, and then exit app until next time. In reviewing the work done so far, the data chart enhancement also fits course outcome three. Course outcome three is, “Design and evaluate computing solutions that solve a given problem using algorithmic principles and computer science practices and standards appropriate to its solution while managing the trade-offs involved in design choices.” The given problem here is the app provides a list of weights that can be read, but not easily analyzed. So the computed solution is to have that same data sorted into two data structures, a line chart and data chart. As for tradeoffs, there aren’t really any obvious ones that come to mind here.

The process for adding in the line/bar charts was interesting, as I had to make use of a specific open-source charting library that I have not used in the past, the MPAndroidChart made by Phillip Jahoda. As per the description from the MPAndroidChart github it is a, “powerful Android chart view / graph view library, supporting line- bar- pie- radar- bubble- and candlestick charts as well as scaling, panning and animations” (PhilJay, 2019). Some of these imports to list a few were com.github.mikephil.charting.charts.Barchart and com.github.mikephil.charting.charts.Barchart which adds each type of chart respectively, and com.github.mikephil.charting.components.LimitLine which allowed me to set goal lines. Thankfully, the imports offered in the library were not too challenging to include as there is a lot of documentation and examples that the MPAndroidChart GitHub had that made the enhancement development simpler to complete. To summarize a bit on the exact changes to the code, I first had to convert the stored data into a workable format to be shown in the charts. The conversion done was from dates stored per user to epoch-days integers. Then the weight entries were parsed through both types of charts, with ranges set using the oldest and newest entries. I then scaled the y-axis based off the min and max weights, forcing the graph to snap toward the nearest 5 unit intervals for readable grid lines. Finally I added a goal weight line using the user’s set goal, and styled the graphs line types and bar widths and set the toggle switch.

The biggest issue for this entire enhancement was easily setting the x-axis properly. I kept running into issues where either the dates shown on the axis were entirely irrelevant, then I accidentally removed them all, then I accidentally printed so many that the axis itself was unreadable. The fix came from forcing some strict settings for how the axis was setup instead of having the date ranges populate automatically. First, I forced the x-axis to work using whole-day units and forcing proper spacing between the days. Second, I used a custom ValueFormatter so only date labels that matched weight entries appeared on the x-axis. But, even when I set this the weight entries were off my a day for each tick on the chart. So third, I made the charts use UTC time zone. Prior to using UTC, entries being put in on Sep 18th and 20th mistakenly matched ticks lining up to the 17th and 19th instead. Much like in enhancement one, I learned the devil is in the details. I thought the most challenging part would be understanding how to use the MPAndroidChart library to make the charts, instead it was just setting the x-axis to work the way I needed it to. But ultimately, I am pleased with how the enhancement worked out and I learned that there are a lot of specialized libraries that programmers to use to accomplish the goals they

need, it just takes a bit of time and research. I also learned that in matters relating to time you have to be very specific with how you set up your code. That way you can prevent offsets in regards to hours or days..

References:

PhilJay. (2019, March 20). *GitHub - PhilJay/MPAndroidChart: A powerful 🚀 Android chart view / graph view library, supporting line- bar- pie- radar- bubble- and candlestick charts as well as scaling, panning and animations.* GitHub. https://github.com/PhilJay/MPAndroidChart?tab=readme-ov-file#report

GeeksforGeeks. (2020, May 20). *LocalDate ofEpochDay() method in Java with Examples*. GeeksforGeeks. https://www.geeksforgeeks.org/java/localdate-ofepochday-method-in-java-with-examples/

Marcel50506. (2016, November 25). *MPAndroidChart x-axis date/time label formatting*. Stack Overflow. https://stackoverflow.com/questions/40803233/mpandroidchart-x-axis-date-time-label-formatting?utm_source=chatgpt.com