Damian Duross CS-499 Computer Science Capstone

9/28/2025 4-2 Milestone Three: Enhancement Two: Algorithms and Data Structure

For this module, I am continuing to work on the same artifact – my CS-360 Android Studio weight tracker app. I am very excited about the way this week's augmentations turned out, and I think they'll be a great addition to my ePortfolio – they reinforce the course outcomes I was already meeting with the first augmentation, such as demonstrating innovative use of tools and techniques, and additionally show off algorithmic problem solving.

This week's augmentations were a new challenge for me, but quite rewarding – getting to see the line graph once it was successfully imported was immensely satisfying. That being said…the implementation was quite difficult, and I had to do a lot of research to make it happen. Afterwards, the algorithm was actually quite easy to implement…however, testing it required some planning.

To start, I decided on creating a dialog-based graph – I didn't want the graph to be on a new screen, because this could be distracting or inconvenient for the user. Rather, the graph will show as a dialog box, and once closed the user will still be on the main dashboard. I created the layout resource for the dialog box (dialog_weight_graph.xml) and included a LineChart view from MPAndroidChart. I additionally left a TextView placeholder to be used for the forecast output for my algorithm.

Next, I had to make sure MPAndroidChart dependencies were properly imported. I had to add it to libs.versions.toml, so I could then add it in my app/build.gradle. I had to resolve some build errors by syncing, cleaning and rebuilding – I also had to go into settings.gradle to

add the jithub.io repository. Once that was working, I added the showGraphDialog() to my DashboardActivity, inflating the dialog_weight_graph.xml. I added the logic to populate the LineChart with the entries from the database, as well as stylizing it to make it more readable. I added a few weights using the FAB, tested it, and the line graph seemed to be working quite well!

Once the line graph was working, I moved on to the linear regression algorithm for goal prediction. This involves calculating slope and intercept from existing data, estimating the day the user reaches their goal, converts the day index to a calendar date, and displays it in the forecast view beneath the line graph. Now, the issue I had was testing the information – all of the dates I added for the line graph were the same date, so I couldn't use that data to test my algorithm. What I did was hardcode some entries in DashboardActivity.java that had different dates and weights, and showed a clear pattern…these changes seemed to create a more reasonable prediction. I have commented the lines out, but saved them for the sake of testing functionality (lines 138-168).

Ultimately, I now have a dynamic graph with a predictive goal date fully integrated into the dashboard. I think the app is coming together quite well; I am still a little daunted by next week's database augmentations, but if my implementations are even half as successful as they were in this module, I'll be quite happy.