Module: SDA
Assignment No.: 5 (final project)
Student Name: Pavel Urusov

Project Report

Introduction

For my final project, I wanted to create something based on my hobby. I am an avid audiophile and record collector, and it is well known that styli used to play vinyl records have limited lifetime (which generally varies between 200 and 800 hours based on the stylus type and the vertical tracking force). I have been using analogue means (mainly a mechanical counter and a notepad) to track stylus usage, but it is not particularly convenient, especially if a user, like me, has multiple cartridges / styli / turntables. In my opinion, this could be achieved with the help of a dedicated mobile application.

Initial Requirements

This section lists the initial requirements/specifications for the application based on the project proposal that was submitted back in January.

— **Application name:** Stylus Timer

— **What does it do:** This application allows audiophiles and record collectors to track the lifetime / usage of their phono cartridges and styli.

— **Application behaviour:** The application would allow users to add multiple cartridges / styli. User would be able to specify the characteristics of the stylus, and the application would suggest the appropriate lifetime based on the stylus type and the vertical tracking force (e.g. 800 hours for a microline stylus used with a VTF of 1.2g or 200 hours for an elliptical stylus used with a VTF of 2g). User would be able to use buttons in the UI to quickly add a record side (0.36 hours) to the counter. The application would display a progress bar showing how much life the stylus has left, and show a warning message when the stylus is approaching end of life. The application would use a backend to quickly download specifications of popular cartridges / styli from the internet.

— **APIs used:** The application will use the specified Android API classes in the following manner:

Databases	The application will use an SQLite database to store user data.
Networks	During the first launch, the application will connect to the backend to download a list of popular / widely used cartridges / styli. On each launch, the application will connect to the backend to see if there is an updated list available, and download it if there is an update. If the user adds a cartridge / stylus that is present in the list, the application will get its specifications from the backend. All data will be downloaded in the JSON format.

Research

I did not consider similar applications although I suspect some might exist because the idea is not exactly new. I did study some literature on the topic, mostly manufacturer's suggested lifetime figures which are widely considered to be too optimistic by experienced vinyl record collectors. For example, Ortofon (the leading cartridge manufacturer today) simply says that phono styli designed for Hi-Fi use can have lifespans of up to 1000 hours:

- https://www.ortofon.com/support/support-hifi/faq-installation/

Audio-Technica, another large cartridge manufacturer, also provides some figures in their literature:

- https://docs.audio-technica.com/eu/VM_Cartridges_EN.pdf

Unfortunately, independently sourced information on exact lifetime of different styli types is very fragmented and mostly exists on forums dedicated to vinyl playback. The following two forum topics were invaluable in providing information used in the app:

- https://www.lencoheaven.net/forum/index.php?topic=31535.0
- https://www.vinylengine.com/turntable_forum/viewtopic.php?f=19&t=22894

At this stage I had to abandon my initial idea of suggesting different lifetime figures based on the differences in tracking force simply because I could not find any evidence-based information confirming that tracking force has a significant effect on the service of a stylus. The values that are used in the app by default are a bit on the conservative side to make sure that users do not damage their records by using worn styli. However, I decided that I also need to provide a facility for overriding default settings in case the user is feeling adventurous.

Design

The following pictures show initial sketches and basic navigation flows for the application.

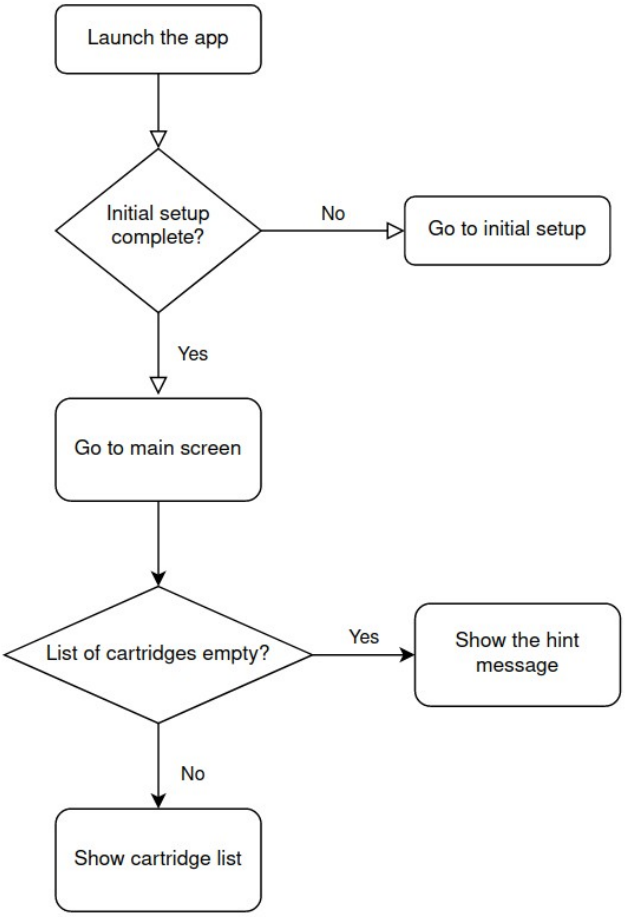


Figure 1: Basic flow — launching the application

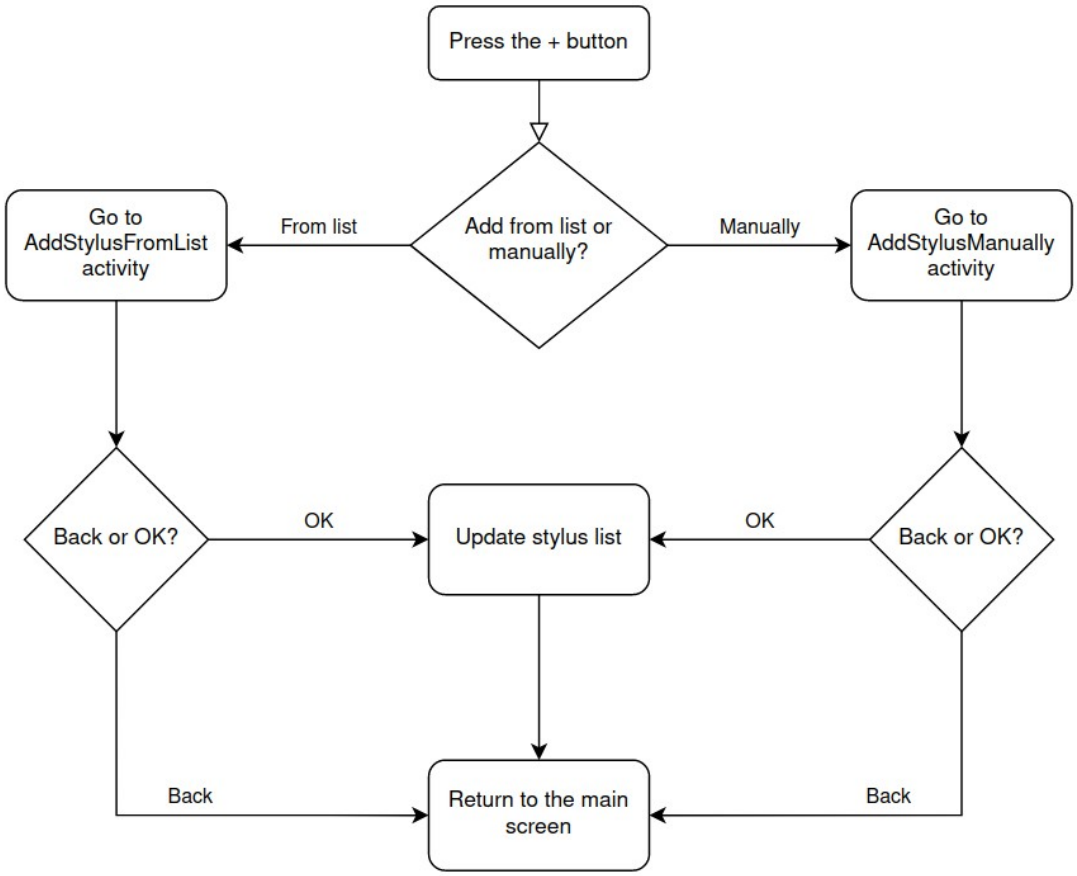


Figure 2: Basic flow — adding a cartridge

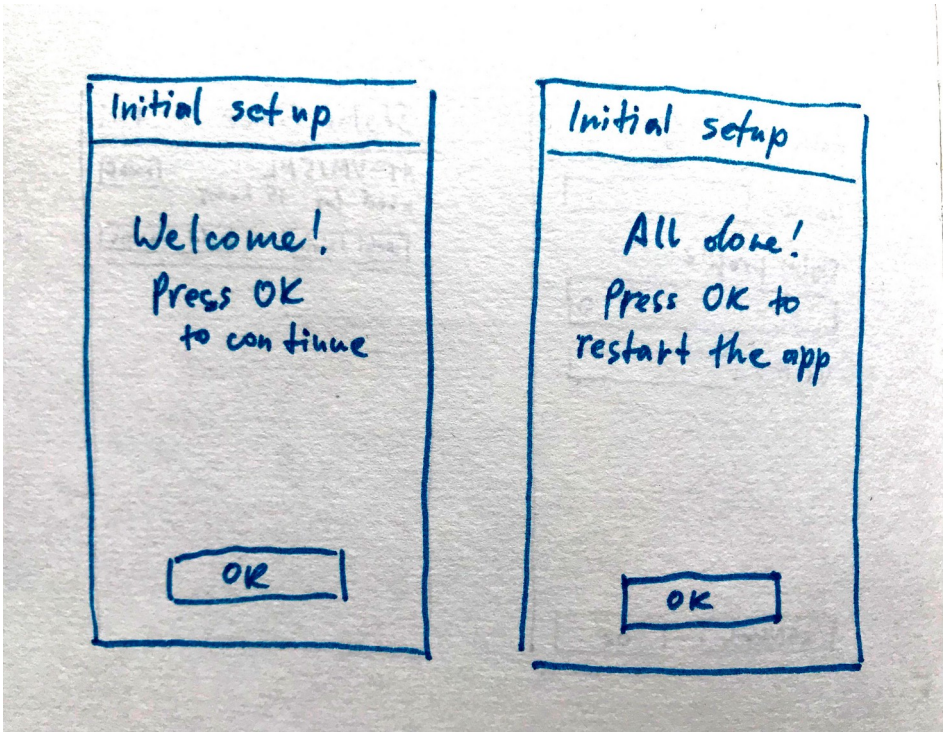


Figure 3: Hand-drawn sketches — initial setup screen

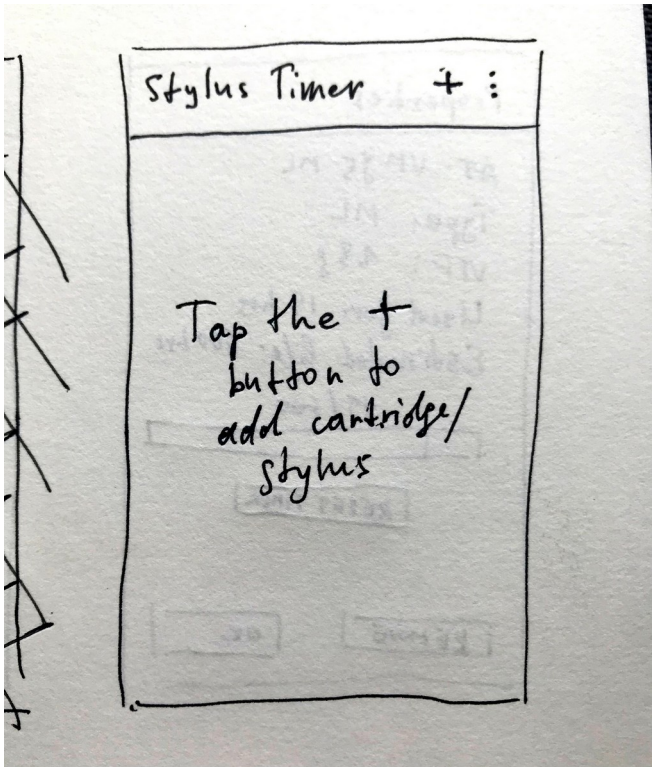


Figure 4: Hand-drawn sketch — main screen (list empty)

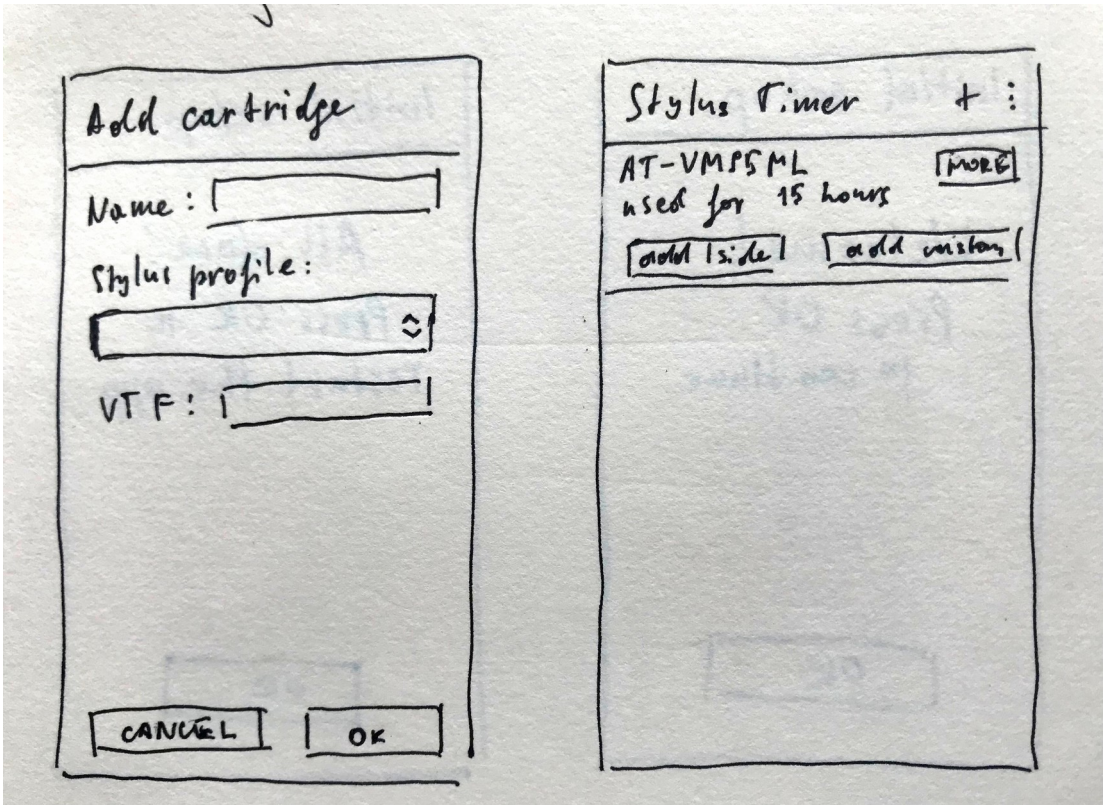


Figure 5: Hand-drawn sketches — "Add cartridge" dialogue and main screen (list not empty)

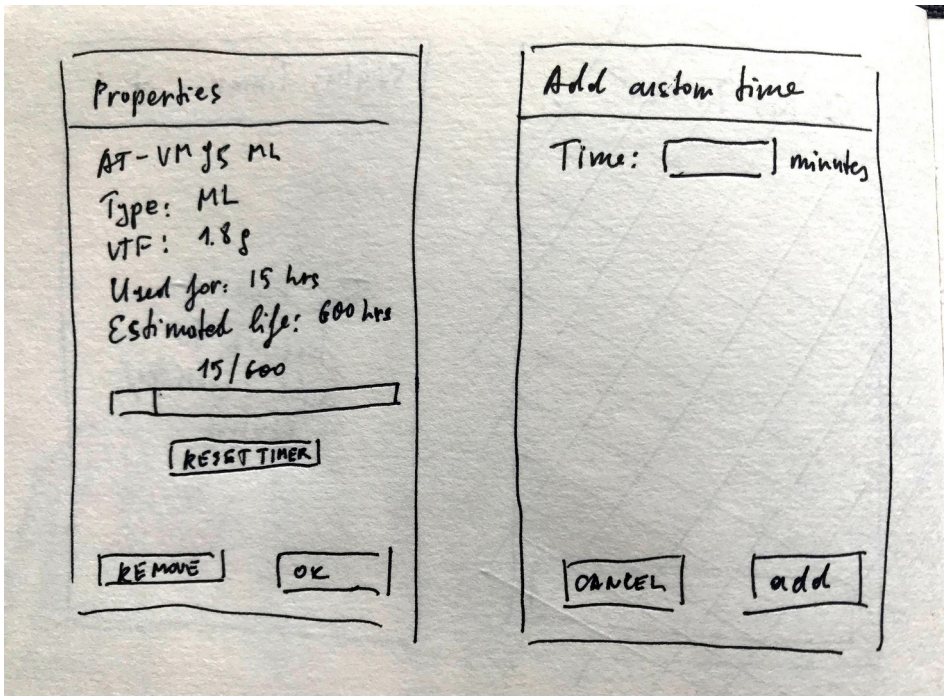


Figure 6: Hand-drawn sketches — cartridge properties screen and custom time dialogue

App architecture

In this section I want to explain some of the choices I made regarding the architecture of the app.

Initially I wanted to create the following separate components:

- Data classes (Stylus and StylusProfile);
- STimerPreferences – a singleton class responsible for working with app preferences;
- DataHelper – a class responsible for working with local SQLite database;
- FirebaseHelper – a class responsible for working with Firebase RealTime Database.
- UI components (basically, the rest of the application).

In my opinion, this architecture makes a lot of sense because it conforms to the Model-View-Controller design pattern and separates the business logic of the application from its UI logic.

Analysis and challenges

Basically, the development process went rather smoothly, although I decided to make some changes and adjustments.

The biggest change I made was that I decided against downloading the precompiled cartridge list and saving it to the local DB during the initial setup process. The cartridge list is needed only when users are adding new cartridges, and it makes more sense to download the latest list of cartridges only when the user opens the respective activity. Instead, during the initial process the application downloads and stores locally only stylus profile data because it's needed even when users are not adding new cartridges to the list.

I also decided against creating a separate class responsible for working with Firebase due to its asynchronous nature. Instead, I implement calls to Firebase RTDB directly in the relevant activities.

The main challenge was making sure that the application can properly handle connection timeouts. As it turns out, Google Firebase API does not include this functionality, so I had to read a lot about Timer, TimerTask and threading in Android:

- https://developer.android.com/reference/java/util/Timer
- https://developer.android.com/reference/java/util/TimerTask
- https://developer.android.com/reference/java/util/concurrent/Executor.html

Testing

Initially I wanted to write as many instrumented tests as possible to test the application. However, it turned out to be really difficult due to the nature of my UI flows (where actions in one activity mostly affect other activities). That's why I decided to test the UI components of the application manually. But at the same time I created proper unit tests for the data helper component (which is responsible for saving data to the database and retrieving it from the database).

The testing process is documented in the file called `test_log.xslx`.

Evaluation

Despite the challenges I had to overcome, I am very pleased with how the application turned out in the end. I was able to reach and even surpass my initial design goals because I also added the ability to override default lifespan values for cartridges and also customise the "+ SIDE" and "+ LP" buttons in the app.

Conclusion

During the project, I learned that developing my own application from scratch is significantly more challenging than completing programming assignments. It requires planning and thinking not only about separate components, but also about the overall architecture of the app, relationships between different components, UI flows, testing plans etc. Completing the project was challenging but I feel that I am now ready to develop useful Android applications on my own.

Appendix A: list of deliveries

1. Source code of the application is located in the `code` subfolder.

2. JavaDoc documentation is located in the `JavaDoc` subfolder.

3. Signed APK is located in the `release` subfolder.

4. Test log is in the `test_log.xlsx` file.

5. GitHub repository: https://github.com/SpinningVinyl/Stylus-Timer

6. Application walkthrough video / screencast: https://www.youtube.com/watch?v=Ap-pxlbE2fw

Appendix B: project diary

Week 1

I spent the last week working on the design of my application, and by this I mean both application design/structure and UI design (doing hand-drawn mockups).

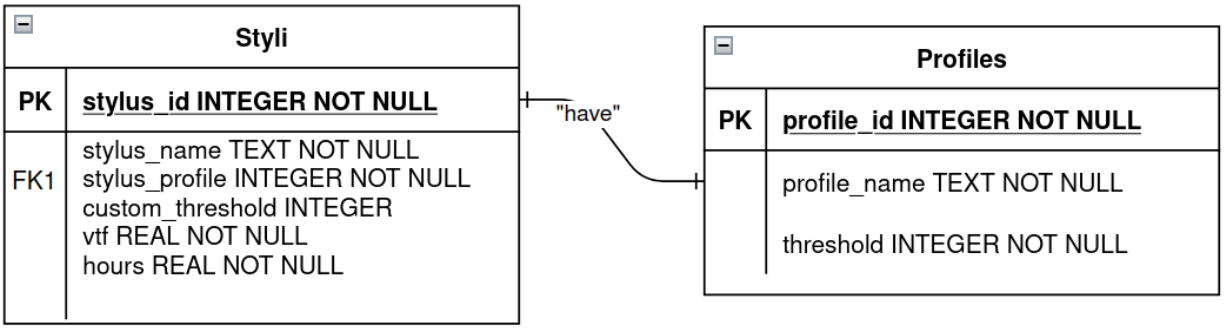
App design:
The application will use a backend (probably Firebase, although I might opt for a custom backend using Python/Flask) to download a database of preconfigured cartridges, although users will still be able to add/configure a cartridge by themselves. I will use SQLite to store data locally, and I will also use an ORM library (probably ORMLite) to implement object persistence.

UI design:
Here are the mockups I came up with so far.

This week I will focus on two things:
1) creating layouts for fragments and activities in Android Studio
2) implementing helper classes to work with data in my app

Week 2

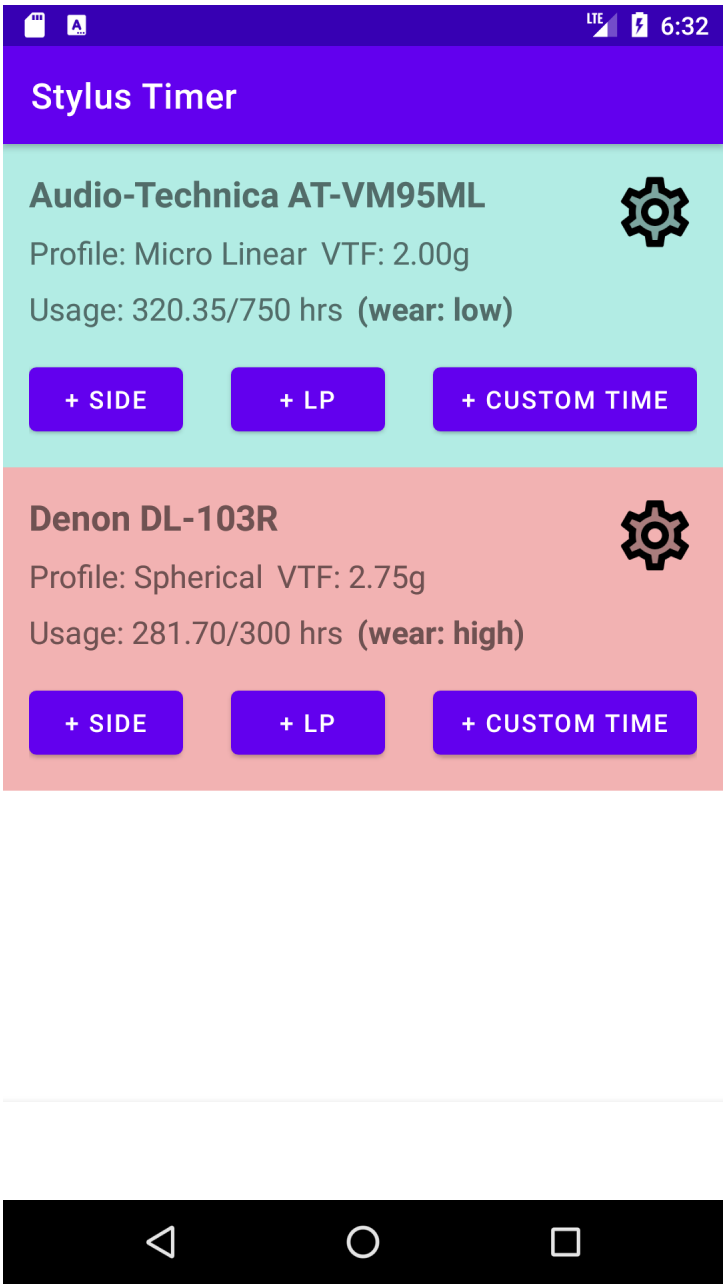
I think I finished the database design for my app, unfortunately I've not been able to do much else but I promise this week's update will be more substantial.



Weeks 3 & 4

I made some real progress during the last two weeks. I have implemented the appropriate data classes, RecyclerViewAdapter and its ViewHolder, and created working versions of most layouts. This is how the main screen of the application looks at present (using some mock data).

The background colour of the individual items in the recycler view show the wear level of the stylus.



Week 5

I feel like I turned the corner this week when it comes to the project. Here is the report of the various components of the app:

Firebase RTDB backend - functional & tested
Data helper class (for saving and retrieving data to/from the local DB) - functional & tested
Initial setup process - functional & tested
RecyclerViewAdapter with its appropriate ViewHolder - functional & tested
Adding cartridges from the pre-compiled list - functional & tested
Main screen - functional & tested
Adding cartridges manually - not implemented yet
Editing cartridge properties - not implemented yet

Appendix C: Bibliography

Advanced Stylus Shapes: Pics, discussion, patents.- Vinyl Engine (no date) *Vinylengine.com*. Available at: https://www.vinylengine.com/turntable_forum/viewtopic.php?f=19&t=22894 (Accessed: April 5, 2023).

Audio-Technica (no date) *VM Cartridges: Technical Guide*, *Audio-technica.com*. Available at: https://docs.audio-technica.com/eu/VM_Cartridges_EN.pdf (Accessed: April 5, 2023).

Ortofon (no date) *FAQ & installation*, *www.ortofon.com*. Available at: https://www.ortofon.com/support/support-hifi/faq-installation/ (Accessed: April 5, 2023).

VTF, Record and Stylus Wear (no date) *Lencoheaven.net*. Available at: https://www.lencoheaven.net/forum/index.php?topic=31535.0 (Accessed: April 5, 2023).https://github.com/SpinningVinyl/Stylus-Timer