Lappeenrannan teknillinen yliopisto

School of Business and Management

Sofware Development Skills

<Insert name here>, <Insert student number here>

LEARNING DIARY, MOBILE MODULE

**LEARNING DIARY**

14.11.2021

I started the course and created a new repository on GitHub. I read through the course information and environment setup pages, which seemed to be almost identical to the ones in the front-end course. I already have Android Studio installed from the object-oriented programming course, so it didn’t take long to get things going.

The first video covers a lot of things that I already knew, but a refresher is always good. I added in the components that were shown in the video. Programming the addition functionality reminded me of how to access the components in code and how to cast a string to an integer in Java.

Breakpoints and the step-by-step debugging were new to me and seem really useful. I also didn’t know of the quick way to turn an integer into a string by concatenating it with an empty string. I used String.format for the same purpose as I think it looks better.

22.11.2021

I watched the second video, which started with a quick terminology portion. It was good to revise what the definitions of activities and intents were, for example. After that I created a new Android Studio project. The video went through a quick tour again showing where things are in Android Studio.

Next I added the buttons and the second activity to the project. The video showed how to use intents, which was good as it was a little unclear to me before. In the object-oriented programming course I used fragments and a singleton to carry information between different screens, so I didn’t have to use intents back then.

After adding and testing the second activity and changing its text, the next task was to make the other button open Google. This didn’t work at first even though I followed the video’s instructions to a tee. After googling the issue, I found out that Android 11 made some changes to package visibility, which made the getPackageManager function return null. Thanks to Stack Overflow (<https://stackoverflow.com/questions/62535856/intent-resolveactivity-returns-null-in-api-30>), I managed to fix the issue by setting the queries in the manifest file:

Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti

29.11.2021

I started watching the third video. The first step was to create another project and add a ListView component. I found out that the component is not in the same place as in the video, because it’s been made a legacy component. Next I added the string arrays to the strings.xml file. After that, I followed the video until I should have had a string array loaded into the ListView. At first, the project would not build due to an error with the compileSdkVersion being wrong. I fixed the issue by editing the app’s Gradle file, and the list was working.

Next up was remaking the layout file I just created by making its root component a RelativeLayout. Getting the TextView set up the same way as in the video was a little weird because the way it works has been updated at some point. I got there in the end. I aligned the description TextView by directly using the layout\_alignBottom and layout\_alignStart properties.

Next was creating the item adapter. It had many steps but the video had pretty clear explanations for what’s happening. Once that was done, I moved on to creating the detail activity with an image of the selected item. I took the images from a site that provides free images for non-commercial use, [www.clipart-library.com](http://www.clipart-library.com). Creating a new activity and switching to it was dealt with in the last video, so not much new there. Instead the process for scaling images to be a proper size according to the device’s screen was new to me and way more convoluted than I would have thought. Not too difficult to wrap my head around though.

5.12.2021

I started work on the course project. I wanted to try doing something with the public API of ScoreSaber, an online leaderboard for the virtual reality rhythm game called Beat Saber. My plan is to create a main view, where you can input a profile ID or select from a list of the top 50 Finnish players, and view information about their profile in another view.

I had to look through my old projects and a tutorial (<https://www.youtube.com/watch?v=5lNQLR53UtY>) to remember how to get and parse JSON responses from an API. I learned that it’s better to use a thread for this instead of modifying the StrictMode.ThreadPolicy to avoid application lock ups. It was quite a bit more complicated this way, for example I had to use the handler object’s messages in order to prompt a refresh to a spinner from the other thread once I fetched and parsed the JSON data.

8.12.2021

Today I added a hashmap for profile names and IDs, so I can easily get the right profile’s information. I switched out the spinner for a ListView because a spinner always has an item selected, which wasn’t what I wanted. I want the user to be able to type/paste in an ID manually or click on a name in the list to fill the ID field automatically.

I customized the app’s colors with the “colors.xml” and “themes.xml” resources. I picked similar colors to what the actual ScoreSaber site uses. Some components like the EditText had color properties that I couldn’t change intuitively, so I confided in Google. The main activity of the project is pretty much completed and functional at this point.

13.12.2021

The goal I set for this session was to create the activity for viewing a profile. I started by creating a new empty activity and making a basic layout for the components. Next I started working on the code for fetching profile information. I felt like I could have made the fetcher its own separate class so I could reuse the same code in both activities, but I might do that refactoring later.

To display profile pictures, I needed to find a way to load an image from an URL. I watched this tutorial to learn how: <https://www.youtube.com/watch?v=oz3uGdi3f8Q>. After implementing this and trying to test it, I found out that the response body for the API has been recently changed, adding a name to the JSON array. Previously it was unnamed, so I had to change my code to accommodate for this.

The next problem was an error that happened because using getText on an EditText doesn’t return a String but an Editable, so I fixed that. After that, I noticed that one of the info pieces I wanted from the JSON was inside another object, so I used getJSONObject twice to get the data. After that I styled the profile activity to be consistent with the main one. I wanted to display a country flag instead of just text on the profiles, so I looked up how to convert a country code to a flag emoji: <https://attacomsian.com/blog/how-to-convert-country-code-to-emoji-in-java>.

14.12.2021

I started working on displaying top scores in a list. I made a new class for a score object for this purpose. I want to show each song’s cover image in the list, so I made my image fetching class its own class file and refactored it so that it returns a bitmap instead of setting a bitmap to an ImageView by itself. I learned about the Thread.join() method, which allows the main thread to wait until the image fetcher is finished before it tries to use the image. I felt like this defeats the entire point of using another thread in the first place, so I opted for another approach: I passed the handler as a parameter so I could send a message once the image has been fetched. This worked nicely.

I faced an issue with my threads looping because I forgot to add break statements to my handler’s switch-case. Correcting this fixed the issue and now I had my list of scores with cover images. The images disappear randomly while scrolling the list, which is apparently an issue with ListViews.

15.12.2021

Upon further research, it seems that the disappearing images problem might not be with the ListView component. It seemed complicated, so I decided to tackle another issue first: showing an error when data can’t be fetched. I did this by checking if the response from the API has a length of 0, and then showing an AlertDialog. I also learned about the finish() method, which finishes the current activity and goes back to the previous one.

After searching for solutions to the earlier problem, I tried resizing the list images to be smaller, but that did not make them stop disappearing. I came to the conclusion that reinventing the wheel is not a productive use of my time right now, so I used an open-source library called Glide. Using it to load images worked perfectly. I still left in my own implementation for loading profile pictures.

I made a new launcher icon for the app with paint.net, replacing the default icon was as easy as creating a new image asset with the same name.