



Mobile App Development 2

Study diary

Jaakko Virtanen

SISÄLLYS

[1 Week exercises 3](#_Toc87035283)

[1.1 Android BMI App 3](#_Toc87035284)

[1.2 Android Weather App GUI 3](#_Toc87035285)

[2 Week excercises 4](#_Toc87035286)

[3 Week exercises 5](#_Toc87035287)

[4 Week exercises 6](#_Toc87035288)

[5 Week exercises 7](#_Toc87035289)

[6 Week exercises 8](#_Toc87035290)

[7 Week exercises 9](#_Toc87035291)

[8 Week exercises 10](#_Toc87035292)

[9 Week exercises 11](#_Toc87035293)

[10 Week exercises 12](#_Toc87035294)

[Final project (not mandatory but required for best grades) 13](#_Toc87035295)

[Sources used with exercises 14](#_Toc87035296)

# Week exercises

## Android BMI App

<https://github.com/jaakko13/mobileAppDev2/tree/master/1.1BmiApp>

No real challenges in this work. Very simple introductory assignment.

## Android Weather App GUI

<https://github.com/jaakko13/mobileAppDev2/tree/master/12_weatherAppGUI>

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity">  
  
 <TextView  
 android:id="@+id/location"  
 android:textSize="24dp"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="164dp"  
 android:layout\_marginTop="98dp"  
 android:text="Tampere"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <TextView  
 android:id="@+id/locationText"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="168dp"  
 android:layout\_marginTop="128dp"  
 android:text="Location"  
 android:textSize="16dp"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <TextView  
 android:id="@+id/temperature"  
 android:textSize="24dp"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="164dp"  
 android:layout\_marginTop="64dp"  
 android:text="10 Celsius"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/location" />  
  
 <TextView  
 android:id="@+id/temperatureText"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="164dp"  
 android:layout\_marginTop="96dp"  
 android:text="Temperature"  
 android:textSize="16dp"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/location" />  
  
 <TextView  
 android:id="@+id/percipitationText"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="56dp"  
 android:layout\_marginTop="212dp"  
 android:text="Percipitation"  
 android:textSize="16dp"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/location" />  
  
 <TextView  
 android:id="@+id/HumidityText"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="240dp"  
 android:layout\_marginTop="212dp"  
 android:text="Humidity"  
 android:textSize="16dp"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/location" />  
  
 <TextView  
 android:id="@+id/percipitation"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="56dp"  
 android:layout\_marginBottom="165dp"  
 android:text="80 Percent"  
 android:textSize="24dp"  
 app:layout\_constraintBottom\_toTopOf="@+id/button"  
 app:layout\_constraintStart\_toStartOf="parent" />  
  
 <TextView  
 android:id="@+id/Humidity"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginEnd="56dp"  
 android:layout\_marginBottom="164dp"  
 android:text="10 Percent"  
 android:textSize="24dp"  
 app:layout\_constraintBottom\_toTopOf="@+id/button"  
 app:layout\_constraintEnd\_toEndOf="parent" />  
  
 <Button  
 android:id="@+id/addLocationButton"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="158dp"  
 android:layout\_marginEnd="159dp"  
 android:layout\_marginBottom="174dp"  
 android:text="Add Location"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent" />  
  
 <Button  
 android:id="@+id/refreshButton"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="75dp"  
 android:layout\_marginTop="98dp"  
 android:layout\_marginEnd="16dp"  
 android:text="Refresh"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toEndOf="@+id/location"  
 app:layout\_constraintTop\_toTopOf="parent" />  
</androidx.constraintlayout.widget.ConstraintLayout>

Again pretty simple introductory assignment.

# Week excercises

2.1 <https://github.com/jaakko13/mobileAppDev2/tree/master/flashlight>

fun switchFlashlight(view: android.view.View){  
 val cameraManager = getSystemService(*CAMERA\_SERVICE*) as CameraManager  
  
 for( id in cameraManager.*cameraIdList*){  
 if(cameraManager.getCameraCharacteristics(id).get(CameraCharacteristics.*FLASH\_INFO\_AVAILABLE*) == true){  
 torch = !torch  
 cameraManager.setTorchMode(id, torch)  
 }  
  
 val background = findViewById<ConstraintLayout>(R.id.*backgroundConstraintLayout*)  
 if(torch){  
 background.setBackgroundColor(Color.*YELLOW*)  
 }  
 else{  
 background.setBackgroundColor(Color.*BLACK*)  
 }  
 }  
   
}

Enjoyed learning the best practices of accessing the sensors. Farly easy overall but useful info. Opens the door for a lot of functionality for future apps and has already given me some ideas. No real challenges in the exercise.

2.2 <https://github.com/jaakko13/mobileAppDev2/tree/master/level>

fun leveler(view: android.view.View){  
 val sensorList = sensorManager.getSensorList(Sensor.*TYPE\_ALL*)  
  
 for( sensor in sensorList){  
 Toast.makeText(this, sensor.*name*, Toast.*LENGTH\_SHORT*).show()  
 }  
  
 val accelerometer = sensorManager.getDefaultSensor(Sensor.*TYPE\_ACCELEROMETER*)  
  
 if(accelerometer != null) {  
 sensorManager.registerListener(this, accelerometer, SensorManager.*SENSOR\_DELAY\_NORMAL*)  
 levelingStarted = true  
 }  
}

Very interesting learning how to access all the different sensor. The function above is how we got the data. But most of the visual code was in the onSensorChanged function as it updated the data on the screen. Not too complicated overall and didn’t face any big challenges.

# Week exercises

3.1

The figure shows the steps a developer should take when thinking about adding permission to their application. First step is to ask yourself if the permissions is really needed. You should not have unecessary permissions in your application. Mainly for security reasons. If possible to create functionality without the use of permissions then do so and that’s it. But if you need permissions then you should declare them in the manifest file of the app. Finally if the permission is a runtime permission then you have to request the access from the user but if not then there is nothing else to do.

if(ActivityCompat.checkSelfPermission(this, android.Manifest.permission.ACCESS\_FINE\_LOCATION) //Make sure permissions are good  
 != PackageManager.PERMISSION\_GRANTED && ActivityCompat  
 .checkSelfPermission(this, android.Manifest.permission.ACCESS\_COARSE\_LOCATION) != PackageManager.PERMISSION\_GRANTED  
){  
 ActivityCompat.requestPermissions(this, arrayOf(android.Manifest.permission.ACCESS\_FINE\_LOCATION), 101)  
 return  
}

The above piece of code is used to check the permissions and ask for them if they have not been granted yet. The initial if statement is what does the checking. While the ActivityCompat.requestPermission inside the brackets of the if statement is what requests permission from the user if the program does not have the required permissions.

3.2 Github: <https://github.com/jaakko13/mobileAppDev2/tree/master/codelab>

Getting coordinates was easy as I’ve don’t this before for a final project and also during my last internship. Worked with maps a lot… The location listener was new and a bit more of a challenge. Very useful as I had just used a button to update the location but now it updates constantly. Also opening the map was fairly simple using intent.

# Week exercises

4.1 **Get familiar with broadcast receivers (eg. from https://developer.android.com/guide/components/broadcasts). Explain briefly what is a Broadcast Receiver and where you can use them?**

It’s a form of communication between components in android devices. Programs can listen for certain events and respond accordingly when they happen, such as when the phone is plugged in or connected to wifi. System broadcasts are automatic and all apps can subsribe to them.

**Asynchronous tasks. Which kind of tasks you should execute in the background thread instead of Main UI thread. Give some examples.**

Any tasks that can take a lot of time such as api calls. Only changes to the UI should be run in the Main UI thread and everything else should have its own thread to increase performance and not put too much strain on the main thread. Your own functions should have their own thread especially when listening for broadcasts or using services.

**What are Services. Give a use case for Service in your application.**

Services are used to run functions that take longer than usual and they run in the background. Can be run even while in another application. A use case for a service could be spotify playing music in the background even when you are not in the app.

4.2 Github: <https://github.com/jaakko13/mobileAppDev2/tree/master/broadcastReceiver>

Did not have any problems setting up the broadcast as there isn’t too many steps involved but did learn how to do it. Also interesting to see the list of all system broadcasts and everything it can be used for. Struggled to access the ui elements in the inner class but figured it out with everyone in class that you need to use the inner keyword when making the inner class to have access.

4.3 Github: <https://github.com/jaakko13/mobileAppDev2/tree/master/Servives>

Service set up was a little tougher because I kept getting the default value and not the one set in the service but figured out the parameters for the broadcastReceiver were messed up and it helped. Also struggled for a while converting it to an array for the lottery numbers and putting that through intent. But that I blame on kotlin because it is very confusing with all the different types of lists and array as some are mutable some aren’t and theres a lot of casting that’s not necessary in the end. But I was able to get it working cleanly in the end so all good.

# Week exercises

Github: <https://github.com/jaakko13/mobileAppDev2/tree/master/51weather>

Assignment was good and fairly easy as I had already done a similar app in the past so I was able to apply those things to this assignment. The way of doing the api call was different so I had to learn that but it was not too complicated. I actually preffered this way of making the call as I think it was more clear and easier to work with.

# Week exercises

<https://github.com/jaakko13/mobileAppDev2/tree/master/61weatherAppp>

6.1 and 6.2

I enjoyed the exercise. Had not used a scroll view before and even though it was fairly easy to use it can be really useful when making apps. Also implementing the 2 volley requests into 1 was very interesting as I have made apps before but never really put a lot of thought into the architecture of the code. Even though it was a small example it showed how things can be simplified.

6.3 Flutter is setup.

<https://github.com/jaakko13/mobileAppDev2/tree/master/HelloWorld>

<https://github.com/jaakko13/mobileAppDev2/tree/master/flutter/ex7_1>

7.1 Classes were fairly easy to setup as we are familiar with them from other languages. So really only needed to learn the new syntax. The syntax in dart is also pretty straightforward and easy to understand

<https://github.com/jaakko13/mobileAppDev2/tree/master/flutter/weather7_2>

7.2 Also easy to work with. But I struggled a little bit with moving the widgets around and getting them where I wanted them which was in the center. It would center vertically but not horizontally even though I had the code for it to do so.

<https://github.com/jaakko13/mobileAppDev2/tree/master/flutter/ex7_3API>

7.3 (Week 8)

Cool to undestand different ways of making api calls. Worked with them a lot in the past but never in flutter. Good to know of the different methods. Also interesting to see the navigation aspect as I think that’s a pretty big part of apps. Struggled with the json reading the most but figured it out eventually.

# Week exercises

# Week exercises

Github: <https://github.com/jaakko13/mobileAppDev2/tree/master/flutter/week9exs>

Struggled a bit with flutter errors with builing the app but figured them out eventually. Nothing specific to this exercise. Learned how to use permissions and ask for them when needed. Also how to setup listeners so the values are constantly changing. Overall the exercise was not too demanding.

# Week exercises

Github: <https://github.com/jaakko13/mobileAppDev2/tree/master/flutter/list_view>

I thought the exercise was good as I think a lot of modern apps use List Views. Mostly on social media apps so they are good to know how to work with them. I struggled with the conditional rendering for a bit because at first my app was just stuck showing the ”waiting data” part and the data would never show but I eventually fixed it by making some adjustments to my api call function.

# Week exercises

# Final project (not mandatory but required for best grades)

Sources used with exercises

List here the possible sources you’ve used with exercises (e.g. stackoverflow, tutorialspoint, github etc.)