



Mobile App Development 2

Study diary

Eric Brown

SISÄLLYS

[1 Week exercises 3](#_Toc86919927)

[1.1 Taskulamppusovellus (Camera HW API) 3](#_Toc86919928)

[1.2 Vatupassi (Sensor API) 3](#_Toc86919929)

[2 Week excercises 4](#_Toc86919930)

[3 Week exercises 5](#_Toc86919931)

[4 Week exercises 6](#_Toc86919932)

[5 Week exercises 7](#_Toc86919933)

[6 Week exercises 8](#_Toc86919934)

[7 Week exercises 9](#_Toc86919935)

[8 Week exercises 10](#_Toc86919936)

[9 Week exercises 11](#_Toc86919937)

[10 Week exercises 12](#_Toc86919938)

[Final project (not mandatory but required for best grades) 13](#_Toc86919939)

[Sources used with exercises 14](#_Toc86919940)

# Week exercises

## Android BMI App

Copy/paste your relevant source, screenshots etc. here.

Source: <https://github.com/evvic/mobile_app_development/tree/main/BMI_app>

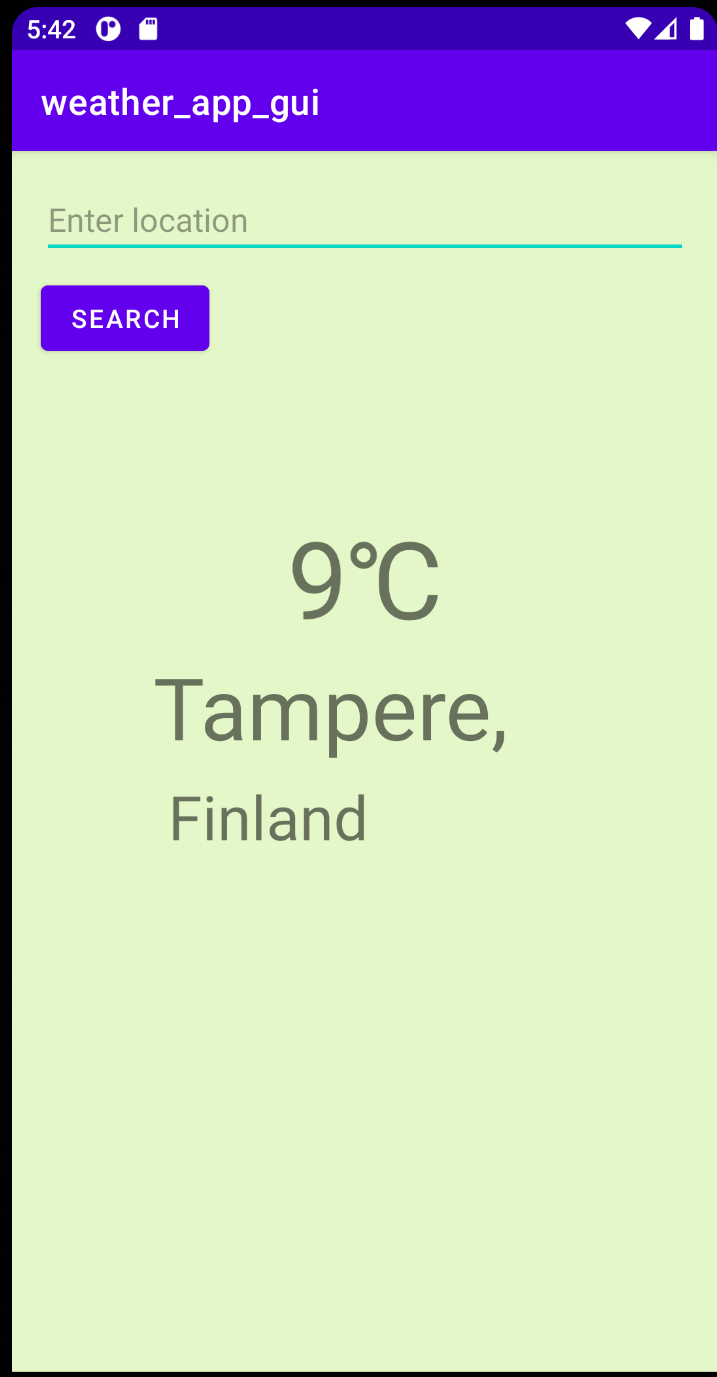
I followed along in class when designing this app. My app looks the exact same as the professors, where when the button is pressed, the BMI is calculated and displayed above the button, near the bottom of the screen.



## Android Weather App GUI

Copy/paste your source, screenshots etc. here

Source:<https://github.com/evvic/mobile_app_development/tree/main/weather_app_gui>

I created the weather GUI design thinking about how a typical weather app may look. Usually very simple and clean, depending on the informatics needed. Basically it could use the user’s location to determine the location and temperature, or the user can enter a location and the app would show the current temperature of that given location.

XML code for project:

<?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" android:background="#AED6F3AE" tools:context=".MainActivity"> <Button android:id="@+id/button" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:layout\_marginStart="16dp" android:layout\_marginTop="8dp" android:text="Search" app:layout\_constraintStart\_toStartOf="parent" app:layout\_constraintTop\_toBottomOf="@+id/searcch" /> <TextView android:id="@+id/locationCity" android:layout\_width="236dp" android:layout\_height="70dp" android:layout\_marginStart="87dp" android:layout\_marginEnd="87dp" android:text="Tampere," android:textSize="48sp" app:layout\_constraintBottom\_toTopOf="@+id/country" app:layout\_constraintEnd\_toEndOf="parent" app:layout\_constraintStart\_toStartOf="parent" /> <TextView android:id="@+id/country" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:layout\_marginStart="87dp" android:layout\_marginBottom="286dp" android:text="Finland" android:textSize="34sp" app:layout\_constraintBottom\_toBottomOf="parent" app:layout\_constraintStart\_toStartOf="parent" /> <TextView android:id="@+id/temperature" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:layout\_marginStart="162dp" android:layout\_marginEnd="162dp" android:text="9℃" android:textSize="60sp" app:layout\_constraintBottom\_toTopOf="@+id/locationCity" app:layout\_constraintEnd\_toEndOf="parent" app:layout\_constraintStart\_toStartOf="parent" /> <EditText android:id="@+id/searcch" android:layout\_width="0dp" android:layout\_height="wrap\_content" android:layout\_marginStart="16dp" android:layout\_marginTop="16dp" android:layout\_marginEnd="16dp" android:ems="10" android:hint="Enter location" android:inputType="textPersonName" app:layout\_constraintEnd\_toEndOf="parent" app:layout\_constraintStart\_toStartOf="parent" app:layout\_constraintTop\_toTopOf="parent" /></androidx.constraintlayout.widget.ConstraintLayout>

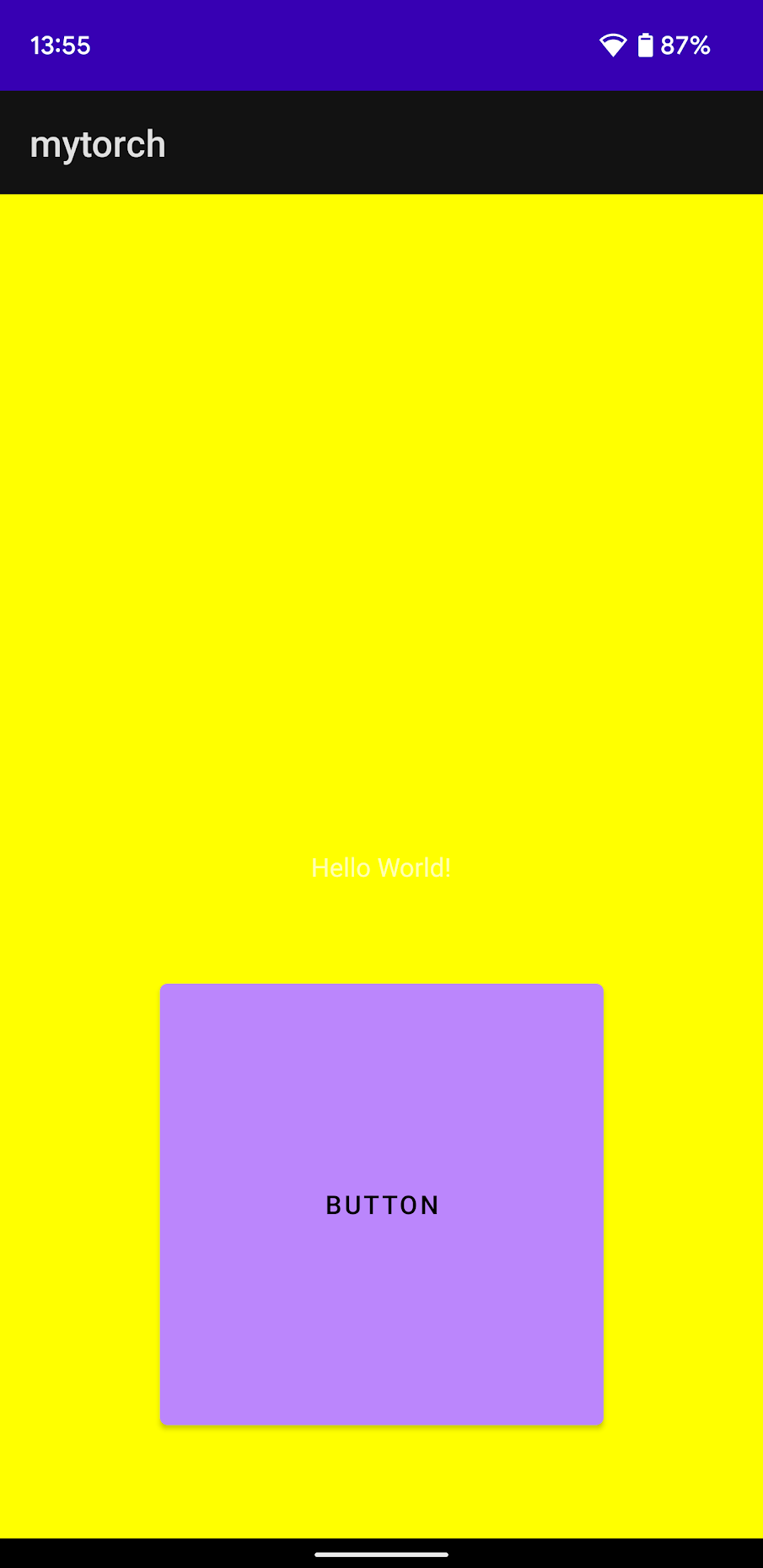
My Gradle wasn’t building correctly so I had to add these extra lines to my build.gradle(:app): https://stackoverflow.com/a/69043734

# Week excercises

**2.1: Synchronous API call example - Camera HW API - "Flashlight APP"**

Source: https://github.com/evvic/mobile\_app\_development/tree/main/mytorch

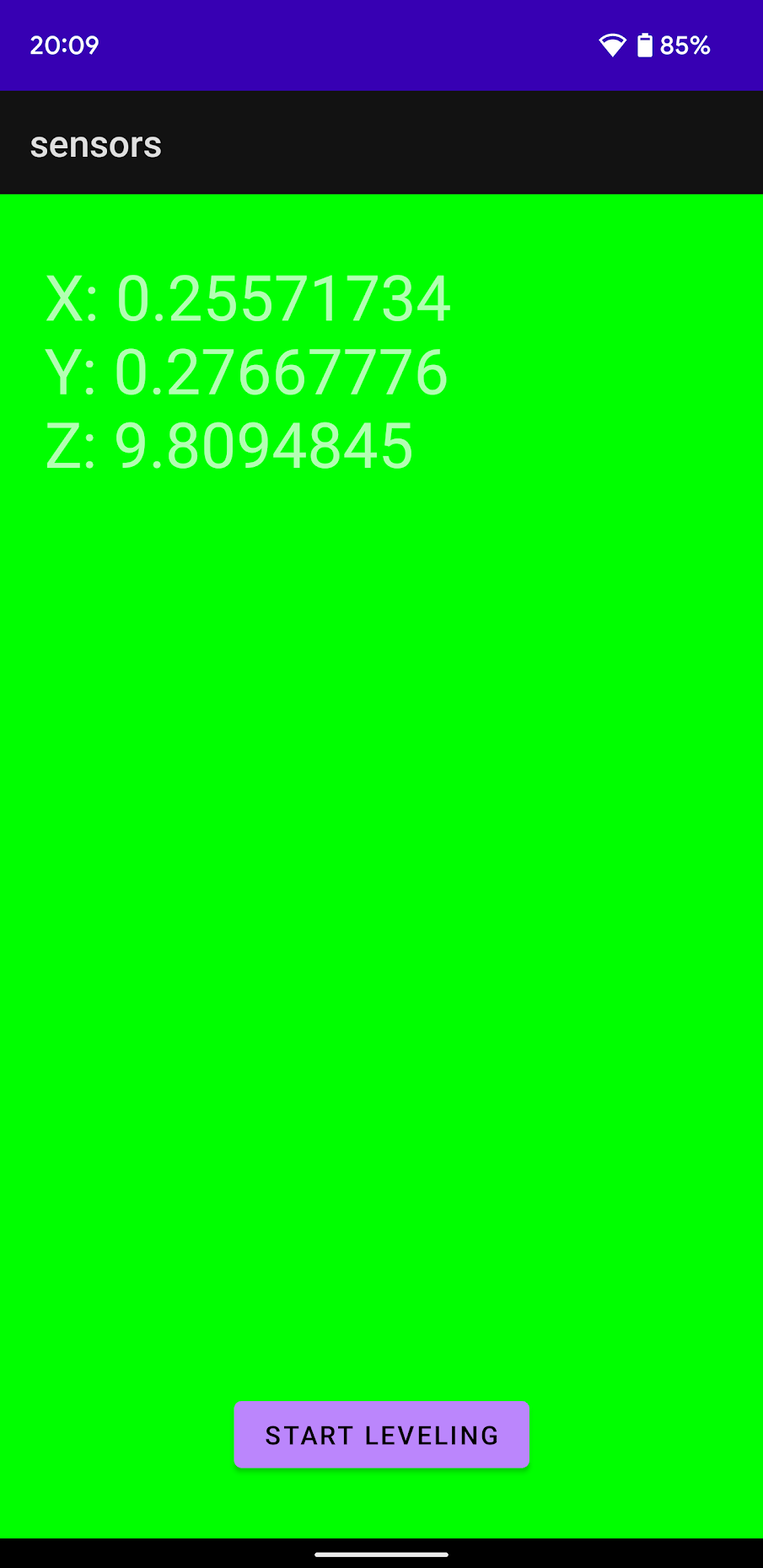
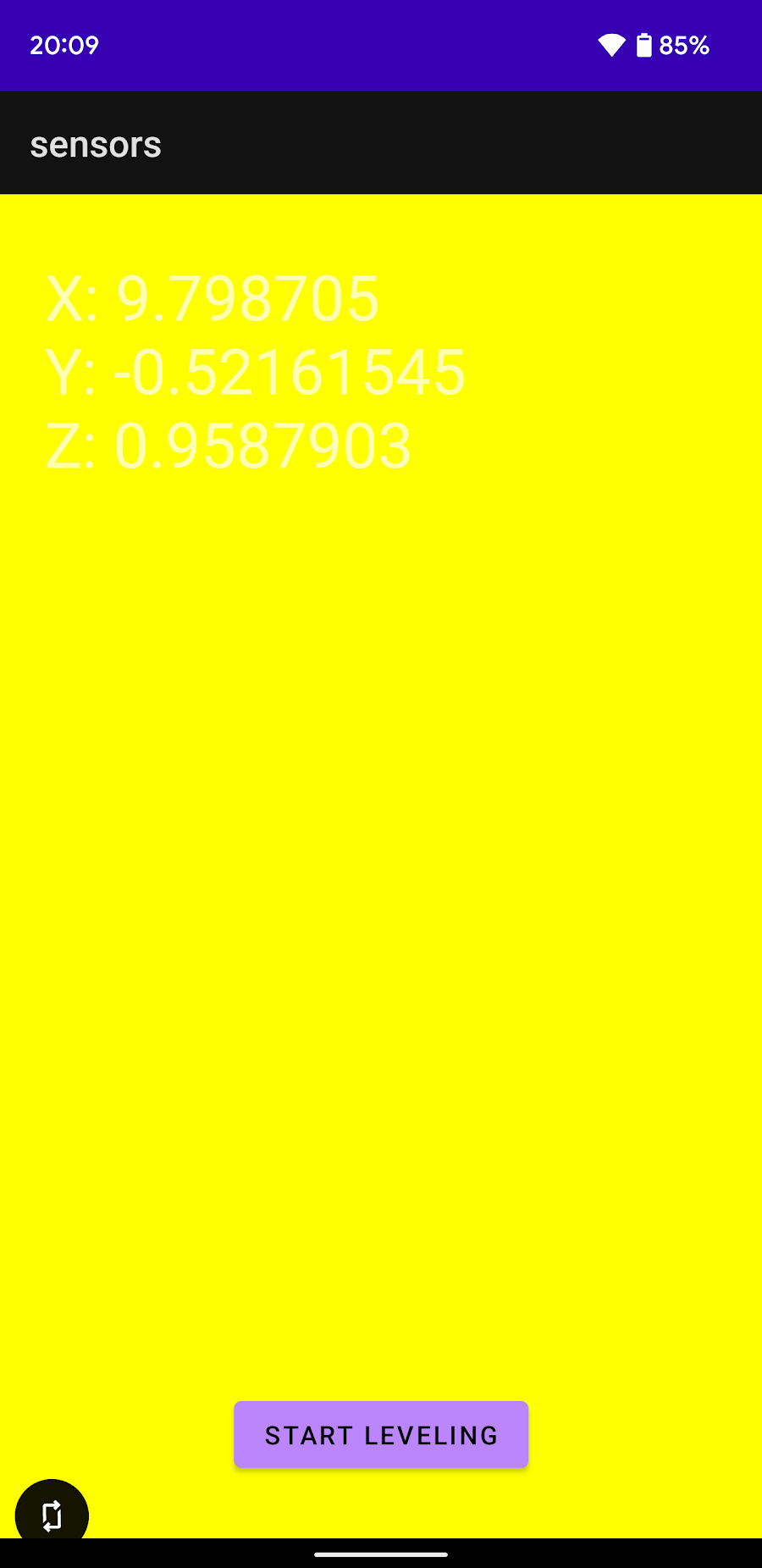
I created a button that toggles the flashlight attached to my back camera to turn on and off. Also when the light is off, the background is black, but when the light is turned on, the background turns yellow.

****

# **2.2: Asynchronous API call example (listener) - Sensor API and "Level APP"**

Source: https://github.com/evvic/mobile\_app\_development/tree/main/sensors

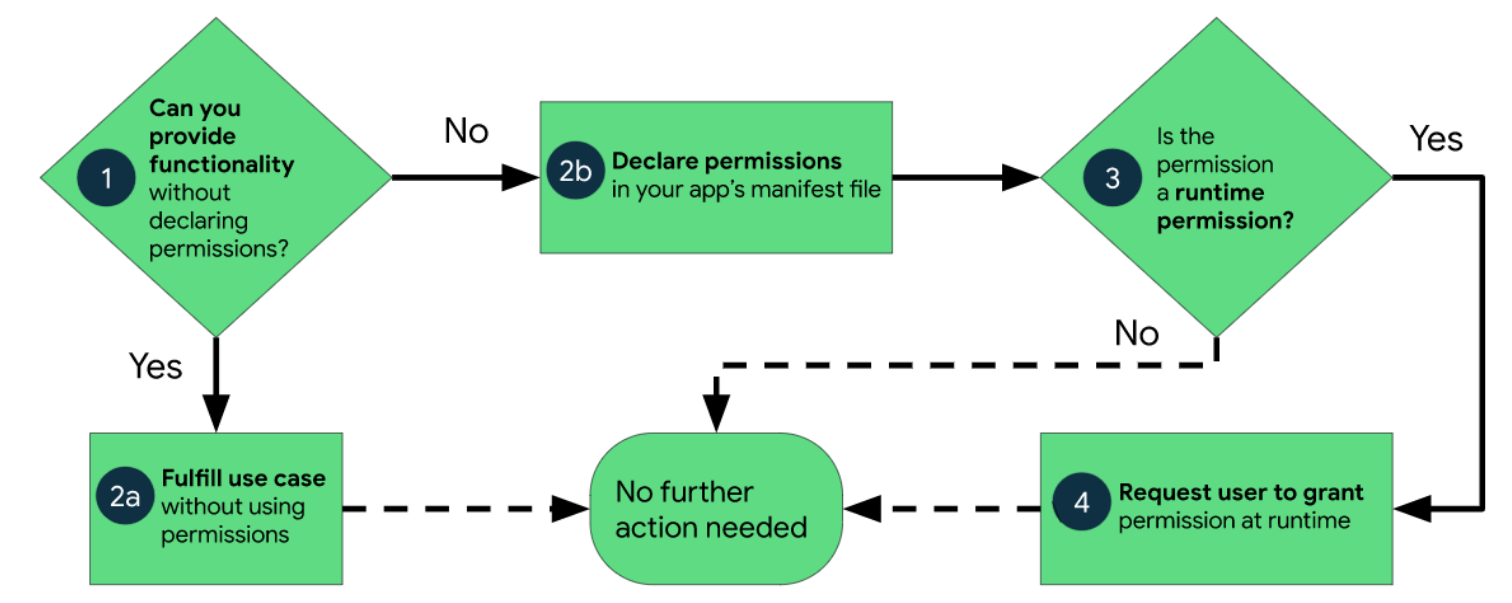
This app uses a listener of the accelerometer to update whenever the phones axis’ change. When the X-axis nears 10, basically the phone is horizontal, the background changes from the default blue to yellow. Then, if the phone is flat on a desk, the x-axis and y-axis are near zero, the background turns green.



# Week exercises

# **Written questions and answers**

The Android App permission work flow:



Basically this flow diagram shows the steps necessary for the program to have access rights to using a user’s devices API. Depending on how sensitive the data is towards the user, the program may need extra permissions before accessing that specific API. The program might not even need to declare the permission in the manifest file if it’s not sensitive data. Or if the permission needs to be declared, it might still need the user’s explicit concent before the API can be accessed by the program.

For example, to access a user’s general and specific location, the manifest file must coontaint the following lines inside the <manifest> tags:

**<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION"></uses-permission>  
<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION"></uses-permission>**

Then, because this is sensitive information, before being able to access the device’s location, it must be granted access by the user through code such as:

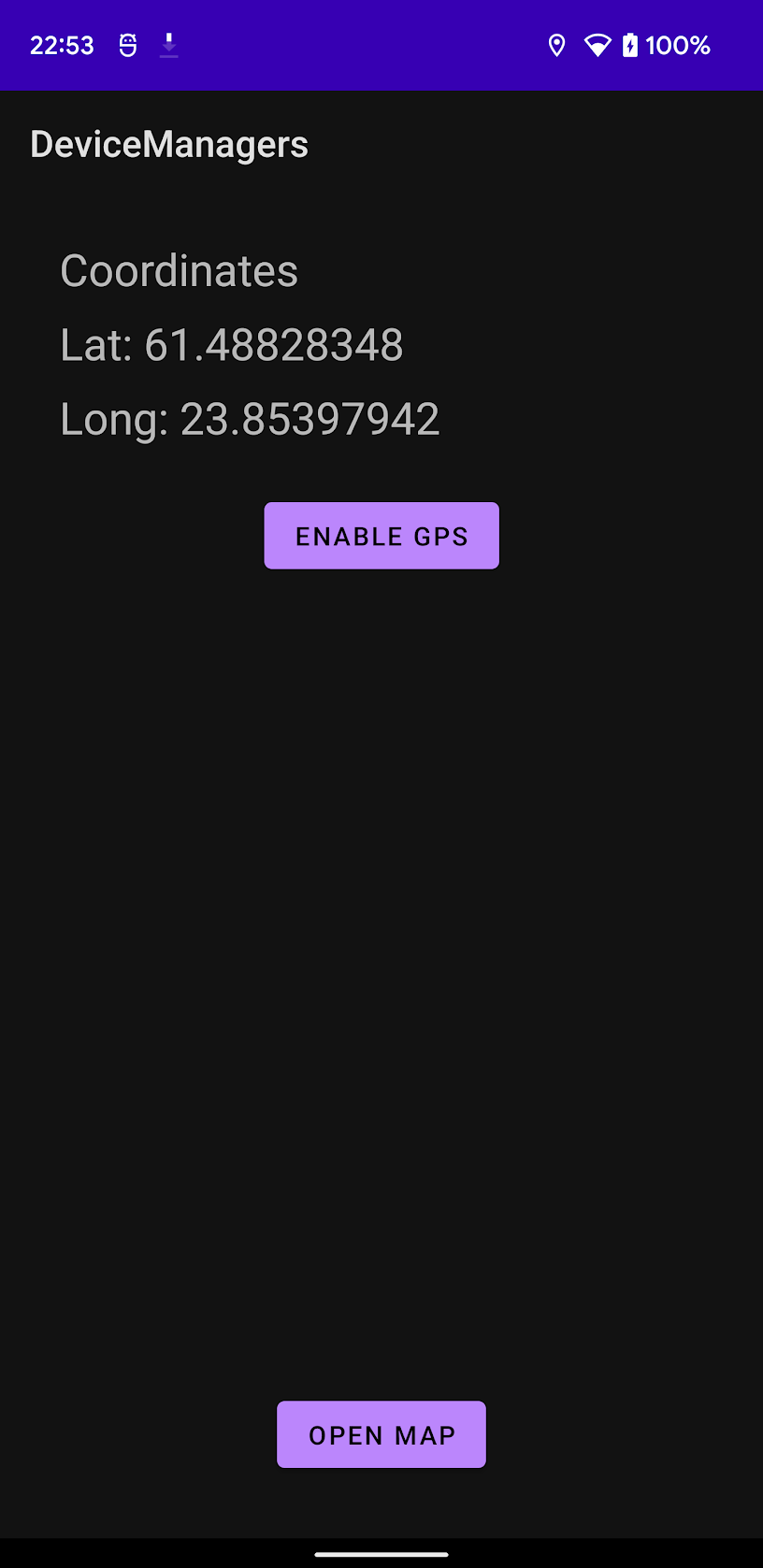
**if(checkSelfPermission(Manifest.permission.*ACCESS\_FINE\_LOCATION*) != PackageManager.*PERMISSION\_GRANTED*  
&& checkSelfPermission(Manifest.permission.*ACCESS\_COARSE\_LOCATION*) != PackageManager.*PERMISSION\_GRANTED*)  
{  
 ActivityCompat.requestPermissions(this,  
 *arrayOf*(Manifest.permission.*ACCESS\_FINE\_LOCATION*, Manifest.permission.*ACCESS\_COARSE\_LOCATION*),  
 0)****}**

The Manifest is used to check if permission has been granted (as shown in the if statement), and if not, a permission request is sent that the user must approve of before the specific device API can be used.

# **Code Lab**

I created the simple app that asks the user for permission to use their specific and general location, then uses that data from the devices API and updates to give the current latitude and longitude of the device.

Source code: <https://github.com/evvic/mobile_app_development/tree/main/DeviceManagers>\

Clicking ”ENABLE GPS” asks the user for location permissions and then begins a listener to any GPS location updates, which updates the UI.

Then clicking open map sends the intent to Google Maps with the current devices location.

# Week Exercises

## 4.1 Written Questions and Answers

The broadcast and broadcast receivers are used by Android apps and the system as a publish-subscribe design pattern. Basically, the system and apps can broadcast data or just notify in general when an event of interest occurs. Apps can register broadcast receivers to a specific broadcast, waiting for an update. The system will route the broadcast to all subscribed receivers in each respective app. There are many instances for using a broadcast receiver, for example, playing music as a service and broadcastign when the song changes.

Any task that takes more than a second, such as API calls to the network or long calculations should be done in a separate thread outside of the main UI thread. If the main UI thread had to freeze and wait for the API request to get the data it wants to display, then the user would have a bad experience.

A service is a background process that can perform long-running operations even when the user switches the application. Services don’t have an interface, so they handle tasks such as network transactions, playing music, or handling the input/output of file transactions.

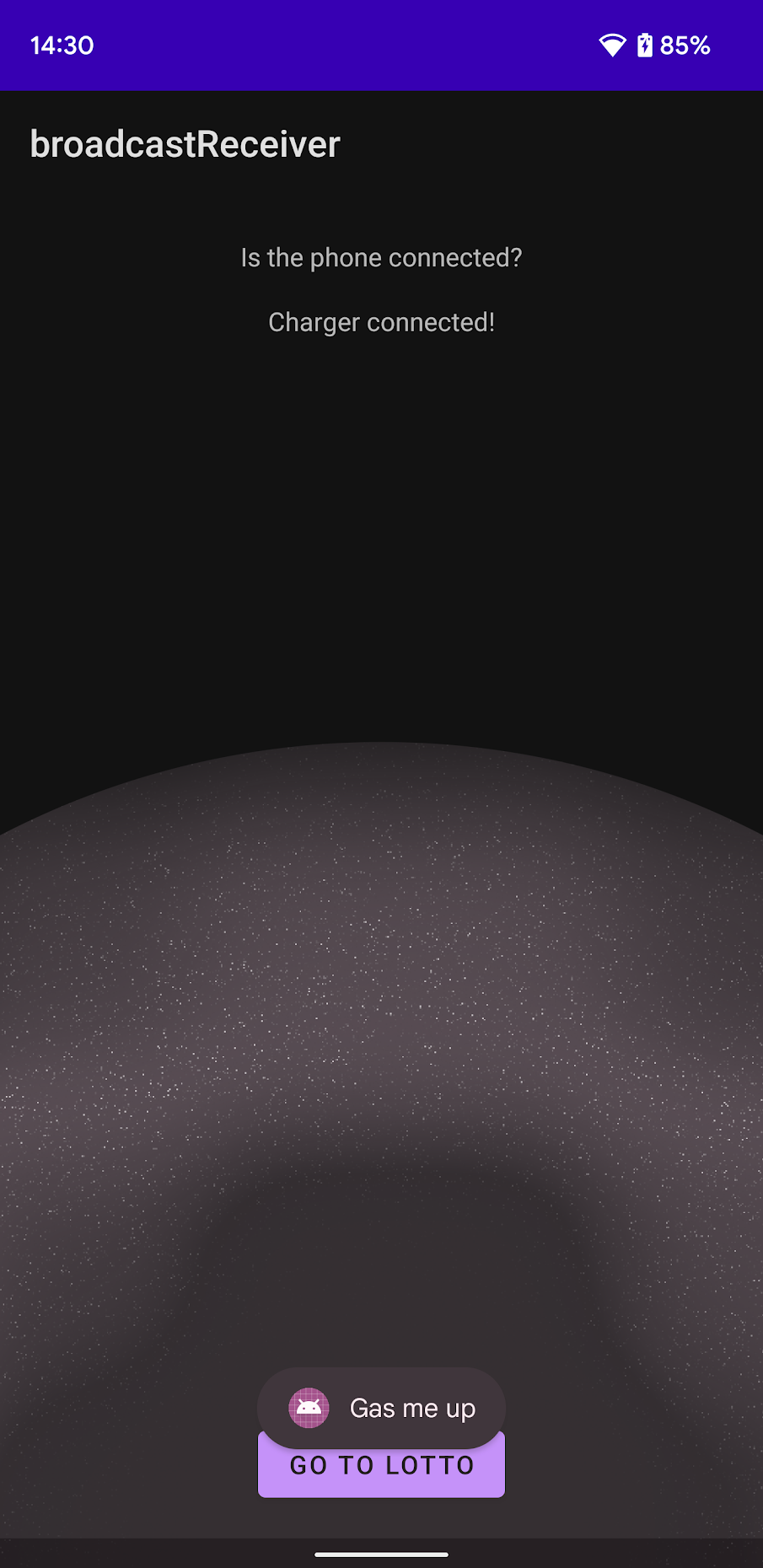
## 4.2 Codelab: Broadcast Receivers

Note: I used the same project for both codelabs this week! I made a button that switches from the main activity to the second activity where the second Codelab exercise occurs.

For the broadcast receiver I made an intetnt filter for ACTION\_POWER\_CONNECTED and ACTION\_POWER\_DISCONNECTED. So I will receive a broadcast everytime the phone is connected and disconnected from a charger. I created a receiver class that implements BroadcastReceiver(), which on receiving a broadcast, runs some code to update the text and send a toast notifcation that the phone was connected/disconnected.

**inner class MyReceiver : BroadcastReceiver() {  
  
 override fun onReceive(context: Context?, intent: Intent?) {  
 if(intent?.*action*.*equals*( Intent.*ACTION\_POWER\_CONNECTED*)) {  
 Toast.makeText(context, "Gas me up", Toast.*LENGTH\_SHORT*).show()  
 val chargerStatus = findViewById<TextView>(R.id.*isConnected*)  
 chargerStatus.*text* = "Charger connected!"  
 }  
 if(intent?.*action*.*equals*( Intent.*ACTION\_POWER\_DISCONNECTED*)) {  
 Toast.makeText(context, "Go down & skip town", Toast.*LENGTH\_SHORT*).show()  
 val chargerStatus = findViewById<TextView>(R.id.*isConnected*)  
 chargerStatus.*text* = "Charger disconnected!"  
 }  
 }**  
}

The app looks like:



## 4.3 Codelab Services

I created a background service (in the same app, but a second activity screen) that when the button ”START LOTTO” is pressed, an array of six integers is randomly generated (between 0 to 40). This is generated in a separate thread where the thread sleeps for about 2 seconds, to simulate a process that isn’t instant. The thread running the code is:

**Thread {  
 Thread.sleep(1500)  
 val lottoNumber = IntArray(7) { Random.nextInt(0, 40) }  
 val lottoBroadcast = Intent("eric.tamk.lotto")  
 // add lotto number data to the broadcast  
 lottoBroadcast.putExtra("LOTTO\_NUMBER", lottoNumber)  
 // send broadcast  
 sendBroadcast( lottoBroadcast )  
}.start() //starts a new thread in the system**

The intent name is eric.tamk.lotto, something unique that wouldn’t already be used by another app or by the system. Here the service sends the broadcast when it is finished creating the integer array.

The screen appears as:

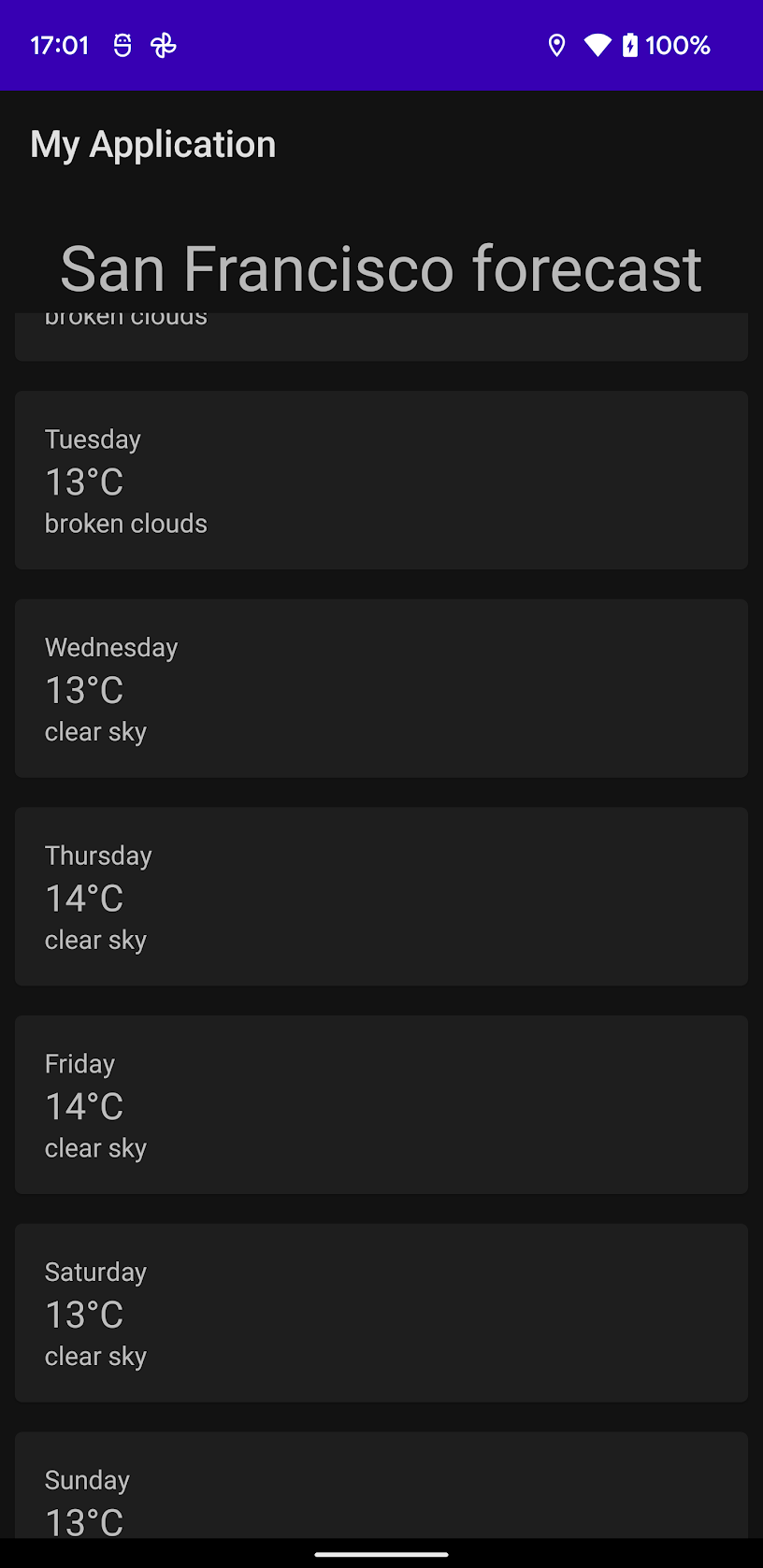
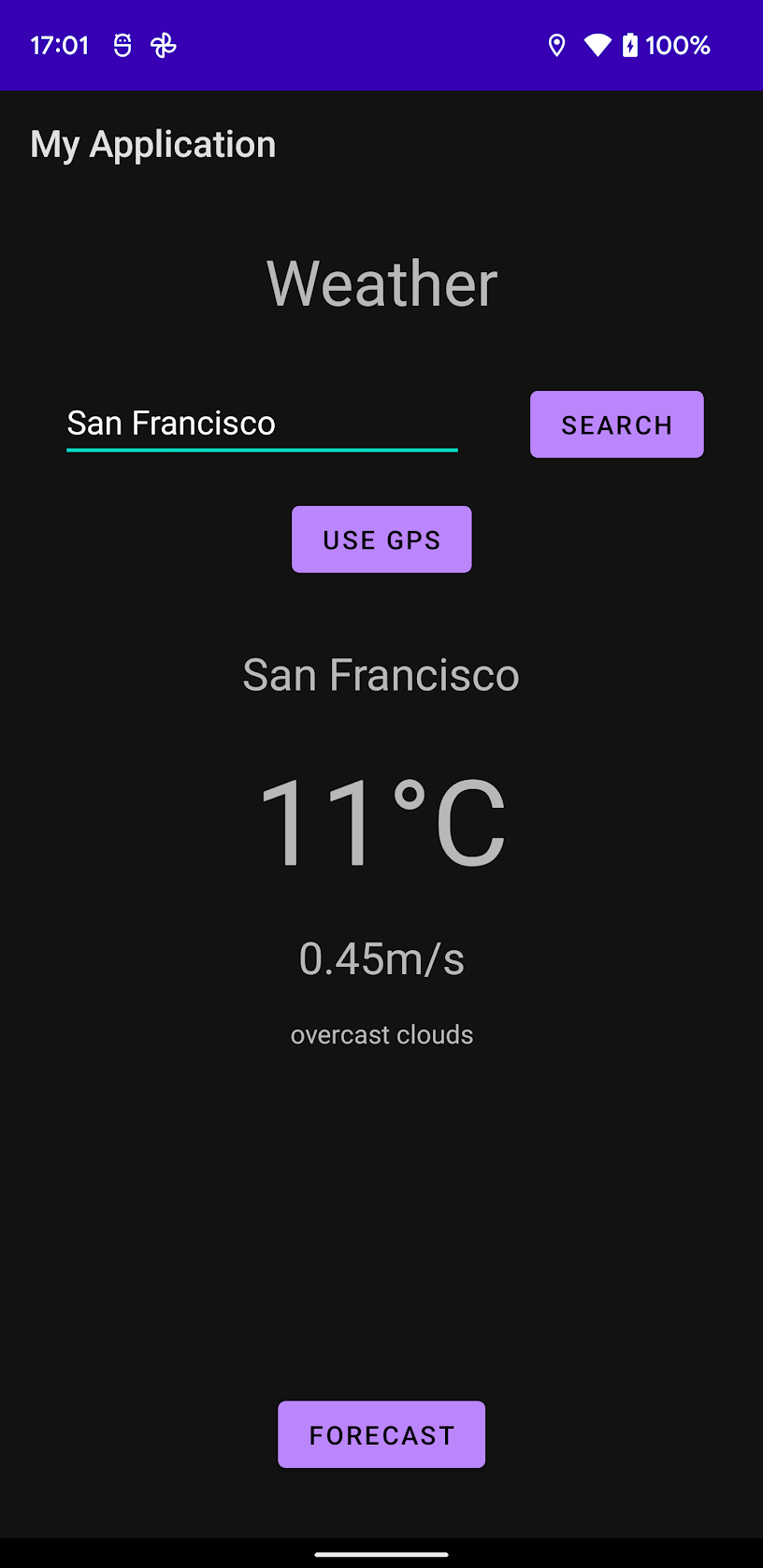


# Week exercises

**5.1 Weather App GUI**

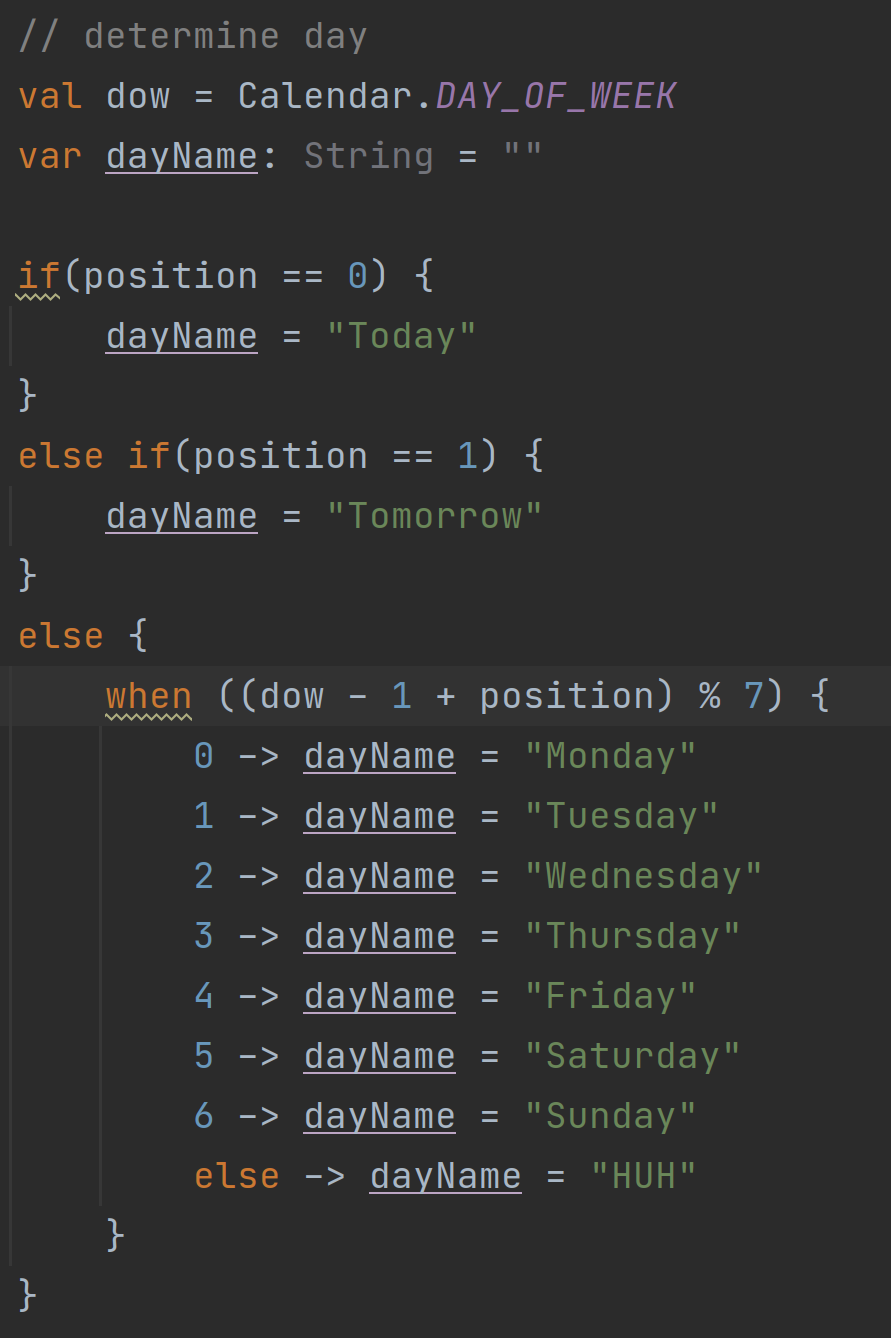
https://github.com/evvic/mobile\_app\_development/tree/weather-gui/weather\_api

I tagged the GUI part, then continued with the functionality. Please note that the GUI was hard to implement wihtout any functionality because I created a recycled list for the elements of the forecast. So ultimately the GUI Looks best in the final product after the functionality was added and successfully working.



Basically the main activity gives the user the option of using the EditText field to enter a city name and press the search button to make the API call (after checking the name); or the user can press the USE GPS button which will then use the devices coordinates to make the API call for the current weather. The main activity displays the city name, current temperature, wind speed, and weather description. The FORECAST button takes the user to second activity.

The second activity, with the forecast displays the next 7 days weather data in cards in a scrollable list. Each card gives the say, the temperature for that day, and the weather description. The name of the city is also displayed.



I like how I implemented the day tracking for each card. I used a switch case statement in the onBindViewHolder() function inside the Adapter for my Item class. My Item class just holds the data for each card, i.e. temperature, and weather description. I used the position to determine which day is today and tomorrow, then filled the rest with some math.

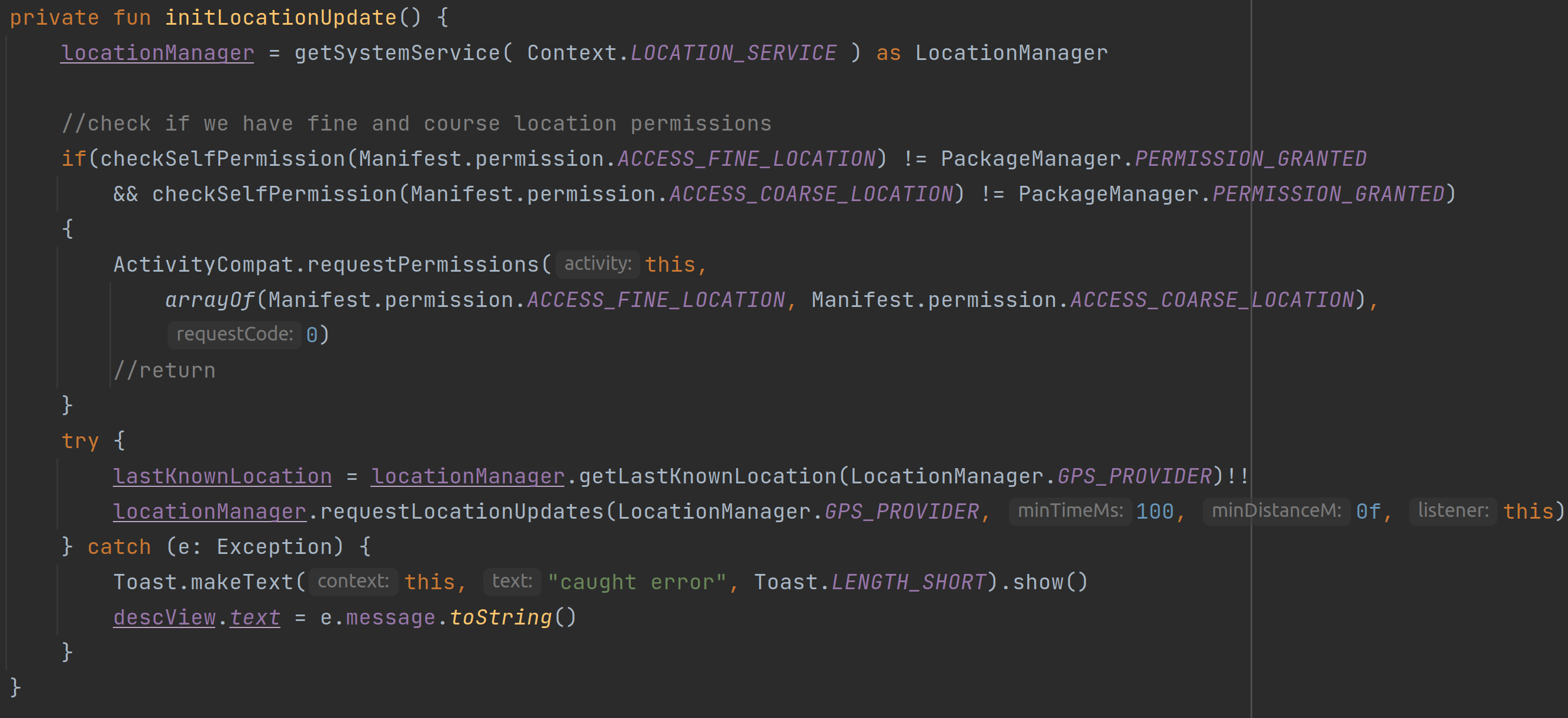
**5.2 Fetch the Weather data**

https://github.com/evvic/mobile\_app\_development/tree/main/weather\_api

The most recent commit contains both the updated GUI and the functionality, making it a working state. I used the Android Volley Library to parse the JSON data into the respective TextViews mentioned in the GUI section (5.1). I had 2 different API calls, one for the current weather (both city and coordinate version) and one for the 7 day forecast (8 days including today).



My JSON object request using Volley (for the forecast) was more complicated because I had to loop through a JSON array for each day of the forecast, then append it to my Mutable List of my class Items. The class items then went through the Adapter to create the recycle view.



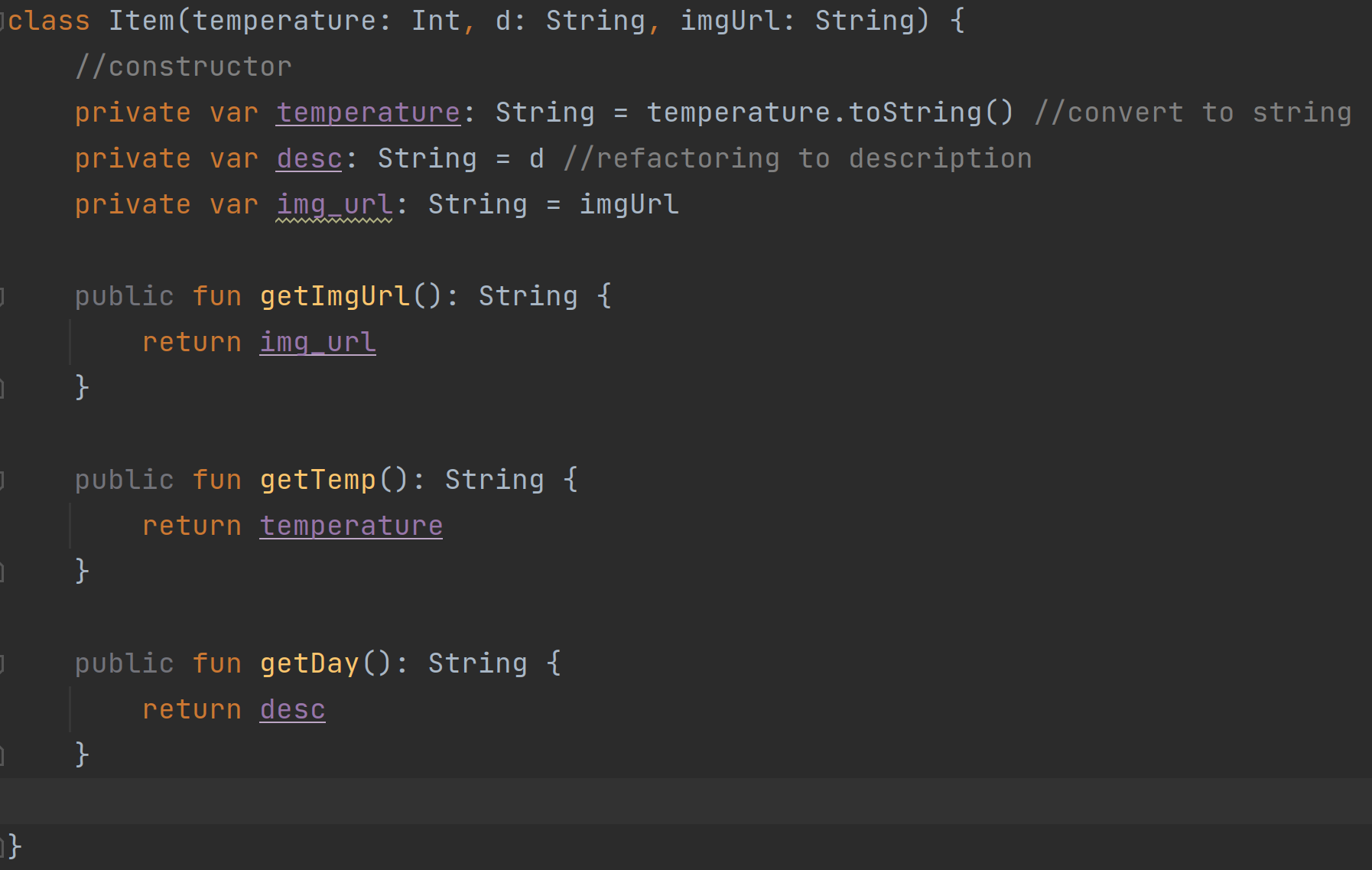
I created a listener for the GPS because unfortunately I couldn’t find a more simple method to capure the current location once. The sitener continuously updates a latitude and longitude variables, but I have another pair that capture the coordinates when the USE GPS button is pressed, also triggering the API call.

# Week Exercises

**6.1 Adding Weather Forecast Screen**

https://github.com/evvic/mobile\_app\_development/tree/main/weather\_api

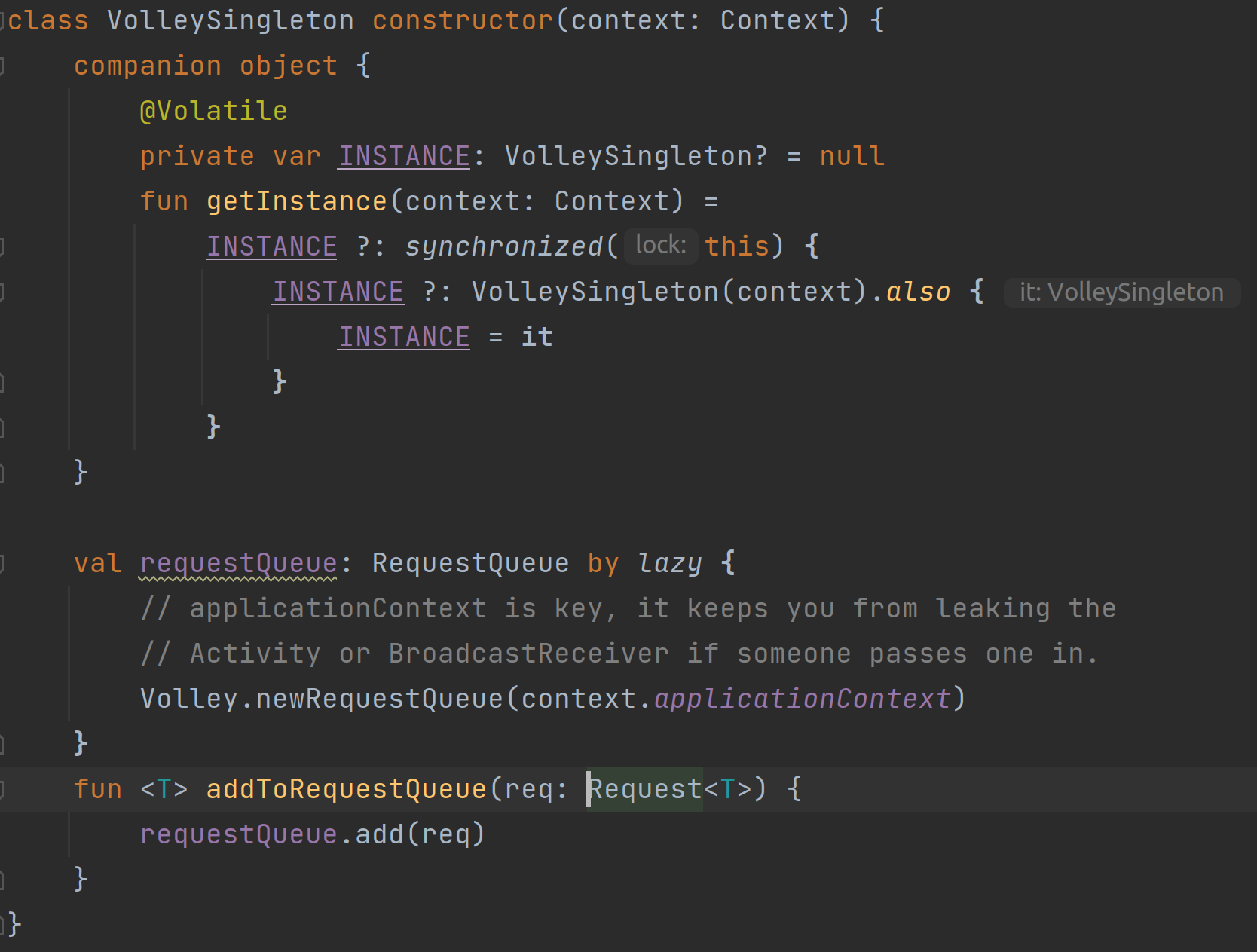
As I mentioned in class, I accidentally did this assignment in last weeks homework, exercise 5.2. So it is the same repo but I will add another screenshot here I guess.

I created an Item class to bring all the relevant data from the API call to the list of cards in the recycle view. I ultimately did not end up using the weather icons in my cards because I couldn’t get the image to appear on the card even though I tested the url in the browser and it was pulling up the icons. So the imgUrl String was never used unfortunately.

**6.2 Singleton and App Architecture**

https://github.com/evvic/mobile\_app\_development/tree/main/weather\_api

This exercise was easier than I thought it would be to implement, the main activiy and forecast activity only needed a few lines changes each; then I had to add a new class for the Volley request.

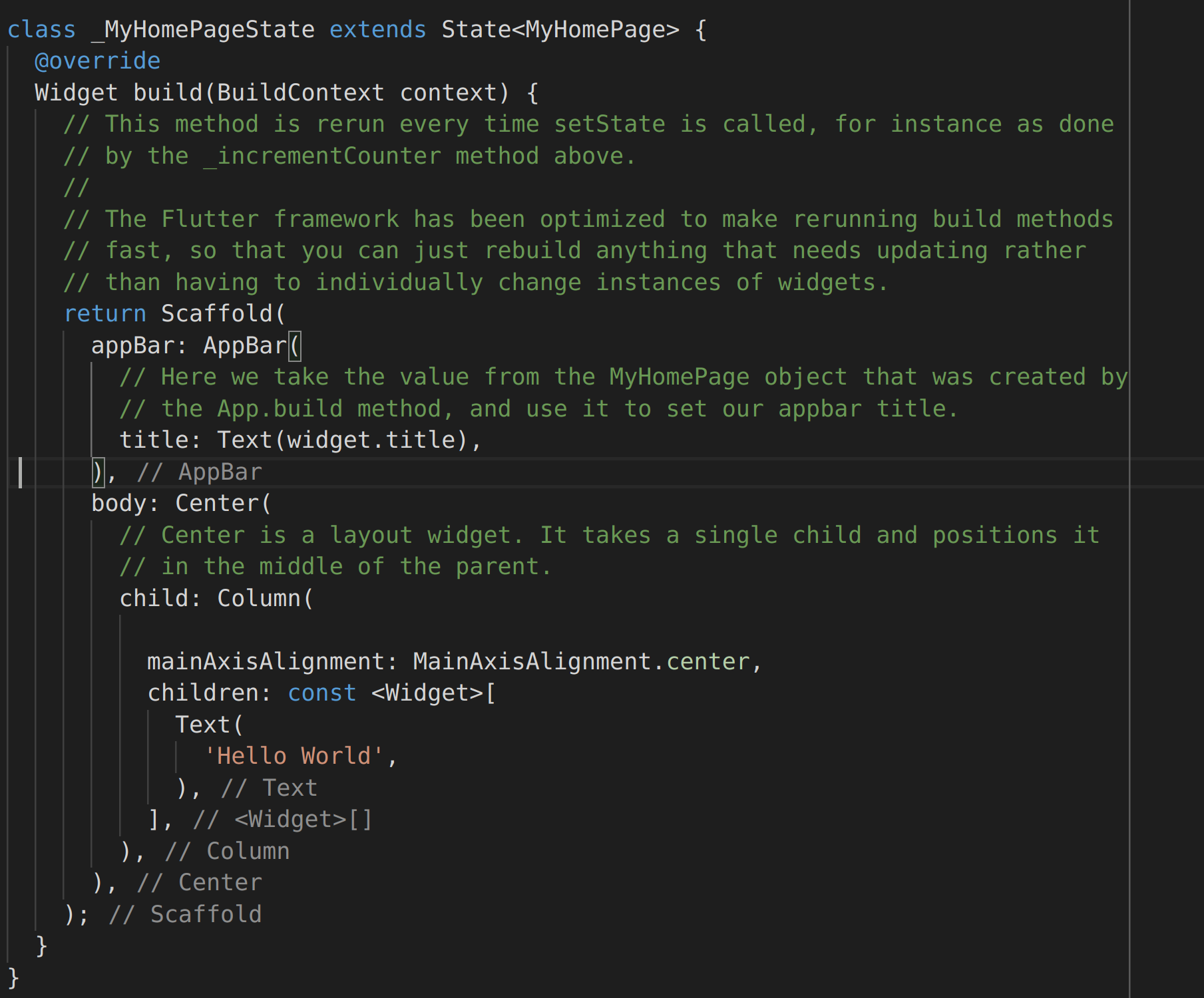


In both activities I had to comment out the requestQueue object that was created and called to queue the API request I make. Then instead, the VolleySingleton class I created was used statically in the activities to queue the requests made.

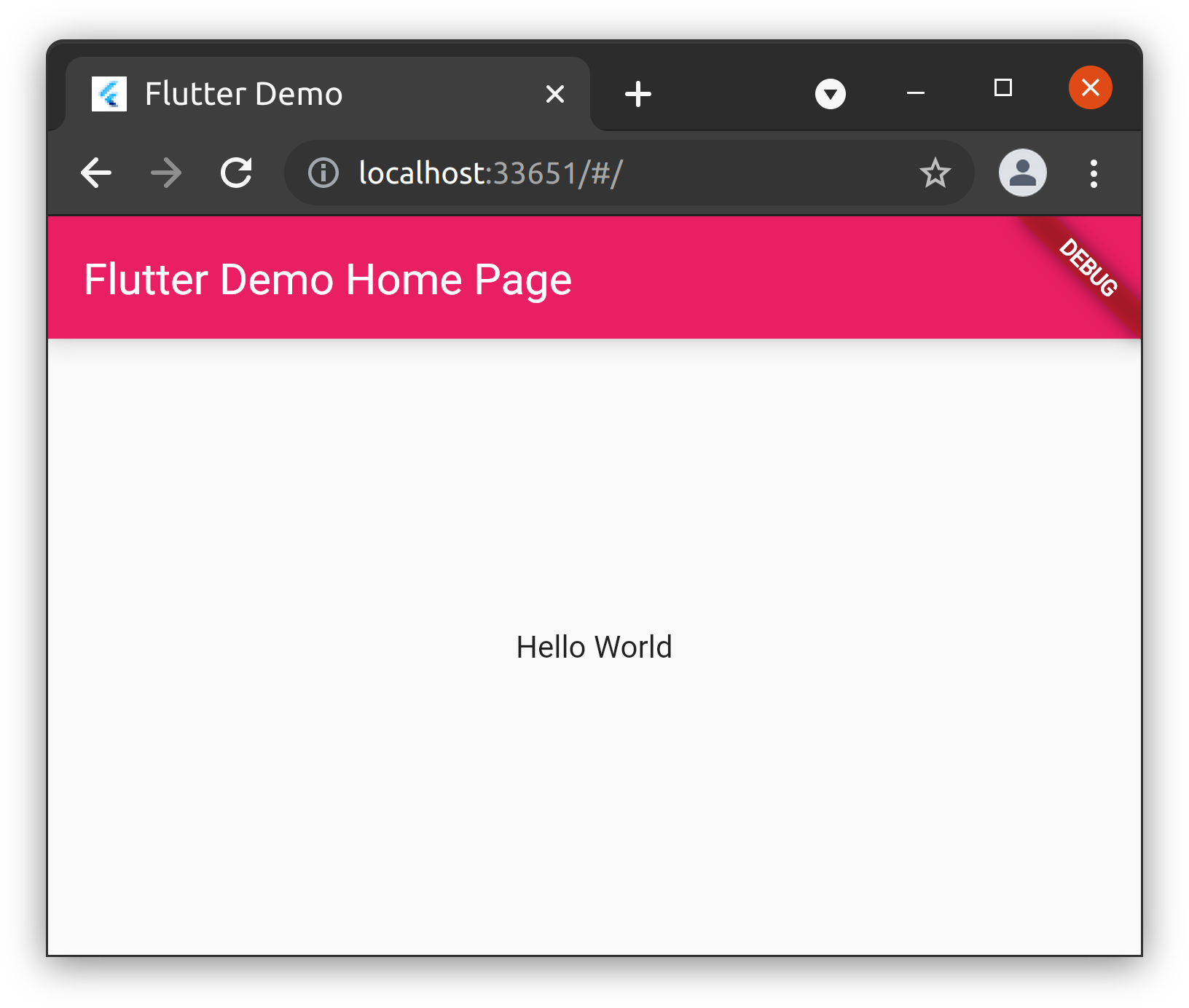
**6.3 Set Up Flutter dev environment and run the Hello World**

https://github.com/evvic/mobile\_app\_development/tree/main/flutter\_intro/hello\_flutter

It was surprisingly hard for me to get Flutter set up but I am happy it is working smoothly now through VS code.



I started a general application that initializes with centered text and a button you can press to increment a counter. I removed this stuff and just changed the centered text to say ”Hello World”, then ran the applcaition.

Then in the console I ran the command ”flutter run” to display the project in the browser, and in the console I pressed the ’r’ key to enable hot reloading.

# Week exercises

# Week exercises

# Week exercises

# Week exercises

# 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.)