Assignment 5

40 points

Due: November 9, 2022 11:59pm

In this assignment you will be adding MVVM architecture to your Jetpack Compose application. The view models will be responsible for making REST API calls to OpenWeatherMap and displaying the data received from there in the views. At a high level these are the steps:

1. Create a ViewModel class for each of your screens: CurrentConditionsViewModel and ForecastViewModel. The view models should not create instances of Moshi, Retrofit, or the Api. Instead, they should receive only the necessary dependencies as constructor arguments. Move your networking calls to your view models and expose the result of those network calls through Flow properties.
2. Enable Hilt for dependency injection. Following the directions in the slide deck #13, set up an Application class, an application module, and use the @Inject annotation correctly. After this step, you should only have one instance of Moshi, Retrofit, and Api in your application. Additionally, you should not be manually creating instances of your CurrentConditionsViewModel or ForecastViewModel, as Hilt should be creating these when you inject them into your Activities.
3. Create and properly annotate your data classes to parse the data received from OpenWeatherMap with Moshi.
4. Use Kotlin Coroutines to make your API calls by setting up an API interface with Retrofit.

Part 0 – Setup

Project build.gradle plugins object:

```
id 'com.google.dagger.hilt.android' version '2.44' apply false
```

Add the dependencies for Coil, Moshi, Retrofit2, and the Moshi Retrofit2 converter to your module level build.gradle file.

```
implementation 'com.squareup.moshi:moshi-kotlin:1.13.0'

implementation 'com.squareup.retrofit2:converter-moshi:2.9.0'

implementation 'com.squareup.retrofit2:retrofit:2.9.0'

implementation "io.coil-kt:coil-compose:2.2.2"

implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-core:1.6.0'
implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-android:1.6.0'

implementation 'androidx.hilt:hilt-navigation-compose:1.0.0'

implementation "com.google.dagger:hilt-android:2.44"
kapt "com.google.dagger:hilt-compiler:2.44"
```

Plugins to add to the plugins object in the build.gradle file:

```
id 'kotlin-kapt'
id 'dagger.hilt.android.plugin'
```

Add the internet permission to the AndroidManifest.xml in the “manifest” tag.

```
<uses-permission android:name="android.permission.INTERNET" />
```

There is currently a bug with using Moshi and Hilt together. Therefore if your application fails to build you must add this line to the end of your gradle.properties file:

```
android.jetifier.ignorelist=moshi-1.13.0
```

Part 1 – Current Conditions

For this part you will use the current weather data endpoint documented here:
https://pro.openweathermap.org/current

Specifically, use the “Built-in API request by ZIP code” for now and hard code the ZIP code to one of your choosing.

Part 2 – Forecast

For this part you will use the forecast data endpoint documented here:

https://openweathermap.org/forecast16

Specifically, use the “Built-in API request by ZIP code” for now and hoard code the ZIP code to one of your choosing.

Requirements:

1. Do not make any UI changes.
2. All functionality from the previous assignments should remain. In other words, on the CurrentConditionsScreen display the current conditions data as provided by a network call to the OpenWeatherMap API. On the ForecastScreen display the forecast items as provided by a network call to the OpenWeatherMap API.
3. You must have two (and only two) view models.
4. Each view model must expose one property each for the appropriate data type a channel exposed as a flow like in the lecture demo.
5. You must create a PR for your submission and submit a link to your PR in D2L
6. Your branch name for this assignment must be “assignment5”
7. You must not have any unused classes or code. Remove commented out code before submitting

Hints:

1. Do each of the above high level steps in turn. Only move on to the next step once you have completed a step after verifying that the application is working.
2. Get started early.
3. Use Logcat to view exceptions and errors. If you don’t understand what the error means copy the error and search for it on Google or StackOverflow.

4. Use the debugger to inspect the runtime state of your application if things aren't working as expected.
5. After creating your Application class, do not forget to update the application tag in your AndroidManifest.xml with the class name of the Application class you created.
6. Merge your assignment 4 branch into main and create a branch for assignment 5 before beginning.
7. After creating your PR, verify that that the code that you expect to see has been pushed to GitHub. I will not accept late submissions for students who fail to do this.
8. Read through the instructions before beginning work. Make sure you understand each task.
9. If you have questions about requirements ask, don't make assumptions.
10. Double check your work after submitting your PR. Go through your PR as though you were a reviewer looking for malformed or unused code.

Point breakdown

- 30 – All requirements met
- 5 – Good code style and formatting
- 5 – Proper branch naming and creation of GitHub PR