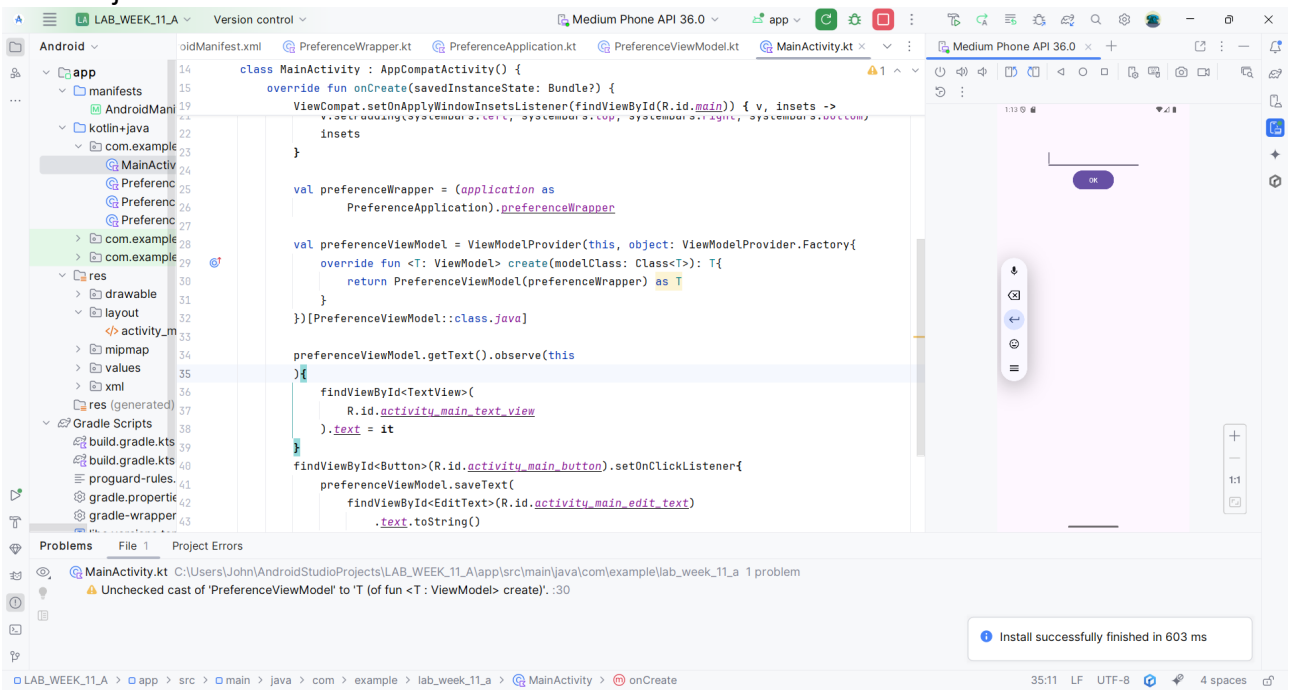
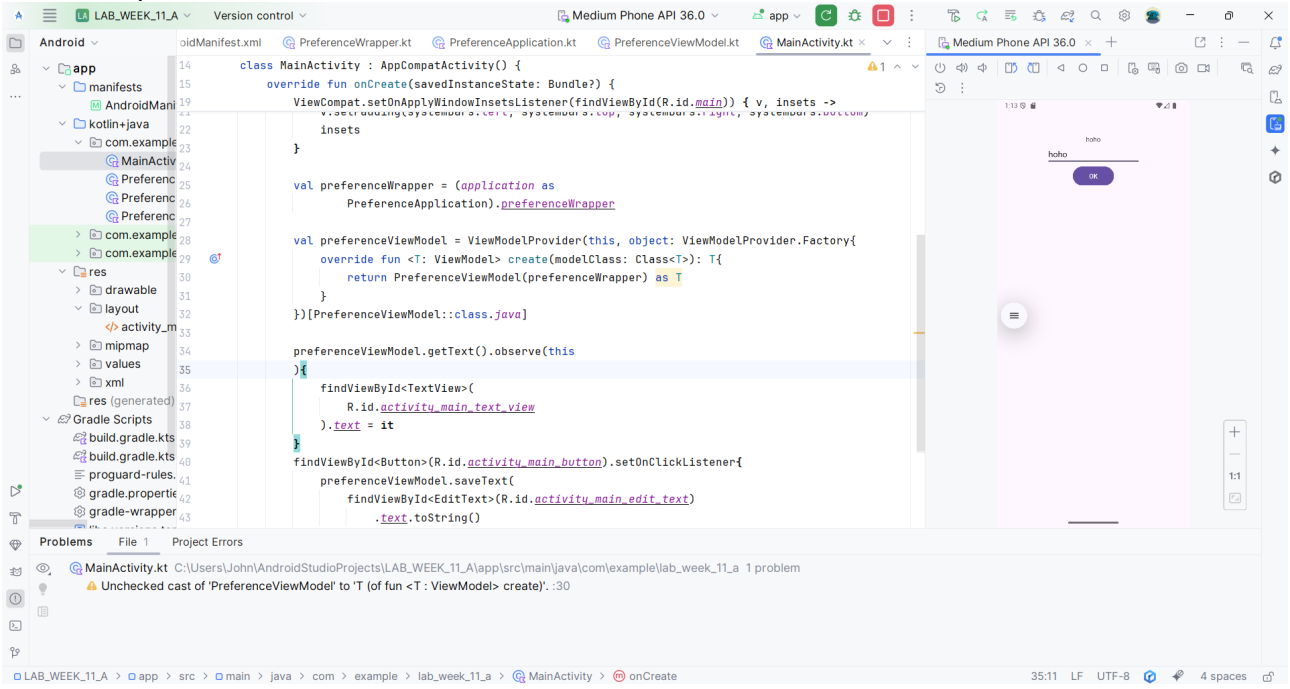
Nama : John Isaac Witness
NIM : 000000 88626
Mobile Application Programming Week 11
Link GitHub untuk LAB_WEEK_11_A: https://github.com/akunjone/LAB_WEEK_11_A
Link GitHub untuk LAB_WEEK_11_B: https://github.com/akunjone/LAB_WEEK_11_B

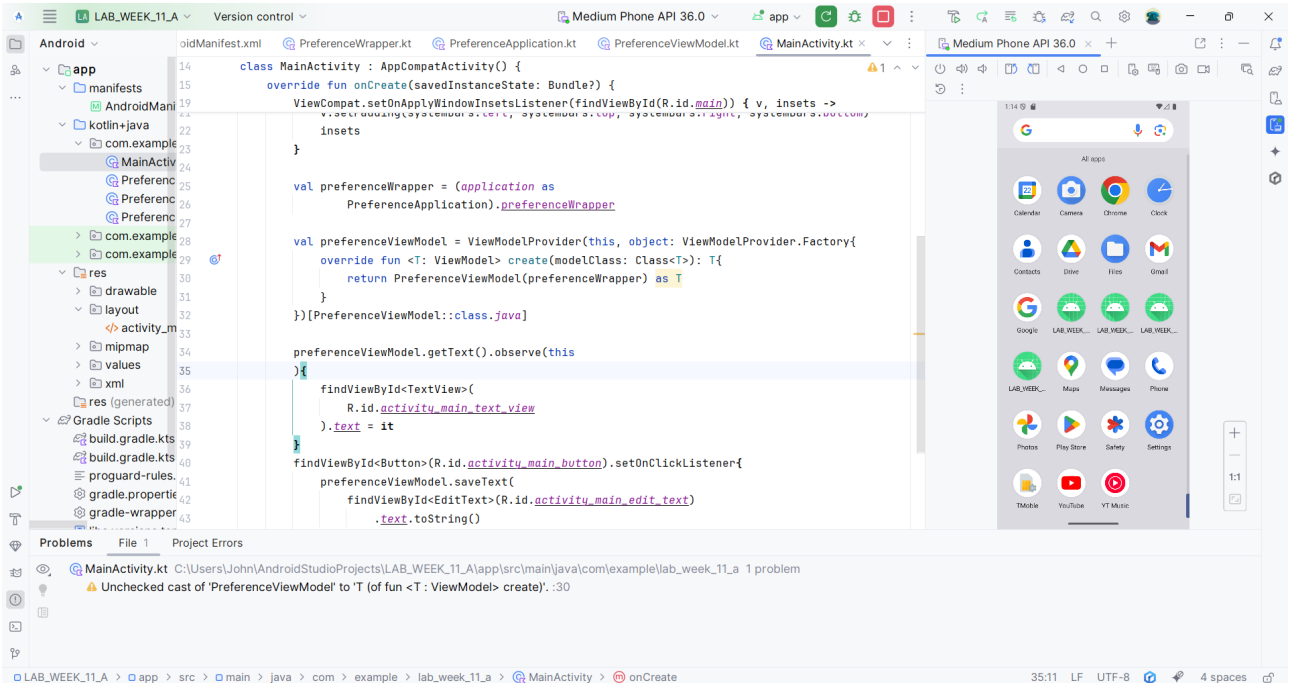
Part 1 - Building App with SharedPreferences
Output 1:
Ketika dijalankan:



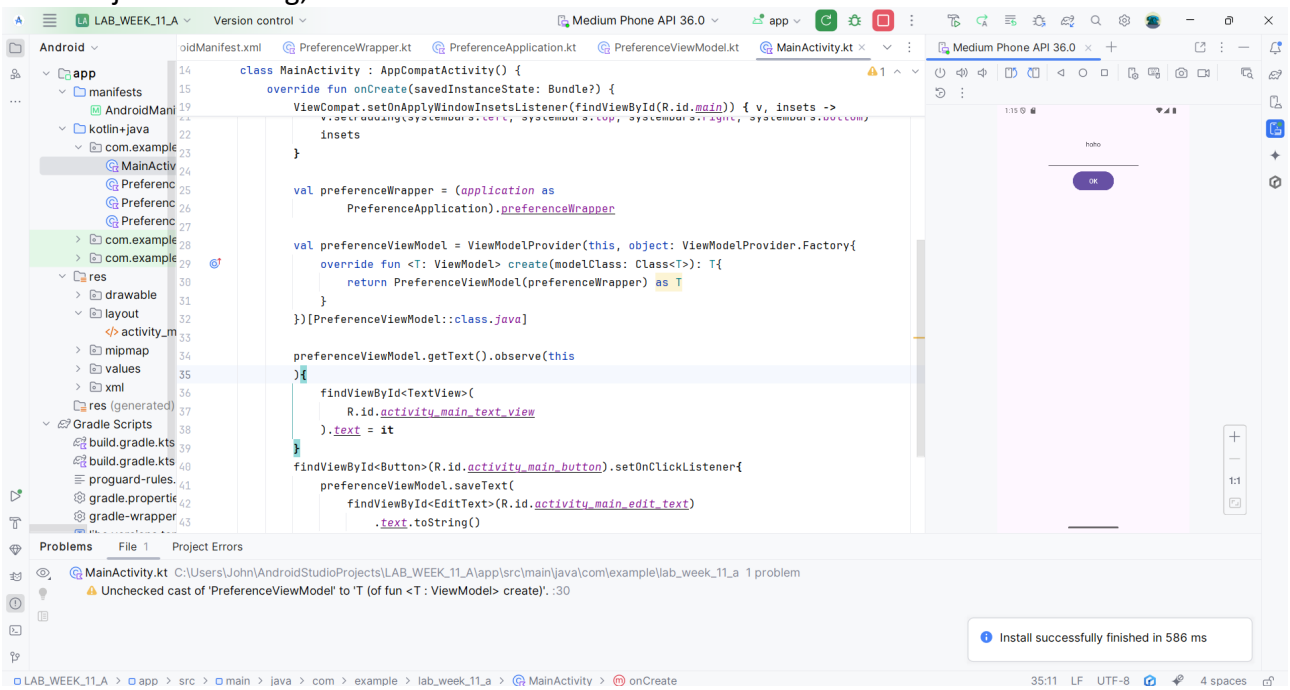
Ketika diinput “hoho”:



Ketika keluar:

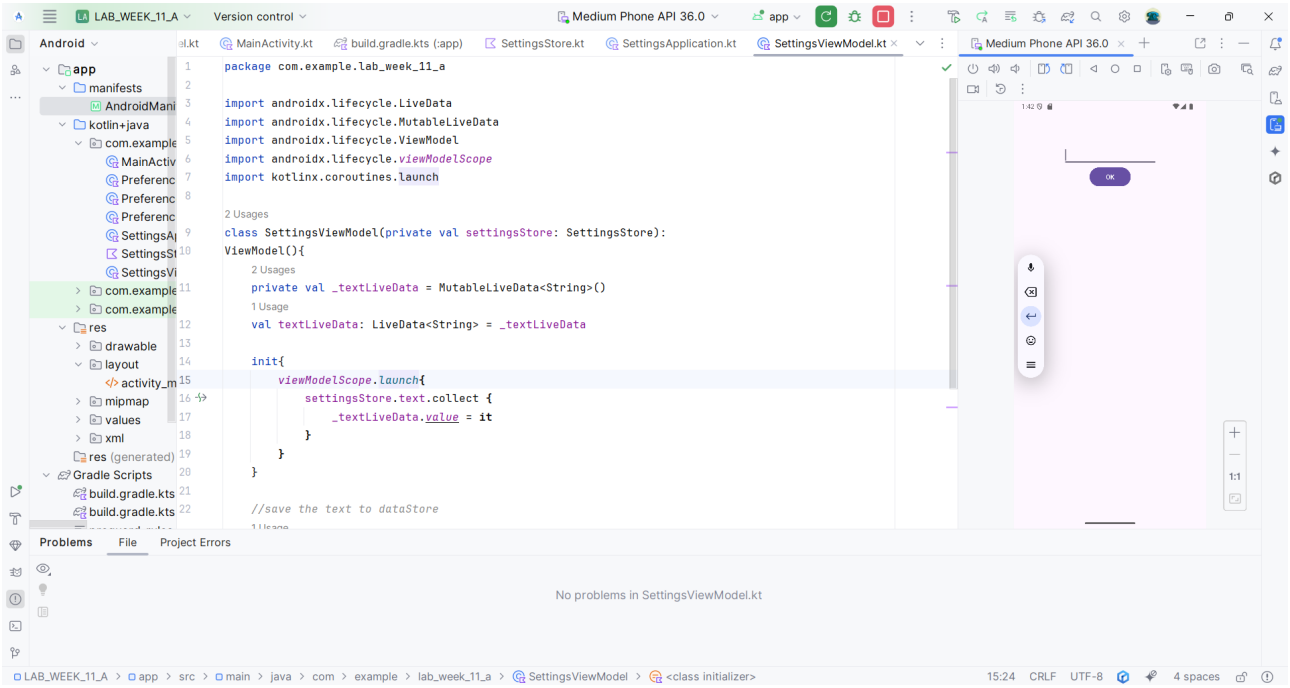


Ketika dijalankan ulang, dia akan kembali ke value “hoho” awal sebelum keluar:

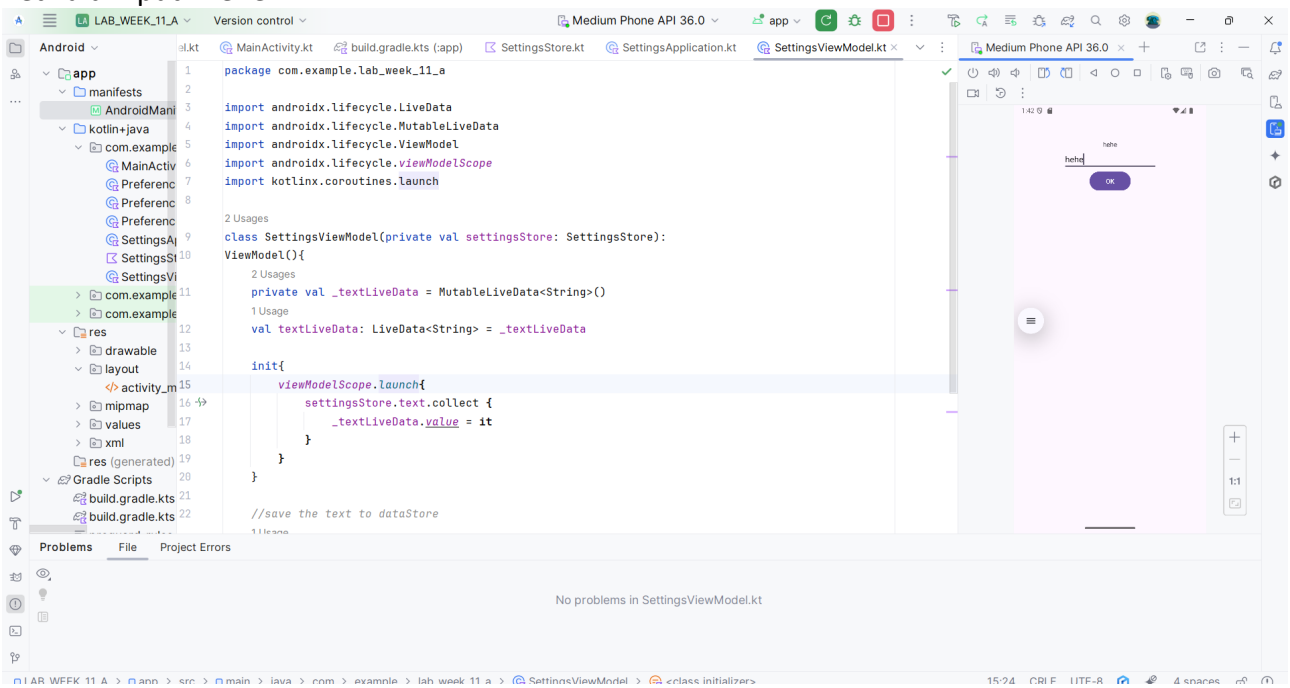


Part 2 - Building App with DataStore

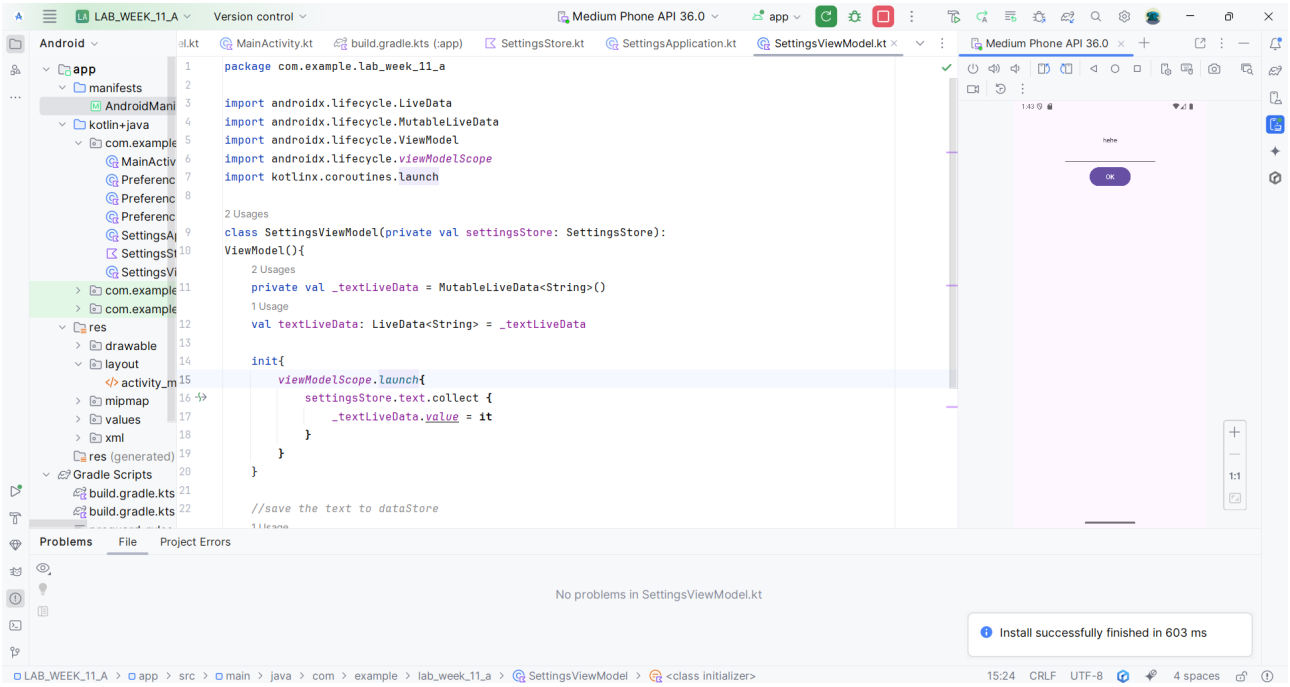
DataStore lebih efisien dan aman daripada SharedPreferences, lebih baru dan modern juga.



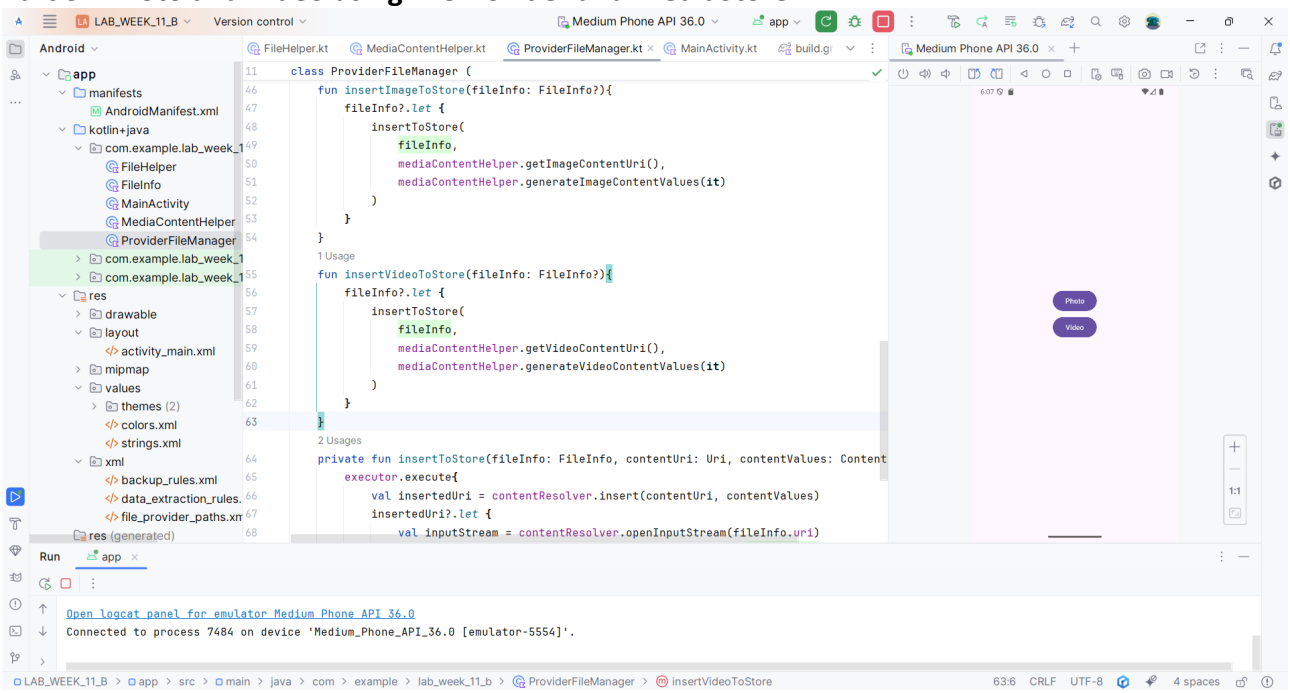
Ketika diinput “hehe”:



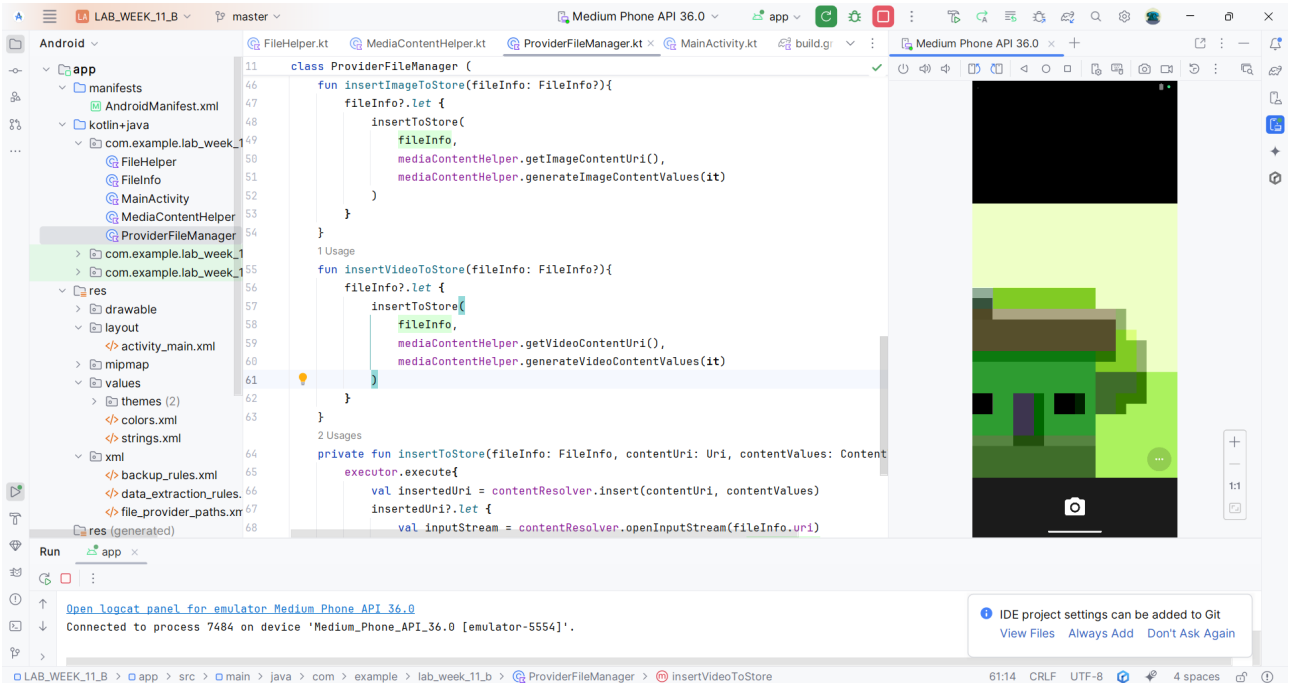
Ketika keluar, kemudian masuk aplikasi lagi, aplikasi akan bekerja seperti sebelumnya:



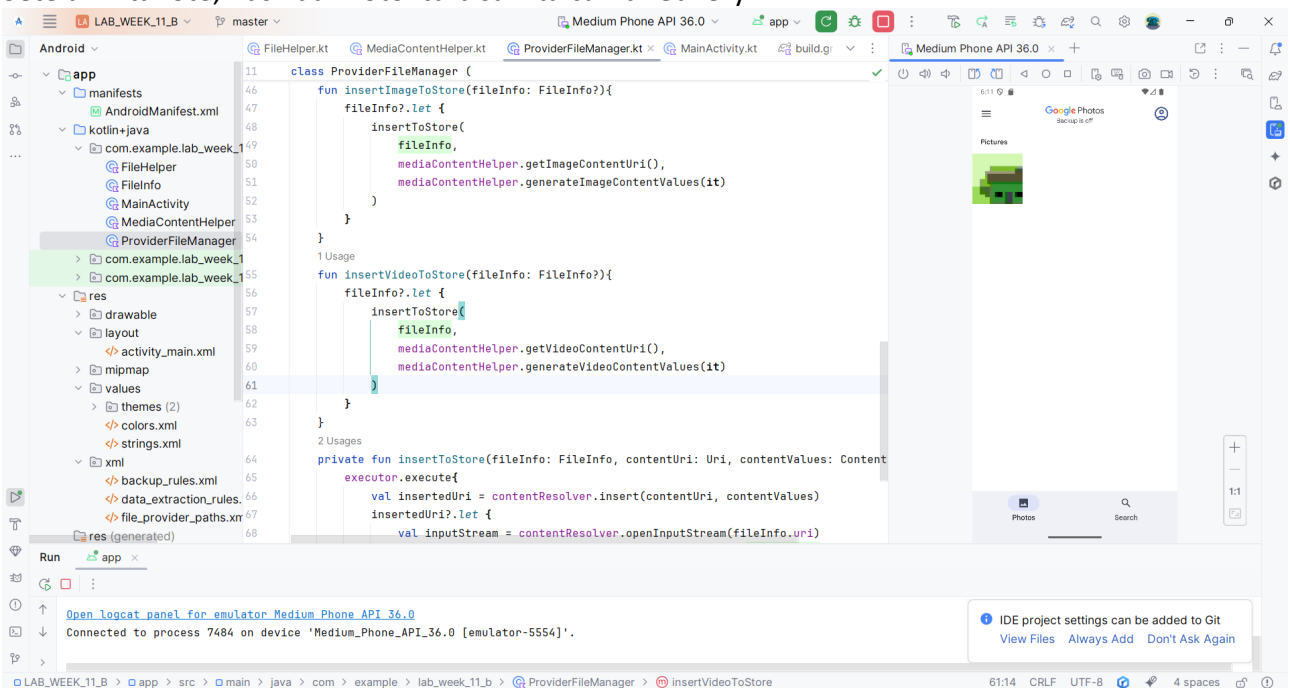
Part 3 - Photo and Video using FileProvider and MediaStore



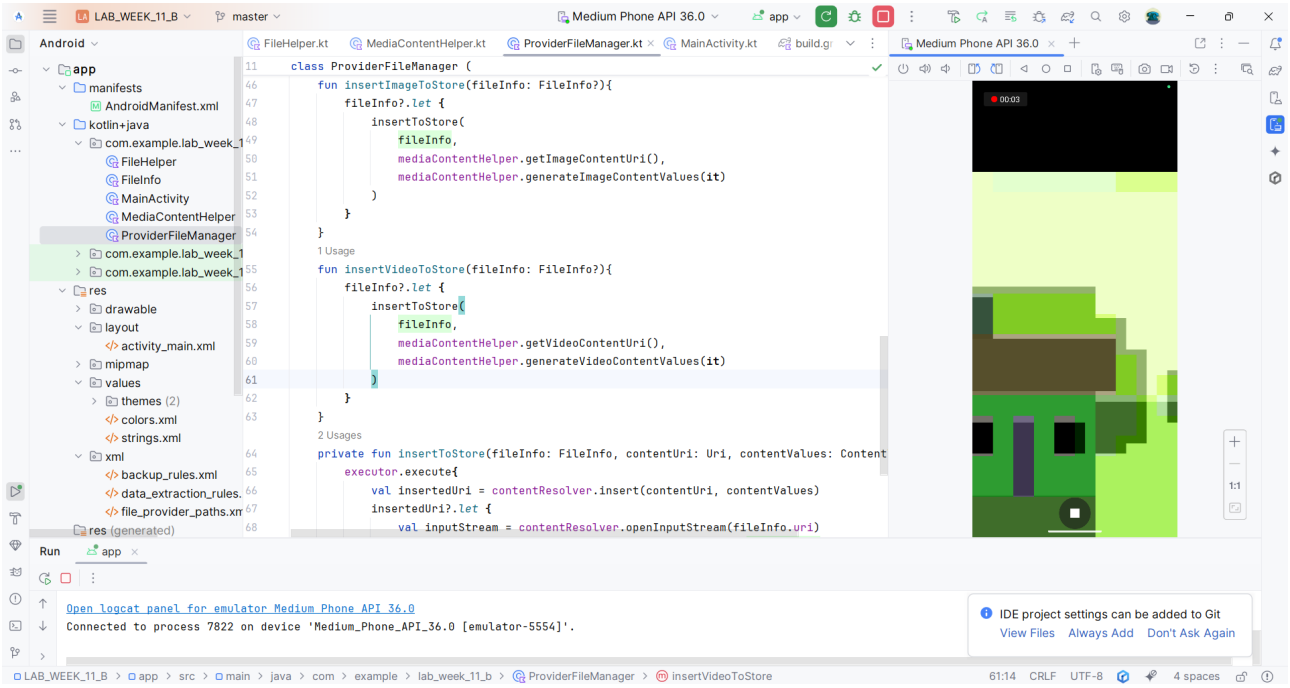
Jika “photo” diklik akan terbuka kamera:



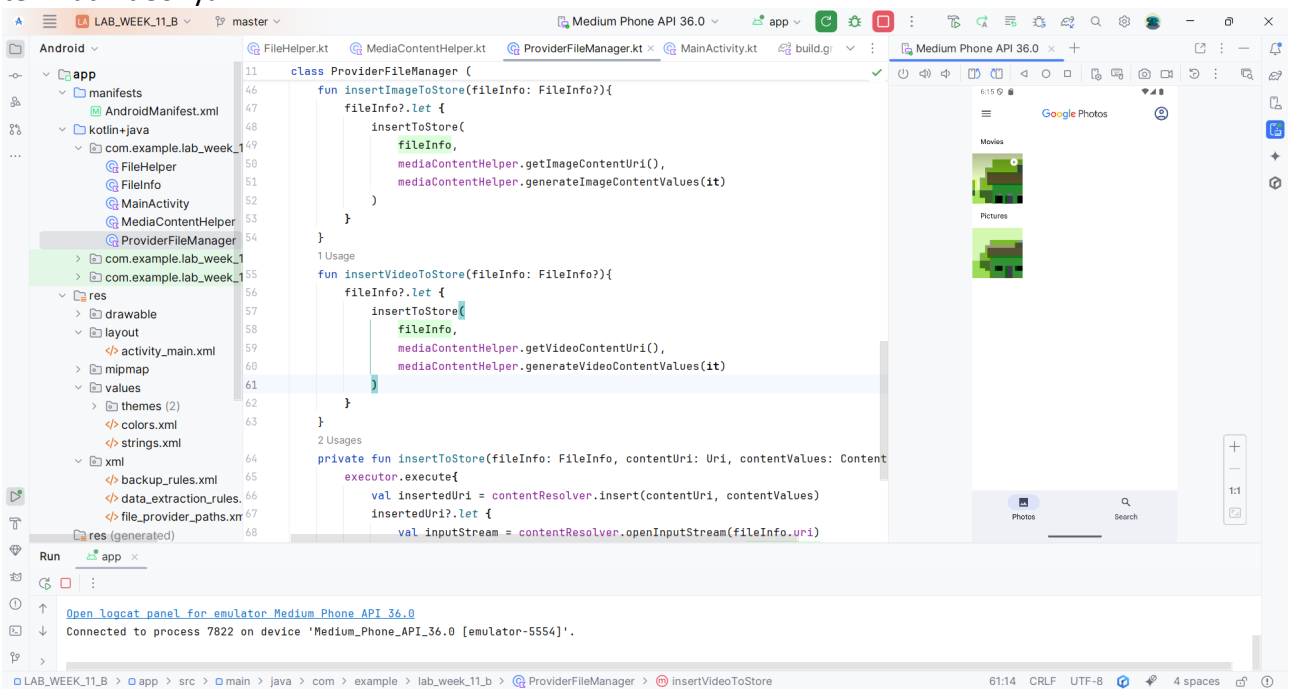
Setelah kita foto, hasil dari foto itu bisa kita cari di Gallery:



Kalau mengklik “video”, akan terbuka layar untuk merekam, ketika mulai rekam:



Setelah selesai rekam dan klik centang (done), maka video/movies akan tersimpan di Gallery, dan terlihat videonya:



Sebenarnya ini harusnya ditulis di txt, tetapi saya akan jawab disini juga (Terkait assignment).
Answer these questions based on Part 3 of the tutorial:

1. When a user takes a picture, that picture is stored in a path based on the given URI. In which part of the code handles this? (Copy that part of the code as the answer)
2. In your FileInfo.kt, there are 5 attributes. On the first attribute, what does the URI refer to? And on the fourth attribute, what does relativePath refer to?
3. [Bonus] Explain the chronological order from when a user takes a picture until the file is stored in the MediaStore.

[JAWAB]

1. The photo is stored in a path based on the given URI in this code:
```
fun generatePhotoUri(time: Long): FileInfo{
    val name = "img_$time.jpg"
    val file = File(
        context.getExternalFilesDir(fileHelper.getPicturesFolder()),
        name
    )
    return FileInfo(
        fileHelper.getUriFromFile(file),
        file,
        name,
        fileHelper.getPicturesFolder(),
        "image/jpeg"
    )
}
```
2. The first Attribute used by generatePhotoUri and generateVideoUri in ProviderFileManager file. They both return the FileInfo with the fileHelper.getUriFromFile(file) as the argument. We know that they refer to the getUriFromFile in the FileHelper class, which is URI, it helps to get the **temporary reference to the file so the app can access the file without reading ALL of the storage.**
   The fourth attribute, the relativePath, which used by generatePhotoUri and generateVideoUri functions in ProviderFileManager class, (fileHelper.getPicturesFolder() and fileHelper.getVideosFolder() to **get the name of the folder where pictures and movies stored**.
3. When the user clicking photo button in MainActivity, the app set isCapturingVideo to false and calls the checkStoragePermission. The permission checked in the MediaContentHelper class, to make sure the app has storage permissions (for android 9 to below) and if the permission is granted or running android 10+, it calls openImageCapture() function. Inside the openImageCapture(), the app called providerFileManager.generatePhotoUri() to generate Uri and make the user get the FileInfo. When the user take a pphoto, the takePictureLauncher.launch(uri) called and the app pass Uri to tell the camera the exact location to save the data. If the user click “done” or a check, the camera app write the image data to the address in Uri, now the getExternalFilesDir folder contains the image. The page activity now back to the MainActivity. And calls the registerForActivityResult to check if the operation successful. If true, it calls providerFileManager.insertImageToStore(photoInfo) to insert the photo to MediaStore. And by providerFileManager.insertImageToStore calls insertToStore, which inside, the mediaContentValues generate contentValues, executor.execute start a background thread, contentResolver.insert make a new empty place in Gallery App using mediastore, it will returns a new insertedUri. Finally, the InputStream is opened from the original photo, OutputStream opened to the MediaStore entry, and IOUtils.copy put the photo from input to output. So the photo’s now can be viewed by user in Gallery App.

Terimakasih.