

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:

The screenshot shows the Android Studio interface with the code editor open to the `MainActivity.kt` file. The code implements a `ViewModelProvider` to observe the preference value and update the UI accordingly.

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
            insets
        }
        val preferenceWrapper = (application as PreferenceApplication).preferenceWrapper
        val preferenceViewModel = ViewModelProvider(this, object: ViewModelProvider.Factory{
            override fun <T: ViewModel> create(modelClass: Class<T>): T{
                return PreferenceViewModel(preferenceWrapper) as T
            }
        })[PreferenceViewModel::class.java]
        preferenceViewModel.getText().observe(this)
        {
            findViewById<TextView>(R.id.activity_main_text_view).text = it
        }
        findViewById<Button>(R.id.activity_main_button).setOnClickListener{
            preferenceViewModel.savetxt()
            findViewById<EditText>(R.id.activity_main_edit_text).text.toString()
        }
    }
}
```

The bottom status bar indicates "Install successfully finished in 603 ms".

Ketika diinput "hoho":

The screenshot shows the same Android Studio setup as above, but with the word "hoho" typed into the text view. The bottom status bar still shows the successful installation message.

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
            insets
        }
        val preferenceWrapper = (application as PreferenceApplication).preferenceWrapper
        val preferenceViewModel = ViewModelProvider(this, object: ViewModelProvider.Factory{
            override fun <T: ViewModel> create(modelClass: Class<T>): T{
                return PreferenceViewModel(preferenceWrapper) as T
            }
        })[PreferenceViewModel::class.java]
        preferenceViewModel.getText().observe(this)
        {
            findViewById<TextView>(R.id.activity_main_text_view).text = it
        }
        findViewById<Button>(R.id.activity_main_button).setOnClickListener{
            preferenceViewModel.savetxt()
            findViewById<EditText>(R.id.activity_main_edit_text).text.toString()
        }
    }
}
```

Ketika keluar:

```

class MainActivity : AppCompatActivity {
    override fun onCreate(savedInstanceState: Bundle?) {
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
            insets
        }
    }

    val preferenceWrapper = (application as
        PreferenceApplication).preferenceWrapper

    val preferenceViewModel = ViewModelProvider(this, object: ViewModelProvider.Factory{
        override fun <T: ViewModel> create(modelClass: Class<T>): T{
            return PreferenceViewModel(preferenceWrapper) as T
        }
    })[PreferenceViewModel::class.java]

    preferenceViewModel.getText().observe(this
    ){ text -
        findViewById<TextView>(R.id.activity_main_text_view)
            .text = it

        findViewById<Button>(R.id.activity_main_button).setOnClickListener{
            preferenceViewModel.savetxt(
                findViewById<EditText>(R.id.activity_main_edit_text)
                    .text.toString()
            )
        }
    }
}

```

Problems File 1 Project Errors

MainActivity.kt C:\Users\John\AndroidStudioProjects\LAB_WEEK_11_A\app\src\main\java\com\example\lab_week_11_a 1 problem

Unchecked cast of 'PreferenceViewModel' to 'T (of fun <T : ViewModel> create)': 30

LAB_WEEK_11_A > app > src > main > java > com > example > lab_week_11_a > MainActivity > onCreate

35:11 LF UTF-8 4 spaces

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

```

class MainActivity : AppCompatActivity {
    override fun onCreate(savedInstanceState: Bundle?) {
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
            insets
        }
    }

    val preferenceWrapper = (application as
        PreferenceApplication).preferenceWrapper

    val preferenceViewModel = ViewModelProvider(this, object: ViewModelProvider.Factory{
        override fun <T: ViewModel> create(modelClass: Class<T>): T{
            return PreferenceViewModel(preferenceWrapper) as T
        }
    })[PreferenceViewModel::class.java]

    preferenceViewModel.getText().observe(this
    ){ text -
        findViewById<TextView>(R.id.activity_main_text_view)
            .text = it

        findViewById<Button>(R.id.activity_main_button).setOnClickListener{
            preferenceViewModel.savetxt(
                findViewById<EditText>(R.id.activity_main_edit_text)
                    .text.toString()
            )
        }
    }
}

```

Problems File 1 Project Errors

MainActivity.kt C:\Users\John\AndroidStudioProjects\LAB_WEEK_11_A\app\src\main\java\com\example\lab_week_11_a 1 problem

Unchecked cast of 'PreferenceViewModel' to 'T (of fun <T : ViewModel> create)': 30

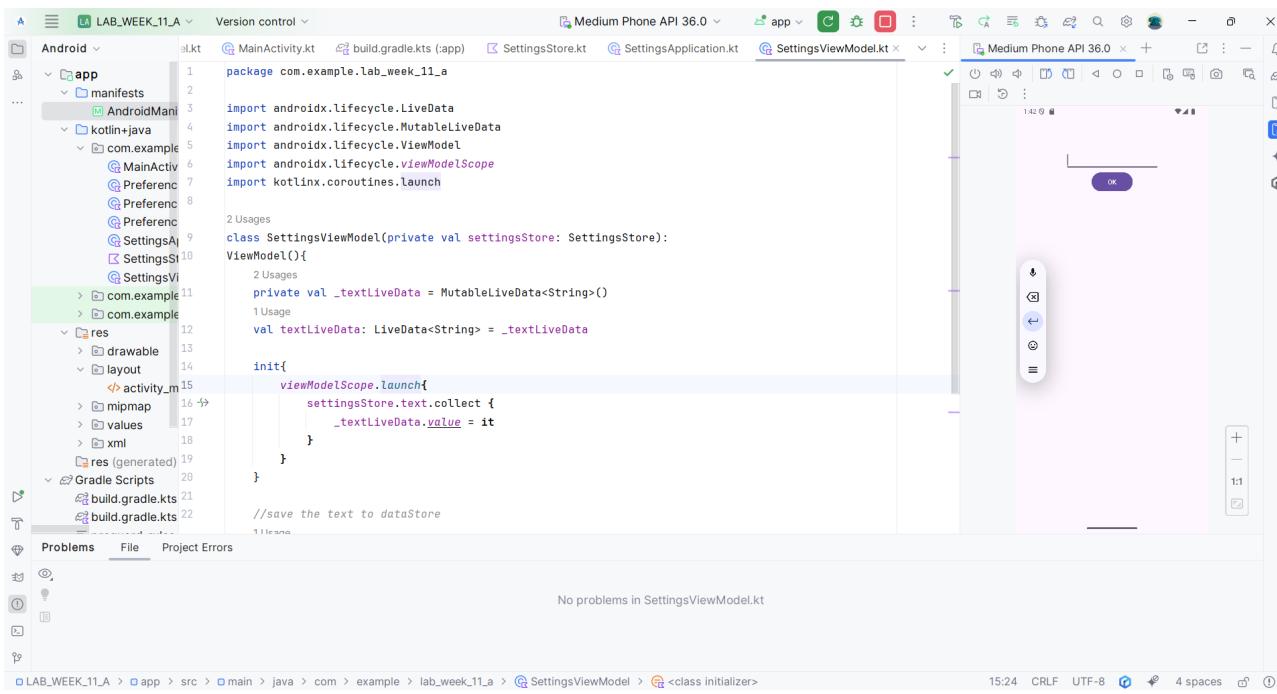
Install successfully finished in 586 ms

LAB_WEEK_11_A > app > src > main > java > com > example > lab_week_11_a > MainActivity > onCreate

35:11 LF UTF-8 4 spaces

Part 2 - Building App with DataStore

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

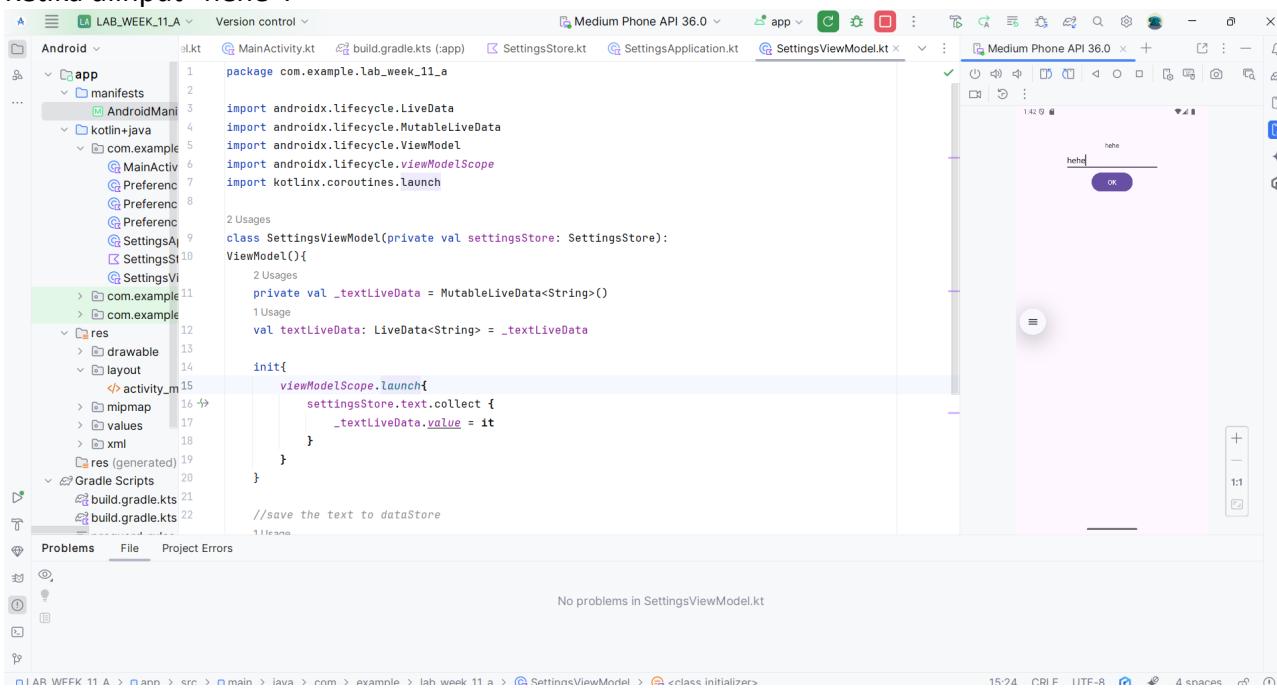


```
1 package com.example.lab_week_11_a
2
3 import androidx.lifecycle.LiveData
4 import androidx.lifecycle.MutableLiveData
5 import androidx.lifecycle.ViewModel
6 import androidx.lifecycle.viewModelScope
7 import kotlinx.coroutines.launch
8
9 class SettingsViewModel(private val settingsStore: SettingsStore):
10 ViewModel() {
11     2 Usages
12     private val _textLiveData = MutableLiveData<String>()
13
14     init{
15         viewModelScope.launch{
16             settingsStore.text.collect {
17                 _textLiveData.value = it
18             }
19         }
20     }
21
22     //save the text to dataStore
23 }
```

No problems in SettingsViewModel.kt

LAB_WEEK_11_A > app > src > main > java > com > example > lab_week_11_a > SettingsViewModel > <class initializers>

Ketika diinput "hehe":



```
1 package com.example.lab_week_11_a
2
3 import androidx.lifecycle.LiveData
4 import androidx.lifecycle.MutableLiveData
5 import androidx.lifecycle.ViewModel
6 import androidx.lifecycle.viewModelScope
7 import kotlinx.coroutines.launch
8
9 class SettingsViewModel(private val settingsStore: SettingsStore):
10 ViewModel() {
11     2 Usages
12     private val _textLiveData = MutableLiveData<String>()
13
14     init{
15         viewModelScope.launch{
16             settingsStore.text.collect {
17                 _textLiveData.value = it
18             }
19         }
20     }
21
22     //save the text to dataStore
23 }
```

No problems in SettingsViewModel.kt

LAB_WEEK_11_A > app > src > main > java > com > example > lab_week_11_a > SettingsViewModel > <class initializers>

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

```
1 package com.example.lab_week_11_a
2
3 import androidx.lifecycle.LiveData
4 import androidx.lifecycle.MutableLiveData
5 import androidx.lifecycle.ViewModel
6 import androidx.lifecycle.viewModelScope
7 import kotlinx.coroutines.launch
8
9 class SettingsViewModel(private val settingsStore: SettingsStore):
10 ViewModel() {
11     2 Usages
12     private val _textLiveData = MutableLiveData<String>()
13
14     init{
15         viewModelScope.launch{
16             settingsStore.text.collect {
17                 _textLiveData.value = it
18             }
19         }
20     }
21     //save the text to dataStore
22 }
```

No problems in SettingsViewModel.kt

Install successfully finished in 603 ms

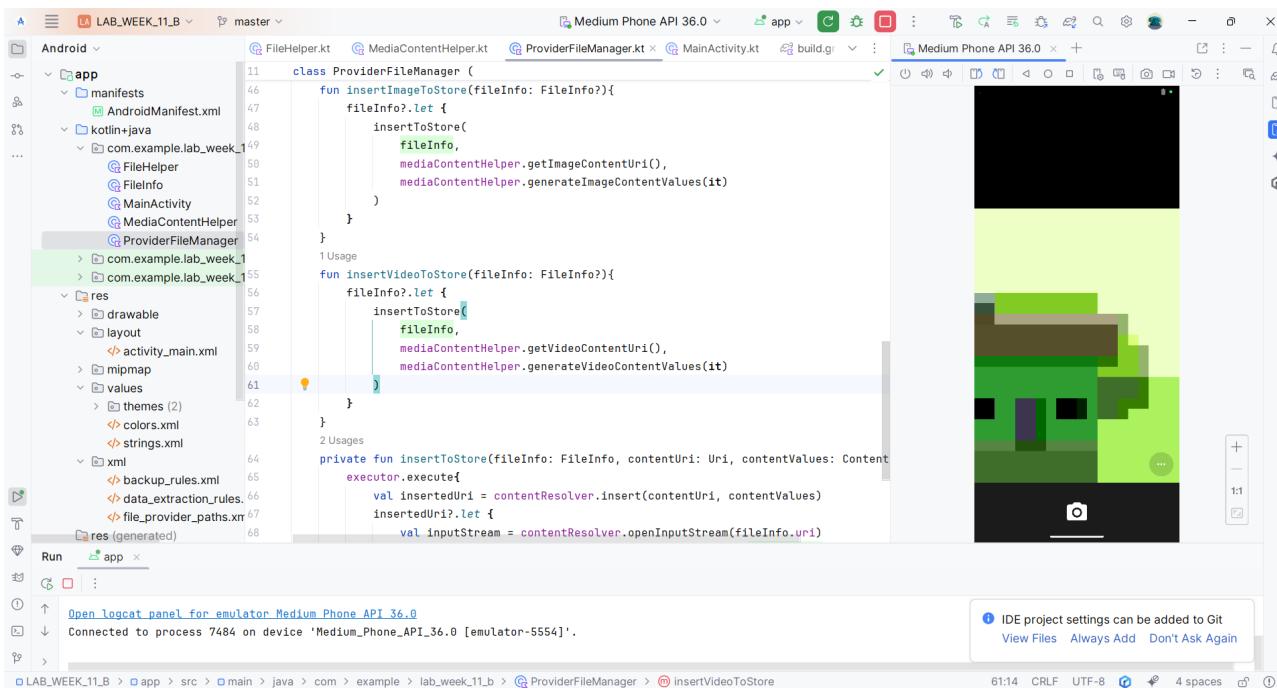
Part 3 - Photo and Video using FileProvider and MediaStore

```
11 class ProviderFileManager {
12     fun insertImageToStore(fileInfo: FileInfo?) {
13         fileInfo?.let {
14             insertToStore(
15                 fileInfo,
16                 mediaContentHelper.getImageContentUri(),
17                 mediaContentHelper.generateImageContentValues(it)
18             )
19         }
20     }
21     1 Usage
22     fun insertVideoToStore(fileInfo: FileInfo?) {
23         fileInfo?.let {
24             insertToStore(
25                 fileInfo,
26                 mediaContentHelper.getVideoContentUri(),
27                 mediaContentHelper.generateVideoContentValues(it)
28             )
29         }
30     }
31     2 Usages
32     private fun insertToStore(fileInfo: FileInfo, contentUri: Uri, contentValues: Content
33             executor.execute{
34                 val insertedUri = contentResolver.insert(contentUri, contentValues)
35                 insertedUri?.let {
36                     val inputStream = contentResolver.openInputStream(fileInfo.uri)
37                 }
38             }
39         }
40     }
```

Photo

Video

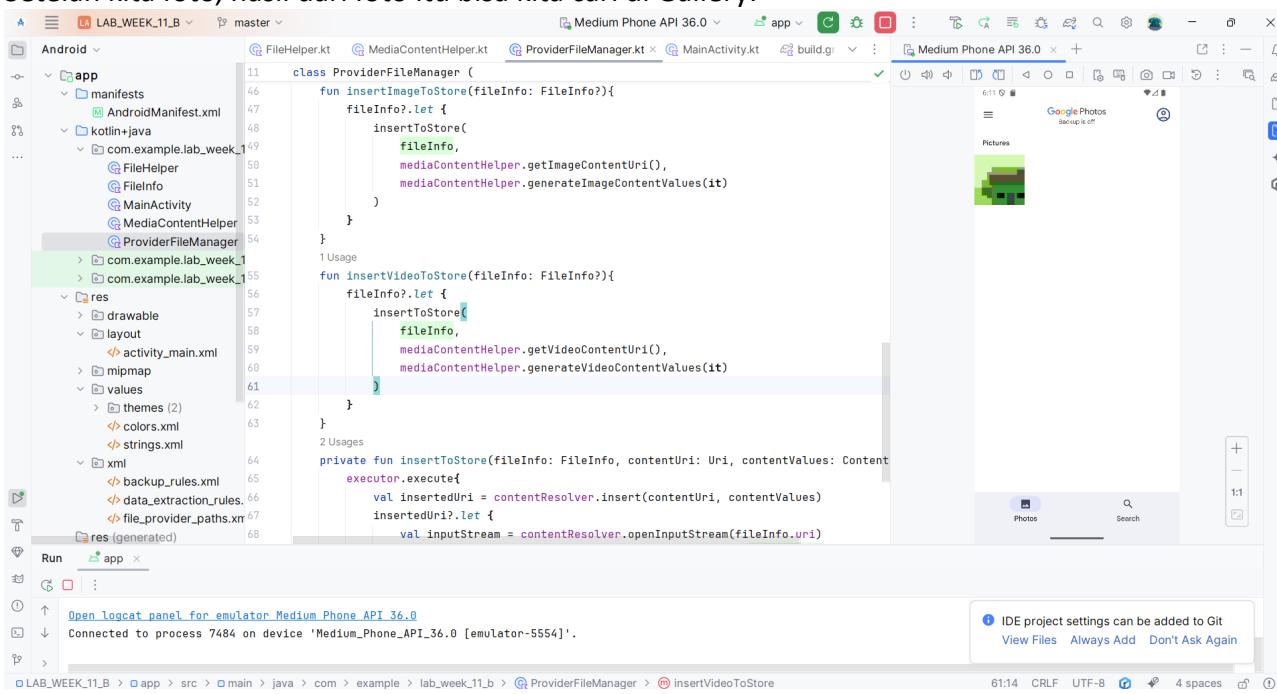
Jika "photo" diklik akan terbuka kamera:



The screenshot shows the Android Studio interface with the ProviderFileManager.kt file open in the editor. The code implements methods for inserting images and videos into a content provider. A camera preview window is displayed on the right side of the screen, showing a green and brown landscape scene.

```
11 class ProviderFileManager {
12     fun insertImageToStore(fileInfo: FileInfo?) {
13         fileInfo?.let {
14             insertToStore(
15                 fileInfo,
16                 mediaContentHelper.getImageContentUri(),
17                 mediaContentHelper.generateImageContentValues(it)
18             )
19         }
20     }
21
22     fun insertVideoToStore(fileInfo: FileInfo?) {
23         fileInfo?.let {
24             insertToStore(
25                 fileInfo,
26                 mediaContentHelper.getVideoContentUri(),
27                 mediaContentHelper.generateVideoContentValues(it)
28             )
29         }
30     }
31
32     private fun insertToStore(fileInfo: FileInfo, contentUri: Uri, contentValues: ContentValues) {
33         executor.execute {
34             val insertedUri = contentResolver.insert(contentUri, contentValues)
35             insertedUri?.let {
36                 val inputStream = contentResolver.openInputStream(insertedUri.uri)
37             }
38         }
39     }
40 }
```

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



The screenshot shows the Android Studio interface with the ProviderFileManager.kt file open in the editor. The code is identical to the previous screenshot. A Google Photos gallery window is displayed on the right side of the screen, showing a small thumbnail of a green and brown landscape photo.

```
11 class ProviderFileManager {
12     fun insertImageToStore(fileInfo: FileInfo?) {
13         fileInfo?.let {
14             insertToStore(
15                 fileInfo,
16                 mediaContentHelper.getImageContentUri(),
17                 mediaContentHelper.generateImageContentValues(it)
18             )
19         }
20     }
21
22     fun insertVideoToStore(fileInfo: FileInfo?) {
23         fileInfo?.let {
24             insertToStore(
25                 fileInfo,
26                 mediaContentHelper.getVideoContentUri(),
27                 mediaContentHelper.generateVideoContentValues(it)
28             )
29         }
30     }
31
32     private fun insertToStore(fileInfo: FileInfo, contentUri: Uri, contentValues: ContentValues) {
33         executor.execute {
34             val insertedUri = contentResolver.insert(contentUri, contentValues)
35             insertedUri?.let {
36                 val inputStream = contentResolver.openInputStream(insertedUri.uri)
37             }
38         }
39     }
40 }
```

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

The screenshot shows the Android Studio interface. On the left is the project structure for 'LAB_WEEK_11_B' with 'app' selected. The 'kotlin+java' section contains files like FileHelper.kt, MediaContentHelper.kt, ProviderFileManager.kt, MainActivity.kt, and build.gradle. The 'res' folder includes drawable, layout, mipmap, values, and xml subfolders. The right side displays the 'ProviderFileManager.kt' code in the editor, specifically the 'insertVideoToStore' function. Below the code is a preview window showing a video frame of a green character. At the bottom, the logcat panel shows 'Connected to process 7822 on device 'Medium_Phone_API_36.0 [emulator-5554]'.

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

This screenshot is similar to the first one but shows the 'Google Photos' application instead of the emulator. The 'Movies' album is open, displaying the recorded video file. The rest of the interface and code editor are identical to the first screenshot.

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

- 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)
- 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?
- [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.