**Module 5: Critical Thinking**

Mohammad Sargazi

Colorado State University Global

CSC475: Platform-Based Development

Dr. Bari

Oct 21, 2024

1. **Pseudocode**

Initialize MainActivity:

* Create MainActivity class that extends ComponentActivity.
* Register permission launcher to request permission for reading images.

On App Launch:

* Call checkAndRequestPermissions() to check if permissions are granted.
* If permission is granted, call loadGallery() to fetch and display images.
* If not, request permission from the user.

Check and Request Permissions:

* If Android version ≥ Tiramisu, request READ\_MEDIA\_IMAGES permission.
* Else, request READ\_EXTERNAL\_STORAGE permission.
* On permission grant, call loadGallery() to load images.

Load Images using Coroutine:

* Use CoroutineScope to fetch images in the background (IO thread).
* Call fetchImages() to retrieve images from MediaStore.
* On completion, call displayGallery() to show the images.

Fetch Images from MediaStore:

* Query MediaStore for image files and add their paths to a list.
* Log the image paths and total count for debugging.
* Return the list of image paths.

Display Images in LazyVerticalGrid:

* Use LazyVerticalGrid to display images in a scrollable grid layout.
* Each image is displayed using AsyncImage.
* When an image is clicked, show it in an AlertDialog.

Navigate between Images in Dialog:

* Use state to track the selected image index.
* Show the selected image with Next, Back, and Close buttons.
* Disable Next/Back buttons at the boundaries (first and last images).

1. **Purpose of Photogallery App**

When I set out to build this photo gallery application, my goal was to create a simple app that displays images from the device’s storage in a grid layout. I also wanted to provide a way for users to click on individual images and view them in full-screen mode, with smooth navigation using Next and Back buttons. The idea was to create an intuitive experience that mimics the way photo galleries work on most smartphones.

The biggest obstacle I faced was managing permissions and making sure the app worked across different Android versions. Android’s permission model can be tricky, especially with newer versions requiring READ\_MEDIA\_IMAGES instead of READ\_EXTERNAL\_STORAGE. At one point, the app crashed because the permissions weren’t requested properly. Additionally, I struggled with scroll conflicts between LazyVerticalGrid and verticalScroll, which caused the app to crash due to "infinite constraints." It took a bit of trial and error, reading through logs, and experimenting with layouts to get things right.

Another challenge I encountered was making sure the images loaded efficiently and didn’t consume too much memory. Using AsyncImage from Coil solved this problem, but I had to carefully manage the grid and prevent unnecessary scroll conflicts. Debugging the application with Logcat was an essential skill I picked up during this process. I learned how to use logging statements to track down issues and see how the app behaved at runtime.

This project taught me a lot about working with Jetpack Compose, handling permissions, and managing coroutines to fetch data in the background. I also learned how to use Coil for image loading, which was a new library for me. The experience was both challenging and rewarding, and it gave me the confidence to tackle real-world problems in Android development. I now feel more comfortable working with layouts, state management, and modern Android tools, and I’m excited to build more complex apps in the future!

1. Source Codes

3.1 : MainActivity.kt

package com.example.photogalary  
  
import android.Manifest  
import android.content.pm.PackageManager  
import android.os.Build  
import android.os.Bundle  
import android.provider.MediaStore  
import android.util.Log  
import android.widget.Toast  
import androidx.activity.ComponentActivity  
import androidx.activity.compose.setContent  
import androidx.activity.result.contract.ActivityResultContracts  
import androidx.compose.foundation.ExperimentalFoundationApi  
import androidx.compose.foundation.layout.padding  
import androidx.compose.foundation.lazy.grid.GridCells  
import androidx.compose.foundation.lazy.grid.LazyVerticalGrid  
import androidx.compose.runtime.Composable  
import androidx.compose.ui.Modifier  
import androidx.compose.ui.unit.dp  
import androidx.compose.material3.Text  
import androidx.core.content.ContextCompat  
import coil.compose.AsyncImage  
import com.example.photogalary.ui.theme.PhotoGalaryTheme  
import kotlinx.coroutines.CoroutineScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
import kotlinx.coroutines.withContext  
import androidx.compose.foundation.clickable  
import androidx.compose.foundation.layout.Row  
import androidx.compose.foundation.layout.Spacer  
import androidx.compose.foundation.rememberScrollState  
import androidx.compose.foundation.verticalScroll  
import androidx.compose.foundation.lazy.grid.items  
import androidx.compose.material3.AlertDialog  
import androidx.compose.material3.Button  
import androidx.compose.runtime.\*  
import androidx.compose.ui.graphics.ColorFilter  
  
class MainActivity : ComponentActivity() {  
  
 // Register for the result of a permission request  
 private val permissionLauncher =  
 registerForActivityResult(ActivityResultContracts.RequestPermission()) **{** isGranted **->** if (isGranted) {  
 loadGallery() // Load the gallery if permission is granted  
 } else {  
 Toast.makeText(this, "Permission denied!", Toast.*LENGTH\_SHORT*).show()  
 }  
 **}** override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 Log.d("PhotoGallery", "App launched, checking permissions...") // Debug log  
 checkAndRequestPermissions()  
 }  
  
 // Check and request permissions based on Android version  
 private fun checkAndRequestPermissions() {  
 val permission = if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*TIRAMISU*) {  
 Manifest.permission.*READ\_MEDIA\_IMAGES* } else {  
 Manifest.permission.*READ\_EXTERNAL\_STORAGE* }  
  
 if (ContextCompat.checkSelfPermission(this, permission) != PackageManager.*PERMISSION\_GRANTED*) {  
 permissionLauncher.launch(permission)  
 } else {  
 loadGallery() // Permission already granted  
 }  
 }  
  
 // Load the gallery in the background using a coroutine  
 private fun loadGallery() {  
 *CoroutineScope*(Dispatchers.Main).*launch* **{** val images = withContext(Dispatchers.IO) **{** fetchImages() // Fetch images on a background thread  
 **}** displayGallery(images) // Display images on the main thread  
 **}** }  
  
 // Fetch images from the device's MediaStore  
 private fun fetchImages(): List<String> {  
 val imageList = *mutableListOf*<String>()  
 val uri = MediaStore.Images.Media.*EXTERNAL\_CONTENT\_URI* val projection = *arrayOf*(MediaStore.Images.Media.*DATA*)  
  
 *contentResolver*.query(uri, projection, null, null, null)?.*use* **{** cursor **->** val columnIndex = cursor.getColumnIndexOrThrow(MediaStore.Images.Media.*DATA*)  
 while (cursor.moveToNext()) {  
 val imagePath = cursor.getString(columnIndex)  
 Log.d("PhotoGallery", "Image found: $imagePath") // Debug log  
 imageList.add(imagePath)  
 }  
 **}** Log.d("PhotoGallery", "Total images fetched: ${imageList.size}") // Debug log  
 return imageList  
 }  
  
 // Display the images using Jetpack Compose's LazyVerticalGrid  
 private fun displayGallery(imageList: List<String>) {  
 *setContent* **{** *PhotoGalaryTheme* **{** *imageGallery*(imageList = imageList) // Use lowercase function name  
 **}  
 }** }  
}  
  
@OptIn(ExperimentalFoundationApi::class)  
@Composable  
fun imageGallery(imageList: List<String>) {  
 // State to manage the current image index  
 var selectedImageIndex by *remember* **{** *mutableStateOf*<Int?>(null) **}** // If an image is selected, show it in a full-screen dialog with Next/Back buttons  
 selectedImageIndex?.*let* **{** index **->** *AlertDialog*(  
 onDismissRequest = **{** selectedImageIndex = null **}**,  
 title = **{** *Text*(text = "Image ${index + 1} of ${imageList.size}") **}**,  
 text = **{** *AsyncImage*(  
 model = imageList[index],  
 contentDescription = "Image $index",  
 modifier = Modifier.*padding*(8.*dp*)  
 )  
 **}**,  
 confirmButton = **{** *Row* **{** // Back Button: Disabled on the first image  
 *Button*(  
 onClick = **{** selectedImageIndex = (selectedImageIndex ?: 0) - 1 **}**,  
 enabled = index > 0 // Disable if on the first image  
 ) **{** *Text*("Back")  
 **}** *Spacer*(modifier = Modifier.*padding*(8.*dp*)) // Space between buttons  
  
 // Next Button: Disabled on the last image  
 *Button*(  
 onClick = **{** selectedImageIndex = (selectedImageIndex ?: 0) + 1 **}**,  
 enabled = index < imageList.size - 1 // Disable if on the last image  
 ) **{** *Text*("Next")  
 **}  
 }  
 }**,  
 dismissButton = **{** *Button*(onClick = **{** selectedImageIndex = null **}**) **{** *Text*("Close")  
 **}  
 }** )  
 **}** // LazyVerticalGrid to display images  
 *LazyVerticalGrid*(  
 columns = GridCells.Fixed(3),  
 modifier = Modifier.*padding*(8.*dp*)  
 ) **{** items(imageList.size) **{** index **->** *AsyncImage*(  
 model = imageList[index],  
 contentDescription = "Image $index",  
 modifier = Modifier  
 .*padding*(4.*dp*)  
 .*clickable* **{** selectedImageIndex = index **}** // Open dialog with selected image  
 )  
 **}  
 }**}

3.2 AndroidManifest.xml

<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.example.photogalary">  
  
 <application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.PhotoGalary">  
  
 <!-- MainActivity with android:exported explicitly set to true -->  
 <activity  
 android:name=".MainActivity"  
 android:exported="true">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 </application>  
  
 <!-- Required permissions -->  
 <uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />  
 <uses-permission android:name="android.permission.READ\_MEDIA\_IMAGES" />  
</manifest>

1. Screenshots



