



# Project code – Image processing App:  
  
1. Mainactivity.kt:  
  
package com.example.rgb2grayscaleconverterapp  
  
import ImageProcessingViewModel  
import androidx.activity.compose.rememberLauncherForActivityResult  
import androidx.activity.compose.setContent  
import androidx.compose.runtime.livedata.observeAsState  
import androidx.compose.ui.platform.*LocalContext*import android.graphics.Bitmap  
import android.graphics.BitmapFactory  
import android.net.Uri  
import android.os.Bundle  
import androidx.activity.ComponentActivity  
import androidx.activity.result.contract.ActivityResultContract  
import androidx.activity.result.contract.ActivityResultContracts  
import androidx.compose.foundation.Image  
import androidx.compose.foundation.layout.\*  
import androidx.compose.material.icons.Icons  
import androidx.compose.material.icons.filled.*Menu*import androidx.compose.material.icons.filled.*Search*import androidx.compose.material3.\*  
import androidx.compose.runtime.\*  
import androidx.compose.ui.Alignment  
import androidx.compose.ui.Modifier  
import androidx.compose.ui.graphics.asImageBitmap  
import androidx.compose.ui.unit.dp  
import androidx.lifecycle.ViewModelProvider  
import androidx.compose.ui.platform.*LocalContext*import androidx.compose.ui.text.font.FontWeight  
import androidx.compose.ui.text.style.TextAlign  
import androidx.compose.ui.unit.sp  
  
class MainActivity : ComponentActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *setContent* **{** ImageProcessingApp()  
 **}** }  
}  
  
@OptIn(ExperimentalMaterial3Api::class)  
@Composable  
fun ImageProcessingApp() {  
 val viewModel: ImageProcessingViewModel = ViewModelProvider(*LocalContext*.current as ComponentActivity).get(ImageProcessingViewModel::class.*java*)  
  
 var inputImageUri by remember **{** *mutableStateOf*<Uri?>(null) **}** var errorMessage by remember **{** *mutableStateOf*<String?>(null) **}** val context = *LocalContext*.current  
 val galleryLauncher = rememberLauncherForActivityResult(ActivityResultContracts.GetContent()) **{** uri **->** uri?.*let* **{** inputImageUri = **it** val inputStream = context.*contentResolver*.openInputStream(**it**)  
 val bitmap = BitmapFactory.decodeStream(inputStream)  
 viewModel.setInputImage(bitmap)  
 viewModel.uploadImage(**it**) // Pass the Uri to upload image and convert to grayscale  
 **}  
 }** val inputImage by viewModel.inputImage.observeAsState()  
 val outputImage by viewModel.outputImage.observeAsState()  
 val error by viewModel.error.observeAsState()  
  
 Scaffold(  
 topBar = **{** TopAppBar(  
 title = **{** Text(  
 text = "Image Processing App",  
 style = MaterialTheme.typography.titleLarge.copy(  
 fontWeight = FontWeight.Bold,  
 fontSize = 18.*sp* // Adjusted font size  
 ),  
 color = MaterialTheme.colorScheme.onPrimary,  
 modifier = Modifier.*fillMaxWidth*(), // Center title horizontally  
 textAlign = TextAlign.Center // Center text  
 )  
 **}**,  
 colors = TopAppBarDefaults.topAppBarColors(  
 containerColor = MaterialTheme.colorScheme.primary,  
 titleContentColor = MaterialTheme.colorScheme.onPrimary  
 ),  
 modifier = Modifier.*fillMaxWidth*()  
 )  
 **}** ) **{** innerPadding **->** Column(  
 modifier = Modifier  
 .*fillMaxSize*()  
 .*padding*(innerPadding)  
 .*padding*(16.*dp*),  
 verticalArrangement = Arrangement.spacedBy(16.*dp*),  
 horizontalAlignment = Alignment.CenterHorizontally // Center content horizontally  
 ) **{** Button(  
 onClick = **{** galleryLauncher.launch("image/\*")  
 **}**,  
 modifier = Modifier.*align*(Alignment.CenterHorizontally) // Center button horizontally  
 ) **{** Text(text = "Select Image from Gallery")  
 **}** inputImage?.*let* **{** bitmap **->** Image(  
 bitmap = bitmap.*asImageBitmap*(),  
 contentDescription = "Input Image",  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*height*(300.*dp*) // Slightly larger size for input image  
 )  
 **}** outputImage?.*let* **{** bitmap **->** Image(  
 bitmap = bitmap.*asImageBitmap*(),  
 contentDescription = "Output Image",  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*height*(300.*dp*) // Slightly larger size for output image  
 )  
 **}** error?.*let* **{** Text(text = "Error: $**it**", color = MaterialTheme.colorScheme.error)  
 **}  
 }  
 }**}  
  
  
  
  
2. ImageProcessingViewModel

//import android.graphics.Bitmap  
// ImageProcessingViewModel.kt  
// ImageProcessingViewModel.kt  
  
import android.app.Application  
import android.graphics.Bitmap  
import android.graphics.BitmapFactory  
import android.net.Uri  
import androidx.lifecycle.AndroidViewModel  
import androidx.lifecycle.LiveData  
import androidx.lifecycle.MutableLiveData  
import androidx.lifecycle.*viewModelScope*import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
import okhttp3.\*  
import okhttp3.MediaType.Companion.toMediaTypeOrNull  
import java.io.ByteArrayOutputStream  
import java.io.IOException  
  
class ImageProcessingViewModel(application: Application) : AndroidViewModel(application) {  
 private val \_inputImage = MutableLiveData<Bitmap>()  
 val inputImage: LiveData<Bitmap> = \_inputImage  
  
 private val \_outputImage = MutableLiveData<Bitmap>()  
 val outputImage: LiveData<Bitmap> = \_outputImage  
  
 private val \_error = MutableLiveData<String>()  
 val error: LiveData<String> = \_error  
  
 private val client = OkHttpClient()  
  
 fun setInputImage(bitmap: Bitmap) {  
 \_inputImage.*value* = bitmap  
 }  
  
 // Updated function to accept only Uri  
 fun uploadImage(imageUri: Uri) {  
 *viewModelScope*.*launch*(Dispatchers.IO) **{** val context = getApplication<Application>().*applicationContext* val inputStream = context.*contentResolver*.openInputStream(imageUri)  
 val bitmap = BitmapFactory.decodeStream(inputStream)  
 val byteArrayOutputStream = ByteArrayOutputStream()  
 bitmap.compress(Bitmap.CompressFormat.*PNG*, 100, byteArrayOutputStream)  
 val requestBody = RequestBody.create("image/png".*toMediaTypeOrNull*(), byteArrayOutputStream.toByteArray())  
  
 val request = Request.Builder()  
 .url("https://3ow99rdgna.execute-api.ap-south-1.amazonaws.com/Dev")  
 .post(requestBody)  
 .build()  
  
 client.newCall(request).enqueue(object : Callback {  
 override fun onFailure(call: Call, e: IOException) {  
 \_error.postValue("Failed to upload image: ${e.message}")  
 }  
  
 override fun onResponse(call: Call, response: Response) {  
 if (!response.isSuccessful) {  
 \_error.postValue("Unexpected code $response")  
 return  
 }  
 response.body?.byteStream()?.*let* **{** stream **->** val outputBitmap = BitmapFactory.decodeStream(stream)  
 \_outputImage.postValue(outputBitmap)  
 **}** ?: \_error.postValue("Error: No response body")  
 }  
 })  
 **}** }  
}

3. RetrofitClient.kt   
package com.example.rgb2grayscaleconverterapp  
  
// RetrofitClient.kt  
import retrofit2.Retrofit  
import retrofit2.converter.scalars.ScalarsConverterFactory  
import okhttp3.OkHttpClient  
  
object RetrofitClient {  
 private const val BASE\_URL = "https://3ow99rdgna.execute-api.ap-south-1.amazonaws.com/Dev/" // Replace with your API URL  
  
 private val client = OkHttpClient.Builder().build()  
  
 val instance: Retrofit by *lazy* **{** Retrofit.Builder()  
 .baseUrl(BASE\_URL)  
 .client(client)  
 .addConverterFactory(ScalarsConverterFactory.create())  
 .build()  
 **}**}

4. Build.gradle app   
   
plugins **{** alias(*libs*.*plugins*.*android*.*application*)  
 alias(*libs*.*plugins*.*jetbrains*.*kotlin*.*android*)  
**}***android* **{** namespace = "com.example.rgb2grayscaleconverterapp"  
 compileSdk = 34  
  
 defaultConfig **{** applicationId = "com.example.rgb2grayscaleconverterapp"  
 minSdk = 24  
 targetSdk = 34  
 versionCode = 1  
 versionName = "1.0"  
  
 testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"  
 vectorDrawables **{** useSupportLibrary = true  
 **}  
 }** buildTypes **{** *release* **{** isMinifyEnabled = false  
 proguardFiles(  
 getDefaultProguardFile("proguard-android-optimize.txt"),  
 "proguard-rules.pro"  
 )  
 **}  
 }** compileOptions **{** sourceCompatibility = JavaVersion.*VERSION\_1\_8* targetCompatibility = JavaVersion.*VERSION\_1\_8* **}** *kotlinOptions* **{** jvmTarget = "1.8"  
 **}** buildFeatures **{** compose = true  
 **}** composeOptions **{** kotlinCompilerExtensionVersion = "1.5.1"  
 **}** packaging **{** resources **{** excludes += "/META-INF/{AL2.0,LGPL2.1}"  
 **}  
 }  
}***dependencies* **{** *implementation*(*libs*.*androidx*.*core*.*ktx*)  
 *implementation*(*libs*.*androidx*.*lifecycle*.*runtime*.*ktx*)  
 *implementation*(*libs*.*androidx*.*activity*.*compose*)  
 *implementation*(platform(*libs*.*androidx*.*compose*.*bom*))  
 *implementation*(*libs*.*androidx*.*ui*)  
 *implementation*(*libs*.*androidx*.*ui*.*graphics*)  
 *implementation*(*libs*.*androidx*.*ui*.*tooling*.*preview*)  
 *implementation*(*libs*.*androidx*.*material3*)  
 *implementation*(*libs*.*androidx*.*appcompat*)  
 *testImplementation*(*libs*.*junit*)  
 *androidTestImplementation*(*libs*.*androidx*.*junit*)  
 *androidTestImplementation*(*libs*.*androidx*.*espresso*.*core*)  
 *androidTestImplementation*(platform(*libs*.*androidx*.*compose*.*bom*))  
 *androidTestImplementation*(*libs*.*androidx*.*ui*.*test*.*junit4*)  
 *debugImplementation*(*libs*.*androidx*.*ui*.*tooling*)  
 *debugImplementation*(*libs*.*androidx*.*ui*.*test*.*manifest*)  
  
 *implementation* ("io.coil-kt:coil-compose:2.1.0") // For image loading  
 *implementation* ("com.squareup.okhttp3:okhttp:4.9.3") // For HTTP calls  
 *implementation* ("androidx.activity:activity-compose:1.4.0")  
 *implementation* ("androidx.lifecycle:lifecycle-viewmodel-compose:2.6.1")  
  
  
  
 *implementation*(*libs*.*ui*)  
 *implementation*(*libs*.*ui*.*tooling*.*preview*)  
 *implementation*(*libs*.*androidx*.*activity*.*compose*.*v180*)  
  
// Image loading and manipulation  
 *implementation*("androidx.compose.foundation:foundation:1.5.0")  
 *implementation*("androidx.compose.foundation:foundation-layout:1.5.0")  
  
  
 // Retrofit  
 *implementation* ("com.squareup.retrofit2:retrofit:2.9.0")  
 *implementation* ("com.squareup.retrofit2:converter-gson:2.9.0")  
 *implementation* ("com.squareup.okhttp3:logging-interceptor:4.9.1")  
  
// Coroutines  
 *implementation* ("org.jetbrains.kotlinx:kotlinx-coroutines-android:1.5.1")  
  
// Coroutines for background tasks  
 *implementation*("org.jetbrains.kotlinx:kotlinx-coroutines-core:1.7.3")  
 *implementation*("org.jetbrains.kotlinx:kotlinx-coroutines-android:1.7.3")  
 *implementation* ("androidx.lifecycle:lifecycle-runtime-ktx:2.6.1")  
// Coil for image loading  
 *implementation*("io.coil-kt:coil-compose:2.4.0")  
  
  
  
 *implementation* (*libs*.*androidx*.*ui*.*text*.*google*.*fonts*) // Replace with the latest version  
  
  
 *implementation*(*libs*.*converter*.*scalars*)  
 *implementation* (*libs*.*androidx*.*ui*.*v140*)  
 *implementation* (*libs*.*androidx*.*material*)  
 *implementation* (*libs*.*androidx*.*ui*.*tooling*.*preview*.*v140*)  
 *implementation* (*libs*.*androidx*.*activity*.*compose*.*v171*)  
 *implementation* (*libs*.*lifecycle*.*runtime*.*ktx*.*v261*)  
 *implementation* (*libs*.*androidx*.*runtime*.*livedata*)  
  
  
**}**

5. android manifest:  
  
<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools">  
  
 <uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE"  
 android:maxSdkVersion="32" />  
 <uses-permission android:name="android.permission.INTERNET" />  
  
  
  
 <application  
 android:allowBackup="true"  
 android:dataExtractionRules="@xml/data\_extraction\_rules"  
 android:fullBackupContent="@xml/backup\_rules"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.AppCompat.DayNight"  
 tools:targetApi="31">  
  
 <activity  
 android:name=".MainActivity"  
 android:exported="true"  
 android:label="@string/app\_name"  
 android:theme="@style/Theme.AppCompat.DayNight">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
  
 </application>  
  
</manifest>