EX 15

ANDROID MANIFEST

```xml
<?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.CAMERA"/>
    <uses-feature android:name="android.hardware.camera"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>


    <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.Ex15"
            tools:targetApi="31">
        <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>

</manifest>
```

ACTIVITY MAIN

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
                xmlns:tools="http://schemas.android.com/tools"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                tools:context=".MainActivity">

    <ImageView
            android:id="@+id/imageView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"

            android:layout_marginTop="16dp"
            android:adjustViewBounds="true"
            android:scaleType="fitCenter"
            android:src="@android:drawable/ic_menu_camera"
    />

    <Button
            android:id="@+id/btnTakePicture"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Take Picture"
            android:layout_below="@id/imageView"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="20dp"/>


    <androidx.camera.view.PreviewView
            android:id="@+id/previewView"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:visibility="invisible"/>

</RelativeLayout>
```

MAIN ACTIVITY

```kotlin
package com.example.ex15

import android.Manifest
import android.content.pm.PackageManager
import android.os.Bundle
import android.util.Log
import android.widget.Button
import android.widget.ImageView
import androidx.activity.result.contract.ActivityResultContracts
import androidx.appcompat.app.AppCompatActivity
import androidx.camera.core.*
import androidx.camera.lifecycle.ProcessCameraProvider
import androidx.camera.view.PreviewView
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat
import androidx.core.net.toUri
import java.io.File
import java.text.SimpleDateFormat
import java.util.*
import java.util.concurrent.ExecutorService
import java.util.concurrent.Executors

class MainActivity : AppCompatActivity() {

    private lateinit var btnTakePicture: Button
    private lateinit var imageView: ImageView
    private lateinit var outputDirectory: File
    private lateinit var cameraExecutor: ExecutorService
    private lateinit var previewView: PreviewView

    private var imageCapture: ImageCapture? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        btnTakePicture = findViewById(R.id.btnTakePicture)
        imageView = findViewById(R.id.imageView)
        previewView=findViewById(R.id.previewView)

        if (allPermissionsGranted()) {
            startCamera()
        } else {
            ActivityCompat.requestPermissions(
                this, REQUIRED_PERMISSIONS, REQUEST_CODE_PERMISSIONS)
        }

        btnTakePicture.setOnClickListener { takePhoto() }

        outputDirectory = getOutputDirectory()
        cameraExecutor = Executors.newSingleThreadExecutor()
    }

    private fun takePhoto() {
        val imageCapture = imageCapture ?: return

        val photoFile = File(
            outputDirectory,
            SimpleDateFormat(FILENAME_FORMAT, Locale.US)
                .format(System.currentTimeMillis()) + ".jpg")

        val outputOptions =
ImageCapture.OutputFileOptions.Builder(photoFile).build()

        imageCapture.takePicture(
            outputOptions, ContextCompat.getMainExecutor(this), object :
ImageCapture.OnImageSavedCallback {
                override fun onError(exception: ImageCaptureException) {
                    Log.e(TAG, "Photo capture failed: ${exception.message}",
exception)
                }

                override fun onImageSaved(outputFileResults:
ImageCapture.OutputFileResults) {
                    val savedUri = outputFileResults.savedUri ?: photoFile.toUri()
                    Log.d(TAG, "Photo capture succeeded: $savedUri")
                    imageView.setImageURI(savedUri)
                }
            })
    }

    private fun startCamera() {
        val cameraProviderFuture = ProcessCameraProvider.getInstance(this)

        cameraProviderFuture.addListener({
            val cameraProvider: ProcessCameraProvider = cameraProviderFuture.get()

            val preview = Preview.Builder()
                .build()
                .also {
                    it.setSurfaceProvider(previewView.surfaceProvider)
                }

            imageCapture = ImageCapture.Builder()
                .build()

            val cameraSelector = CameraSelector.DEFAULT_BACK_CAMERA

            try {
                cameraProvider.unbindAll()
                cameraProvider.bindToLifecycle(
                    this, cameraSelector, preview, imageCapture)
            } catch (exc: Exception) {
                Log.e(TAG, "Use case binding failed", exc)
            }

        }, ContextCompat.getMainExecutor(this))
    }

    private fun allPermissionsGranted() = REQUIRED_PERMISSIONS.all {
        ContextCompat.checkSelfPermission(
            baseContext, it) == PackageManager.PERMISSION_GRANTED
    }

    private fun getOutputDirectory(): File {
```

```kotlin
        val mediaDir = externalMediaDirs.firstOrNull()?.let {
            File(it, resources.getString(R.string.app_name)).apply { mkdirs() } }
        return if (mediaDir != null && mediaDir.exists())
            mediaDir else filesDir
    }

    override fun onDestroy() {
        super.onDestroy()
        cameraExecutor.shutdown()
    }

    override fun onRequestPermissionsResult(requestCode: Int, permissions:
Array<String>, grantResults: IntArray) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults)
        if (requestCode == REQUEST_CODE_PERMISSIONS) {
            if (allPermissionsGranted()) {
                startCamera()
            } else {
                Log.d(TAG, "Permissions not granted by the user.")
            }
        }
    }

    companion object {
        private const val TAG = "CameraXBasic"
        private const val FILENAME_FORMAT = "yyyy-MM-dd-HH-mm-ss-SSS"
        private const val REQUEST_CODE_PERMISSIONS = 10
        private val REQUIRED_PERMISSIONS = arrayOf(Manifest.permission.CAMERA)
    }
}
```