Experiment 10: Using the Camera

1. Objective

Develop an application that can take pictures using the device's camera and display them within the app. This introduces camera API usage and file storage.

2. Steps to Complete the Experiment

1. Update Android Manifest:

   Add the necessary permissions to access the camera and write to external storage in your AndroidManifest.xml file:

```
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

2. Request Runtime Permissions:

   Since accessing the camera and writing to storage are considered dangerous permissions, request them at runtime in your activity, especially if targeting Android 6.0 (API level 23) or higher.

3. Design the UI:

   Create a layout with a Button to open the camera and an ImageView to display the captured image.

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

    <Button
        android:id="@+id/buttonCapture"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Capture Image"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="32dp"/>

    <ImageView
        android:id="@+id/imageViewCaptured"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/buttonCapture"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="32dp"

android:contentDescription="@string/captured_image_desc"/>

</RelativeLayout>
```

4. Capture Image:

Use an Intent to capture an image with the camera. In the OnClickListener for the camera button, create an intent with MediaStore.ACTION_IMAGE_CAPTURE.

Check if there's a camera activity available to handle the intent using resolveActivity(getPackageManager()) before starting the intent.

5. Save the Captured Image:

   Optionally, to save the image, specify a file URI where the photo should be saved and pass it to the camera intent. This involves creating a file in the external storage directory and using FileProvider to get a content URI for passing it securely.

6. Handle the Activity Result:

   Override onActivityResult to receive the result from the camera activity. If the result is RESULT_OK, retrieve and display the captured image in the ImageView.

```
package com.yourpackage.name; // Replace with your actual
package name

import androidx.activity.result.ActivityResultLauncher;
import
androidx.activity.result.contract.ActivityResultContracts;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;
import androidx.core.content.FileProvider;

import android.content.Intent;
import android.graphics.Bitmap;
import android.net.Uri;
import android.os.Bundle;

import android.os.Environment;

import android.provider.MediaStore;

import android.view.View;

import android.widget.Button;

import android.widget.ImageView;


import java.io.File;

import java.io.IOException;

import java.text.SimpleDateFormat;

import java.util.Date;


public class MainActivity extends AppCompatActivity {


    private ImageView imageViewCaptured;

    private String currentPhotoPath;


    private        final        ActivityResultLauncher<Intent>
takePictureActivityResultLauncher = registerForActivityResult(
            new
ActivityResultContracts.StartActivityForResult(),
            result -> {
                if (result.getResultCode() == RESULT_OK) {
                    setPic();
                }
            });
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button                 buttonCapture                  =
findViewById(R.id.buttonCapture);
        imageViewCaptured                                     =
findViewById(R.id.imageViewCaptured);

        buttonCapture.setOnClickListener(view                ->
dispatchTakePictureIntent());
    }

    private File createImageFile() throws IOException {
        // Create an image file name
        String          timeStamp           =           new
SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
        String imageFileName = "JPEG_" + timeStamp + "_";
        File                   storageDir                     =
getExternalFilesDir(Environment.DIRECTORY_PICTURES);
        File image = File.createTempFile(
                imageFileName,  /* prefix */
                ".jpg",         /* suffix */
                storageDir      /* directory */
        );

        // Save a file: path for use with ACTION_VIEW intents
        currentPhotoPath = image.getAbsolutePath();
        return image;
    }

    private void dispatchTakePictureIntent() {
        Intent         takePictureIntent         =         new
Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        // Ensure that there's a camera activity to handle the
intent
        if
(takePictureIntent.resolveActivity(getPackageManager())      !=
null) {
            // Create the File where the photo should go
            File photoFile = null;
            try {
                photoFile = createImageFile();
            } catch (IOException ex) {
                // Error occurred while creating the File
            }
            //  Continue  only  if  the  File  was  successfully
created
            if (photoFile != null) {
                Uri                 photoURI                  =
FileProvider.getUriForFile(this,
                        "com.yourpackage.name.fileprovider",
// Update with your package name and applicationId +
.fileprovider
                        photoFile);

takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, photoURI);

takePictureActivityResultLauncher.launch(takePictureIntent);
            }
        }
    }

    private void setPic() {
        // Get the dimensions of the View
        int targetW = imageViewCaptured.getWidth();
        int targetH = imageViewCaptured.getHeight();

        // Get the dimensions of the bitmap
        BitmapFactory.Options       bmOptions       =       new
BitmapFactory.Options();
        bmOptions.inJustDecodeBounds = true;
        BitmapFactory.decodeFile(currentPhotoPath, bmOptions);
        int photoW = bmOptions.outWidth;
        int photoH = bmOptions.outHeight;
        // Determine how much to scale down the image
        int scaleFactor = Math.max(1, Math.min(photoW / targetW, photoH / targetH));

        // Decode the image file into a Bitmap sized to fill the View
        bmOptions.inJustDecodeBounds = false;
        bmOptions.inSampleSize = scaleFactor;
        bmOptions.inPurgeable = true;

        Bitmap bitmap = BitmapFactory.decodeFile(currentPhotoPath, bmOptions);
        imageViewCaptured.setImageBitmap(bitmap);
    }
}
```

7. Testing:

   Test the application on a real device (camera hardware is required) to ensure the camera opens, captures an image, and the image is displayed within the app as expected.

3. Explanation

Button (buttonCapture): When clicked, this button will initiate the process to open the camera and capture an image. It's positioned at the top center of the screen.

ImageView (imageViewCaptured): This view will display the image captured by the camera. It's placed below the button and centered horizontally in the parent layout.

Permissions and FileProvider: Ensure you have the correct permissions in your AndroidManifest.xml and have set up a FileProvider in the manifest to share the photo file securely with the camera app.

Taking a Picture: The dispatchTakePictureIntent method starts an intent to take a picture. It checks if there's a camera app that can handle the intent and creates a file for the picture. The URI of this file is passed to the camera app.

Saving the Image: The captured image is saved to the file created by createImageFile. The file's location is stored in currentPhotoPath.

Displaying the Image: The setPic method sets the captured image in the ImageView. It scales the image to fit the ImageView efficiently to conserve memory.

Remember to replace "com.yourpackage.name.fileprovider" with your actual package name and the .fileprovider authority you've defined in your AndroidManifest.xml. This authority should match the one specified in your <provider> tag within the manifest.