1. Add permission:

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

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>

1. Edit Layout File
2. Add Texture View + button + onClick
3. Add new java class CameraCapture.java
4. Edit manifest file: <activity android:name=".CameraCapture"></activity>
5. Add CheckPermission in MainActivity

if(ActivityCompat.*checkSelfPermission*(this, Manifest.permission.*CAMERA*)  
 != PackageManager.*PERMISSION\_GRANTED*){  
 ActivityCompat.*requestPermissions*(this, new String[]{  
 Manifest.permission.*CAMERA*,  
 Manifest.permission.*WRITE\_EXTERNAL\_STORAGE* }, *REQUEST\_CAMERA\_PERMISSION*);  
}

@Override  
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
 if (requestCode == *REQUEST\_CAMERA\_PERMISSION*){  
 if (grantResults[0] != PackageManager.*PERMISSION\_GRANTED*) {  
 Toast.*makeText*(this, "You cant use this application without permission", Toast.*LENGTH\_SHORT*).show();  
 finish();  
 }  
 }  
}

1. Add takePic Method on MainActivity
2. Add scanQR Method on MainActivity
3. Edit class extends AppCompactActivity
4. @Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_camera);

}

1. Initialise views

private TextureView cameraCapture;  
 private Button captureBtn;

//OnCreate

cameraCapture = (TextureView) findViewById(R.id.*textureView*);  
assert cameraCapture != null; //check expression

captureBtn = (Button)findViewById(R.id.*capture\_btn*);

1. Take picture Method initialize + set onclick for button

captureBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 takePicture ();  
 }  
});

private void takePicture(){  
   
}

1. Initialise Variables needed for the app

private String cameraId; //id of the camera  
private CameraDevice cameraDevice;  
private CameraCaptureSession cameraCaptureSession;  
private CaptureRequest.Builder captureRequestBuilder;  
private Size imageDimension;  
private ImageReader imageReader;  
  
//Save to file  
private File file;  
private boolean mFlashSupported;  
private Handler mBackgroundHandler;  
private HandlerThread mBackgroundThread;

1. TakePicture method

private void takePicture(){  
 if(cameraDevice == null){  
 return; //if there is no camera  
 }  
 CameraManager manager = (CameraManager)getSystemService(*CAMERA\_SERVICE*);  
 try{  
 CameraCharacteristics characteristics = manager.getCameraCharacteristics(cameraDevice.getId());  
 Size[] jpegSizes = null;  
 if(characteristics != null){  
 jpegSizes = characteristics.get(CameraCharacteristics.*SCALER\_STREAM\_CONFIGURATION\_MAP*)  
 .getOutputSizes(ImageFormat.*JPEG*);  
 }  
  
 //capture with custom size  
 int width = 640;  
 int height = 480;  
  
 if(jpegSizes != null && jpegSizes.length > 0){  
 width = jpegSizes[0].getWidth();  
 height = jpegSizes[0].getHeight();  
 }  
  
 final ImageReader reader = ImageReader.*newInstance*(width,height,ImageFormat.*JPEG*,1);  
 List<Surface> outputSurface = new ArrayList<>(2);  
 outputSurface.add(reader.getSurface());  
 outputSurface.add(new Surface(cameraCapture.getSurfaceTexture()));  
  
 final CaptureRequest.Builder captureBuilder = cameraDevice.createCaptureRequest(CameraDevice.*TEMPLATE\_STILL\_CAPTURE*);  
 captureBuilder.addTarget(reader.getSurface());  
 captureBuilder.set(CaptureRequest.*CONTROL\_MODE*, CameraMetadata.*CONTROL\_MODE\_AUTO*);  
  
 file = new File(Environment.*getExternalStorageDirectory*() + "/" +  
 UUID.*randomUUID*().toString() + ".jpg");  
  
 ImageReader.OnImageAvailableListener readerListener = new ImageReader.OnImageAvailableListener() {  
 @Override  
 public void onImageAvailable(ImageReader imageReader) {  
 Image image = null;  
 try{  
 image = reader.acquireLatestImage();  
 ByteBuffer buffer = image.getPlanes()[0].getBuffer();  
 byte[] bytes = new byte[buffer.capacity()];  
 buffer.get(bytes);  
 save(bytes);  
 } catch (FileNotFoundException e1){  
 e1.printStackTrace();  
 } catch (IOException e2){  
 e2.printStackTrace();  
 } finally {  
 if(image != null){  
 image.close();  
 }  
 }  
 }  
  
 private void save(byte[] bytes) throws IOException{  
 OutputStream outputStream = null;  
 try{  
 outputStream = new FileOutputStream(file);  
 outputStream.write(bytes);  
 } finally {  
 if(outputStream != null){  
 outputStream.close();  
 }  
 }  
 }  
 };  
  
 reader.setOnImageAvailableListener(readerListener, mBackgroundHandler);  
 final CameraCaptureSession.CaptureCallback captureListener = new CameraCaptureSession.CaptureCallback() {  
 @Override  
 public void onCaptureCompleted(@NonNull CameraCaptureSession session, @NonNull CaptureRequest request, @NonNull TotalCaptureResult result) {  
 super.onCaptureCompleted(session, request, result);  
 Toast.*makeText*(getApplicationContext(), "Saved " + file, Toast.*LENGTH\_SHORT*).show();  
 createCameraPreview();  
 }  
 };  
  
 cameraDevice.createCaptureSession(outputSurface, new CameraCaptureSession.StateCallback() {  
 @Override  
 public void onConfigured(@NonNull CameraCaptureSession cameraCaptureSession) {  
 try{  
 cameraCaptureSession.capture(captureBuilder.build(), captureListener, mBackgroundHandler);  
 } catch (CameraAccessException e){  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public void onConfigureFailed(@NonNull CameraCaptureSession cameraCaptureSession) {  
  
 }  
 }, mBackgroundHandler);  
  
 } catch (CameraAccessException e){  
 e.printStackTrace();  
 }  
}

1. Save method

private void save(byte[] bytes) throws IOException{  
 OutputStream outputStream = null;  
 try{  
 outputStream = new FileOutputStream(file);  
 outputStream.write(bytes);  
 } finally {  
 if(outputStream != null){  
 outputStream.close();  
 }  
 }  
}

1. CreateCameraPreview Method

private void createCameraPreview(){  
 try{  
 SurfaceTexture texture = cameraCapture.getSurfaceTexture();  
 assert texture != null;  
 texture.setDefaultBufferSize(imageDimension.getWidth(), imageDimension.getHeight());  
 Surface surface = new Surface(texture);  
 captureRequestBuilder = cameraDevice.createCaptureRequest(CameraDevice.*TEMPLATE\_PREVIEW*);  
 captureRequestBuilder.addTarget(surface);  
 cameraDevice.createCaptureSession(Arrays.*asList*(surface), new CameraCaptureSession.StateCallback() {  
 @Override  
 public void onConfigured(@NonNull CameraCaptureSession cameraCaptureSession1) {  
 if(cameraDevice == null){  
 return;  
 }  
 cameraCaptureSession = cameraCaptureSession1;  
 updatePreview();  
 }  
  
 @Override  
 public void onConfigureFailed(@NonNull CameraCaptureSession cameraCaptureSession) {  
 Toast.*makeText*(getApplicationContext(), "Changed", Toast.*LENGTH\_SHORT*).show();  
 }  
 }, null);  
  
 } catch (CameraAccessException e){  
 e.printStackTrace();  
 }  
}

1. UpdatePreview Method
2. private void updatePreview(){  
    if(cameraDevice == null){  
    Toast.*makeText*(getApplicationContext(), "Error", Toast.*LENGTH\_SHORT*).show();  
    }  
    captureRequestBuilder.set(CaptureRequest.*CONTROL\_MODE*, CaptureRequest.*CONTROL\_MODE\_AUTO*);  
    try{  
    cameraCaptureSession.setRepeatingRequest(captureRequestBuilder.build(),  
    null, mBackgroundHandler);  
    } catch(CameraAccessException e){  
    e.printStackTrace();  
    }  
   }
3. OpenCamera Method

private void openCamera(){  
 CameraManager manager = (CameraManager)getSystemService(*CAMERA\_SERVICE*);  
 try{  
 cameraId = manager.getCameraIdList()[0];  
 CameraCharacteristics characteristics = manager.getCameraCharacteristics(cameraId);  
 StreamConfigurationMap map = characteristics.get(CameraCharacteristics.*SCALER\_STREAM\_CONFIGURATION\_MAP*);  
 assert map != null;  
 imageDimension = map.getOutputSizes(SurfaceTexture.class)[0];  
  
 manager.openCamera(cameraId, stateCallBack, null);  
  
 } catch (CameraAccessException e) {  
 e.printStackTrace();  
 } catch (SecurityException e1){  
 e1.printStackTrace();  
 }  
}

1. StateCallBack Method + Variable

CameraDevice.StateCallback stateCallBack = new CameraDevice.StateCallback() {  
 @Override  
 public void onOpened(@NonNull CameraDevice cameraDevice1) {  
 cameraDevice = cameraDevice1;  
 createCameraPreview();  
 }  
  
 @Override  
 public void onDisconnected(@NonNull CameraDevice cameraDevice) {  
 cameraDevice.close();  
 }  
  
 @Override  
 public void onError(@NonNull CameraDevice cameraDevice, int i) {  
 cameraDevice.close();  
 cameraDevice = null;  
 }  
};

1. Add onResume on CameraCapture.java

@Override  
protected void onResume() {  
 super.onResume();  
 startBackgroundThread();  
 if(cameraCapture.isAvailable()){  
 openCamera();  
 }else{  
 cameraCapture.setSurfaceTextureListener(textureListener);  
 }  
}

1. Add textureListener
2. TextureView.SurfaceTextureListener textureListener = new TextureView.SurfaceTextureListener() {  
    @Override  
    public void onSurfaceTextureAvailable(SurfaceTexture surfaceTexture, int i, int i1) {  
    openCamera();  
    }  
     
    @Override  
    public void onSurfaceTextureSizeChanged(SurfaceTexture surfaceTexture, int i, int i1) {  
     
    }  
     
    @Override  
    public boolean onSurfaceTextureDestroyed(SurfaceTexture surfaceTexture) {  
    return false;  
    }  
     
    @Override  
    public void onSurfaceTextureUpdated(SurfaceTexture surfaceTexture) {  
     
    }  
   };
3. startBackgroundThread Method

private void startBackgroundThread(){  
 mBackgroundThread = new HandlerThread("Camera Background");  
 mBackgroundThread.start();  
 mBackgroundHandler = new Handler(mBackgroundThread.getLooper());  
}

1. onPause Method

@Override  
protected void onPause() {  
 stopBackgroundThread();  
 super.onPause();  
}

1. stopBackgroundThread Method

private void stopBackgroundThread() {  
 mBackgroundThread.quitSafely();  
 try{  
 mBackgroundThread.join();  
 mBackgroundThread = null;  
 mBackgroundHandler = null;  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
}

=================QR================

1. Add dependency to gradle

compile 'me.dm7.barcodescanner:zxing:1.8.4'

1. Add Permission to Manifest

<uses-permission android:name="android.permission.VIBRATE"/>

1. Edit the permission method we wrote before, add vibrate in

Manifest.permission.*VIBRATE*

1. Implement resultHandler in MainActivity

public class MainActivity extends AppCompatActivity implements ZXingScannerView.ResultHandler

1. Add some ZXingScanner variables

private ZXingScannerView zXingScannerView;   
private ZXingScannerView.ResultHandler handler;

1. Edit sytle, add a new style for alertDialog

<style name="myDialog" parent="Theme.AppCompat.Dialog">  
 <item name="colorPrimary">@color/colorPrimary</item>  
 <item name="colorPrimaryDark">@color/colorPrimaryDark</item>  
 <item name="colorAccent">@color/colorAccent</item>  
</style>

1. ScanQr method

public void scanQR(View view){  
 zXingScannerView = new ZXingScannerView(getApplicationContext());  
 setContentView(zXingScannerView);  
 zXingScannerView.setResultHandler(this);  
 zXingScannerView.startCamera();  
}

1. Add override HandleResult method to MainActivity

@Override  
public void handleResult(Result result) {  
 handler = this;  
 if(result != null){  
 //show a dialog  
 AlertDialog.Builder alertShow = new AlertDialog.Builder(MainActivity.this,  
 R.style.*myDialog*);  
 alertShow.setTitle("Scanned Result");  
 alertShow.setMessage(result.getText());  
 alertShow.setCancelable(false);  
 alertShow.setPositiveButton("OK", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialogInterface, int i) {  
 zXingScannerView.resumeCameraPreview(handler);  
 dialogInterface.dismiss();  
 }  
 });  
 AlertDialog dialog = alertShow.create();  
 dialog.show();  
 }else{  
 zXingScannerView.resumeCameraPreview(this);  
 }  
}

1. Add onResume method

@Override  
protected void onResume() {  
 super.onResume();  
 if(zXingScannerView != null){  
 zXingScannerView.startCamera();  
 }  
}

1. Add onBackPressed Method

@Override  
public void onBackPressed() {  
 Intent intent = new Intent(MainActivity.this, MainActivity.class);  
 finish();  
 overridePendingTransition(0,0);  
 intent.addFlags(Intent.*FLAG\_ACTIVITY\_NO\_ANIMATION*);  
 startActivity(intent);  
 super.onBackPressed();  
}