1 Introduction, QR Code Scanner App

1.1 Using ZXing (Zebra Crossing) Scanner Library

Some facts about Quick Response(QR) code

- a two dimensional barcode(matrix codes) that allows content to be decoded at a high speed;
- invented in 1994 by japanese company Denso-Wave.
- license-free,
- consists of black modules (square dots) arranged in a square grid on a white background, readable by an imaging device (such as a camera, scanner, etc.)
- dimensions, 21×21 pixel size is version 1, 25×25 is version 2, and so on. The 177×177 size is version 40.

1.2 Generated QR block

```
%%HTML
<video width="320" height="240" controls>
  <source src="data/QR.mp4" type="video/mp4">
</video>
```

115

```
  0x0  0x1  0x2  0x3  0x4  0x5  0x6  0x7  0x8  0x9  0xa
0xb  0xc  0xd  0xe  0xf




       !    "    #    $    %    &    '    (    )    *
+    ,    -    .    /
  0    1    2    3    4    5    6    7    8    9    :
;    <    =    >    ?
  @    A    B    C    D    E    F    G    H    I    J
K    L    M    N    O
  P    Q    R    S    T    U    V    W    X    Y    Z
[    \    ]    ^    _
  `    a    b    c    d    e    f    g    h    i    j
k    l    m    n    o
  p    q    r    s    t    u    v    w    x    y    z
{    |    }    ~





       ¡    ¢    £    ¤    ¥    ¦    §    ¨    ©    ª
«    ¬        ®    ¯
  °    ±    ²    ³    ´    µ    ¶    ·    ¸    ¹    º
»    ¼    ½    ¾    ¿
  À    Á    Â    Ã    Ä    Å    Æ    Ç    È    É    Ê
Ë    Ì    Í    Î    Ï
  Ð    Ñ    Ò    Ó    Ô    Õ    Ö    ×    Ø    Ù    Ú
Û    Ü    Ý    Þ    ß
  à    á    â    ã    ä    å    æ    ç    è    é    ê
ë    ì    í    î    ï
  ð    ñ    ò    ó    ô    õ    ö    ÷    ø    ù    ú
û    ü    ý    þ    ÿ
```

1.3 Subject

Using **ZXing Library** to implement the process of scanning the image of the QR Code on the click of Button and **QRGen** to create our QRCode.

1.4 Steps

1. implement the **ZXingScannerView.ResultHandler** class to handle the scanned result in the **MainActivity.java** file
2. initialize the **ZXingScannerView** in **MainActivity.java** file. It will start your camera and scan the image of QR Code to decode the QR code.
3. After complete scanning of QR Code, result is handled by **handleResult()** method.

1.5 Demo Project

- Create a new Project with Empty Activity,
  ```
  Project: QrCodeApp
  company domain: com.kotlin.qrcode
  ```
  in Android Studio, Also enalbed with kotlin support; goto ```[Start a New Project]```, .

1. AndroifManifest.xml. In the first demo part, there are two Intents created, MainActivity.kt and BarcodeScanningActivity.kt:

- one is main UI, MainActivity.kt, including
  - `TextView` , describes what the purpose of the App is,
  - `View` , a narrow seperation line,
  - `Button` , function to jump to the other Inden to do QR code scanning
- the QrCode Scanning Intent, BarcodeScanningActivity.kt.
  💻, And one Indent for exercise, BarcodeGenActivity.kt`, at which to generate the QR code online.

To scan QR code, need require Camera, add permisions as follows:

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res
/android"
    package="qrcodescanner.com.qrcodescannerproject">
    <uses-permission android:name="android.permission.CAMER
A" />
    <uses-permission android:name="android.permission.WRITE
_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.INTER
NET" />

    <uses-feature android:name="android.hardware.Camera" />
    <uses-feature android:name= "android.hardware.camera.au
tofocus " />

    <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/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action
.MAIN" />

                <category android:name="android.intent.cate
gory.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".BarcodeScanningActivity" /
>
    </application>

</manifest>
```

2. Adding Library in the dependencies:

To use QRGen, open **build.gradle (Project xxx)**, here xxx represents what app we define in the project, here **QRCodeApp**, and add the following link site:

```
...
allprojects {
    repositories {
        jcenter()
        maven { url "https://jitpack.io" }
    }
}
```

To compile the **ZXing** library, we need to add the library in the app's dependencies. Open the app's `build.gradle (Module app)` file. Add the library in the dependencies. It will compile the library at run time.

```
...
dependencies {
    ...
    implementation 'me.dm7.barcodescanner:zbar:1.9.8'
}
```

Click [sync] to let Android studio automatically download the **zxing** and **QRGen** libraries we added.

Note. Keyword had been changed from `compile` to `implementation` .

3. Layout

change the Layout style to RelativeLayout and add one button and one one TextView:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="@string/barcode_demo"
        android:textSize="20sp" />

    <View
        android:layout_width="match_parent"
        android:layout_height="5dp"
        android:layout_marginBottom="16dp"
        android:layout_marginTop="16dp"
        android:background="@android:color/darker_gray" />

    <Button
        android:id="@+id/scanBarcodeButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/scan_barcode" />
</LinearLayout>
```

In `res/values/strings.xml` , define the id's as follows (folows the help of Android Studio):

```
<resources>
    <string name="app_name">QRCodeApp</string>
    <string name="barcode_demo">Barcode Scanning ...</strin
g>
    <string name="scan_barcode">Scanning Barcode</string>
</resources>
```

Exercise: 💻, To Complete the whole app, add another button to jump to BarCodeActivity.kt

4. **MainActivity.java**

Processing Flow

```
     (1)                       (2)
Click button    --->   activate Camera   --->  focus the QR or BarCode
                                                |
                                                |  (3)
                                                |
                                                ▼
             response      < ------      return the Info's
```

1. Waiting for button clicked

```
button.setOnClickListener(new View.OnClickListener() {
              // clicked
           });
```

2. start Scanning

```
             public void onClick(View v) {
                IntentIntegrator scanIntegrator = new In
tentIntegrator(mainactivity);
                scanIntegrator.initiateScan();
             }
```

3. return info's

```
 public void onActivityResult(int requestCode, int resul
tCode, Intent intent){
      IntentResult scanningResult = IntentIntegrator.par
seActivityResult(requestCode, resultCode, intent);

      if(scanningResult!=null){
          // if something
      }else{
          // if nothing
      }
  }
```

1.6 Main part of Code, MainActivity.java

1. import necessary packages and change the Activity,
2. auto-detetect whethe `camera permission` had been enabled; allow if not
3. set up the button UI and listen for jumping to another Intent,

```
package qrcode.kotlin.com.qrcodeapp

import android.support.v7.app.AppCompatActivity
import android.support.v4.app.ActivityCompat
import android.os.Bundle
import android.content.Intent
import android.content.pm.PackageManager
import android.widget.Button

class MainActivity : AppCompatActivity() {

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

        if (ActivityCompat.checkSelfPermission(this, androi
d.Manifest.permission.CAMERA) != PackageManager.PERMISSION_
GRANTED) {
            ActivityCompat.requestPermissions(this, arrayOf
(android.Manifest.permission.CAMERA), 0)
        }

        val scanningCodeButton = findViewById<Button>(R.id.
scanBarcodeButton)
        scanningCodeButton.setOnClickListener {
            val intent = Intent(this@MainActivity, BarcodeS
canningActivity::class.java)
            startActivity(intent)
        }
    }
}
```

1.7 BracodeScanningActivity.kt

Create the kotlin and related by `[File]`➔`[New]`➔`[Activity]`➔`[Empty Activity]` with BracodeScanning.kt and activity_barcode_scanning.xml. The new Intent should be added in AndroidManifest.xzm, (check it):

- activity_barcode_scanning.xml, only LinearLayout acclaimed:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
    android:orientation="vertical"
    tools:context=".BarcodeScanningActivity">
</LinearLayout>
```

- BarcodeScanningActivity.kt, 1. override `onResume()/onPause()` functions to start/stop camera; 2. implement `handleResult()` to display the scanning result and resume the camera preview:

```kotlin
package qrcode.kotlin.com.qrcodeapp

import android.support.v7.app.AppCompatActivity
import android.app.Activity
import android.os.Bundle
import android.widget.Toast
import me.dm7.barcodescanner.zbar.Result
import me.dm7.barcodescanner.zbar.ZBarScannerView

class BarcodeScanningActivity : AppCompatActivity(), ZBarSc
annerView.ResultHandler {

    private lateinit var mScannerView: ZBarScannerView

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        mScannerView = ZBarScannerView(this)
        setContentView(mScannerView)
    }

    override fun onResume() {
        super.onResume()
        mScannerView.setResultHandler(this)
        mScannerView.startCamera()
    }

    override fun onPause() {
        super.onPause()
        mScannerView.stopCamera()
    }

     override fun handleResult(result: Result?) {
        Toast.makeText(this, result?.contents, Toast.LENGTH
_SHORT).show()
        mScannerView.resumeCameraPreview(this)
    }

}
```

1.8 BarcodGenActivity.kt and related (Exercise)

0. implement
    - activity_main.xml: add button,
    - MainActivity.kt: Intent for jump.
1. barcode_gen_activity.xml:

```
<?xml version="1.0" encoding="utf-8"?>
/*
  LinearLayout
  EditText:  input (with id: contentEditText)
  Button: Click to generate barcode image (with id: generat
eButton)
*/
    <ImageView
        android:id="@+id/generationImageView"
        android:layout_width="match_parent"
        android:layout_marginTop="16dp"
        android:layout_height="wrap_content" />
```

2. BarcodGenActivity.kt: EditText wait for any input, and generate its QRcode image; show a warning if no input:

```kotlin
package qrcode.kotlin.com.qrcodeapp

import android.support.v7.app.AppCompatActivity
import android.os.Bundle

import android.graphics.BitmapFactory
import android.widget.Button
import android.widget.EditText
import android.widget.ImageView
import android.widget.Toast
import net.glxn.qrgen.android.QRCode

class BarcodeGenActivity : AppCompatActivity() {

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


        val contentEditText = findViewById<EditText>(R.id.c
ontentEditText)
        val generateButton = findViewById<Button>(R.id.gene
rateButton)
        val generationImageView = findViewById<ImageView>(R
.id.generationImageView)

        generateButton.setOnClickListener {
            val text = contentEditText.text.toString()

            if (text.isEmpty()) {
                Toast.makeText(this, "Enter something to cr
eate a barcode", Toast.LENGTH_SHORT).show()
                return@setOnClickListener
            }

            val bitmap = QRCode.from(text).withSize(600, 60
0).bitmap()
            generationImageView.setImageBitmap(bitmap)
        }
    }
}
```