QR Code Scanner App Documentation

Overview

This QR Code Scanner app is built using Kotlin and integrates Android CameraX and ML Kit's Barcode Scanning API to scan and decode QR codes. The scanned URL or text is displayed, styled dynamically, and can be opened in a WebView if it's a URL.

Features

- **Real-time QR code scanning** using CameraX.
- **Dynamic TextView** that underlines and changes color for detected URLs.
- **WebView integration** to open scanned URLs directly within the app.
- Handles permissions for camera access gracefully.

Code Explanation

1. MainActivity

This is the entry point of the application where the camera is initialized, QR code scanning is handled, and results are displayed.

a. Variables Initialization

```
private lateinit var previewView: PreviewView
private lateinit var resultTextView: TextView
private lateinit var cameraExecutor: ExecutorService
private lateinit var barcodeScanner: BarcodeScanner
```

- **`previewView`**: Displays the camera preview.
- **`resultTextView`**: Displays the scanned QR code text or URL.
- **`cameraExecutor`**: Handles background threading for image processing.
- **`barcodeScanner`**: Instance of ML Kit's Barcode Scanner.

b. Permissions Handling

```
val requestPermissionLauncher = registerForActivityResult(
    ActivityResultContracts.RequestPermission()
) { isGranted ->
    if (isGranted) {
        startCamera()
    } else {
        resultTextView.text = "Camera permission is required"
    }
}
```

- Uses **ActivityResultContracts.RequestPermission** to request camera permissions dynamically.
- Displays a message if permissions are denied.

c. Camera Setup

```
private fun startCamera() {
    val cameraProviderFuture = ProcessCameraProvider.getInstance(this)
    val screenSize = Size(1280, 720)
    val resolutionSelector =
ResolutionSelector.Builder().setResolutionStrategy(
        ResolutionStrategy(screenSize,
ResolutionStrategy.FALLBACK_RULE_NONE)
    ).build()

    cameraProviderFuture.addListener({
        val cameraProvider = cameraProviderFuture.get()
        val preview =
Preview.Builder().setResolutionSelector(resolutionSelector)
            .build()
            .also {
                it.setSurfaceProvider(previewView.surfaceProvider)
            }

        val imageAnalyzer =
ImageAnalysis.Builder().setBackpressureStrategy(ImageAnalysis.STRATEGY_KEEP
_ONLY_LATEST)
            .build().also {
                it.setAnalyzer(cameraExecutor, { imageProxy ->
                    processImageProxy(imageProxy)
                })
            }

        val cameraSelector = CameraSelector.DEFAULT_BACK_CAMERA
        cameraProvider.bindToLifecycle(this, cameraSelector, preview,
imageAnalyzer)
    }, ContextCompat.getMainExecutor(this))
}
```

- **Camera Provider**: Initializes and binds the camera lifecycle.
- **Preview**: Configures the camera preview.
- **Image Analyzer**: Processes frames to detect QR codes.

d. Image Processing

```
@OptIn(ExperimentalGetImage::class)
private fun processImageProxy(imageProxy: ImageProxy) {
    val mediaImage = imageProxy.image
    if (mediaImage != null) {
        val image = InputImage.fromMediaImage(mediaImage,
imageProxy.imageInfo.rotationDegrees)
        barcodeScanner.process(image)
            .addOnSuccessListener { barcodes ->
                for (barcode in barcodes) {
                    handleBarcode(barcode)
                }
            }
            .addOnFailureListener {
                resultTextView.text = "Failed to scan QR code"
            }
            .addOnCompleteListener {
                imageProxy.close()
            }
    }
}
```

- Converts the camera's frame into an **InputImage** for ML Kit to process.
- Handles results via **Success** and **Failure** listeners.

e. Handling QR Code Data

```
private fun handleBarcode(barcode: Barcode) {
    val url = barcode.url?.url ?: barcode.displayValue
    if (url != null) {
        resultTextView.apply {
            text = url
            setTextColor(Color.BLUE) // Set text color to blue
            paintFlags = paintFlags or Paint.UNDERLINE_TEXT_FLAG // Add
underline
            setOnClickListener {
                val intent = Intent(this@MainActivity,
WebVIewActivity::class.java)
                intent.putExtra("url", url)
                startActivity(intent)
            }
        }
    } else {
        resultTextView.apply {
            text = "No QR code Detected"
            setTextColor(Color.BLACK) // Reset to default color
            paintFlags = 0 // Remove underline
        }
    }
}
```

- Displays the decoded QR code.
- Dynamically styles the TextView for URLs (blue text and underlined).
- Opens URLs in **WebViewActivity** when clicked.

---

2. WebViewActivity

Handles the display of URLs in a WebView.

WebView Setup

```
val webView = findViewById<WebView>(R.id.webView)
val url = intent.getStringExtra("url")

webView.webViewClient = WebViewClient()
webView.settings.javaScriptEnabled = true

if (url != null) {
    webView.loadUrl(url)
}
```

- Configures the WebView to handle JavaScript and load the URL passed via Intent.

Key Learning Points

1. **CameraX Integration**: Learn how to use CameraX to display a live camera feed.
2. **ML Kit**: Integrate barcode scanning for real-time QR code detection.
3. **Dynamic UI Styling**: Update TextView properties (color, underline) dynamically.
4. **Permissions Handling**: Manage runtime permissions for camera access.
5. **WebView**: Use WebView to display URLs in-app.
6. **Threading**: Use ExecutorService for background tasks like image processing.

Dependencies

Add the following dependencies to your `build.gradle` file:

```
implementation 'androidx.camera:camera-core:1.4.0'
implementation 'androidx.camera:camera-lifecycle:1.4.0'
implementation 'androidx.camera:camera-view:1.4.0'
implementation 'com.google.mlkit:barcode-scanning:17.0.2'
```

Layout Files

activity_main.xml

Defines the UI for the camera preview and result display.

```
<androidx.camera.view.PreviewView
    android:id="@+id/previewView"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1" />

<TextView
    android:id="@+id/resultTextView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Scan a QR Code"
    android:gravity="center"
    android:textSize="18sp"
    android:padding="16dp" />
```

activity_web_view.xml

Defines the UI for the WebView.

```
<WebView
    android:id="@+id/webView"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

Conclusion

This project demonstrates the integration of CameraX and ML Kit to create a functional QR code scanner app with dynamic UI updates and WebView integration. It provides a solid foundation for learning modern Android development practices.