Date: 06 March 2025

MLKitDocumentScannerSample Application

VERSION: 1.0.0

**Overview:**

The MLKitDocumentScannerSample application can identify and scan documents. Under the hood, the component contains NFIs that use the Google MLKit Document Scanner API. The component also supports Auto and manual cropping, applying filters and retake option.

**Requirements:**

* Volt MX Iris

**Devices:**

* Mobile
* Tablet

**Platforms:**

* Android

**Features:**

1. The component utilizes Google MLKit Document Scanner SDK for auto detecting and scanning documents.

2. It supports all types of documents.

**A. App Functionality:**

1. When you build and run the app, you can scan the document. And get the scanned Image in the form of image and pdf.
2. You can either manually adjust the document to fit the screen or auto-detect it.

**B. Importing the app:**  
  
To import the MLKitDocumentScannerSample application into your workspace, follow these steps:

1. Open Volt MX Iris
2. On the main menu select Forge/Browse.
3. Search for the MLKitDocumentScannerSample application, and then click Import to Workspace. The app is imported to your workspace. A dialog box appears, confirming that the app has been imported. Click OK.

4. Switch to your project containing the MLKitDocumentScannerSample application. To switch to your project, click File/Open/Reference Architecture /<project name>

**C. Building the app:**

After performing all the above steps, you can build your app and run it on your device. For more information, you can refer to the Building and Viewing an Application section of the Volt MX User Guide.

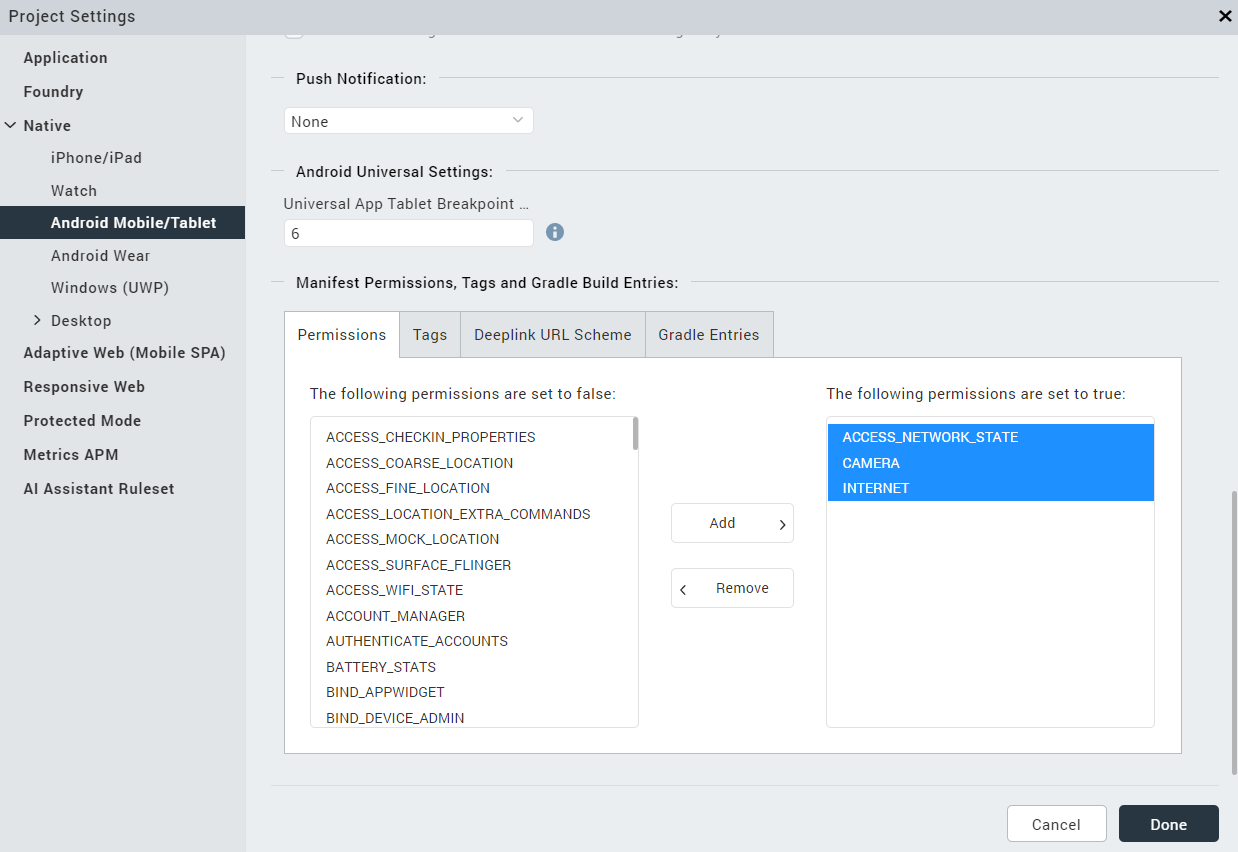
You can then run your app to see the Document Scanner work in real time.

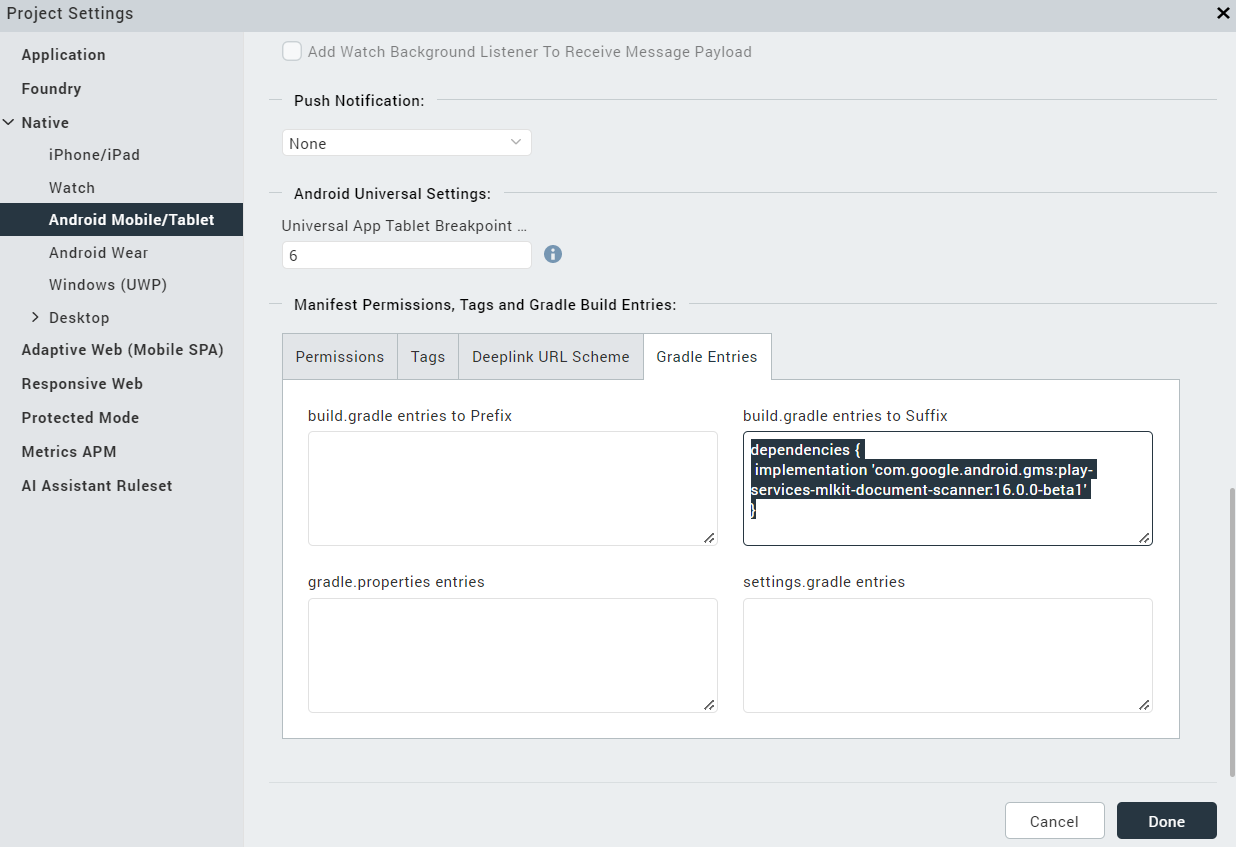
**D. Configure the UI and settings of the MLKitDocumentScannerSample application**

The MLKitDocumentScannerSample application consists of the following component:

**Configuring Native Settings (Android)**  
  
Ensure the below settings are reflected in the sample app:

1. Check camera permission at **Project Settings/Native/Android Mobile/Tablet/Manifest Permissions, Tags and Gradle Build Entries/Permissions**

  
To set a permission to true, select the permission from the left panel, and then click Add

1. Check the following entries are available at **Project Settings/Native/Android Mobile/Tablet/Manifest Permissions, Tags and Gradle Build Entries/Gradle Entries/build.gradle entries to Suffix** 

dependencies {

implementation 'com.google.android.gms: play-services-mlkit-document-scanner:16.0.0-beta1'

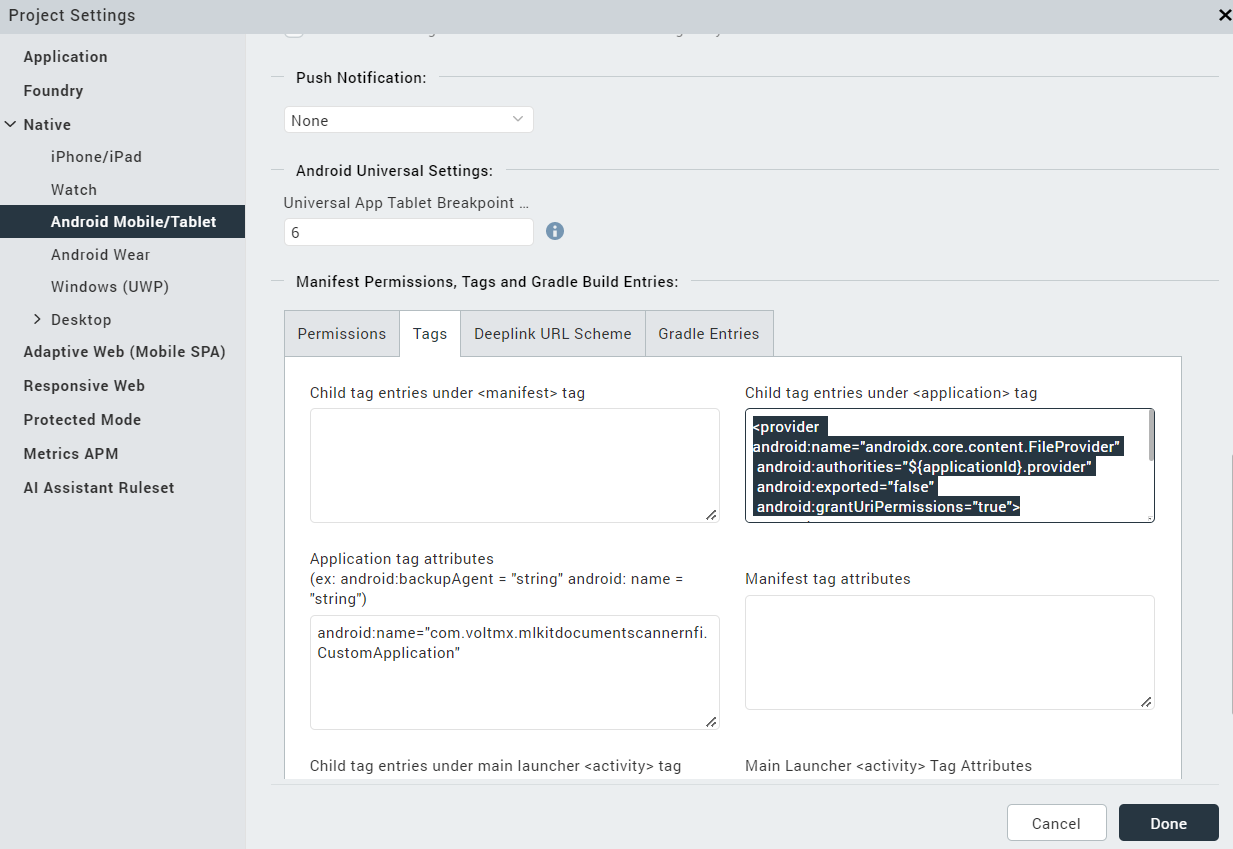
}

3. Check the following entries are available at **Project Settings/NativeAndroid Mobile/Tablet/Manifest Permissions, Tags and Gradle Build Entries/Tags/Child tag entries under <application>tag**

<provider android:name="androidx.core.content.FileProvider" android:authorities="${applicationId}.provider" android:exported="false" android:grantUriPermissions="true">

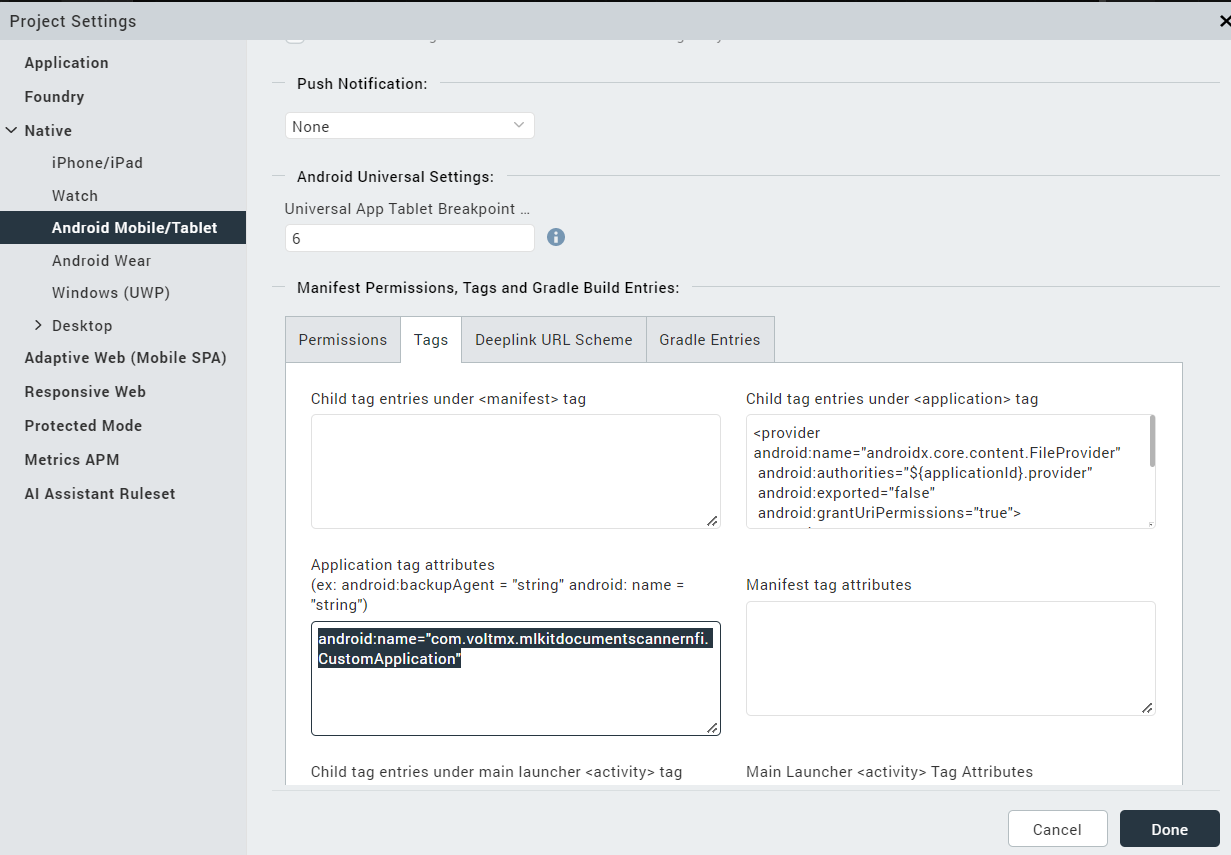
<meta-data android:name="android.support.FILE\_PROVIDER\_PATHS" android:resource="@xml/filepaths" />

</provider>

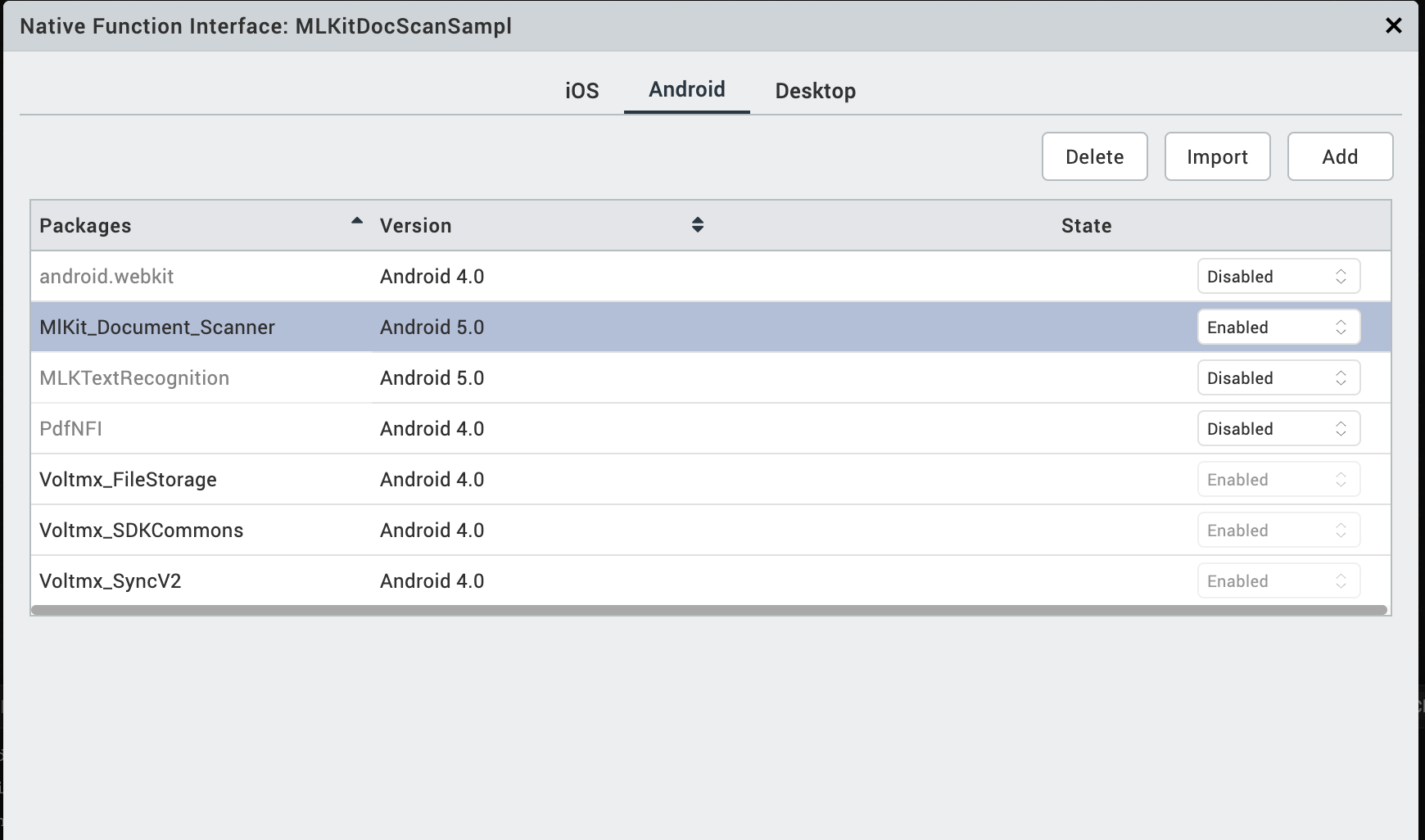


4. Check the following entry is available at **Project Settings/Native/Android Mobile/Tablet/Manifest Permissions, Tags and Gradle Build Entries/Tags/Application tag attributes.**

android:name="com.voltmx.mlkitdocumentscannernfi.CustomApplication"



5. Check the MLKit document scanner NFI added and enabled at **Iris/Edit/Manage Native Function API(s)/Android/**

  
  
**NOTE:**  
  
· If you do not add the **Gradle entries** to your project, the app crashes.  
  
  
After you configure the native settings, you can Build and Run your app to see the Document Scanner component work in real time.

**MLKitDocumentScannerSample** **application:**  
  
The following are screen shots of the app:

1. **Home Screen:**

Consists of a scan button, when tapped directly to the scanning screen.

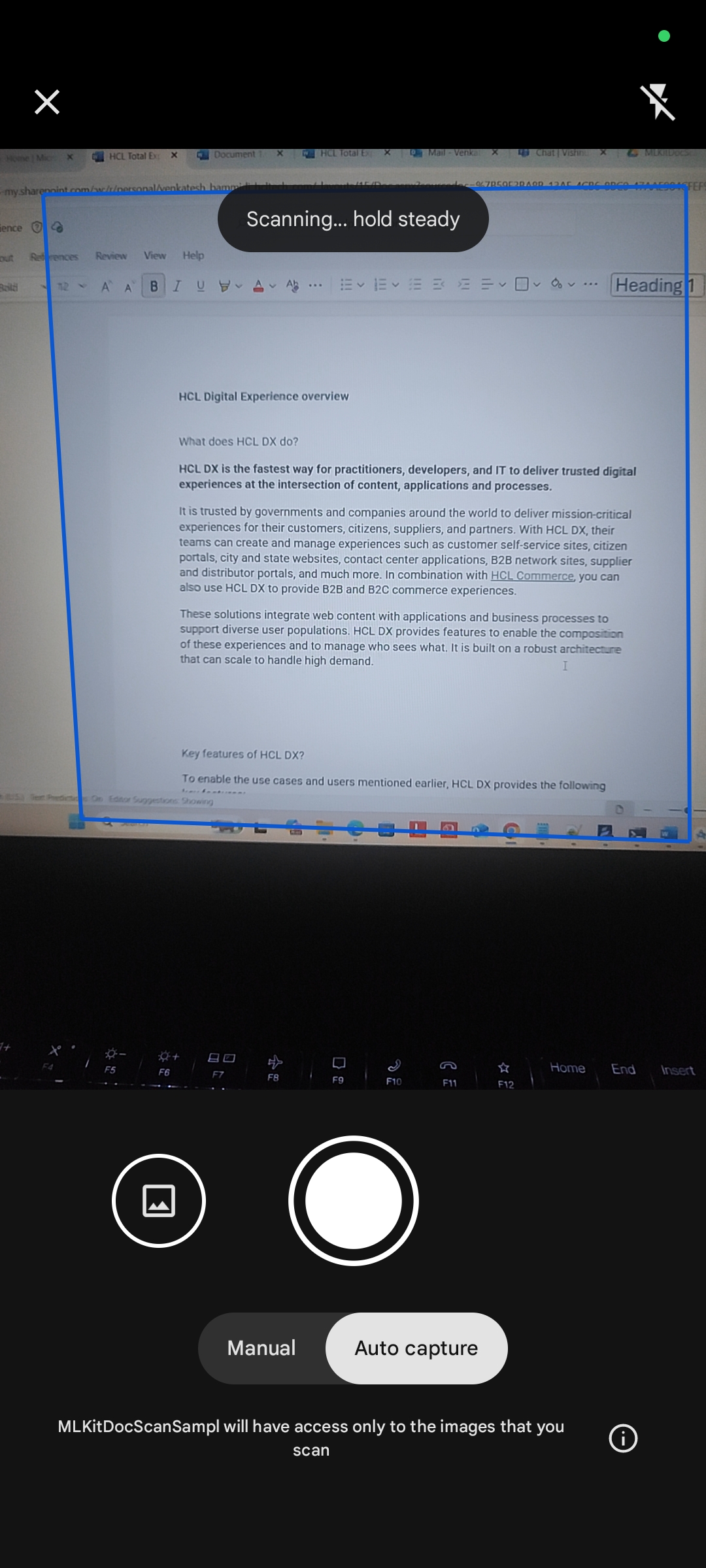


btnScanOnClick: function() {

this.mlkitBarcodeScannerNFI.scanDocument(enableGalleryImport, FULL\_MODE, pageLimit, this.documentScannerCallback);},

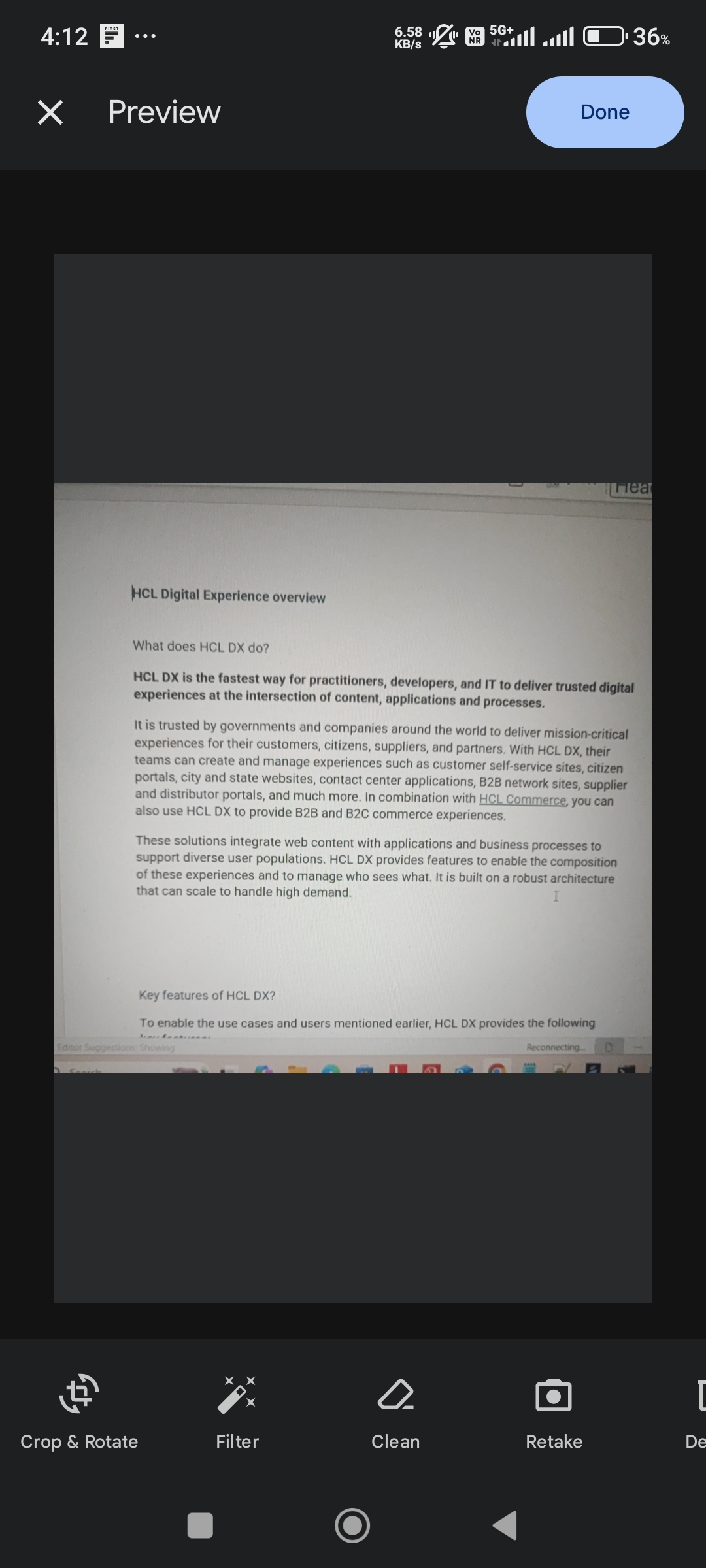
**2. Capture Screen:**

Camera is used to capture the Document. It scans and captures the document automatically



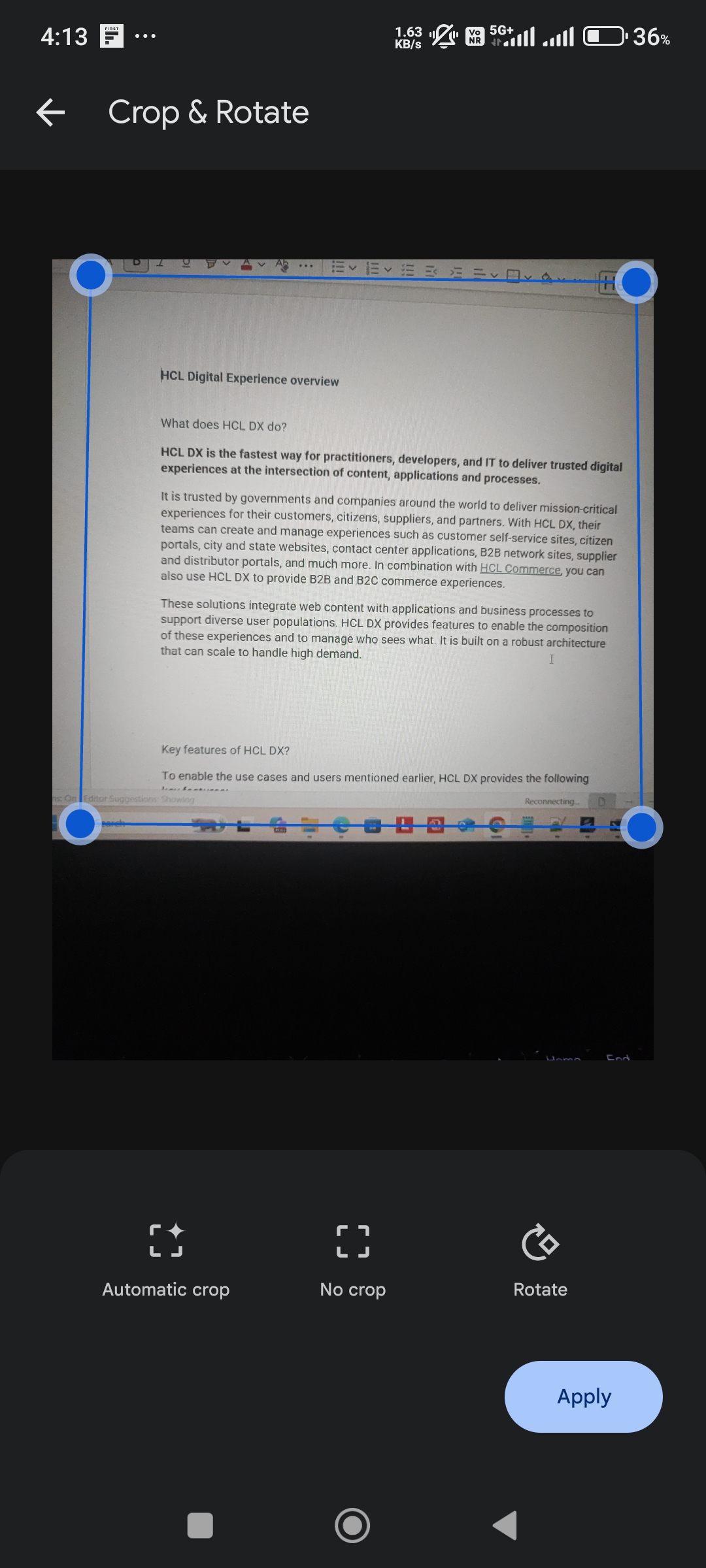
**3. Preview Screen:**

After the successful capture of the document, Next will be the preview screen where we will see the preview of the scanned document provided with options like crop, filter, clean, rotate and delete.



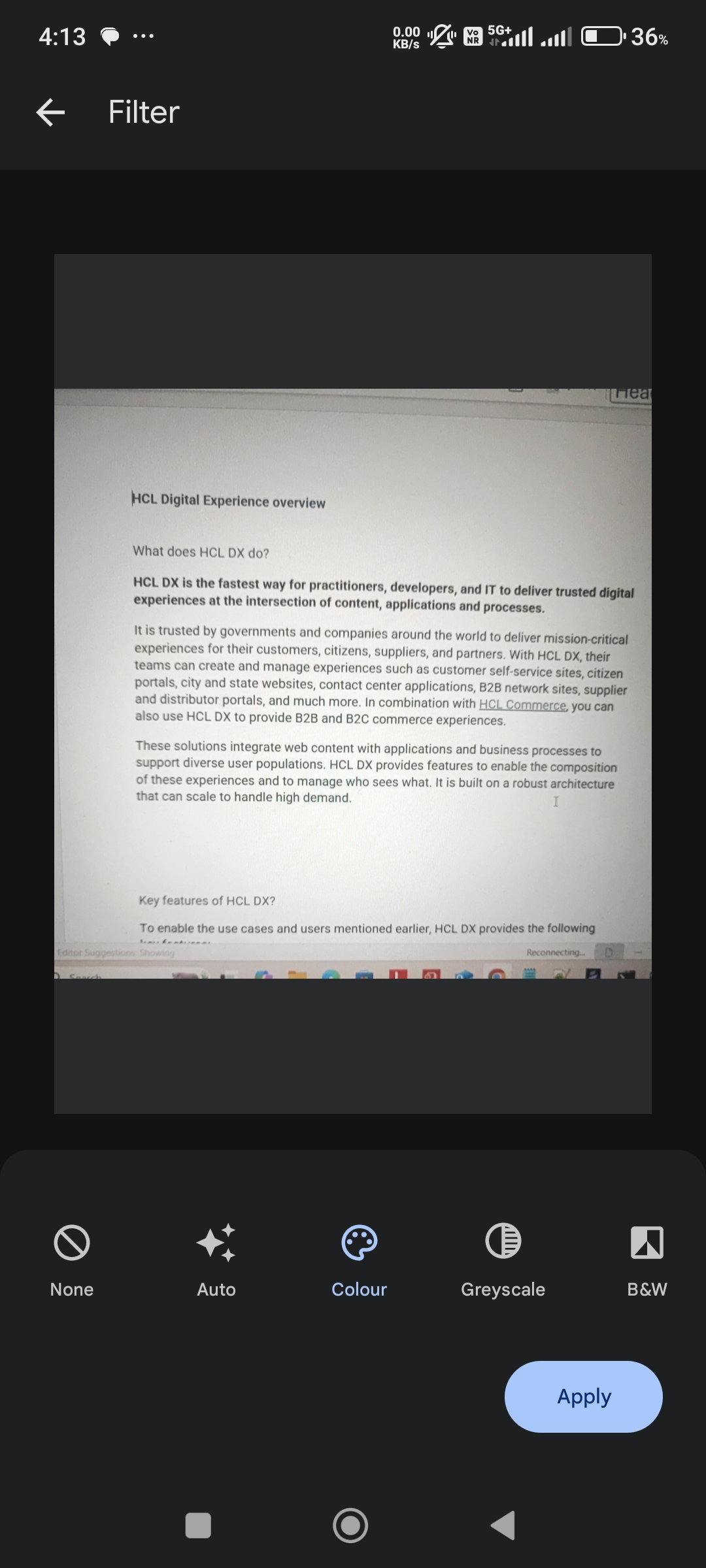
**4. Cropping Option:**

The cropping option helps in cropping the document to the desired level. Here the cropping can be done either manually or with the provided auto crop feature.



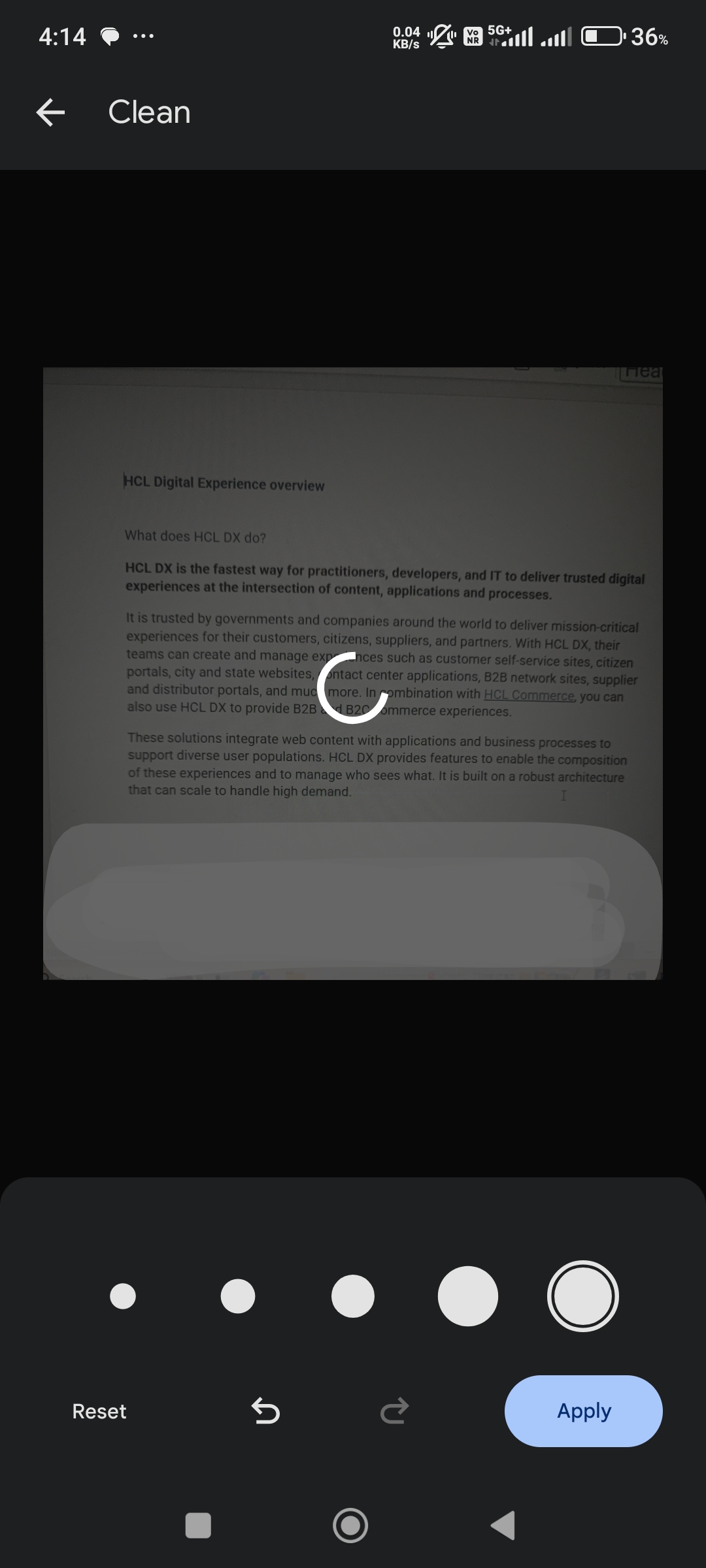
**5. Filters Option:**

We have multiple filters which can be applied to the scanned document like greyscale, b&w, Color, and auto filter.



**6. Clean Option:**

The clean option helps in removing unwanted Moir.



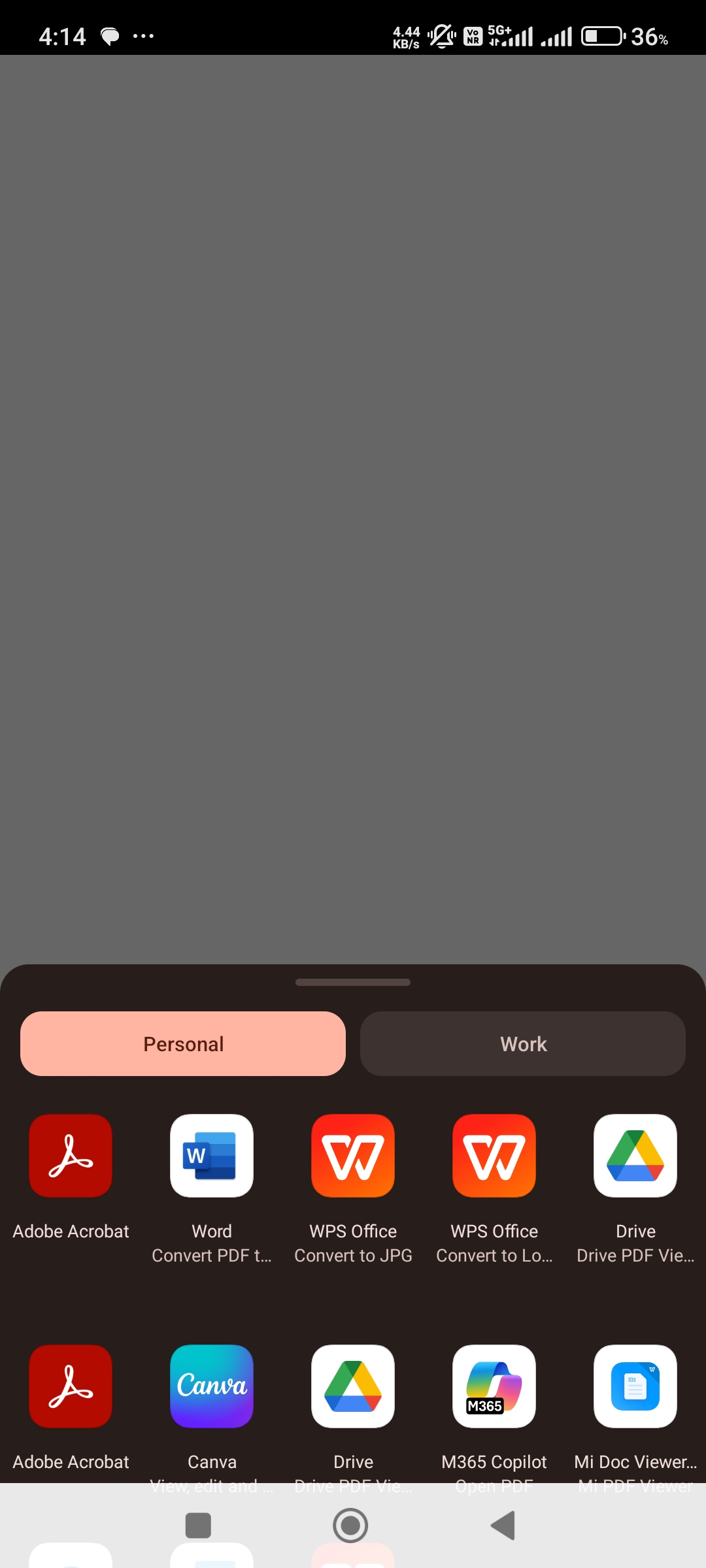
**7. Output Screen:**

The Callback method gives the result

a. Image (in base64 string) and its path

b. PDF (in base64 string) and its path

The App suggests the available PDF Viewer Apps as the output PDF cannot be displayed in webview of mobile APP.



documentScannerCallback: function(status, result) {

// status can be the following, RESULT\_OK, RESULT\_CANCELED, RESULT\_SCAN\_FAILED and RESULT\_LISTENER\_FAILED

if (status === "RESULT\_OK") {

voltmx.print("DS: imagePath: " + result["imagePath"]); // for Image path

voltmx.print("DS: imageBase64: " + result["imageBase64"]); // Image base64

voltmx.print("DS: result: " + result["result"]); voltmx.print("DS: pdfPath: " + result["pdfPath"]); //for pdf path

voltmx.print("DS: pdfBase64: " + result["pdfBase64"]); //pdf base64

// call this method in a required place to view the pdf. this.mlkitBarcodeScannerNFI.openPDF(result["pdfPath"]);

}

**Limitations:**

1. This component supports only 64-bit devices

**Known issues:**

-N.A.-