Umair Afzal

PROJECT  21F-9612

SOFTWARE FOR MOBILE DEVICES

SOFTWARE FOR MOBILE DEVICES

PROJECT

**Manifest:**<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.example.project">  
  
 <uses-permission android:name="android.permission.CAMERA" />  
 <uses-feature android:name="android.hardware.camera" android:required="true" />  
 <uses-permission android:name="android.permission.INTERNET" />  
  
 <application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true">  
  
 <!-- Activities without intent-filters can remain as-is -->  
 <activity android:name=".ObjectDetectionActivity"  
 android:exported="false" /> <!-- Not accessible externally -->  
  
 <activity android:name=".BarcodeScanningActivity"  
 android:exported="false" /> <!-- Not accessible externally -->  
  
 <activity android:name=".FaceDetectionActivity"  
 android:exported="false" /> <!-- Not accessible externally -->  
  
 <activity android:name=".TextRecognitionActivity"  
 android:exported="false" /> <!-- Not accessible externally -->  
  
 <!-- MainActivity (LAUNCHER) MUST be exported -->  
 <activity android:name=".MainActivity"  
 android:exported="true"> <!-- Required for launcher -->  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 </application>  
</manifest>

**Build.gradle:**plugins {  
 id 'com.android.application'  
 id 'com.google.gms.google-services'  
}  
  
android {  
 namespace 'com.example.project'  
 compileSdk 34  
  
 defaultConfig {  
 applicationId "com.example.project"  
 minSdk 21  
 targetSdk 34  
 versionCode 1  
 versionName "1.0"  
 multiDexEnabled true  
 }  
  
 buildTypes {  
 release {  
 minifyEnabled false  
 proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'  
 }  
 }  
  
 compileOptions {  
 sourceCompatibility JavaVersion.VERSION\_1\_8  
 targetCompatibility JavaVersion.VERSION\_1\_8  
 }  
  
 packagingOptions {  
 exclude 'META-INF/DEPENDENCIES'  
 exclude 'META-INF/LICENSE'  
 exclude 'META-INF/LICENSE.txt'  
 exclude 'META-INF/NOTICE'  
 exclude 'META-INF/NOTICE.txt'  
 pickFirst 'lib/\*\*/libc++\_shared.so'  
 pickFirst 'lib/\*\*/libfbjni.so'  
 }  
}  
  
dependencies {  
 implementation 'com.google.mlkit:text-recognition:16.0.0'  
 implementation 'com.google.mlkit:face-detection:16.1.5' // Corrected version  
 implementation 'com.google.mlkit:barcode-scanning:17.3.0'  
 implementation 'com.google.mlkit:object-detection:17.0.0' // Corrected version  
 implementation 'androidx.constraintlayout:constraintlayout:2.1.4'  
 implementation 'com.google.mlkit:text-recognition-latin:16.0.0'  
  
 // For the executor  
 implementation 'com.google.guava:guava:31.1-android'  
  
// CameraX dependencies  
 implementation 'androidx.camera:camera-core:1.3.4'  
 implementation 'androidx.camera:camera-camera2:1.3.4'  
 implementation 'androidx.camera:camera-lifecycle:1.3.4'  
 implementation 'androidx.camera:camera-view:1.3.4'  
  
}  
configurations.all {  
 exclude group: 'org.jetbrains.kotlin', module: 'kotlin-stdlib-jdk8'  
}

**Barcode\_Scannning.xml:**<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="@color/white">  
  
 <com.example.project.CameraSourcePreview  
 android:id="@+id/camera\_preview"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_above="@+id/barcode\_result\_panel" />  
  
 <com.example.project.GraphicOverlay  
 android:id="@+id/graphic\_overlay"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_above="@+id/barcode\_result\_panel" />  
  
 <View  
 android:id="@+id/scan\_box\_guide"  
 android:layout\_width="200dp"  
 android:layout\_height="200dp"  
 android:layout\_centerInParent="true"  
 android:background="#80FFFFFF"  
 android:alpha="0.5" />  
  
 <TextView  
 android:id="@+id/barcode\_result\_panel"  
 android:layout\_width="match\_parent"  
 android:layout\_height="150dp"  
 android:layout\_alignParentBottom="true"  
 android:background="@color/gray"  
 android:padding="16dp"  
 android:scrollbars="vertical"  
 android:textColor="@color/black"  
 android:textSize="16sp"  
 android:isScrollContainer="true" />  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_above="@id/barcode\_result\_panel"  
 android:layout\_marginBottom="16dp"  
 android:gravity="center"  
 android:orientation="horizontal">  
  
 <Button  
 android:id="@+id/open\_button"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginEnd="8dp"  
 android:backgroundTint="@color/teal\_200"  
 android:text="@string/open"  
 android:textColor="@color/white" />  
  
 <Button  
 android:id="@+id/copy\_button"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginEnd="8dp"  
 android:backgroundTint="@color/teal\_200"  
 android:text="@string/copy"  
 android:textColor="@color/white" />  
  
 <Button  
 android:id="@+id/save\_button"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:backgroundTint="@color/teal\_200"  
 android:text="@string/save"  
 android:textColor="@color/white" />  
  
 </LinearLayout>  
  
</RelativeLayout>

**Barcode\_Scanning.java:**  
package com.example.project;  
  
import android.Manifest;  
import android.content.ClipData;  
import android.content.ClipboardManager;  
import android.content.Context;  
import android.content.Intent;  
import android.content.pm.PackageManager;  
import android.net.Uri;  
import android.os.Bundle;  
import android.widget.Button;  
import android.widget.TextView;  
import android.widget.Toast;  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.camera.core.ImageAnalysis;  
import androidx.core.app.ActivityCompat;  
  
public class BarcodeScanningActivity extends AppCompatActivity {  
 private CameraSourcePreview cameraPreview;  
 private GraphicOverlay graphicOverlay;  
 private TextView barcodeResultPanel;  
 private String barcodeValue = "";  
 private static final int CAMERA\_PERMISSION\_CODE = 100;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_barcode\_scanning);  
  
 cameraPreview = findViewById(R.id.camera\_preview);  
 graphicOverlay = findViewById(R.id.graphic\_overlay);  
 barcodeResultPanel = findViewById(R.id.barcode\_result\_panel);  
 Button openButton = findViewById(R.id.open\_button);  
 Button copyButton = findViewById(R.id.copy\_button);  
 Button saveButton = findViewById(R.id.save\_button);  
  
 if (ActivityCompat.checkSelfPermission(this, Manifest.permission.CAMERA) != PackageManager.PERMISSION\_GRANTED) {  
 ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.CAMERA}, CAMERA\_PERMISSION\_CODE);  
 } else {  
 startCamera();  
 }  
  
 openButton.setOnClickListener(v -> {  
 if (!barcodeValue.isEmpty()) {  
 Intent browserIntent = new Intent(Intent.ACTION\_VIEW, Uri.parse(barcodeValue));  
 startActivity(browserIntent);  
 }  
 });  
  
 copyButton.setOnClickListener(v -> {  
 if (!barcodeValue.isEmpty()) {  
 ClipboardManager clipboard = (ClipboardManager) getSystemService(Context.CLIPBOARD\_SERVICE);  
 ClipData clip = ClipData.newPlainText("Barcode Value", barcodeValue);  
 clipboard.setPrimaryClip(clip);  
 Toast.makeText(this, "Barcode copied to clipboard", Toast.LENGTH\_SHORT).show();  
 }  
 });  
  
 saveButton.setOnClickListener(v -> {  
 // Placeholder for save functionality  
 Toast.makeText(this, "Save functionality not implemented", Toast.LENGTH\_SHORT).show();  
 });  
 }  
  
 private void startCamera() {  
 ImageAnalysis imageAnalysis = new ImageAnalysis.Builder()  
 .setBackpressureStrategy(ImageAnalysis.STRATEGY\_KEEP\_ONLY\_LATEST)  
 .build();  
  
 cameraPreview.startCamera(this, graphicOverlay, imageAnalysis);  
 }  
  
 @Override  
 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
 super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
 if (requestCode == CAMERA\_PERMISSION\_CODE && grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION\_GRANTED) {  
 startCamera();  
 } else {  
 Toast.makeText(this, "Camera permission denied", Toast.LENGTH\_SHORT).show();  
 finish();  
 }  
 }  
  
 @Override  
 protected void onPause() {  
 super.onPause();  
 cameraPreview.stop();  
 }  
  
 @Override  
 protected void onResume() {  
 super.onResume();  
 if (ActivityCompat.checkSelfPermission(this, Manifest.permission.CAMERA) == PackageManager.PERMISSION\_GRANTED) {  
 startCamera();  
 }  
 }  
  
 @Override  
 protected void onDestroy() {  
 super.onDestroy();  
 cameraPreview.stop();  
 }  
}

**Face\_Detection.xml:**<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="@color/white">  
  
 <com.example.project.CameraSourcePreview  
 android:id="@+id/camera\_preview"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_above="@+id/face\_data\_panel" />  
  
 <com.example.project.GraphicOverlay  
 android:id="@+id/graphic\_overlay"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_above="@+id/face\_data\_panel" />  
  
 <TextView  
 android:id="@+id/face\_data\_panel"  
 android:layout\_width="match\_parent"  
 android:layout\_height="150dp"  
 android:layout\_alignParentBottom="true"  
 android:background="@color/gray"  
 android:padding="16dp"  
 android:scrollbars="vertical"  
 android:textColor="@color/black"  
 android:textSize="16sp"  
 android:isScrollContainer="true" />  
  
 <Button  
 android:id="@+id/capture\_button"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_above="@id/face\_data\_panel"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginBottom="16dp"  
 android:backgroundTint="@color/purple\_500"  
 android:text="@string/capture"  
 android:textColor="@color/white" />  
  
</RelativeLayout>

**Face\_Detection.java:**  
package com.example.project;  
  
import android.Manifest;  
import android.content.pm.PackageManager;  
import android.os.Bundle;  
import android.widget.Button;  
import android.widget.TextView;  
import android.widget.Toast;  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.camera.core.ImageAnalysis;  
import androidx.core.app.ActivityCompat;  
  
public class FaceDetectionActivity extends AppCompatActivity {  
 private CameraSourcePreview cameraPreview;  
 private GraphicOverlay graphicOverlay;  
 private TextView faceDataPanel;  
 private static final int *CAMERA\_PERMISSION\_CODE* = 100;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_face\_detection*);  
  
 cameraPreview = findViewById(R.id.*camera\_preview*);  
 graphicOverlay = findViewById(R.id.*graphic\_overlay*);  
 faceDataPanel = findViewById(R.id.*face\_data\_panel*);  
 Button captureButton = findViewById(R.id.*capture\_button*);  
  
 if (ActivityCompat.*checkSelfPermission*(this, Manifest.permission.*CAMERA*) != PackageManager.*PERMISSION\_GRANTED*) {  
 ActivityCompat.*requestPermissions*(this, new String[]{Manifest.permission.*CAMERA*}, *CAMERA\_PERMISSION\_CODE*);  
 } else {  
 startCamera();  
 }  
  
 captureButton.setOnClickListener(v -> processImage());  
 }  
  
 private void startCamera() {  
 ImageAnalysis imageAnalysis = new ImageAnalysis.Builder()  
 .setBackpressureStrategy(ImageAnalysis.*STRATEGY\_KEEP\_ONLY\_LATEST*)  
 .build();  
  
 cameraPreview.startCamera(this, graphicOverlay, imageAnalysis);  
 }  
  
 private void processImage() {  
 // Placeholder for image processing logic  
 Toast.*makeText*(this, "Image processing not implemented", Toast.*LENGTH\_SHORT*).show();  
 }  
  
 @Override  
 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
 super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
 if (requestCode == *CAMERA\_PERMISSION\_CODE* && grantResults.length > 0 && grantResults[0] == PackageManager.*PERMISSION\_GRANTED*) {  
 startCamera();  
 } else {  
 Toast.*makeText*(this, "Camera permission denied", Toast.*LENGTH\_SHORT*).show();  
 finish();  
 }  
 }  
  
 @Override  
 protected void onPause() {  
 super.onPause();  
 cameraPreview.stop();  
 }  
  
 @Override  
 protected void onResume() {  
 super.onResume();  
 if (ActivityCompat.*checkSelfPermission*(this, Manifest.permission.*CAMERA*) == PackageManager.*PERMISSION\_GRANTED*) {  
 startCamera();  
 }  
 }  
  
 @Override  
 protected void onDestroy() {  
 super.onDestroy();  
 cameraPreview.stop();  
 }  
}

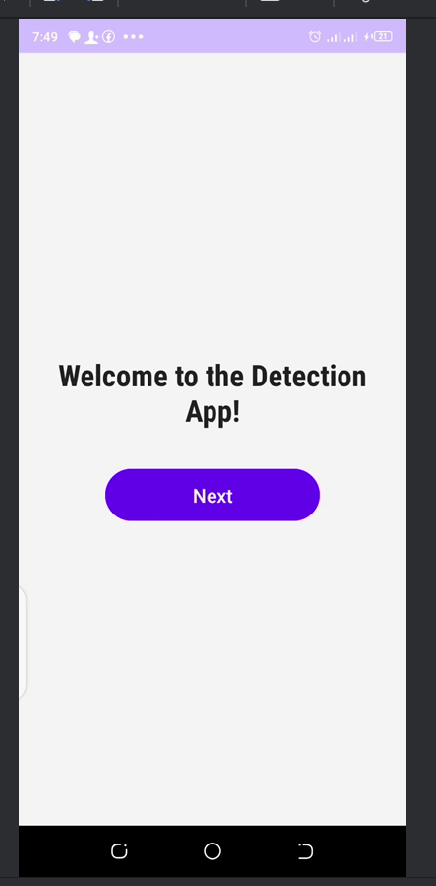
**Object\_Detection.xml:**<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="@color/white">  
  
 <com.example.project.CameraSourcePreview  
 android:id="@+id/camera\_preview"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_above="@+id/label\_list" />  
  
 <com.example.project.GraphicOverlay  
 android:id="@+id/graphic\_overlay"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_above="@+id/label\_list" />  
  
 <TextView  
 android:id="@+id/label\_list"  
 android:layout\_width="match\_parent"  
 android:layout\_height="150dp"  
 android:layout\_alignParentBottom="true"  
 android:background="@color/gray"  
 android:padding="16dp"  
 android:scrollbars="vertical"  
 android:textColor="@color/black"  
 android:textSize="16sp"  
 android:isScrollContainer="true" />  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_above="@id/label\_list"  
 android:layout\_marginBottom="16dp"  
 android:gravity="center"  
 android:orientation="horizontal">  
  
 <Button  
 android:id="@+id/capture\_button"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginEnd="8dp"  
 android:backgroundTint="@color/purple\_500"  
 android:text="@string/capture"  
 android:textColor="@color/white" />  
  
 <CheckBox  
 android:id="@+id/track\_toggle"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginEnd="8dp"  
 android:text="@string/track"  
 android:textColor="@color/black" />  
  
 <Button  
 android:id="@+id/settings\_button"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:backgroundTint="@color/teal\_200"  
 android:text="@string/settings"  
 android:textColor="@color/white" />  
  
 </LinearLayout>  
  
</RelativeLayout>

**Object\_Detection.java:**package com.example.project;  
  
import android.Manifest;  
import android.content.pm.PackageManager;  
import android.os.Bundle;  
import android.widget.Button;  
import android.widget.CheckBox;  
import android.widget.TextView;  
import android.widget.Toast;  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.camera.core.ImageAnalysis;  
import androidx.core.app.ActivityCompat;  
  
public class ObjectDetectionActivity extends AppCompatActivity {  
 private CameraSourcePreview cameraPreview;  
 private GraphicOverlay graphicOverlay;  
 private TextView labelList;  
 private CheckBox trackToggle;  
 private static final int *CAMERA\_PERMISSION\_CODE* = 100;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_object\_detection*);  
  
 cameraPreview = findViewById(R.id.*camera\_preview*);  
 graphicOverlay = findViewById(R.id.*graphic\_overlay*);  
 labelList = findViewById(R.id.*label\_list*);  
 Button captureButton = findViewById(R.id.*capture\_button*);  
 trackToggle = findViewById(R.id.*track\_toggle*);  
 Button settingsButton = findViewById(R.id.*settings\_button*);  
  
 if (ActivityCompat.*checkSelfPermission*(this, Manifest.permission.*CAMERA*) != PackageManager.*PERMISSION\_GRANTED*) {  
 ActivityCompat.*requestPermissions*(this, new String[]{Manifest.permission.*CAMERA*}, *CAMERA\_PERMISSION\_CODE*);  
 } else {  
 startCamera();  
 }  
  
 captureButton.setOnClickListener(v -> processImage());  
  
 trackToggle.setOnCheckedChangeListener((buttonView, isChecked) -> {  
 // Placeholder for tracking toggle logic  
 Toast.*makeText*(this, "Tracking " + (isChecked ? "enabled" : "disabled"), Toast.*LENGTH\_SHORT*).show();  
 });  
  
 settingsButton.setOnClickListener(v -> {  
 // Placeholder for settings  
 Toast.*makeText*(this, "Settings not implemented", Toast.*LENGTH\_SHORT*).show();  
 });  
 }  
  
 private void startCamera() {  
 ImageAnalysis imageAnalysis = new ImageAnalysis.Builder()  
 .setBackpressureStrategy(ImageAnalysis.*STRATEGY\_KEEP\_ONLY\_LATEST*)  
 .build();  
  
 cameraPreview.startCamera(this, graphicOverlay, imageAnalysis);  
 }  
  
 private void processImage() {  
 // Placeholder for image processing logic  
 Toast.*makeText*(this, "Image processing not implemented", Toast.*LENGTH\_SHORT*).show();  
 }  
  
 @Override  
 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
 super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
 if (requestCode == *CAMERA\_PERMISSION\_CODE* && grantResults.length > 0 && grantResults[0] == PackageManager.*PERMISSION\_GRANTED*) {  
 startCamera();  
 } else {  
 Toast.*makeText*(this, "Camera permission denied", Toast.*LENGTH\_SHORT*).show();  
 finish();  
 }  
 }  
  
 @Override  
 protected void onPause() {  
 super.onPause();  
 cameraPreview.stop();  
 }  
  
 @Override  
 protected void onResume() {  
 super.onResume();  
 if (ActivityCompat.*checkSelfPermission*(this, Manifest.permission.*CAMERA*) == PackageManager.*PERMISSION\_GRANTED*) {  
 startCamera();  
 }  
 }  
  
 @Override  
 protected void onDestroy() {  
 super.onDestroy();  
 cameraPreview.stop();  
 }  
}

**Text\_Recognition.xml:**<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="@color/white">  
  
 <com.example.project.CameraSourcePreview  
 android:id="@+id/camera\_preview"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_above="@+id/text\_output" />  
  
 <com.example.project.GraphicOverlay  
 android:id="@+id/graphic\_overlay"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_above="@+id/text\_output" />  
  
 <TextView  
 android:id="@+id/text\_output"  
 android:layout\_width="match\_parent"  
 android:layout\_height="150dp"  
 android:layout\_alignParentBottom="true"  
 android:background="@color/gray"  
 android:padding="16dp"  
 android:scrollbars="vertical"  
 android:textColor="@color/black"  
 android:textSize="16sp"  
 android:isScrollContainer="true" />  
  
 <Button  
 android:id="@+id/capture\_button"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_above="@id/text\_output"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginBottom="16dp"  
 android:backgroundTint="@color/purple\_500"  
 android:text="@string/capture"  
 android:textColor="@color/white" />  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_above="@id/capture\_button"  
 android:layout\_marginBottom="16dp"  
 android:gravity="center"  
 android:orientation="horizontal">  
  
 <Button  
 android:id="@+id/copy\_button"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginEnd="8dp"  
 android:backgroundTint="@color/teal\_200"  
 android:text="@string/copy"  
 android:textColor="@color/white" />  
  
 <Button  
 android:id="@+id/share\_button"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:backgroundTint="@color/teal\_200"  
 android:text="@string/share"  
 android:textColor="@color/white" />  
  
 </LinearLayout>  
  
</RelativeLayout>

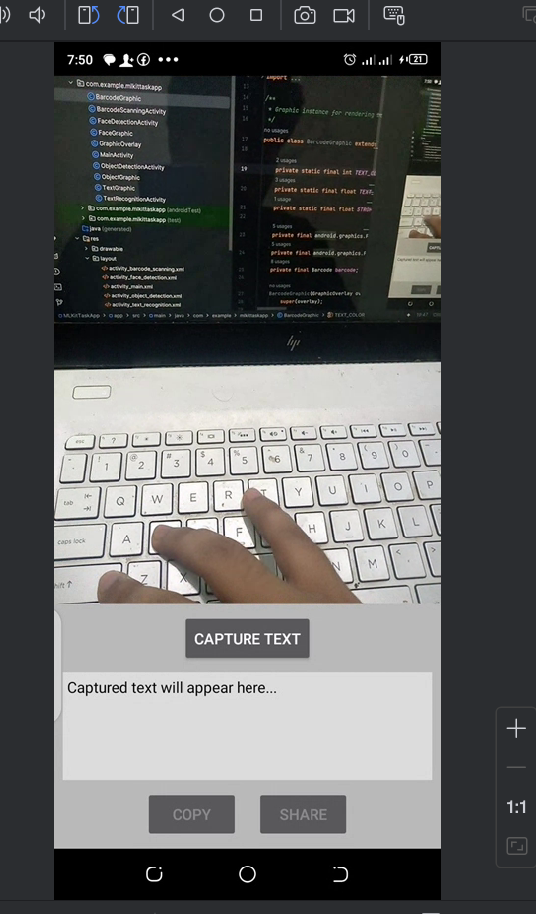
**Text\_Recognition.java:**package com.example.project;  
  
import android.Manifest;  
import android.content.ClipData;  
import android.content.ClipboardManager;  
import android.content.Context;  
import android.content.Intent;  
import android.content.pm.PackageManager;  
import android.os.Bundle;  
import android.widget.Button;  
import android.widget.TextView;  
import android.widget.Toast;  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.camera.core.Camera;  
import androidx.camera.core.CameraSelector;  
import androidx.camera.core.ImageCapture;  
import androidx.camera.core.ImageCaptureException;  
import androidx.camera.core.ImageProxy;  
import androidx.camera.core.Preview;  
import androidx.camera.lifecycle.ProcessCameraProvider;  
import androidx.camera.view.PreviewView;  
import androidx.core.app.ActivityCompat;  
import androidx.core.content.ContextCompat;  
import com.google.mlkit.vision.common.InputImage;  
import com.google.mlkit.vision.text.TextRecognition;  
import com.google.mlkit.vision.text.TextRecognizer;  
import com.google.mlkit.vision.text.latin.TextRecognizerOptions;  
import com.google.common.util.concurrent.ListenableFuture;  
import java.util.concurrent.ExecutorService;  
import java.util.concurrent.Executors;  
  
public class TextRecognitionActivity extends AppCompatActivity {  
 private PreviewView previewView;  
 private TextView textOutput;  
 private String recognizedText = "";  
 private static final int CAMERA\_PERMISSION\_CODE = 100;  
 private ImageCapture imageCapture;  
 private ExecutorService cameraExecutor;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_text\_recognition);  
  
 previewView = findViewById(R.id.camera\_preview);  
 textOutput = findViewById(R.id.text\_output);  
 Button captureButton = findViewById(R.id.capture\_button);  
 Button copyButton = findViewById(R.id.copy\_button);  
 Button shareButton = findViewById(R.id.share\_button);  
  
 cameraExecutor = Executors.newSingleThreadExecutor();  
  
 if (ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA)  
 != PackageManager.PERMISSION\_GRANTED) {  
 ActivityCompat.requestPermissions(this,  
 new String[]{Manifest.permission.CAMERA},  
 CAMERA\_PERMISSION\_CODE);  
 } else {  
 startCamera();  
 }  
  
 captureButton.setOnClickListener(v -> captureAndProcessImage());  
 copyButton.setOnClickListener(v -> copyTextToClipboard());  
 shareButton.setOnClickListener(v -> shareRecognizedText());  
 }  
  
 private void startCamera() {  
 ListenableFuture<ProcessCameraProvider> cameraProviderFuture =  
 ProcessCameraProvider.getInstance(this);  
  
 cameraProviderFuture.addListener(() -> {  
 try {  
 ProcessCameraProvider cameraProvider = cameraProviderFuture.get();  
 Preview preview = new Preview.Builder().build();  
 preview.setSurfaceProvider(previewView.getSurfaceProvider());  
  
 imageCapture = new ImageCapture.Builder()  
 .setCaptureMode(ImageCapture.CAPTURE\_MODE\_MINIMIZE\_LATENCY)  
 .build();  
  
 CameraSelector cameraSelector = new CameraSelector.Builder()  
 .requireLensFacing(CameraSelector.LENS\_FACING\_BACK)  
 .build();  
  
 cameraProvider.unbindAll();  
 cameraProvider.bindToLifecycle(  
 this, cameraSelector, preview, imageCapture);  
  
 } catch (Exception e) {  
 Toast.makeText(this, "Camera setup failed: " + e.getMessage(),  
 Toast.LENGTH\_SHORT).show();  
 }  
 }, ContextCompat.getMainExecutor(this));  
 }  
  
 private void captureAndProcessImage() {  
 if (imageCapture == null) {  
 Toast.makeText(this, "Camera not ready", Toast.LENGTH\_SHORT).show();  
 return;  
 }  
  
 imageCapture.takePicture(  
 cameraExecutor,  
 new ImageCapture.OnImageCapturedCallback() {  
 @Override  
 public void onCaptureSuccess(@NonNull ImageProxy imageProxy) {  
 try {  
 InputImage inputImage = getInputImageFromProxy(imageProxy);  
 if (inputImage != null) {  
 processImageWithMLKit(inputImage);  
 } else {  
 runOnUiThread(() -> Toast.makeText(TextRecognitionActivity.this,  
 "Failed to process image: Image data unavailable", Toast.LENGTH\_SHORT).show());  
 }  
 } finally {  
 imageProxy.close();  
 }  
 }  
  
 @Override  
 public void onError(@NonNull ImageCaptureException exception) {  
 runOnUiThread(() -> Toast.makeText(TextRecognitionActivity.this,  
 "Capture failed: " + exception.getMessage(),  
 Toast.LENGTH\_SHORT).show());  
 }  
 });  
 }  
  
 private InputImage getInputImageFromProxy(ImageProxy imageProxy) {  
 android.media.Image mediaImage = imageProxy.getImage();  
 if (mediaImage == null) {  
 return null;  
 }  
  
 try {  
 return InputImage.fromMediaImage(  
 mediaImage,  
 imageProxy.getImageInfo().getRotationDegrees()  
 );  
 } catch (IllegalArgumentException e) {  
 runOnUiThread(() -> Toast.makeText(this, "Error creating input image: " + e.getMessage(),  
 Toast.LENGTH\_SHORT).show());  
 return null;  
 }  
 }  
  
 private void processImageWithMLKit(InputImage inputImage) {  
 TextRecognizer recognizer = TextRecognition.getClient(  
 TextRecognizerOptions.DEFAULT\_OPTIONS);  
  
 recognizer.process(inputImage)  
 .addOnSuccessListener(text -> {  
 recognizedText = text.getText();  
 runOnUiThread(() -> {  
 textOutput.setText(recognizedText);  
 Toast.makeText(TextRecognitionActivity.this,  
 "Text recognized", Toast.LENGTH\_SHORT).show();  
 });  
 })  
 .addOnFailureListener(e -> {  
 recognizedText = "";  
 runOnUiThread(() -> {  
 textOutput.setText("");  
 Toast.makeText(TextRecognitionActivity.this,  
 "Recognition failed: " + e.getMessage(),  
 Toast.LENGTH\_SHORT).show();  
 });  
 });  
 }  
  
 private void copyTextToClipboard() {  
 if (recognizedText.isEmpty()) {  
 Toast.makeText(this, "No text to copy", Toast.LENGTH\_SHORT).show();  
 return;  
 }  
  
 ClipboardManager clipboard =  
 (ClipboardManager) getSystemService(Context.CLIPBOARD\_SERVICE);  
 ClipData clip = ClipData.newPlainText("Recognized Text", recognizedText);  
 clipboard.setPrimaryClip(clip);  
 Toast.makeText(this, "Text copied to clipboard", Toast.LENGTH\_SHORT).show();  
 }  
  
 private void shareRecognizedText() {  
 if (recognizedText.isEmpty()) {  
 Toast.makeText(this, "No text to share", Toast.LENGTH\_SHORT).show();  
 return;  
 }  
  
 Intent shareIntent = new Intent(Intent.ACTION\_SEND);  
 shareIntent.setType("text/plain");  
 shareIntent.putExtra(Intent.EXTRA\_TEXT, recognizedText);  
 startActivity(Intent.createChooser(shareIntent, "Share text via"));  
 }  
  
 @Override  
 public void onRequestPermissionsResult(int requestCode,  
 @NonNull String[] permissions, @NonNull int[] grantResults) {  
 super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
  
 if (requestCode == CAMERA\_PERMISSION\_CODE &&  
 grantResults.length > 0 &&  
 grantResults[0] == PackageManager.PERMISSION\_GRANTED) {  
 startCamera();  
 } else {  
 Toast.makeText(this, "Camera permission required",  
 Toast.LENGTH\_SHORT).show();  
 finish();  
 }  
 }  
  
 @Override  
 protected void onDestroy() {  
 super.onDestroy();  
 if (cameraExecutor != null) {  
 cameraExecutor.shutdown();  
 }  
 }  
}

**OUTPUT:**



A screenshot of a device

AI-generated content may be incorrect.



A screenshot of a phone

AI-generated content may be incorrect.