package com.example.opencvcam;  
  
import androidx.annotation.NonNull;  
import androidx.core.app.ActivityCompat;  
import androidx.core.content.ContextCompat;  
  
import android.Manifest;  
import android.app.Activity;  
import android.content.Intent;  
import android.content.SharedPreferences;  
import android.content.pm.PackageManager;  
import android.graphics.Color;  
import android.media.CamcorderProfile;  
import android.media.MediaRecorder;  
import android.os.Bundle;  
import android.os.Environment;  
import android.speech.RecognitionListener;  
import android.speech.RecognizerIntent;  
import android.speech.SpeechRecognizer;  
import android.util.Log;  
import android.view.MotionEvent;  
import android.view.SurfaceView;  
import android.view.View;  
import android.view.Window;  
import android.view.WindowManager;  
import android.view.textclassifier.TextClassifierEvent;  
import android.widget.EditText;  
import android.widget.ImageView;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import com.google.android.gms.tasks.OnFailureListener;  
import com.google.android.gms.tasks.OnSuccessListener;  
import com.google.firebase.ml.naturallanguage.FirebaseNaturalLanguage;  
import com.google.firebase.ml.naturallanguage.languageid.FirebaseLanguageIdentification;  
import com.google.mlkit.nl.translate.NaturalLanguageTranslateRegistrar;  
import com.google.mlkit.nl.translate.TranslateLanguage;  
import com.google.mlkit.nl.translate.Translation;  
import com.google.mlkit.nl.translate.Translator;  
import com.google.mlkit.nl.translate.TranslatorOptions;  
  
import org.opencv.android.BaseLoaderCallback;  
import org.opencv.android.CameraBridgeViewBase;  
import org.opencv.android.LoaderCallbackInterface;  
import org.opencv.android.OpenCVLoader;  
import org.opencv.core.Core;  
import org.opencv.core.CvType;  
import org.opencv.core.Mat;  
import org.opencv.core.Point;  
import org.opencv.core.Scalar;  
import org.opencv.imgcodecs.Imgcodecs;  
import org.opencv.imgproc.Imgproc;  
  
import java.io.File;  
import java.io.IOException;  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.Locale;  
  
public class CameraActivity extends Activity implements CameraBridgeViewBase.CvCameraViewListener2 {  
 private static final String *TAG*="MainActivity";  
 private Mat mRgba;  
 private Mat mGrey;  
 private CameraBridgeViewBase mOpenCvCameraView;  
 private ImageView flip\_camera;  
 // integer that represents camera  
 // 0 = back camera  
 // 1 = front camera  
 // initially it will start with back camera  
 private int mCameraId = 0;  
 private ImageView take\_picture\_button;  
 private int take\_image = 0;  
 private ImageView image\_gallery\_icon;  
 private MediaRecorder recorder;  
 private ImageView video\_camera\_button;  
 private int video\_or\_photo = 0;  
 // if 1 video mode  
 // if 0 photo mode  
 private int take\_video\_or\_not = 0;  
 private SpeechRecognizer speechRecognizer;  
 private String speechText ;  
 //private TextView subtitles;  
 String option;  
 Translator englishUrdu;  
 Translator urduEnglish;  
 String sourceLang;  
 //String TransUrdu;  
 //String TransEng;  
  
   
  
 private BaseLoaderCallback mLoaderCallback = new BaseLoaderCallback(this) {  
 @Override  
 public void onManagerConnected(int status) {  
 switch(status) {  
 case LoaderCallbackInterface  
 .*SUCCESS*:{  
 Log.*i*(*TAG*,"OpenCV is loaded");  
 mOpenCvCameraView.enableView();  
 }  
 default:{  
 super.onManagerConnected(status);  
 }  
 break;  
 }  
 }  
 };  
  
 public CameraActivity(){  
 Log.*i*(*TAG*,"Instantiated new "+this.getClass());  
 }  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 requestWindowFeature(Window.*FEATURE\_NO\_TITLE*);  
 getWindow().addFlags(WindowManager.LayoutParams.*FLAG\_KEEP\_SCREEN\_ON*);  
  
 int MY\_PERMISSIONS\_REQUEST\_CAMERA = 0;  
  
 if (ContextCompat.*checkSelfPermission*(CameraActivity.this, Manifest.permission.*CAMERA*) == PackageManager.*PERMISSION\_DENIED*) {  
 ActivityCompat.*requestPermissions*(CameraActivity.this, new String[] {Manifest.permission.*CAMERA*}, MY\_PERMISSIONS\_REQUEST\_CAMERA);  
 }  
  
 if (ContextCompat.*checkSelfPermission*(CameraActivity.this, Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*) == PackageManager.*PERMISSION\_DENIED*) {  
 ActivityCompat.*requestPermissions*(CameraActivity.this, new String[] {Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*}, MY\_PERMISSIONS\_REQUEST\_CAMERA);  
 }  
  
 if (ContextCompat.*checkSelfPermission*(CameraActivity.this, Manifest.permission.*READ\_EXTERNAL\_STORAGE*) == PackageManager.*PERMISSION\_DENIED*) {  
 ActivityCompat.*requestPermissions*(CameraActivity.this, new String[] {Manifest.permission.*READ\_EXTERNAL\_STORAGE*}, MY\_PERMISSIONS\_REQUEST\_CAMERA);  
 }  
  
 if (ContextCompat.*checkSelfPermission*(CameraActivity.this, Manifest.permission.*RECORD\_AUDIO*) == PackageManager.*PERMISSION\_DENIED*) {  
 ActivityCompat.*requestPermissions*(CameraActivity.this, new String[] {Manifest.permission.*RECORD\_AUDIO*}, MY\_PERMISSIONS\_REQUEST\_CAMERA);  
 }  
  
 if (ContextCompat.*checkSelfPermission*(CameraActivity.this, Manifest.permission.*INTERNET*) == PackageManager.*PERMISSION\_DENIED*) {  
 ActivityCompat.*requestPermissions*(CameraActivity.this, new String[] {Manifest.permission.*INTERNET*}, MY\_PERMISSIONS\_REQUEST\_CAMERA);  
 }  
  
 setContentView(R.layout.*activity\_camera*);  
  
 mOpenCvCameraView = (CameraBridgeViewBase) findViewById(R.id.*frame\_surface*);  
 mOpenCvCameraView.setVisibility(SurfaceView.*VISIBLE*);  
 mOpenCvCameraView.setCvCameraViewListener(this);  
 mOpenCvCameraView.enableFpsMeter();  
  
 flip\_camera = findViewById(R.id.*flip\_camera*);  
 flip\_camera.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 swapCamera();  
 }  
 });  
  
  
  
 image\_gallery\_icon = findViewById(R.id.*image\_gallery\_icon*);  
 image\_gallery\_icon.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 startActivity(new Intent(CameraActivity.this,GalleryActivity.class).addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TASK* | Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP*));  
 }  
 });  
  
 recorder = new MediaRecorder();  
 video\_camera\_button = findViewById(R.id.*video\_camera\_button*);  
 video\_camera\_button.setOnTouchListener(new View.OnTouchListener() {  
 @Override  
 public boolean onTouch(View v, MotionEvent event) {  
 if (event.getAction() == MotionEvent.*ACTION\_DOWN*) {  
 video\_camera\_button.setColorFilter(Color.*DKGRAY*);  
 return true;  
 }  
  
 if (event.getAction() == MotionEvent.*ACTION\_UP*) {  
 video\_camera\_button.setColorFilter(Color.*WHITE*);  
 if (video\_or\_photo == 0) {  
 take\_picture\_button.setImageResource(R.drawable.*circle\_button*);  
 take\_picture\_button.setColorFilter(Color.*WHITE*);  
 video\_or\_photo = 1;  
 }  
  
 else {  
 take\_picture\_button.setImageResource(R.drawable.*take\_picture\_icon*);  
 video\_or\_photo = 0;  
 }  
 return true;  
 }  
 return false;  
 }  
 });  
  
 take\_picture\_button = findViewById(R.id.*take\_picture\_button*);  
 final EditText subtitles = findViewById(R.id.*subtitles*);  
  
 final Intent speechIntent = new Intent(RecognizerIntent.*ACTION\_RECOGNIZE\_SPEECH*);  
  
 speechIntent.putExtra("android.speech.extra.DICTATION\_MODE", true);  
 speechIntent.putExtra(RecognizerIntent.*EXTRA\_LANGUAGE\_MODEL*, RecognizerIntent.*LANGUAGE\_MODEL\_FREE\_FORM*);  
 speechIntent.putExtra(RecognizerIntent.*EXTRA\_PARTIAL\_RESULTS*, true);  
 speechIntent.putExtra(RecognizerIntent.*EXTRA\_SPEECH\_INPUT\_MINIMUM\_LENGTH\_MILLIS*, 9000);  
 speechIntent.putExtra(RecognizerIntent.*EXTRA\_LANGUAGE*, Locale.*getDefault*());  
  
 speechRecognizer = SpeechRecognizer.*createSpeechRecognizer*(getApplicationContext());  
  
 speechRecognizer.setRecognitionListener(new RecognitionListener() {  
  
 public void start(View view) {  
 speechRecognizer.startListening(speechIntent);  
 }  
  
 public void stop(View view) {  
 speechRecognizer.stopListening();  
 }  
  
 @Override  
 public void onReadyForSpeech(Bundle params) {  
  
 }  
  
 @Override  
 public void onBeginningOfSpeech() {  
  
 }  
  
 @Override  
 public void onRmsChanged(float rmsdB) {  
  
 }  
  
 @Override  
 public void onBufferReceived(byte[] buffer) {  
  
 }  
  
 @Override  
 public void onEndOfSpeech() {  
  
 }  
  
 @Override  
 public void onError(int error) {  
  
 }  
  
 @Override  
 public void onResults(Bundle results) {  
 take\_picture\_button.setImageResource(R.drawable.*circle\_button*);  
 ArrayList<String> arrayList = results.getStringArrayList(SpeechRecognizer.*RESULTS\_RECOGNITION*);  
 speechText = arrayList.get(0);  
 SharedPreferences sharedPreferences = getSharedPreferences("myOption", *MODE\_PRIVATE*);  
 option = sharedPreferences.getString("option","");  
 detectLanguage(speechText);  
 if (sourceLang.compareTo("EN") >= 1){  
 switch (option) {  
 case "English": {  
 subtitles.setText(speechText);  
 break;  
 }  
 case "Urdu": {  
 prepareModel();  
 break;  
 }  
 }  
 speechRecognizer.startListening(speechIntent);  
 }  
 else if (sourceLang.compareTo("UR") >= 1){  
 switch (option) {  
 case "Urdu": {  
 subtitles.setText(speechText);  
 break;  
 }  
 case "English": {  
 prepareModel();  
 break;  
 }  
 }  
 speechRecognizer.startListening(speechIntent);  
 }  
  
 }  
  
 @Override  
 public void onPartialResults(Bundle partialResults) {  
  
 }  
  
 @Override  
 public void onEvent(int eventType, Bundle params) {  
  
 }  
 });  
  
 take\_picture\_button.setOnTouchListener(new View.OnTouchListener() {  
 @Override  
 public boolean onTouch(View v, MotionEvent event) {  
 if (event.getAction() == MotionEvent.*ACTION\_DOWN*) {  
 if (video\_or\_photo == 0) {  
 if (take\_image == 0) {  
 take\_picture\_button.setColorFilter(Color.*DKGRAY*);  
 }  
 }  
  
 return true;  
 }  
  
 if (event.getAction() == MotionEvent.*ACTION\_UP*) {  
 if (video\_or\_photo == 1){  
 if (take\_video\_or\_not == 0) {  
 try {  
 File folder = new File(Environment.*getExternalStorageDirectory*().getPath()+"/LiveSubtitles");  
 boolean success = true;  
 if (!folder.exists()) {  
 success = folder.mkdirs();  
 }  
 take\_picture\_button.setImageResource(R.drawable.*circle\_button*);  
 take\_picture\_button.setColorFilter(Color.*RED*);  
 recorder.setAudioSource(MediaRecorder.AudioSource.*VOICE\_RECOGNITION*);  
 recorder.setVideoSource(MediaRecorder.VideoSource.*SURFACE*);  
 CamcorderProfile camcorderProfile = CamcorderProfile.*get*(CamcorderProfile.*QUALITY\_HIGH*);  
 recorder.setProfile(camcorderProfile);  
  
 SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd\_HH-mm-ss");  
 String current\_Date\_and\_Time = sdf.format(new Date());  
 String filename = Environment.*getExternalStorageDirectory*().getPath()+"/LiveSubtitles/"+current\_Date\_and\_Time+".mp4";  
 recorder.setOutputFile(filename);  
 recorder.setVideoSize(480, 720);  
 recorder.prepare();  
 mOpenCvCameraView.setRecorder(recorder);  
 recorder.start();  
 speechRecognizer.startListening(speechIntent);  
  
 }  
 catch(IOException e) {  
 e.printStackTrace();  
 }  
 take\_video\_or\_not = 1;  
 }  
  
 else {  
 take\_picture\_button.setImageResource(R.drawable.*circle\_button*);  
 take\_picture\_button.setColorFilter(Color.*WHITE*);  
 mOpenCvCameraView.setRecorder(null);  
 speechRecognizer.stopListening();  
 recorder.stop();  
  
  
 try {  
 Thread.*sleep*(1000);  
 }  
 catch (InterruptedException e) {  
 throw new RuntimeException(e);  
 }  
 take\_video\_or\_not = 0;  
 }  
 }  
  
 else {  
 take\_picture\_button.setColorFilter(Color.*WHITE*);  
 if (take\_image == 0) {  
 take\_image = 1;  
 }  
 else {  
 take\_image = 0;  
 }  
 }  
  
 return true;  
 }  
 return false;  
 }  
  
 });  
 }  
  
 private void detectLanguage(String speechText) {  
 FirebaseLanguageIdentification languageIdentifier = FirebaseNaturalLanguage.*getInstance*().getLanguageIdentification();  
  
 // adding method to detect language using identify language method.  
 sourceLang= languageIdentifier.identifyLanguage(speechText).toString();  
  
 }  
  
  
 private void prepareModel() {  
 switch (option) {  
 case "Urdu": {  
 TranslatorOptions option1 = new TranslatorOptions.Builder().setSourceLanguage(TranslateLanguage.*ENGLISH*).setTargetLanguage(TranslateLanguage.*URDU*).build();  
 englishUrdu = Translation.*getClient*(option1);  
 englishUrdu.downloadModelIfNeeded();  
 final EditText subtitles = findViewById(R.id.*subtitles*);  
 englishUrdu.translate(speechText).addOnSuccessListener(new OnSuccessListener<String>() {  
 @Override  
 public void onSuccess(String s) {  
 subtitles.setText(s);  
 }  
 });  
  
 }  
 }  
 switch (option) {  
 case "English": {  
 TranslatorOptions option2 = new TranslatorOptions.Builder().setSourceLanguage(TranslateLanguage.*URDU*).setTargetLanguage(TranslateLanguage.*ENGLISH*).build();  
 urduEnglish = Translation.*getClient*(option2);  
 urduEnglish.downloadModelIfNeeded();  
 final EditText subtitles = findViewById(R.id.*subtitles*);  
 urduEnglish.translate(speechText).addOnSuccessListener(new OnSuccessListener<String>() {  
 @Override  
 public void onSuccess(String s) {  
 subtitles.setText(s);  
 }  
 });  
 }  
 }  
 }  
  
  
 private void swapCamera() {  
 mCameraId = mCameraId^1;  
 mOpenCvCameraView.disableView();  
 mOpenCvCameraView.setCameraIndex(mCameraId);  
 mOpenCvCameraView.enableView();  
 }  
  
 @Override  
 protected void onResume() {  
 super.onResume();  
 if(OpenCVLoader.*initDebug*()){  
 Log.*d*(*TAG*, "OpenCV initialization is done");  
 mLoaderCallback.onManagerConnected(LoaderCallbackInterface.*SUCCESS*);  
 }  
 else{  
 Log.*d*(*TAG*, "OpenCV is not loaded. Try again");  
 OpenCVLoader.*initAsync*(OpenCVLoader.*OPENCV\_VERSION\_3\_4\_0*,this,mLoaderCallback);  
 }  
 }  
  
 @Override  
 protected void onPause() {  
 super.onPause();  
 if (mOpenCvCameraView != null){  
 mOpenCvCameraView.disableView();  
 }  
 }  
  
 public void onDestroy() {  
 super.onDestroy();  
 if (mOpenCvCameraView != null){  
 mOpenCvCameraView.disableView();  
 }  
 speechRecognizer.destroy();  
 }  
  
 public void onCameraViewStarted(int width, int height) {  
 mRgba = new Mat(height, width, CvType.*CV\_8UC4*);  
 mGrey = new Mat(height, width, CvType.*CV\_8UC1*);  
 }  
  
 public void onCameraViewStopped() {  
 mRgba.release();  
 }  
  
 public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame) {  
 mRgba = inputFrame.rgba();  
 mGrey = inputFrame.gray();  
  
 if (mCameraId == 1) {  
 Core.*flip*(mRgba, mRgba, 1);  
 Core.*flip*(mGrey, mGrey, 1);  
 }  
  
 take\_image = take\_picture\_function\_rgb(take\_image, mRgba);  
  
 return mRgba;  
 }  
  
  
 private int take\_picture\_function\_rgb(int take\_image, Mat mRgba) {  
 if (take\_image==1) {  
 Mat save\_mat = new Mat();  
 Core.*flip*(mRgba.t(), save\_mat,1);  
 Imgproc.*cvtColor*(save\_mat, save\_mat, Imgproc.*COLOR\_RGBA2BGRA*);  
 File folder = new File(Environment.*getExternalStorageDirectory*().getPath()+"/LiveSubtitles");  
  
 boolean success = true;  
  
 if (!folder.exists()) {  
 success = folder.mkdirs();  
 }  
  
 SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd\_HH-mm-ss");  
 String currentDateAndTime = sdf.format(new Date());  
 String fileName = Environment.*getExternalStorageDirectory*().getPath()+"/LiveSubtitles/"+currentDateAndTime+".jpg";  
 Imgcodecs.*imwrite*(fileName, save\_mat);  
 take\_image = 0;  
 }  
  
 return take\_image;  
 }  
}