name: my\_new\_app

description: "A new Flutter project."

# The following line prevents the package from being accidentally published to

# pub.dev using `flutter pub publish`. This is preferred for private packages.

publish\_to: 'none' # Remove this line if you wish to publish to pub.dev

# The following defines the version and build number for your application.

# A version number is three numbers separated by dots, like 1.2.43

# followed by an optional build number separated by a +.

# Both the version and the builder number may be overridden in flutter

# build by specifying --build-name and --build-number, respectively.

# In Android, build-name is used as versionName while build-number used as versionCode.

# Read more about Android versioning at https://developer.android.com/studio/publish/versioning

# In iOS, build-name is used as CFBundleShortVersionString while build-number is used as CFBundleVersion.

# Read more about iOS versioning at

# https://developer.apple.com/library/archive/documentation/General/Reference/InfoPlistKeyReference/Articles/CoreFoundationKeys.html

# In Windows, build-name is used as the major, minor, and patch parts

# of the product and file versions while build-number is used as the build suffix.

version: 1.0.0+1

environment:

  sdk: '>=3.4.3 <4.0.0'

# Dependencies specify other packages that your package needs in order to work.

# To automatically upgrade your package dependencies to the latest versions

# consider running `flutter pub upgrade --major-versions`. Alternatively,

# dependencies can be manually updated by changing the version numbers below to

# the latest version available on pub.dev. To see which dependencies have newer

# versions available, run `flutter pub outdated`.

dependencies:

  flutter:

    sdk: flutter

  flutter\_native\_splash: ^2.4.0

  # The following adds the Cupertino Icons font to your application.

  # Use with the CupertinoIcons class for iOS style icons.

  cupertino\_icons: ^1.0.6

  image\_picker: ^1.1.2

  tflite\_flutter: ^0.10.4

  google\_fonts: ^6.2.1

dev\_dependencies:

  flutter\_test:

    sdk: flutter

  # The "flutter\_lints" package below contains a set of recommended lints to

  # encourage good coding practices. The lint set provided by the package is

  # activated in the `analysis\_options.yaml` file located at the root of your

  # package. See that file for information about deactivating specific lint

  # rules and activating additional ones.

  flutter\_lints: ^3.0.0

# For information on the generic Dart part of this file, see the

# following page: https://dart.dev/tools/pub/pubspec

# The following section is specific to Flutter packages.

flutter:

  # The following line ensures that the Material Icons font is

  # included with your application, so that you can use the icons in

  # the material Icons class.

  uses-material-design: true

  # To add assets to your application, add an assets section, like this:

  assets:

    - assets/Grape icon2.png

    - assets/grape\_disease\_detection\_model.tflite

  # An image asset can refer to one or more resolution-specific "variants", see

  # https://flutter.dev/assets-and-images/#resolution-aware

  # For details regarding adding assets from package dependencies, see

  # https://flutter.dev/assets-and-images/#from-packages

  # To add custom fonts to your application, add a fonts section here,

  # in this "flutter" section. Each entry in this list should have a

  # "family" key with the font family name, and a "fonts" key with a

  # list giving the asset and other descriptors for the font. For

  # example:

  # fonts:

  #   - family: Schyler

  #     fonts:

  #       - asset: fonts/Schyler-Regular.ttf

  #       - asset: fonts/Schyler-Italic.ttf

  #         style: italic

  #   - family: Trajan Pro

  #     fonts:

  #       - asset: fonts/TrajanPro.ttf

  #       - asset: fonts/TrajanPro\_Bold.ttf

  #         weight: 700

  #

  # For details regarding fonts from package dependencies,

  # see https://flutter.dev/custom-fonts/#from-packages

flutter\_native\_splash:

  # This package generates native code to customize Flutter's default white native splash screen

  # with background color and splash image.

  # Customize the parameters below, and run the following command in the terminal:

  # dart run flutter\_native\_splash:create

  # To restore Flutter's default white splash screen, run the following command in the terminal:

  # dart run flutter\_native\_splash:remove

  # IMPORTANT NOTE: These parameter do not affect the configuration of Android 12 and later, which

  # handle splash screens differently that prior versions of Android.  Android 12 and later must be

  # configured specifically in the android\_12 section below.

  # color or background\_image is the only required parameter.  Use color to set the background

  # of your splash screen to a solid color.  Use background\_image to set the background of your

  # splash screen to a png image.  This is useful for gradients. The image will be stretch to the

  # size of the app. Only one parameter can be used, color and background\_image cannot both be set.

  color: "#42a5f5"

  background\_image: ".idea/libraries/assets/Grape icon2.png"

  # Optional parameters are listed below.  To enable a parameter, uncomment the line by removing

  # the leading # character.

  # The image parameter allows you to specify an image used in the splash screen.  It must be a

  # png file and should be sized for 4x pixel density.

  #image: assets/splash.png

  # The branding property allows you to specify an image used as branding in the splash screen.

  # It must be a png file. It is supported for Android, iOS and the Web.  For Android 12,

  # see the Android 12 section below.

  #branding: assets/dart.png

  # To position the branding image at the bottom of the screen you can use bottom, bottomRight,

  # and bottomLeft. The default values is bottom if not specified or specified something else.

  #branding\_mode: bottom

  # The color\_dark, background\_image\_dark, image\_dark, branding\_dark are parameters that set the background

  # and image when the device is in dark mode. If they are not specified, the app will use the

  # parameters from above. If the image\_dark parameter is specified, color\_dark or

  # background\_image\_dark must be specified.  color\_dark and background\_image\_dark cannot both be

  # set.

  #color\_dark: "#042a49"

  #background\_image\_dark: "assets/dark-background.png"

  #image\_dark: assets/splash-invert.png

  #branding\_dark: assets/dart\_dark.png

  # From Android 12 onwards, the splash screen is handled differently than in previous versions.

  # Please visit https://developer.android.com/guide/topics/ui/splash-screen

  # Following are specific parameters for Android 12+.

  android\_12:

    # The image parameter sets the splash screen icon image.  If this parameter is not specified,

    # the app's launcher icon will be used instead.

    # Please note that the splash screen will be clipped to a circle on the center of the screen.

    # App icon with an icon background: This should be 960×960 pixels, and fit within a circle

    # 640 pixels in diameter.

    # App icon without an icon background: This should be 1152×1152 pixels, and fit within a circle

    # 768 pixels in diameter.

    #image: assets/android12splash.png

    # Splash screen background color.

    #color: "#42a5f5"

    # App icon background color.

    #icon\_background\_color: "#111111"

    # The branding property allows you to specify an image used as branding in the splash screen.

    #branding: assets/dart.png

    # The image\_dark, color\_dark, icon\_background\_color\_dark, and branding\_dark set values that

    # apply when the device is in dark mode. If they are not specified, the app will use the

    # parameters from above.

    #image\_dark: assets/android12splash-invert.png

    #color\_dark: "#042a49"

    #icon\_background\_color\_dark: "#eeeeee"

  # The android, ios and web parameters can be used to disable generating a splash screen on a given

  # platform.

  #android: false

  #ios: false

  #web: false

  # Platform specific images can be specified with the following parameters, which will override

  # the respective parameter.  You may specify all, selected, or none of these parameters:

  #color\_android: "#42a5f5"

  #color\_dark\_android: "#042a49"

  #color\_ios: "#42a5f5"

  #color\_dark\_ios: "#042a49"

  #color\_web: "#42a5f5"

  #color\_dark\_web: "#042a49"

  #image\_android: assets/splash-android.png

  #image\_dark\_android: assets/splash-invert-android.png

  #image\_ios: assets/splash-ios.png

  #image\_dark\_ios: assets/splash-invert-ios.png

  #image\_web: assets/splash-web.gif

  #image\_dark\_web: assets/splash-invert-web.gif

  #background\_image\_android: "assets/background-android.png"

  #background\_image\_dark\_android: "assets/dark-background-android.png"

  #background\_image\_ios: "assets/background-ios.png"

  #background\_image\_dark\_ios: "assets/dark-background-ios.png"

  #background\_image\_web: "assets/background-web.png"

  #background\_image\_dark\_web: "assets/dark-background-web.png"

  #branding\_android: assets/brand-android.png

  #branding\_dark\_android: assets/dart\_dark-android.png

  #branding\_ios: assets/brand-ios.png

  #branding\_dark\_ios: assets/dart\_dark-ios.png

  #branding\_web: assets/brand-web.gif

  #branding\_dark\_web: assets/dart\_dark-web.gif

  # The position of the splash image can be set with android\_gravity, ios\_content\_mode, and

  # web\_image\_mode parameters.  All default to center.

  #

  # android\_gravity can be one of the following Android Gravity (see

  # https://developer.android.com/reference/android/view/Gravity): bottom, center,

  # center\_horizontal, center\_vertical, clip\_horizontal, clip\_vertical, end, fill, fill\_horizontal,

  # fill\_vertical, left, right, start, or top.

  #android\_gravity: center

  #

  # ios\_content\_mode can be one of the following iOS UIView.ContentMode (see

  # https://developer.apple.com/documentation/uikit/uiview/contentmode): scaleToFill,

  # scaleAspectFit, scaleAspectFill, center, top, bottom, left, right, topLeft, topRight,

  # bottomLeft, or bottomRight.

  #ios\_content\_mode: center

  #

  # web\_image\_mode can be one of the following modes: center, contain, stretch, and cover.

  #web\_image\_mode: center

  # The screen orientation can be set in Android with the android\_screen\_orientation parameter.

  # Valid parameters can be found here:

  # https://developer.android.com/guide/topics/manifest/activity-element#screen

  #android\_screen\_orientation: sensorLandscape

  # To hide the notification bar, use the fullscreen parameter.  Has no effect in web since web

  # has no notification bar.  Defaults to false.

  # NOTE: Unlike Android, iOS will not automatically show the notification bar when the app loads.

  #       To show the notification bar, add the following code to your Flutter app:

  #       WidgetsFlutterBinding.ensureInitialized();

  #       SystemChrome.setEnabledSystemUIMode(SystemUiMode.manual, overlays: [SystemUiOverlay.bottom, SystemUiOverlay.top], );

  #fullscreen: true

  # If you have changed the name(s) of your info.plist file(s), you can specify the filename(s)

  # with the info\_plist\_files parameter.  Remove only the # characters in the three lines below,

  # do not remove any spaces:

  #info\_plist\_files:

  #  - 'ios/Runner/Info-Debug.plist'

  #  - 'ios/Runner/Info-Release.plist'

import 'package:flutter/material.dart';

import 'package:google\_fonts/google\_fonts.dart';

import 'package:image\_picker/image\_picker.dart';

import 'package:tflite\_flutter/tflite\_flutter.dart';

import 'dart:io';

import 'dart:typed\_data';

import 'package:image/image.dart' as img;

class MySplash extends StatefulWidget {

  @override

  \_MySplashState createState() => \_MySplashState();

}

class \_MySplashState extends State<MySplash> {

  File? \_image;

  String? \_diseaseResult;

  Interpreter? \_interpreter;

  List<String> labels = ["ESCA", "Leaf Blight", "Black rot", "Healthy"];

  @override

  void initState() {

    super.initState();

    loadModel();

  }

  Future<void> loadModel() async {

    try {

      \_interpreter = await Interpreter.fromAsset(

          'assets/grape\_disease\_detection\_model.tflite');

      print("Model loaded successfully");

    } catch (e) {

      print("Failed to load model: $e");

    }

  }

  Future<void> \_pickImage(ImageSource source) async {

    final picker = ImagePicker();

    final pickedFile = await picker.pickImage(source: source);

    if (pickedFile != null) {

      setState(() {

        \_image = File(pickedFile.path);

        \_diseaseResult = null; // Reset output when a new image is picked

      });

    }

  }

  Uint8List \_preprocessImage(File imageFile) {

    final originalImage =

        img.decodeImage(imageFile.readAsBytesSync())!;

    final resizedImage =

        img.copyResize(originalImage, width: 300, height: 300);

    return resizedImage.getBytes();

  }

  Future<void> \_detectDiseases() async {

    print("Detect Diseases button pressed.");

    if (\_image == null) {

      print("No image selected.");

      return;

    }

    if (\_interpreter == null) {

      print("Model interpreter not initialized.");

      return;

    }

 //   try {

      final input = \_preprocessImage(\_image!);

      final output = List.filled(labels.length, 0).reshape([1, labels.length]);

      \_interpreter!.run(input, output);

      // Debug print the output tensor for verification

      print("Output tensor: $output");

      setState(() {

        final detectedIndex = output[0].indexWhere((element) => element == output[0].reduce((a, b) => a > b ? a : b));

        final detectedLabel = labels[detectedIndex];

        if (detectedLabel == "Healthy") {

          \_diseaseResult = "The leaf is healthy.";

        } else {

          \_diseaseResult = "The leaf is affected by $detectedLabel.";

        }

      });

    // // } catch (e) {

    //   print("Failed to run model: $e");

    //   setState(() {

    //     \_diseaseResult = "Failed to detect disease.";

    //   });

   // }

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text(

          'LEAF HEALTH CHECKER',

          style: GoogleFonts.poppins(

            textStyle: TextStyle(

              color: Colors.white,

              fontSize: 22,

              fontWeight: FontWeight.bold,

            ),

          ),

        ),

        centerTitle: true,

        backgroundColor: Colors.black,

      ),

      body: Stack(

        children: [

          Container(

            width: double.infinity,

            height: double.infinity,

            decoration: BoxDecoration(

              gradient: LinearGradient(

                begin: Alignment.topCenter,

                end: Alignment.bottomCenter,

                colors: [Color(0xFF7B4397), Color(0xFFDC2430)], // Grape violet gradient

              ),

            ),

            child: Column(

              mainAxisAlignment: MainAxisAlignment.center,

              children: <Widget>[

                \_image != null

                    ? Image.file(

                        \_image!,

                        width: 200,

                        height: 200,

                      )

                    : Image.asset(

                        'assets/Grape icon2.png',

                        width: 200,

                        height: 200,

                      ),

                SizedBox(height: 50),

                \_buildButton(

                  context,

                  'Capture Image',

                  Icons.camera\_alt,

                  () {

                    \_pickImage(ImageSource.camera);

                  },

                  Colors.purpleAccent,

                ),

                SizedBox(height: 20),

                \_buildButton(

                  context,

                  'Browse Gallery',

                  Icons.photo\_library,

                  () {

                    \_pickImage(ImageSource.gallery);

                  },

                  Colors.deepPurpleAccent,

                ),

                SizedBox(height: 20),

                \_buildButton(

                  context,

                  'Detect Diseases',

                  Icons.local\_hospital,

                  \_detectDiseases,

                  Colors.pinkAccent,

                ),

                SizedBox(height: 20),

                \_diseaseResult != null

                    ? Text(

                        \_diseaseResult!,

                        style: TextStyle(

                          color: Colors.white,

                          fontSize: 18,

                        ),

                      )

                    : Container(),

              ],

            ),

          ),

          // You can place other widgets or animations here if needed

        ],

      ),

    );

  }

  Widget \_buildButton(BuildContext context, String text, IconData icon,

      VoidCallback onPressed, Color color) {

    return ElevatedButton.icon(

      style: ElevatedButton.styleFrom(

        backgroundColor: color,

        shape: RoundedRectangleBorder(

          borderRadius: BorderRadius.circular(20.0),

        ),

        padding: EdgeInsets.symmetric(horizontal: 30, vertical: 15),

      ),

      icon: Icon(icon, size: 30, color: Colors.white), // Ensure icon color is white for visibility

      label: Text(

        text,

        style: TextStyle(

            fontSize: 18, color: Colors.white), // Ensure text color is white for visibility

      ),

      onPressed: onPressed,

    );

  }

  @override

  void dispose() {

    \_interpreter?.close();

    super.dispose();

  }

}

void main() {

  runApp(MaterialApp(

    title: 'Leaf Health Checker',

    theme: ThemeData(

      primarySwatch: Colors.purple,

      visualDensity: VisualDensity.adaptivePlatformDensity,

    ),

    home: MySplash(),

  ));

}