# Dart code for Capturing Image from Camera and Picking Image from Gallery:

class Screen1 extends StatefulWidget { @override

\_Screen1State createState() => new \_Screen1State();

}

final cropKey = GlobalKey<CropState>();

class \_Screen1State extends State<Screen1> with AutomaticKeepAliveClientMixin<Screen1> {

@override

bool get wantKeepAlive => true; String enDisease;

List data; List lang;

String disease;

Future getImageGallery() async {

var image = await ImagePicker.pickImage(source: ImageSource.gallery);

recognizeImage(image);

\_cropImage(image);

setState(() {

// \_image = image;

});

}

Future getImageCamera() async{

File image=await ImagePicker.pickImage(source: ImageSource.camera);

\_cropImage(image);

//recognizeImage(image); setState((){

// \_image=image;

});

}

Future \_cropImage(File imageFile) async {

File croppedFile = await ImageCropper.cropImage( sourcePath: imageFile.path,

ratioX: 1.0,

ratioY: 1.0,

maxWidth: 224,

maxHeight: 224,

);

# Dart code to Load tflite model and Recognizing the Disease in Leaves :

void initState() { super.initState(); loadModel();

}

Future loadModel() async { try {

String res = await Tflite.loadModel( model: "assets/Disease2.tflite", labels: "assets/Disease2.txt",

);

print(res);

} on PlatformException { print('Failed to load model.');

}

}

Uint8List imageToByteList(

img.Image image, int inputSize, double mean, double std) { var convertedBytes = Float32List(1 \* inputSize \* inputSize \* 3); var buffer = Float32List.view(convertedBytes.buffer);

int pixelIndex = 0;

for (var i = 0; i < inputSize; i++) { for (var j = 0; j < inputSize; j++) {

var pixel = image.getPixel(i, j);

buffer[pixelIndex++] = (((pixel >> 16) & 0xFF) - mean) / std; buffer[pixelIndex++] = (((pixel >> 8) & 0xFF) - mean) / std; buffer[pixelIndex++] = (((pixel) & 0xFF) - mean) / std;

}

}

return convertedBytes.buffer.asUint8List();

}

Future recognizeImage(File image) async {

var recognitions = await Tflite.runModelOnImage( path: image.path,

numResults: 6,

threshold: 0.05,

imageMean: 127.5,

imageStd: 127.5,

);

setState(() {

\_recognitions = recognitions;

});

}

Future recognizeImageBinary(File image) async {

var imageBytes = (await rootBundle.load(image.path)).buffer; img.Image oriImage = img.decodeJpg(imageBytes.asUint8List()); img.Image resizedImage = img.copyResize(oriImage, 224, 224); var recognitions = await Tflite.runModelOnBinary(

binary: imageToByteList(resizedImage, 224, 127.5, 127.5),

numResults: 6,

threshold: 0.05,

);

print(recognitions); setState(() {

\_recognitions = recognitions;

});

}

recognizeImage(croppedFile); enDisease= \_recognitions[0]["label"]; if(enDisease=='2')

{

showDialog(context: context, builder:(BuildContext context)

{

return AlertDialog(

shape: RoundedRectangleBorder(borderRadius: BorderRadius.all(Radius.circular(15))), content: Container(

height: 100,

child: Center(child: new Text( "Healthy leaf. No Diseases found ", style: TextStyle(fontSize: 18),),),

),

);

},

);

}

else if(enDisease==null)

{

showDialog(context: context, builder:(BuildContext context)

{

return AlertDialog(

shape: RoundedRectangleBorder(borderRadius: BorderRadius.all(Radius.circular(15))), content: Container(

height: 100,

child: Center(child: new Text(

"Dataset Not found. Please refer in Text section", style: TextStyle(fontSize: 18),),),

),

);

},

);

}

if(enDisease=='0' || enDisease=='1')

{

showDialog(context: context, builder:(BuildContext context)

{

return AlertDialog(

shape: RoundedRectangleBorder(borderRadius: BorderRadius.all(Radius.circular(15))), content: Container(

height: 100,

child: Center(child: new Text( "Disease Not available..",

style: TextStyle(fontSize: 18),),),

),

);

},

);

}

else {

data = Lib.getData(language[0]);

lang = Lib.getData(language[\_lanIndex]);

SecondRoute s = new SecondRoute(en: data, data: lang); disease = lang[Lib.getIndex(enDisease,

'dis\_name')[0]]['dis\_name']; print(data); print(disease); print(enDisease);

s.newTaskModalBottomSheet1(context, enDisease, disease);

}

}

# Dart code to Load the plant disease library and display the Diseases and the respective images for the crop :

class ListTutorial extends StatelessWidget { final String sip;

ListTutorial ({ this.sip}); @override

Widget build(BuildContext context) { return new Container(

height: 75, child:Row( children:[

new Expanded( child:Container(

height: 200.0,

width: 120.0, child: Column(

children:[ SizedBox(height: 5,),

Chip(

15),),

label: Text(sip,style: TextStyle(color: Colors.white,fontSize:

backgroundColor: Colors.purple[400],

padding: const EdgeInsets.only(left: 20.0,right: 20.0,top:

20.0,bottom: 20.0),

),],),

),),],),

); }

}

class ListTutorial1 extends StatelessWidget { final String sipp;

ListTutorial1 ({ this.sipp}); @override

Widget build(BuildContext context) { return new Container(

height: 220, child: Row( children: [

new Expanded( child:Container( height: 200.0,

width: 120.0,

margin: EdgeInsets.all(5.0), decoration: BoxDecoration(

borderRadius: BorderRadius.circular(10.0), image: DecorationImage(

image: AssetImage(sipp), fit: BoxFit.cover

),

),

),),],),

); }

}

class SecondRoute extends StatelessWidget

{

final String plant; final List data; final List en;

SecondRoute({this.plant,this.data,this.en}); @override

Widget build(BuildContext context) {

// TODO: implement build

List diseases=Lib.getDiseases(plant,data); List enDiseases=Lib.getDiseases(plant,en);

List images=Lib.getDiseaseImages(enDiseases,data); var bd=ListView.builder (

itemCount: diseases.length ,

itemBuilder: (BuildContext context,int index)

{

return new GestureDetector ( onTap:(){

\_newTaskModalBottomSheet(context,enDiseases[index],diseases[index]);

},

child: new Container( color: Colors.white,

margin: new EdgeInsets.only(top: 10.0, bottom: 10.0,left: 10.0,right: 10),

child: new Column( children:[

new ListTutorial(sip: diseases[index],), new SizedBox(height: 10,),

GridView.builder(

primary:false, gridDelegate:new

SliverGridDelegateWithFixedCrossAxisCount(crossAxisCount: 2,

),

shrinkWrap: true,

itemCount: images[index].length,

// scrollDirection: Axis.horizontal, itemBuilder: (BuildContext context,int i)

{

return new Container ( height: 200,

child: new ListTutorial1(sipp: images[index][i],),

);

}

)

],

), ),

);

},

);

return new Scaffold( backgroundColor: Colors.white, appBar: AppBar(

),

body: bd,

);

}

void \_newTaskModalBottomSheet (context,String disease,String

lanDisease){ showBottomSheet(

context: context,

builder: (BuildContext bc){

return Container( color: Colors.white, child: new Wrap(

children: <Widget>[ Container(

color:Colors.green, alignment: Alignment.center, height:40, width:double.infinity,

child: Text("Check Symptoms",style: TextStyle(color: Colors.white,fontSize: 20),),

),

new SizedBox( height:10,

),

Container(

color: Colors.blue, alignment: Alignment.center, width:double.infinity, child:new SizedBox(

height:30,

child:Text(lanDisease,style: TextStyle(color: Colors.white,fontSize: 20)),

),),

new SingleChildScrollView( child: Container(

decoration: new BoxDecoration( borderRadius:new BorderRadius.circular(25.0), border: new Border.all(

width: 10.0,

color: Colors.white,

),

gradient: new LinearGradient(

colors: [Colors.orange[50], Colors.cyan[50]],

)

),

child:Text(Lib.getProp(disease,data,'symptoms'),style:TextStyle(color: Colors.black,fontSize: 17,fontStyle: FontStyle.italic)),

),),

new CheckboxListTile(

title: const Text('Check if symptoms are matched',style:TextStyle(color: Colors.purple,fontSize: 20,)),

value: false,

onChanged: (bool value) { newTaskModalBottomSheet1(context,disease,lanDisease);

},

),

],

), );

}

);

}

void finalsheet(context,String lanDisease,String prop,String name,String disease){

showBottomSheet( context: context,

builder: (BuildContext bc){ return Container(

child: new Wrap( children: <Widget>[

Container( color:Colors.green,

alignment: Alignment.center, height:40,

width:double.infinity,

child: Text(name,style: TextStyle(color: Colors.white,fontSize: 20),),

),

new Wrap( children:[

Container( color:Colors.blue,

alignment: Alignment.center, width:double.infinity, child:new SizedBox(

height:30,

child:Text(lanDisease,style: TextStyle(color: Colors.white,fontSize: 20)),

),),

Container(

decoration: new BoxDecoration( borderRadius:new BorderRadius.circular(25.0), border: new Border.all(

width: 10.0,

color: Colors.white,

),

gradient: new LinearGradient(

colors: [Colors.green[50], Colors.cyan[50]],

)

),

child:Text(Lib.getProp(disease,data,prop),style:TextStyle(color: Colors.black,fontSize: 17,fontStyle: FontStyle.italic)),

),],),

],),);

},

);

}

void newTaskModalBottomSheet1(context,String disease,String lanDisease){

print(Lib.getProp(disease,data,'nutshell'));

List nutshell=Lib.getProp(disease,data,'nutshell');

final markDownData = nutshell.map((x) => "- $x\n").reduce((x, y) => "$x$y");

showBottomSheet( context: context,

builder: (BuildContext bc){ return Container(

color: Colors.white, child: new Wrap(

children: <Widget>[ Container(

alignment: Alignment.center, height:40, width:double.infinity,

color: Colors.green,

child: Text("Summary",style: TextStyle(color: Colors.white,fontSize: 20),),

),

Container(

alignment: Alignment.center, color:Colors.blue, width:double.infinity, child:new SizedBox( height:25,

child:Text(lanDisease,style: TextStyle(color: Colors.white,fontSize: 18)),

),),

Container(

alignment: Alignment.centerLeft, width:double.infinity,

color: Colors.white, height:180,

child:Markdown(data: markDownData)), new SizedBox(width: 20,),

new RaisedButton( color: Colors.blue,

padding: EdgeInsets.only(left: 10,right: 10), child: new Text("Chemical Control",style:

TextStyle(decorationColor: Colors.black,color: Colors.white,fontSize: 20),),

shape: new RoundedRectangleBorder(borderRadius: new BorderRadius.circular(10.0),),

onPressed: (){finalsheet(context,lanDisease,'c\_control',"Chemical Control",disease);}

),

new SizedBox(width: 20,), new RaisedButton(

color: Colors.blue,

child: new Text("Natural Control",style: TextStyle(decorationColor: Colors.black,color: Colors.white,fontSize: 20),),

padding: EdgeInsets.only(left: 10,right: 10),

shape: new RoundedRectangleBorder(borderRadius: new BorderRadius.circular(10.0),),

onPressed: (){finalsheet(context,lanDisease,'b\_control',"Natural Control",disease);}

),

new SizedBox(width: 80,),

new RaisedButton( color: Colors.green,

child: new Text("Preventive Measures",style: TextStyle(decorationColor: Colors.blue,color: Colors.white,fontSize: 20),),

padding: EdgeInsets.only(left: 30,right: 30),

shape: new RoundedRectangleBorder(borderRadius: new BorderRadius.circular(10.0),),

onPressed: (){finalsheet(context,lanDisease,'measures',"Preventive Measures",disease);}

),

],

),

);}

);

}

}

class ThirdRoute extends StatelessWidget{ @override

Widget build(BuildContext context) {

// TODO: implement build return Scaffold(

appBar:new AppBar( title: Text("Text solution"),), body:new Container(

width: 100,

height: 100,

),);

}

}

# Dart code for displaying the output in a selected language:

class Lib {

static jsonToList(File f) {

List x = JSON.jsonDecode(f.readAsStringSync()); RegExp regExp = new RegExp(r"', '");

for (int i = 0; i < x.length; i++) {

List r = x[i]['sci\_name'].split(regExp); r[0] = r[0].substring(1);

r[r.length - 1] = r[r.length - 1].replaceAll("'", ""); x[i]['sci\_name'] = r;

r = x[i]['nutshell'].split(regExp);

r[0] = r[0].substring(1);

r[r.length - 1] = r[r.length - 1].replaceAll("'", ""); x[i]['nutshell'] = r;

x[i]['file\_name'] = x[i]['file\_name'].split(r",");

}

return x;

}

static getDiseases(String plant, List data) { List diseases = new List();

for (int i in getIndex(plant, 'sci\_name')) { diseases.add(data[i]['dis\_name']);

}

return diseases;

}

static getPlants(List x) { List plant = new List();

for (int i = 0; i < x.length; i++) { x[i]['sci\_name'].forEach((e) => plant.add(e));

}

return plant.toSet().toList();

}

static getProp(String disease, List data, String prop) { int index = getIndex(disease, 'dis\_name')[0];

return data[index][prop];

}

static getFromZip(String lang) { Stringpath='/data/user/0/sq.flutter.tfliteexample/app\_flutter/.jsonFile/'; List<int> bytes = new File(path + 'json.zip').readAsBytesSync(); Archive archive = new ZipDecoder().decodeBytes(bytes);

lang = lang + '.json';

for (ArchiveFile file in archive) { String filename = file.name;

if (filename == lang &&

(FileSystemEntity.typeSync(path + 'json/' + filename) == FileSystemEntityType.notFound)) {

List<int> data = file.content;

new File(path + 'json/' + filename)

..createSync(recursive: true)

..writeAsBytesSync(data);

}

}

return path + 'json/' + lang;

}

static getData(String lang) {

return jsonToList(new File(getFromZip(lang)));

}

static getIndex(String item, String col) { List data = getData('en');

List index = new List();

for (int i = 0; i < data.length; i++) {

if (data[i][col].contains(item)) index.add(i);

}

return index;

}

static formIndex(int i, List data, String prop) {

return data[i][prop];

}

static getDiseaseImages(List diseases, List data) { List images = new List();

List newImages = new List();

for (int i = 0; i < diseases.length; i++) { images.add(getProp(diseases[i], data, 'file\_name'));

}

for (int i = 0; i < images.length; i++) { List list = new List();

list = images[i];

for (int j = 0; j < list.length; j++) {

if (list[j].substring(0, 6) == "assets") break;

list[j] = list[j].substring(1, list[j].length - 1); list[j] = 'assets/' + list[j];

}

newImages.add(list);

}

return newImages;

}

static getPlantImages(List plants) { List images =new List();

for(int i=0;i<plants.length;i++){ images.add('assets/plants/'+plants[i]+'.png');

}

return images;

}

}