يا دكتور موجود ملفات الفلاتر و الموبايل ابليكيشن

لاكن توجد مشكلة عرض الصوره على التطبيق و بسبب عدم وجود خبره فى المجال محدش عرف يحلها

مش عارف ارفع ملف التطبيق,

الطلاب

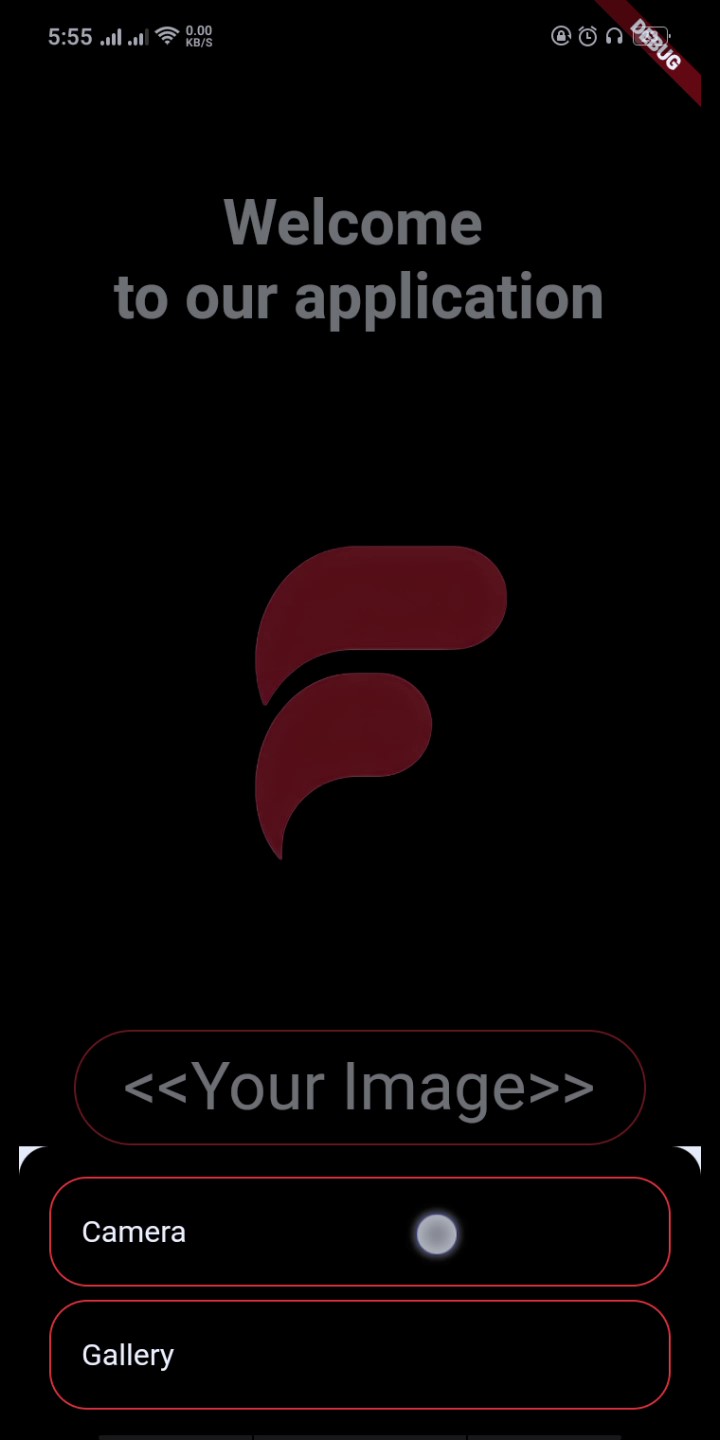
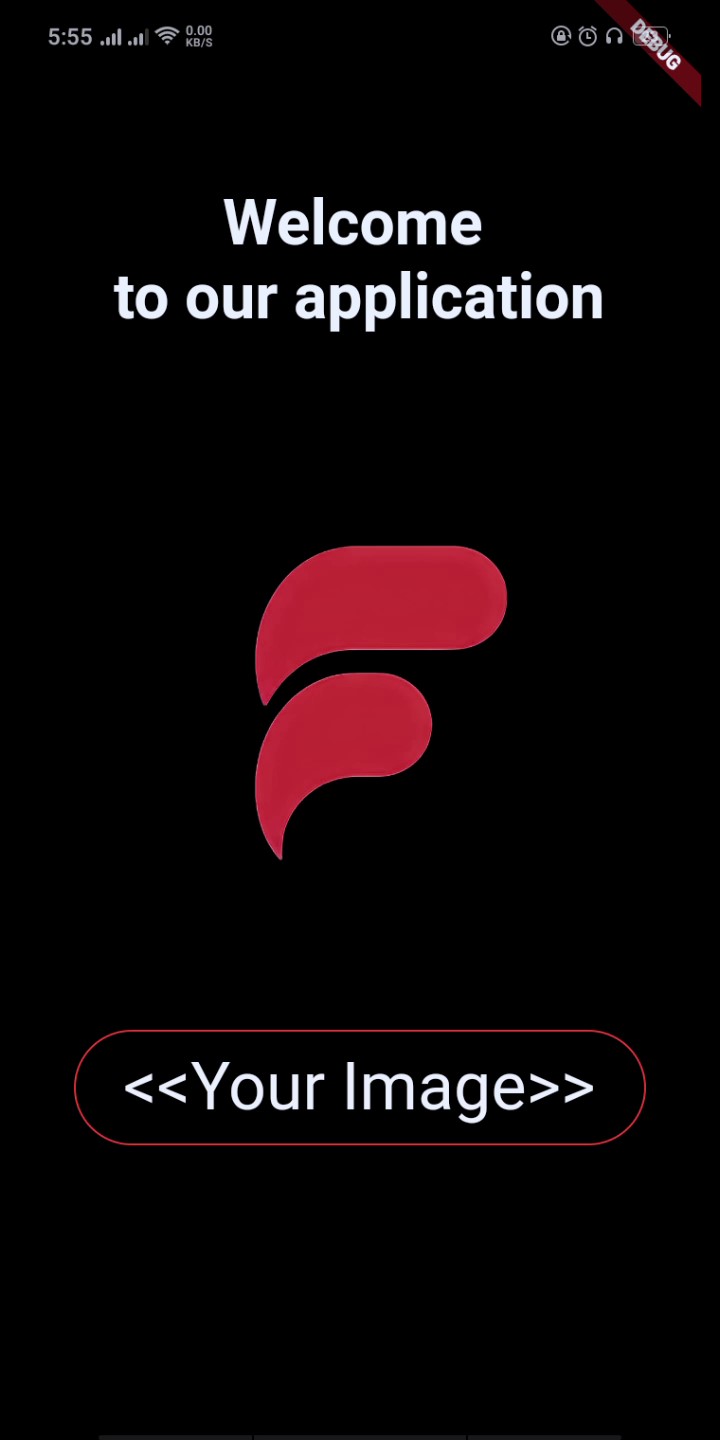
احمد عيد القطب

مصطفى محمود سنجاب

إبراهيم عبدالسميع شعيب

احمد محمد ابوبكر

كريم احمد عبدالعليم



الاكواد

smoothing filter

import 'dart:typed\_data';

import 'dart:ui' as ui;

import 'package:flutter/material.dart';

import 'package:flutter/services.dart' show rootBundle;

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

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: MyHomePage(),

);

}

}

class MyHomePage extends StatefulWidget {

@override

\_MyHomePageState createState() => \_MyHomePageState();

}

class \_MyHomePageState extends State<MyHomePage> {

ui.Image? \_image;

@override

void initState() {

super.initState();

\_loadImage();

}

Future<void> \_loadImage() async {

final ByteData data = await rootBundle.load('assets/sample.jpg');

final List<int> bytes = data.buffer.asUint8List();

final img.Image image = img.decodeImage(Uint8List.fromList(bytes))!;

final img.Image smoothedImage = img.gaussianBlur(image, 10); // تطبيق smoothing filter

final Completer<ui.Image> completer = Completer();

ui.decodeImageFromPixels(

smoothedImage.getBytes(),

smoothedImage.width,

smoothedImage.height,

ui.PixelFormat.rgba8888,

(ui.Image img) {

setState(() {

\_image = img;

});

completer.complete(img);

},

);

return completer.future;

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Smoothing Filter Example'),

),

body: Center(

child: \_image == null

? CircularProgressIndicator()

: Image(image: ImageInfo(image: \_image!, scale: 1.0)),

),

);

}

}

----------------------------------------------------------------------------------------

Sharpening Filter

import 'dart:io';

import 'dart:typed\_data';

import 'package:image/image.dart';

void main() {

String inputImagePath = 'input\_image.jpg';

String outputImagePath = 'sharpened\_output.jpg';

File inputFile = File(inputImagePath);

List<int> imageBytes = inputFile.readAsBytesSync();

Image image = decodeImage(Uint8List.fromList(imageBytes));

var kernel = [

[-1, -1, -1],

[-1, 9, -1],

[-1, -1, -1]

];

Image sharpenedImage = convolve(image, kernel);

File(outputImagePath).writeAsBytesSync(encodeJpg(sharpenedImage));

}

Image convolve(Image image, List<List<int>> kernel) {

int rows = image.height;

int cols = image.width;

for (int y = 1; y < rows - 1; y++) {

for (int x = 1; x < cols - 1; x++) {

int sum = 0;

for (int i = -1; i <= 1; i++) {

for (int j = -1; j <= 1; j++) {

int pixel = image.getPixel(x + i, y + j);

int value = getRed(pixel);

sum += kernel[j + 1][i + 1] \* value;

}

}

sum = sum.clamp(0, 255);

image.setPixel(x, y, getColor(sum, sum, sum));

}

}

return image;

----------------------------------------------

Blur Filter

import 'dart:ui' as ui;

import 'dart:typed\_data';

import 'dart:ui' show Image;

import 'dart:io';

void main() {

// Replace 'path/to/your/image.jpg' with the actual path to your image file

String imagePath = 'path/to/your/image.jpg';

// Load the image

ui.Image image = decodeImage(imagePath);

// Apply blur filter

ui.Image blurredImage = applyBlur(image, 5.0); // You can adjust the blur radius

// Save or display the blurred image as needed

saveImage(blurredImage, 'path/to/save/blurred\_image.jpg');

}

ui.Image decodeImage(String path) {

ByteData data = ByteData.sublistView(new File(path).readAsBytesSync());

return ui.decodeImageFromList(Uint8List.view(data.buffer));

}

ui.Image applyBlur(ui.Image image, double sigma) {

final ui.Recorder recorder = ui.Recorder();

final ui.Paint paint = ui.Paint()

..colorFilter = ui.ColorFilter.mode(

const ui.Color(0xFFFFFFFF),

ui.TransferMode.clear,

)

..imageFilter = ui.ImageFilter.blur(sigmaX: sigma, sigmaY: sigma);

final ui.Rect rect = ui.Rect.fromPoints(

ui.Offset(0.0, 0.0),

ui.Offset(image.width.toDouble(), image.height.toDouble()),

);

recorder

..clipRect(rect)

..drawImage(image, ui.Offset.zero, paint);

final ui.Picture picture = recorder.endRecording();

final ui.PictureRecorder pictureRecorder = ui.PictureRecorder();

final ui.Canvas canvas = ui.Canvas(pictureRecorder);

canvas.drawImage(picture, ui.Offset.zero, paint);

return pictureRecorder.endRecording().toImage(image.width, image.height);

}

void saveImage(ui.Image image, String path) {

final ByteData data =

ByteData.sublistView(Uint8List(image.width \* image.height \* 4));

final buffer = data.buffer.asUint8List();

final cullRect = ui.Rect.fromPoints(

const ui.Offset(0.0, 0.0),

ui.Offset(image.width.toDouble(), image.height.toDouble()),

);

final paint = ui.Paint();

final recorder = ui.Recorder();

recorder

..clipRect(cullRect)

..drawImage(image, ui.Offset.zero, paint);

final ui.Picture picture = recorder.endRecording();

final ui.PictureRecorder pictureRecorder = ui.PictureRecorder();

final ui.Canvas canvas = ui.Canvas(pictureRecorder);

canvas.drawImage(picture, ui.Offset.zero, paint);

final ui.Image img = pictureRecorder.endRecording().toImage(image.width, image.height);

img.toByteData(format: ui.ImageByteFormat.png).then((ByteData byteData) {

buffer.setAll(0, byteData.buffer.asUint8List());

File(path).writeAsBytesSync(Uint8List.sublistView(buffer));

});

}

-------------------------------------------------------------

blackandwhite filter

import 'dart:typed\_data';

import 'dart:ui' as ui;

import 'dart:io';

import 'dart:convert';

import 'package:flutter/material.dart';

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

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: Scaffold(

appBar: AppBar(

title: Text('Black and White Image Filter'),

),

body: Center(

child: ImageFilterWidget(),

),

),

);

}

}

class ImageFilterWidget extends StatefulWidget {

@override

\_ImageFilterWidgetState createState() => \_ImageFilterWidgetState();

}

class \_ImageFilterWidgetState extends State<ImageFilterWidget> {

img.Image? \_filteredImage;

@override

void initState() {

super.initState();

\_loadImage();

}

Future<void> \_loadImage() async {

// Replace 'assets/sample.jpg' with the path to your image

ByteData data = await rootBundle.load('assets/sample.jpg');

List<int> bytes = data.buffer.asUint8List();

img.Image originalImage = img.decodeImage(Uint8List.fromList(bytes))!;

// Apply black and white filter

\_filteredImage = img.grayscale(originalImage);

setState(() {});

}

@override

Widget build(BuildContext context) {

return \_filteredImage == null

? CircularProgressIndicator()

: Image.memory(Uint8List.fromList(img.encodePng(\_filteredImage!)));

}

}

----------------------------------------------------------------

اكواد بناء التطبيق

main.dart

import 'dart:io';

import 'package:flutter/material.dart';

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

import 'package:permission\_handler/permission\_handler.dart';

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

import 'permission\_service.dart';

import 'media\_service\_interface.dart';

import 'display\_image\_widget.dart';

import 'dart:typed\_data';

import 'dart:ui';

import 'dart:async';

import 'package:flutter/services.dart';

import 'package:flutter\_image\_filters/flutter\_image\_filters.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

const MyApp({Key? key}) : super(key: key);

@override

Widget build(BuildContext context) {

return MaterialApp(

home: MyHomePage(),

);

}

}

class MyHomePage extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

backgroundColor: Colors.black,

body: SafeArea(

child: Stack(

children: [

Positioned(

top: 70,

left: 50,

right: 50,

child: Container(

decoration: BoxDecoration(

color: Colors.black,

borderRadius: BorderRadius.circular(10.0),

),

padding: EdgeInsets.symmetric(vertical: 0.0),

child: Center(

child: Text(

' Welcome to our application',

style: TextStyle(

fontSize: 33,

fontWeight: FontWeight.bold,

color: Colors.white,

),

),

),

),

),

Center(

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Image(

image: AssetImage("Images/fci.jpg"),

),

SizedBox(height: 0),

SizedBox(height: 0),

AvatarUploader(),

],

),

),

],

),

),

);

}

}

class AvatarUploader extends StatefulWidget {

@override

\_AvatarUploaderState createState() => \_AvatarUploaderState();

}

class \_AvatarUploaderState extends State<AvatarUploader> {

final MediaServiceInterface \_mediaService = PermissionServiceImpl();

File? imageFile;

bool \_isLoadingGettingImage = false;

Future<void> \_pickImageSource() async {

await \_showCupertinoModalPopup();

}

Future<void> \_pickImageFromCamera() async {

try {

final pickedFile = await ImagePicker().pickImage(source: ImageSource.camera);

if (pickedFile != null) {

File pickedImageFile = File(pickedFile.path);

await \_getImageFromFile(pickedImageFile);

}

} catch (e) {

print("Error picking image from camera: $e");

}

}

Future<void> \_pickImageFromGallery() async {

try {

FilePickerResult? result = await FilePicker.platform.pickFiles(

type: FileType.image,

allowMultiple: false,

);

if (result != null && result.files.isNotEmpty) {

File pickedFile = File(result.files.first.path!);

await \_getImageFromFile(pickedFile);

}

} catch (e) {

print("Error picking image from gallery: $e");

}

}

Future<void> \_getImageFromFile(File file) async {

setState(() => \_isLoadingGettingImage = true);

final \_pickedImageFile = await \_mediaService.compressFile(file);

setState(() => \_isLoadingGettingImage = false);

if (\_pickedImageFile != null) {

setState(() => imageFile = \_pickedImageFile);

}

}

Future<void> \_showCupertinoModalPopup() async {

showModalBottomSheet(

context: context,

builder: (BuildContext context) {

return Container(

padding: EdgeInsets.all(16.0),

decoration: BoxDecoration(

borderRadius: BorderRadius.vertical(top: Radius.circular(16.0)),

color: Colors.black,

),

child: Column(

mainAxisSize: MainAxisSize.min,

children: [

\_buildListTile('Camera', \_pickImageFromCamera),

SizedBox(height: 7.0),

\_buildListTile('Gallery', \_pickImageFromGallery),

],

),

);

},

);

}

Widget \_buildListTile(String title, Function() onTapFunction) {

return Container(

decoration: BoxDecoration(

border: Border.all(color: Colors.red),

borderRadius: BorderRadius.circular(20.0),

),

child: ListTile(

title: Text(

title,

style: TextStyle(color: Colors.white),

),

onTap: () async {

Navigator.pop(context);

await onTapFunction();

},

),

);

}

@override

Widget build(BuildContext context) {

return Container(

child: ElevatedButton(

style: ElevatedButton.styleFrom(

primary: Colors.black54, // لون الخلفية

onPrimary: Colors.white, // لون النص

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(50.0), // شكل الحواف

side: BorderSide(color: Colors.red), // لون الحدود

),

),

onPressed: () async {

await \_pickImageSource();

setState(() {});

},

child: Padding(

padding: const EdgeInsets.all(10.0),

child: Text(

'<<Your Image>>',

style: TextStyle(

fontSize: 35.0,

),

),

),

),

);

}

}

class PermissionServiceImpl implements PermissionService, MediaServiceInterface {

@override

Future requestPhotosPermission() async {

// Implement the logic for requesting photos permission

}

@override

Future<bool> handlePhotosPermission(BuildContext context) async {

// Implement the logic for handling photos permission

return true; // Replace with the actual logic

}

@override

Future requestCameraPermission() async {

// Implement the logic for requesting camera permission

}

@override

Future<bool> handleCameraPermission(BuildContext context) async {

// Implement the logic

return true; // Replace with the actual logic

}

@override

Future<File?> compressFile(File file, {int quality = 30}) async {

// Provide an empty implementation or the actual logic if needed

return null;

}

@override

Future<File?> uploadImage(BuildContext context, AppImageSource appImageSource, {bool shouldCompress = true}) async {

// Provide an empty implementation or the actual logic if needed

return null;

}

}

-----------------------------------------------------

media\_service\_interface

// media\_service\_interface.dart

import 'dart:io';

import 'package:flutter/cupertino.dart';

enum AppImageSource {

camera,

gallery,

}

abstract class MediaServiceInterface {

Future<File?> uploadImage(

BuildContext context,

AppImageSource appImageSource, {

bool shouldCompress = true,

});

Future<File?> compressFile(File file, {int quality = 30});

}

---------------------------------------------------------------

permission\_service

import 'package:flutter/material.dart';

abstract class PermissionService {

Future requestPhotosPermission();

Future<bool> handlePhotosPermission(BuildContext context);

Future requestCameraPermission();

Future<bool> handleCameraPermission(BuildContext context);

}

class PermissionServiceImpl implements PermissionService {

@override

Future requestPhotosPermission() async {

// Implement the logic for requesting photos permission

}

@override

Future<bool> handlePhotosPermission(BuildContext context) async {

// Implement the logic for handling photos permission

return true; // Replace with the actual logic

}

@override

Future requestCameraPermission() async {

// Implement the logic for requesting camera permission

}

@override

Future<bool> handleCameraPermission(BuildContext context) async {

// Implement the logic for handling camera permission

return true; // Replace with the actual logic

}

}