# ใบงาน 10

รหัสน.ศ.

ชื่อ-สกุล

กลุ่มเรียน

## การออกแบบโปรแกรมด้วยตนเอง (mini project)

วัตถุประสงค์ เพื่อให้นักศึกษาเข้าใจ สามารถออกแบบ เขียนโปรแกรม Flutter และสามารถสร้าง Application ที่ใช้งานได้ด้วยตนเอง และสามารถอธิบายขั้นตอนวิธีการสร้างได้

คำอธิบาย เขียนคำอธิบายขั้นตอนการสร้างส่วนต่าง ๆ ของ code โปรแกรมแต่ละบรรทัดโดยละเอียด

วิธีการ

1. ให้นักศึกษาออกแบบ Application ด้วยตนเอง เพื่อใช้ประโยชน์จากอุปกรณ์ smart phone ให้ชีวิตในคุณภาพมากขึ้น โดยมีเงื่อนไขดังนี้
   1. ที่ออกแบบต้องไม่ใช่หัวข้อเกมส์
   2. ต้องมีพื้นฐานจากใบงานที่ผ่านมาอย่างน้อย 2 ใบงาน จากใบงานที่ 5 ถึง 8
2. หัวข้อ Application ที่นักศึกษาออกแบบคือ

App drawing

1. ประโยชน์ในการใช้ Application นี้คือ

ใช้ในการวาดเขียน

1. **ภาพร่าง**หน้าจอสำหรับ Application ที่นักศึกษาออกแบบ (**ไม่ใช่**ภาพผลงานสุดท้ายที่จัดทำเสร็จแล้ว)

A screenshot of a video game

Description automatically generated

1. Application นี้ออกแบบโดยมีพื้นฐานจากใบงานใดบ้าง ระบุหมายเลขใบงาน และหมายเลขข้ออ้างอิงในใบงาน

ใบงานที่ 5,7,8

1. ขั้นตอนวิธีการเขียนโปรแกรมสร้าง Application โดยละเอียด (ตัวอย่างจากขั้นตอนการเขียนโปรแกรมทุกใบงานที่ผ่านมา)

ติดตั้ง package

dependencies:

  flutter:

    sdk: flutter

  flutter\_colorpicker: ^1.1.0

  flutter\_drawing\_board: ^0.9.3

  material\_design\_icons\_flutter: ^7.0.7296

  path\_provider: ^2.1.4

  photo\_manager: ^3.3.0

  zoom\_tap\_animation: ^1.1.0

  json\_annotation:

  intl:

dev\_dependencies:

  flutter\_test:

    sdk: flutter

  flutter\_lints: ^4.0.0

  build\_runner:

  json\_serializable:

drawingBoard(

*background*: Container(

*width*: *constraints*.maxWidth,

*height*: *constraints*.maxHeight,

*color*: Colors.white,

              ),

*showDefaultTools*: false,

*showDefaultActions*: false,

            ),

1. บันทึก code ในขั้นตอนแรก และเขียนอธิบาย code

สร้างไฟล์สำหรับกดหนดสีในแอพ

*import* 'package:flutter/material.dart';

*// Define color constants*

*class* ColorPalette { *//กำหนดตัวแปรสำหรับสีที่ใช้ในแอพ*

*static* *const* Color black = Colors.black;

*static* *const* Color white = Colors.white;

*static* *const* Color red = Colors.red;

*static* *const* Color green = Colors.green;

*static* *const* Color blue = Colors.blue;

*static* *const* Color yellow = Colors.yellow;

*static* *const* Color pink = Colors.pink;

*static* *const* Color purple = Colors.purple;

*static* *const* Color cyanAccent = Colors.cyanAccent;

*static* *const* Color brown = Colors.brown;

*static* *const* Color orange = Colors.orange;

*static* *const* Color grey = Colors.grey;

*static* *const* Color darkGrey = Color.fromARGB(255, 133, 133, 133);

*static* *const* Color primary = Color.fromARGB(255, 219, 219, 219);

*static* *const* Color toolBg = Color.fromARGB(255, 32, 32, 32);

}

สร้างไฟล์ที่ไว้กำหนดตัวแปรแบบ global

*import 'package:flutter/material.dart';*

*import 'package:flutter\_drawing\_board/flutter\_drawing\_board.dart';*

*List<DrawingController> savedControllers =*

*[]; //ตัวแปรที่ไว้เก็บการ save drawingControllers*

*final GlobalKey globalKey = GlobalKey(); //key ในการบันทึกรูป*

*DrawingController drawingController =*

*DrawingController(); //ตัวแปร drawingController*

*DrawingController jsonCon =*

*DrawingController(); //ตัวแปรที่ไว้เก็บ json ของ drawingController ที่โหลดมาจากการบันทึก*

*String editFileName =*

*""; //ตัวแปรที่ไว้เก็บชื่อ ไฟล์ json ที่จะใช้ในการแก้ไขรูปวาด*

*// bool isActive = false; //ไว้ ในการเช็คว่าเลือก เครื่องมือในการวาดตัวไหน*

*Color selectedColor = Colors.blue; //ตัวแปรในการเลือกสีในการวาด*

*double strokeWidth = 2.0; //ตัวแปรขนาดของเส้นในการวาด*

*enum ToolType {*

*//ตัวแปรที่ไว้เก็บประเภทของเครื่องมือ*

*none,*

*eraser,*

*strokeWidth,*

*pencil*

*}*

สร้างไฟล์สำหรับกระดาน การวาด

*import* 'package:drawnig\_flutter/globalvariable.dart';

*import* 'package:flutter/material.dart';

*import* 'package:flutter\_drawing\_board/flutter\_drawing\_board.dart';

*void* main() {

  runApp(DrawingBoardPage());

}

*class* DrawingBoardPage *extends* StatefulWidget {

  @override

  \_DrawingBoardPageState createState() => \_DrawingBoardPageState();

}

*class* \_DrawingBoardPageState *extends* State<DrawingBoardPage> {

  @override

  Widget build(BuildContext *context*) {

*return* Scaffold(

*body*: LayoutBuilder(

*builder*: (*context*, *constraints*) {

*return* RepaintBoundary(

*//ใช้ในการบันทึกรูปที่เขียน*

*key*: globalKey, *//key ที่ใช้ในการบันทึกรูป*

*child*: DrawingBoard(

*controller*: drawingController, *//ตัว controller สำหรับการวาด*

*background*: Container(

*width*: *constraints*.maxWidth, *//กำหนดขนาด กระดานวาด*

*height*: *constraints*.maxHeight,

*color*: Colors.white, *//กำหนดสีพื้นหลังกระดานวาด*

              ),

*showDefaultTools*: false, *//ปิดการใช้ เครื่องมือพื้นฐาน ต่างๆ*

*showDefaultActions*: true,

            ),

          );

        },

      ),

    );

  }

}

สร้างไฟล์สำหรับเครืองมื่อการวาด

*import* 'package:drawnig\_flutter/app\_colors.dart';

*import* 'package:flutter/material.dart';

*import* 'package:drawnig\_flutter/globalvariable.dart';

*import* 'package:flutter\_drawing\_board/paint\_contents.dart';

*import* 'package:material\_design\_icons\_flutter/material\_design\_icons\_flutter.dart';

*import* 'package:flutter\_colorpicker/flutter\_colorpicker.dart';

*class* ToolDrawing *extends* StatefulWidget {

  @override

  \_ToolDrawingState createState() => \_ToolDrawingState();

}

*class* \_ToolDrawingState *extends* State<ToolDrawing> {

  ToolType \_activeTool = ToolType.pencil; *//กำหนดตัวแปร ToolType ให้เป็นปากกา*

  @override

  Widget build(BuildContext *context*) {

*return* Column(

*children*: [

        Row(

*mainAxisAlignment*: MainAxisAlignment.spaceBetween,

*children*: [

            Row(

*children*: [

*//สร้างเครื่องมือ ปากกา*

                Tooltip(

*message*: "Pencil",

*child*: IconButton(

*icon*: Icon(MdiIcons.pencil),

*color*: \_activeTool ==

                            ToolType

                                .pencil *//เช็คว่าเลือกไหมถ้าใช่ ให้เป็นสีที่เลือก*

                        ? selectedColor

                        : ColorPalette.darkGrey,

*onPressed*: () {

                      setState(() {

*//ใช้  setState เพื่อ อัปเดตการเลือกการแสดงผล*

                        drawingController.setPaintContent(

                            SimpleLine()); *//กำหนดให้เครื่องมือเป็น SimpleLine คือเส้นการวาดทั่วไป*

                        \_activeTool = ToolType.pencil;

                        drawingController.setStyle(

*//กำหนดสไตล์ ของเส้น*

*color*:

                              selectedColor, *//กำหนดสี ให้เป็นสีที่เลือกจาก selectedColor*

*strokeWidth*:

                              strokeWidth, *//กำหนดขนาดเส้นให้เป็นการเลือกจาก strokeWidth*

*isAntiAlias*: true, *//โหมดป้องกันรอยหยัก*

*style*: PaintingStyle

                              .fill, *//มีแบบ PaintingStyle.fill ลงสีทึบ*

                        );

                      });

                    },

                  ),

                ),

                Tooltip(

*message*: "Eraser",

*child*: IconButton(

*icon*: Icon(MdiIcons.eraserVariant),

*color*: \_activeTool ==

                            ToolType

                                .eraser *//เช็คว่าเลือกไหมถ้าใช่ ให้เป็นสีที่เลือก*

                        ? ColorPalette.primary

                        : ColorPalette.darkGrey,

*onPressed*: () {

                      setState(() {

                        \_activeTool = ToolType.eraser;

                        setState(() {

                          drawingController.setPaintContent(

                              Eraser()); *//กำหนดให้เครื่องมือเป็น ยางลบ*

                        });

                      });

                    },

                  ),

                ),

                Tooltip(

*message*: "StrokeWidth",

*child*: Container(

*width*: 100,

*child*: Slider(

*//สไลล์*

*value*: strokeWidth,

*min*: 0, *//ค่าของสไลล์ต่ำสุด*

*max*: 20, *//สูงสุด*

*divisions*:

                            20, *//การช่วง เช่น ในค่ามากสุด 0- 20 แบ่งเป็น 20 จากตัวค่า 1,2,3,4...20 แต่ถ้าเป็น 40 จะได้ 0.5,1,1.5,2...20*

*onChanged*: (*value*) {

                          setState(() {

                            strokeWidth =

*value*; *//กำหนด strokeWidth ให้ทำกับค่าที่เลือกจาก  สไลล์*

                            drawingController.setStyle(

*color*: selectedColor,

*strokeWidth*:

                                  strokeWidth, *//กำนาดขนาดเส้นเป็น strokeWidth*

*isAntiAlias*: true,

*style*: PaintingStyle.stroke,

                            );

                          });

                        },

*label*: strokeWidth.toStringAsFixed(2),

                      ),

                    )),

                SizedBox(

*width*: 50,

*child*: Tooltip(

*message*: "Selete Color",

*child*: ElevatedButton(

*style*: ElevatedButton.styleFrom(

*//กำสไตล์ให้ Tooltip ElevatedButton*

*backgroundColor*: Colors.transparent,

*shadowColor*: Colors.transparent,

                      ),

*onPressed*:

                          \_openColorPicker, *//เรียกใช้ฟังชั่น \_openColorPicker*

*child*: Icon(MdiIcons.palette, *//กำสไตล์ให้ icon*

*size*: 30,

*color*: selectedColor),

                    ),

                  ),

                ),

              ],

            ),

            Row(

*children*: [

                SizedBox(

*width*: 50,

*child*: Tooltip(

*message*: "Undo",

*child*: IconButton(

*icon*: Icon(MdiIcons.undo),

*color*: ColorPalette.darkGrey,

*onPressed*: () {

                        setState(() {

                          drawingController.undo(); *//กำหนดเครื่องมือให้เป็น undo*

                        });

                      },

                    ),

                  ),

                ),

                SizedBox(

*width*: 50,

*child*: Tooltip(

*message*: "Redo",

*child*: IconButton(

*icon*: Icon(MdiIcons.redo),

*color*: ColorPalette.darkGrey,

*onPressed*: () {

                        setState(() {

                          drawingController.redo();*//กำหนดเครื่องมือให้เป็น redo*

                        });

                      },

                    ),

                  ),

                ),

              ],

            )

          ],

        ),

      ],

    );

  }

*void* \_openColorPicker() {*//ฟังชั่นในการเลือกสี*

    showDialog(

*context*: context,

*builder*: (BuildContext *context*) {

*return* AlertDialog( *//เป็น popup เด้นขึ้นมา*

*title*: Text('Select a color'),

*content*: SingleChildScrollView( *//ทำให้เลือนได้*

*child*: BlockPicker(*//ถาดสีจาก package flutter\_colorpicker*

*pickerColor*: selectedColor, *// สีที่แสดงตอนเริ่ม*

*onColorChanged*: (Color *color*) {

                setState(() { *//กำหนด setState*

                  selectedColor = *color*; *//กำหนดสีให้เป็นสีที่เลือก*

                  drawingController.setStyle(

*color*: selectedColor, *//กำหนดสี*

*strokeWidth*: strokeWidth,

*isAntiAlias*: true,

*style*: PaintingStyle.stroke,

                  );

                });

              },

            ),

          ),

*actions*: <Widget>[ *// actions กำหนด ข้อความ Done เป็น TextButton เมือกด ปิดหน้า selectedColor หรือ ย้อนกลับมาหน้าวาดรูป*

            TextButton(

*child*: Text('Done'),

*onPressed*: () {

                Navigator.of(*context*).pop();

              },

            ),

          ],

        );

      },

    );

  }

}

สร้างหน้าเริ่มต้นการวาด

*import* 'package:drawnig\_flutter/main.dart';

*import* 'package:drawnig\_flutter/model/image\_save.dart';

*import* 'package:flutter/cupertino.dart';

*import* 'package:flutter/material.dart';

*import* 'package:flutter/widgets.dart';

*import* 'package:flutter\_drawing\_board/flutter\_drawing\_board.dart';

*import* 'package:material\_design\_icons\_flutter/material\_design\_icons\_flutter.dart';

*import* 'package:drawnig\_flutter/app\_colors.dart';

*import* 'package:drawnig\_flutter/globalvariable.dart';

*import* 'package:drawnig\_flutter/model/drawing\_board\_page.dart';

*import* 'package:drawnig\_flutter/model/tool\_drawing.dart';

*import* 'package:zoom\_tap\_animation/zoom\_tap\_animation.dart';

*void* main() {

  runApp(DrawingApp());

}

*class* DrawingApp *extends* StatelessWidget {

  @override

  Widget build(BuildContext *context*) {

*return* MaterialApp(

*home*:

          DrawingAppPage(), *// เปลี่ยนชื่อคลาสให้ตรงกับ StatefulWidget ที่ถูกต้อง*

    );

  }

}

*class* DrawingAppPage *extends* StatefulWidget {

*// เปลี่ยนชื่อคลาส StatefulWidget เป็น DrawingAppPage*

  @override

  \_DrawingAppPageState createState() => \_DrawingAppPageState();

}

*class* \_DrawingAppPageState *extends* State<DrawingAppPage> {

  @override

*void* initState() {

*super*.initState();

    drawingController = DrawingController(); *//กำหนด drawingController เป็นค่าเริ่มต้น*

    \_resetState();

  }

*void* \_resetState() { *//รีเช็ต การกระดานวาด*

    drawingController.setStyle(

*color*: selectedColor,

*strokeWidth*: strokeWidth,

*isAntiAlias*: true,

*style*: PaintingStyle.stroke,

    );

  }

*void* \_saveDrawing() {

*// เพิ่ม drawingController ลงในลิสต์*

    savedControllers.add(drawingController);

*// บันทึกภาพตามที่คุณต้องการ*

    saveDrawing(*context*: context, *globalKey*: globalKey, *newDrawing*: true); *//ส่งข้อมูลต่างๆไปยังการ save*

  }

  @override

  Widget build(BuildContext *context*) {

*return* Scaffold(

*appBar*: AppBar( *//appbar*

*leading*: IconButton(

*icon*: Icon(*//ปุ่มย้อนกลับ*

            MdiIcons.arrowLeftCircle,

*color*: ColorPalette.white,

*size*: 40,

          ),

*onPressed*: () {

            Navigator.pushReplacement(

*context*,

              MaterialPageRoute(

*builder*: (*context*) => MainPage(),

              ),

            );

          },

        ),

*title*: *const* Text(

          'Drawing Board',

*style*:

              TextStyle(*color*: ColorPalette.white, *fontWeight*: FontWeight.bold),

        ),

*backgroundColor*: ColorPalette.black,

*actions*: [

          ZoomTapAnimation(*//กำหนดอนิเมชั่นการกดเป็นการซูมเมื่อกด จาก แพกเก็จ ZoomTapAnimation*

*end*: 0.8, *// เมื่อกดย่อ icon = 0.8*

*begin*: 1,*//เริ่มต้นของขนาด icon*

*child*: IconButton( *//ปุ่ม save*

*icon*: Icon(

                MdiIcons.contentSaveAll,

*color*: ColorPalette.grey, *// ใช้สีพื้นหลังจาก ColorPalette*

              ),

*onPressed*: \_saveDrawing, *//เรียกใช้ฟั่งชั่น save*

            ),

          )

        ],

      ),

*body*: Stack( *//Stack ใช้ในการจัด  Positioned ของหน้าจอต่างๆ เช่น ตำแหน่งเครื่องมือ*

*children*: [

          Positioned(

*child*: DrawingBoardPage(),*//เรียกใช้ กระดานวาด จาก  drawing\_board\_page.dart ที่สร้างไว้*

          ),

          Positioned( *//กำหนดตำแหนน่งของเครื่องมือ*

*bottom*: 0,

*child*: Container(

*padding*: *const* EdgeInsets.all(2),

*width*: MediaQuery.of(*context*).size.width,

*color*: ColorPalette.toolBg, *// ใช้สีพื้นหลังจาก ColorPalette*

*child*: ToolDrawing(),*//เรียกใช้เครื่องมือ tool\_drawing.dart ที่เตรียมไว้*

            ),

          ),

        ],

      ),

    );

  }

}

1. บันทึกภาพผลการ run ในขั้นตอนแรก



1. ขันตอนวิธีการเขียนโปรแกรม ในขั้นตอนถัดไปโดยละเอียด

สร้างไฟล์ สำหรับบันทึกรูปและ json

*import* 'dart:convert';

*import* 'dart:io';

*import* 'dart:ui' *as* ui;

*import* 'package:drawnig\_flutter/globalvariable.dart';

*import* 'package:drawnig\_flutter/main.dart';

*import* 'package:flutter/rendering.dart';

*import* 'package:path\_provider/path\_provider.dart';

*import* 'package:photo\_manager/photo\_manager.dart';

*import* 'package:flutter/material.dart';

*import* 'package:intl/intl.dart';

Future<*void*> saveDrawing({

*required* BuildContext *context*,

*required* bool *newDrawing*,

}) *async* {

*try* {

*// รับข้อมูลภาพจาก DrawingController*

*final* byteData = *await* drawingController.getImageData();

*// ตรวจสอบว่า byteData เป็น null หรือไม่*

*if* (byteData != *null*) {

*final* buffer =

          byteData.buffer.asUint8List(); *// แปลง ByteData เป็น Uint8List*

*// ดึงที่อยู่ของ directory ที่ใช้ในการบันทึกรูป*

*final* directory = *await* getTemporaryDirectory();

*final* now = DateTime.now(); *// รับวันที่และเวลา*

*final* timestamp =

          '${*now*.*year*}-${*now*.*month*}-${*now*.*day*}\_${*now*.*hour*}-${*now*.*minute*}-${*now*.*second*}';

*final* path =

          '${*directory*.*path*}/drawing\_image\_$*timestamp*.png'; *// กำหนดเส้นทางไฟล์*

*final* file = File(path); *// สร้างไฟล์ที่เส้นทางกำหนด*

*await* file.writeAsBytes(buffer); *// เขียนข้อมูลลงในไฟล์*

*// ใช้ PhotoManager ในการบันทึกรูปลงในอัลบั้มภาพ*

*final* result = *await* PhotoManager.editor.saveImage(

        buffer, *// ใช้ buffer ที่ได้จาก getImageData*

*filename*: '/drawing\_image\_$*timestamp*.png', *// ตั้งชื่อไฟล์*

      );

*if* (result != *null*) {

*// เช็คว่าการบันทึกสำเร็จหรือไม่*

        ScaffoldMessenger.of(*context*).showSnackBar(

          SnackBar(*content*: Text('Save image')),

        );

*// ตรวจสอบว่าเป็นรูปใหม่หรือไม่*

*if* (*newDrawing*) {

*// สร้างวันที่และเวลาปัจจุบัน*

*final* formatter = DateFormat('yyyyMMdd\_HHmmss');

*final* dateTimeString = formatter.format(DateTime.now());

          \_SaveJsonListNew(dateTimeString); *// บันทึก JSON พร้อมชื่อใหม่*

        } *else* {

          \_SaveJsonListNew(editFileName); *// บันทึก JSON ด้วยชื่อเดิม*

        }

        Future.delayed(Duration(*seconds*: 2), () {

          Navigator.pushReplacement(

*context*,

            MaterialPageRoute(

*builder*: (*context*) => Main()), *// เปลี่ยนหน้าหลังจากบันทึก*

          );

        });

      } *else* {

        ScaffoldMessenger.of(*context*).showSnackBar(

          SnackBar(*content*: Text('Failed to save image')),

        );

      }

    } *else* {

      ScaffoldMessenger.of(*context*).showSnackBar(

        SnackBar(*content*: Text('Failed to retrieve image data')),

      );

    }

  } *catch* (e) {

    ScaffoldMessenger.of(*context*).showSnackBar(

      SnackBar(*content*: Text('Failed to save image: $*e*')),

    );

  }

}

*//ฟังชั่นในการ save json*

Future<*void*> \_SaveJsonListNew(String *name*) *async* {

*try* {

*// รับข้อมูล JSON จาก DrawingController*

    String jsonString = *const* JsonEncoder.withIndent('  ')

        .convert(drawingController.getJsonList());

*// กำหนดพาธสำหรับบันทึกไฟล์ (เปลี่ยนพาธตามต้องการ)*

*final* directory = Directory.systemTemp; *// ใช้ไดเรกทอรี temp ของระบบ*

*final* filePath = '${*directory*.*path*}/$*name*.json';

*// สร้างไฟล์และเขียนข้อมูล JSON ลงในไฟล์*

*final* file = File(filePath);

*await* file.writeAsString(jsonString);

    print('ข้อมูล JSON ถูกบันทึกลงในไฟล์ที่ $*filePath*');

  } *catch* (e) {

    print('เกิดข้อผิดพลาดในการบันทึกข้อมูล JSON: $*e*');

  }

}

สร้างไฟล์ *หน้าในการ แปลงไฟล์ json เป็น obj ของ dart โดยใช้ json\_serializable*

*import* 'package:json\_annotation/json\_annotation.dart';

*part* 'drawing\_model.g.dart'; //ชื่อต้องตรงกับชื่อไฟล์

*//หน้าในการ แปลงไฟล์ json เป็น obj ของ dart โดยใช้ json\_serializable*

@JsonSerializable()

*class* PaintData { *//ค่าต่างๆในไฟล์ json ที่วาด เช่น สี ขนาดเส้น*

*final* int blendMode;

*final* int color;

*final* int filterQuality;

*final* bool invertColors;

*final* bool isAntiAlias;

*final* int strokeCap;

*final* int strokeJoin;

*final* double strokeWidth;

*final* int style;

  PaintData({

*required* *this*.blendMode,

*required* *this*.color,

*required* *this*.filterQuality,

*required* *this*.invertColors,

*required* *this*.isAntiAlias,

*required* *this*.strokeCap,

*required* *this*.strokeJoin,

*required* *this*.strokeWidth,

*required* *this*.style,

  });

*factory* PaintData.fromJson(Map<String, dynamic> *json*) => \_$PaintDataFromJson(*json*);

  Map<String, dynamic> toJson() => \_$PaintDataToJson(*this*);

}

@JsonSerializable()

*class* StepData { *//ประเภท //ตำแหน่ง x y*

*final* String type;

*final* double x;

*final* double y;

  StepData({

*required* *this*.type,

*required* *this*.x,

*required* *this*.y,

  });

*factory* StepData.fromJson(Map<String, dynamic> *json*) => \_$StepDataFromJson(*json*);

  Map<String, dynamic> toJson() => \_$StepDataToJson(*this*);

}

@JsonSerializable()

*class* PathData {

*final* int fillType;

*final* List<StepData> steps;

  PathData({

*required* *this*.fillType,

*required* *this*.steps,

  });

*factory* PathData.fromJson(Map<String, dynamic> *json*) => \_$PathDataFromJson(*json*); *//แปลงข้อมูลต่างๆ*

  Map<String, dynamic> toJson() => \_$PathDataToJson(*this*);

}

@JsonSerializable()

*class* Simple {

*final* String type;

*final* PathData path;

*final* PaintData paint;

  Simple({

*required* *this*.type,

*required* *this*.path,

*required* *this*.paint,

  });

*factory* Simple.fromJson(Map<String, dynamic> *json*) => \_$SimpleFromJson(*json*);

  Map<String, dynamic> toJson() => \_$SimpleToJson(*this*);

}

ใช้คำสั่ง flutter pub run build\_runner build ในการ build การสร้างไฟล์

สร้างไฟล์สำหรับการ เพิ่มข้อมูลที่ต้ำการแก้สำหรับการวาด

*import* 'dart:convert';

*import* 'dart:io';

*import* 'package:drawnig\_flutter/globalvariable.dart';

*import* 'package:drawnig\_flutter/model/drawing\_model.dart';

*import* 'package:flutter\_drawing\_board/flutter\_drawing\_board.dart';

*import* 'package:flutter\_drawing\_board/paint\_contents.dart';

Future<*void*> GetJsonDrawing(String *name*) *async* {

*// กำหนดพาธของไฟล์ JSON ที่จะอ่าน (เปลี่ยนพาธตามต้องการ)*

*final* directory = Directory.systemTemp; *// ใช้ไดเรกทอรี temp ของระบบ*

*final* filePath = '${*directory*.*path*}/${*name*}.json';

*final* file = File(filePath);

  drawingController = DrawingController();

*// ตรวจสอบว่าไฟล์มีอยู่หรือไม่*

*if* (*await* file.exists()) {

*// อ่านข้อมูลจากไฟล์*

    String jsonString = *await* file.readAsString();

*// แปลงข้อมูล JSON เป็น Dart object*

*final* List<dynamic> jsonData = jsonDecode(jsonString);

*// แสดงข้อมูล JSON*

    print(jsonData);

*// final List<dynamic> jsonData = jsonDecode(jsonString);*

*final* List<Simple> drawings =

        jsonData.map((*item*) => Simple.fromJson(*item*)).toList();

*//แปลงข้อมูลจาก json ที่ได้มาเพื่อทำให้ข้อมูลที่ได้ ตรงกันกับ ฟังชั่น addContent ของ flutter\_drawing\_board*

*for* (*var* drawing *in* drawings) {

*final* Map<String, dynamic> \_testLineSimple = {

        "type": drawing.type,

        "path": {

          "fillType": drawing.path.fillType,

          "steps": drawing.path.steps

              .map((*step*) => {

                    "type": *step*.type,

                    "x": *step*.x,

                    "y": *step*.y,

                  })

              .toList(),

        },

        "paint": {

          "blendMode": drawing.paint.blendMode,

          "color": drawing.paint.color,

          "filterQuality": drawing.paint.filterQuality,

          "invertColors": drawing.paint.invertColors,

          "isAntiAlias": drawing.paint.isAntiAlias,

          "strokeCap": drawing.paint.strokeCap,

          "strokeJoin": drawing.paint.strokeJoin,

          "strokeWidth": drawing.paint.strokeWidth,

          "style": drawing.paint.style,

        }

      };

*// print(\_testLineSimple);*

*// ตรวจสอบประเภทและแปลงเป็นวัตถุที่เหมาะสม*

*if* (\_testLineSimple["type"] == "SimpleLine") {

        drawingController.addContent(SimpleLine.fromJson(\_testLineSimple));

      } *else* *if* (\_testLineSimple["type"] == "Eraser") {

        drawingController.addContent(Eraser.fromJson(\_testLineSimple));

      }

    }

  } *else* {

    print('ไฟล์ JSON ไม่พบที่ $*filePath*');

  }

}

สร้างไฟล์ สำหรับหน้าแก้ไขการวาด

*import* 'package:drawnig\_flutter/main.dart';

*import* 'package:drawnig\_flutter/model/json\_drawing.dart';

*import* 'package:flutter/material.dart';

*import* 'package:flutter/services.dart';

*import* 'package:material\_design\_icons\_flutter/material\_design\_icons\_flutter.dart';

*import* 'package:drawnig\_flutter/app\_colors.dart';

*import* 'package:drawnig\_flutter/globalvariable.dart';

*import* 'package:drawnig\_flutter/model/drawing\_board\_page.dart';

*import* 'package:drawnig\_flutter/model/image\_save.dart';

*import* 'package:drawnig\_flutter/model/tool\_drawing.dart';

*import* 'package:zoom\_tap\_animation/zoom\_tap\_animation.dart';

*void* main() {

  runApp(EditDrawingApp());

}

*class* EditDrawingApp *extends* StatelessWidget {

  @override

  Widget build(BuildContext *context*) {

*return* MaterialApp(

*home*: DrawingAppPage(), *// เปลี่ยนชื่อคลาสเป็น DrawingAppPage*

    );

  }

}

Future<*void*> \_getJsonList() *async* {

  print((*await* drawingController.getImageData())?.buffer.asInt8List());

}

*class* DrawingAppPage *extends* StatefulWidget {

  @override

  \_DrawingAppPageState createState() => \_DrawingAppPageState();

}

*class* \_DrawingAppPageState *extends* State<DrawingAppPage> {

  @override

*void* initState() {

*//ฟั่งที่จะทำงานเมื่อเริ่มโปรแกรมหรือเริ่มหน้า*

*super*.initState();

    \_resetState();

    GetJsonDrawing(editFileName); *//ส่งชื่อไฟล์สำหรับการแก้ไขรูป*

  }

*void* \_resetState() {

*//รีเช็ต drawingController*

    drawingController.setStyle(

*color*: selectedColor,

*strokeWidth*: strokeWidth,

*isAntiAlias*: true,

*style*: PaintingStyle.stroke,

    );

  }

*void* \_saveDrawing() {

*// เพิ่ม drawingController ลงในลิสต์*

    savedControllers.add(drawingController);

*// บันทึกภาพตามที่คุณต้องการ*

    saveDrawing(

*context*: context,

*globalKey*: globalKey,

*newDrawing*: false); *//ส่งข้อมูลต่างๆไปยังการ save*

  }

  @override

  Widget build(BuildContext *context*) {

*return* Scaffold(

*appBar*: AppBar(

*leading*: IconButton(

*icon*: Icon(

            MdiIcons.arrowLeftCircle,

*color*: ColorPalette.white,

*size*: 40,

          ),

*onPressed*: () {

            Navigator.pushReplacement(

*context*,

              MaterialPageRoute(

*builder*: (*context*) => MainPage(),

              ),

            );

          },

        ),

*title*: *const* Text(

          'Drawing Board Edit',

*style*:

              TextStyle(*color*: ColorPalette.white, *fontWeight*: FontWeight.bold),

        ),

*backgroundColor*: ColorPalette.black,

*actions*: [

          ZoomTapAnimation(

*//กำหนดอนิเมชั่นการกด*

*end*: 0.8,

*begin*: 1,

*child*: IconButton(

*icon*: Icon(

                MdiIcons.contentSaveAll,

*color*: ColorPalette.grey,

              ),

*onPressed*: \_saveDrawing,

            ),

          )

        ],

      ),

*body*: Stack(

*children*: [

          Positioned(

*child*:

                DrawingBoardPage(), *// เรียกใช้ DrawingBoardPage จากที่กำหนดไว้ จาก drawing\_board\_page.dart*

          ),

          Positioned(

*//ตำแหน่งเครื่องเมื่อ*

*bottom*: 0,

*child*: Container(

*padding*: *const* EdgeInsets.all(2),

*width*: MediaQuery.of(*context*).size.width,

*color*: ColorPalette.toolBg,

*child*: ToolDrawing(),

            ),

          ),

        ],

      ),

    );

  }

}

สร้างหน้า main หรือหน้าหลัก หรือหน้า index

*import* 'dart:io';

*import* 'package:drawnig\_flutter/app\_colors.dart';

*import* 'package:drawnig\_flutter/drawing.dart';

*import* 'package:drawnig\_flutter/edit\_drawing.dart';

*import* 'package:drawnig\_flutter/globalvariable.dart';

*import* 'package:flutter/material.dart';

*import* 'package:material\_design\_icons\_flutter/material\_design\_icons\_flutter.dart';

*import* 'package:zoom\_tap\_animation/zoom\_tap\_animation.dart';

*void* main() {

  runApp(Main());

}

*class* Main *extends* StatelessWidget {

  @override

  Widget build(BuildContext *context*) {

*return* MaterialApp(

*home*: MainPage(),

    );

  }

}

*class* MainPage *extends* StatefulWidget {

  @override

  \_MainPageState createState() => \_MainPageState();

}

*class* \_MainPageState *extends* State<MainPage> {

  @override

*void* initState() {

*super*.initState();

    \_loadFileNames();

  }

  List<String> \_fileNames = [];

  Future<*void*> \_loadFileNames() *async* {

*//โหลดชื่อไฟล์ json ที่บบันทึกไว้*

*try* {

*final* directory = Directory.systemTemp; *// ใช้ไดเรกทอรี temp ของระบบ*

*final* files = directory.listSync();

      setState(() {

        \_fileNames = files

            .where((*file*) => *file* *is* File && *file*.path.endsWith('.json'))

            .map((*file*) {

*final* fileNameWithExtension = *file*.uri.pathSegments.last;

*// แยกชื่อไฟล์จากนามสกุล*

*final* fileName = fileNameWithExtension.split('.').first;

*return* fileName;

        }).toList();

      });

    } *catch* (e) {

      print('เกิดข้อผิดพลาดในการดึงรายชื่อไฟล์: $*e*');

    }

  }

*//ฟังชั่นในการ ลบ ไฟล์ json ที่บันทึกไว้*

  Future<*void*> \_deleteFile(String *fileName*) *async* {

*try* {

*final* directory = Directory.systemTemp; *// ใช้ไดเรกทอรี temp ของระบบ*

*final* filePath = '${*directory*.*path*}/$*fileName*.json';

*final* file = File(filePath);

*// ตรวจสอบว่าไฟล์มีอยู่หรือไม่*

*if* (*await* file.exists()) {

*await* file.delete();

        print('ไฟล์ $*fileName*.json ถูกลบเรียบร้อยแล้ว');

*// อัปเดตรายชื่อไฟล์หลังจากลบ*

*await* \_loadFileNames();

      } *else* {

        print('ไฟล์ $*fileName*.json ไม่พบ');

      }

    } *catch* (e) {

      print('เกิดข้อผิดพลาดในการลบไฟล์: $*e*');

    }

  }

  @override

  Widget build(BuildContext *context*) {

*return* Scaffold(

*appBar*: AppBar(

*backgroundColor*: ColorPalette.black,

*title*: *const* Text('Drawing',

*style*: TextStyle(

*color*: ColorPalette.white, *fontWeight*: FontWeight.bold)),

      ),

*body*: ListView.builder(

*itemCount*: \_fileNames.length,

*itemBuilder*: (*context*, *index*) {

*return* ListTile(

*//แสดง list ไฟล์ ใช้ ListTile ที่ให้สำมารถเลือนได้*

*title*: Container(

*padding*: EdgeInsets.all(10),

*decoration*: *const* BoxDecoration(

*borderRadius*: BorderRadius.all(Radius.circular(10)),

*color*: ColorPalette.black,

            ),

*child*: Row(

*mainAxisAlignment*: MainAxisAlignment.spaceBetween,

*children*: [

                Container(

*decoration*: *const* BoxDecoration(

*borderRadius*: BorderRadius.all(Radius.circular(15)),

*color*: ColorPalette.black,

                  ),

*child*: TextButton.icon(

*//เมื่อกดเลือกไฟล์ที่จะแก้ไข*

*onPressed*: () {

                      editFileName =

                          \_fileNames[*index*]; *//กำหนดชื่อไฟล์ที่จะใช้ในการแก้ไข*

                      Navigator.pushReplacement(

*//ลิ้งไปหน้าแก้ไข*

*context*,

                        MaterialPageRoute(

*builder*: (*context*) => EditDrawingApp(),

                        ),

                      );

                    },

*icon*: Icon(

                      MdiIcons.imageEdit,

*color*: ColorPalette.white.withOpacity(0.8),

                    ),

*label*: Text(

*//แสดงชื่อไฟล์ที่จะแก้ไข*

                      \_fileNames[*index*],

*overflow*: TextOverflow.ellipsis,

*maxLines*: 1,

*style*: *const* TextStyle(*color*: ColorPalette.white),

                    ),

                  ),

                ),

                Container(

*//การลบ*

*decoration*: BoxDecoration(

*borderRadius*: BorderRadius.all(Radius.circular(15)),

*color*: ColorPalette.grey.withOpacity(0.3),

                  ),

*child*: IconButton(

*onPressed*: () {

                        \_deleteFile(\_fileNames[*index*]); *//ลบไฟล์ทีี่เลือก*

                      },

*icon*: Icon(

*//แสดง  iconการลบ*

                        MdiIcons.deleteAlert,

*color*: ColorPalette.red.withOpacity(0.8),

*size*: 28,

                      )),

                )

              ],

            ),

          ));

        },

      ),

*floatingActionButtonLocation*:

          FloatingActionButtonLocation.centerDocked, *//ปุ่มการสร้างหน้าวาด*

*floatingActionButton*: FloatingActionButton(

*onPressed*: () {

            Navigator.pushReplacement(

*//ลิ้งไปหน้าการวาด*

*context*,

                MaterialPageRoute(*builder*: (*context*) => DrawingApp()));

          },

*backgroundColor*: ColorPalette.black,

*foregroundColor*: ColorPalette.white,

*elevation*: 0,

*shape*: *const* RoundedRectangleBorder(

*borderRadius*: BorderRadius.all(Radius.circular(22)), *// กำหนดขอบโค้ง*

          ),

*child*: *const* ZoomTapAnimation(

*end*: 0.8,

*begin*: 1,

*child*: Icon(

                Icons.add\_to\_photos\_rounded,

*size*: 30,

              ))),

*bottomNavigationBar*: *const* BottomAppBar(

*shape*: CircularNotchedRectangle(),

*color*: ColorPalette.black,

*notchMargin*: 15, *//การเว้นตรงกลาง สำหรับ floatingActionButtonLocation*

*child*: Row(

*mainAxisAlignment*: MainAxisAlignment.spaceBetween,

*children*: [],

        ),

      ),

    );

  }

}

1. ปัญหาอุปสรรค และวิธีการแก้ไข ที่พบในขั้นตอนต่างๆ ในการจัดทำ

ปัญหาเกี่ยวกับproblem occurred configuring project ':gallery\_saver'. วิธีแก้ ใช้ตัวอื่น

1. บันทึก Code ทั้งหมด
2. *import* 'dart:io';
3. *import* 'package:drawnig\_flutter/app\_colors.dart';
4. *import* 'package:drawnig\_flutter/drawing.dart';
5. *import* 'package:drawnig\_flutter/edit\_drawing.dart';
6. *import* 'package:drawnig\_flutter/globalvariable.dart';
7. *import* 'package:flutter/material.dart';
8. *import* 'package:material\_design\_icons\_flutter/material\_design\_icons\_flutter.dart';
9. *import* 'package:zoom\_tap\_animation/zoom\_tap\_animation.dart';
10. *void* main() {
11. runApp(Main());
12. }
13. *class* Main *extends* StatelessWidget {
14. @override
15. Widget build(BuildContext *context*) {
16. *return* MaterialApp(
17. *home*: MainPage(),
18. );
19. }
20. }
21. *class* MainPage *extends* StatefulWidget {
22. @override
23. \_MainPageState createState() => \_MainPageState();
24. }
25. *class* \_MainPageState *extends* State<MainPage> {
26. @override
27. *void* initState() {
28. *super*.initState();
29. \_loadFileNames();
30. }
31. List<String> \_fileNames = [];
32. Future<*void*> \_loadFileNames() *async* {
33. *//โหลดชื่อไฟล์ json ที่บบันทึกไว้*
34. *try* {
35. *final* directory = Directory.systemTemp; *// ใช้ไดเรกทอรี temp ของระบบ*
36. *final* files = directory.listSync();
37. setState(() {
38. \_fileNames = files
39. .where((*file*) => *file* *is* File && *file*.path.endsWith('.json'))
40. .map((*file*) {
41. *final* fileNameWithExtension = *file*.uri.pathSegments.last;
42. *// แยกชื่อไฟล์จากนามสกุล*
43. *final* fileName = fileNameWithExtension.split('.').first;
44. *return* fileName;
45. }).toList();
46. });
47. } *catch* (e) {
48. print('เกิดข้อผิดพลาดในการดึงรายชื่อไฟล์: $*e*');
49. }
50. }
51. *//ฟังชั่นในการ ลบ ไฟล์ json ที่บันทึกไว้*
52. Future<*void*> \_deleteFile(String *fileName*) *async* {
53. *try* {
54. *final* directory = Directory.systemTemp; *// ใช้ไดเรกทอรี temp ของระบบ*
55. *final* filePath = '${*directory*.*path*}/$*fileName*.json';
56. *final* file = File(filePath);
57. *// ตรวจสอบว่าไฟล์มีอยู่หรือไม่*
58. *if* (*await* file.exists()) {
59. *await* file.delete();
60. print('ไฟล์ $*fileName*.json ถูกลบเรียบร้อยแล้ว');
61. *// อัปเดตรายชื่อไฟล์หลังจากลบ*
62. *await* \_loadFileNames();
63. } *else* {
64. print('ไฟล์ $*fileName*.json ไม่พบ');
65. }
66. } *catch* (e) {
67. print('เกิดข้อผิดพลาดในการลบไฟล์: $*e*');
68. }
69. }
70. @override
71. Widget build(BuildContext *context*) {
72. *return* Scaffold(
73. *appBar*: AppBar(
74. *backgroundColor*: ColorPalette.black,
75. *title*: *const* Text('Drawing',
76. *style*: TextStyle(
77. *color*: ColorPalette.white, *fontWeight*: FontWeight.bold)),
78. ),
79. *body*: ListView.builder(
80. *itemCount*: \_fileNames.length,
81. *itemBuilder*: (*context*, *index*) {
82. *return* ListTile(
83. *//แสดง list ไฟล์ ใช้ ListTile ที่ให้สำมารถเลือนได้*
84. *title*: Container(
85. *padding*: EdgeInsets.all(10),
86. *decoration*: *const* BoxDecoration(
87. *borderRadius*: BorderRadius.all(Radius.circular(10)),
88. *color*: ColorPalette.black,
89. ),
90. *child*: Row(
91. *mainAxisAlignment*: MainAxisAlignment.spaceBetween,
92. *children*: [
93. Container(
94. *decoration*: *const* BoxDecoration(
95. *borderRadius*: BorderRadius.all(Radius.circular(15)),
96. *color*: ColorPalette.black,
97. ),
98. *child*: TextButton.icon(
99. *//เมื่อกดเลือกไฟล์ที่จะแก้ไข*
100. *onPressed*: () {
101. editFileName =
102. \_fileNames[*index*]; *//กำหนดชื่อไฟล์ที่จะใช้ในการแก้ไข*
103. Navigator.pushReplacement(
104. *//ลิ้งไปหน้าแก้ไข*
105. *context*,
106. MaterialPageRoute(
107. *builder*: (*context*) => EditDrawingApp(),
108. ),
109. );
110. },
111. *icon*: Icon(
112. MdiIcons.imageEdit,
113. *color*: ColorPalette.white.withOpacity(0.8),
114. ),
115. *label*: Text(
116. *//แสดงชื่อไฟล์ที่จะแก้ไข*
117. \_fileNames[*index*],
118. *overflow*: TextOverflow.ellipsis,
119. *maxLines*: 1,
120. *style*: *const* TextStyle(*color*: ColorPalette.white),
121. ),
122. ),
123. ),
124. Container(
125. *//การลบ*
126. *decoration*: BoxDecoration(
127. *borderRadius*: BorderRadius.all(Radius.circular(15)),
128. *color*: ColorPalette.grey.withOpacity(0.3),
129. ),
130. *child*: IconButton(
131. *onPressed*: () {
132. \_deleteFile(\_fileNames[*index*]); *//ลบไฟล์ทีี่เลือก*
133. },
134. *icon*: Icon(
135. *//แสดง  iconการลบ*
136. MdiIcons.deleteAlert,
137. *color*: ColorPalette.red.withOpacity(0.8),
138. *size*: 28,
139. )),
140. )
141. ],
142. ),
143. ));
144. },
145. ),
146. *floatingActionButtonLocation*:
147. FloatingActionButtonLocation.centerDocked, *//ปุ่มการสร้างหน้าวาด*
148. *floatingActionButton*: FloatingActionButton(
149. *onPressed*: () {
150. Navigator.pushReplacement(
151. *//ลิ้งไปหน้าการวาด*
152. *context*,
153. MaterialPageRoute(*builder*: (*context*) => DrawingApp()));
154. },
155. *backgroundColor*: ColorPalette.black,
156. *foregroundColor*: ColorPalette.white,
157. *elevation*: 0,
158. *shape*: *const* RoundedRectangleBorder(
159. *borderRadius*: BorderRadius.all(Radius.circular(22)), *// กำหนดขอบโค้ง*
160. ),
161. *child*: *const* ZoomTapAnimation(
162. *end*: 0.8,
163. *begin*: 1,
164. *child*: Icon(
165. Icons.add\_to\_photos\_rounded,
166. *size*: 30,
167. ))),
168. *bottomNavigationBar*: *const* BottomAppBar(
169. *shape*: CircularNotchedRectangle(),
170. *color*: ColorPalette.black,
171. *notchMargin*: 15, *//การเว้นตรงกลาง สำหรับ floatingActionButtonLocation*
172. *child*: Row(
173. *mainAxisAlignment*: MainAxisAlignment.spaceBetween,
174. *children*: [],
175. ),
176. ),
177. );
178. }
179. }
180. *import* 'package:drawnig\_flutter/main.dart';
181. *import* 'package:drawnig\_flutter/model/json\_drawing.dart';
182. *import* 'package:flutter/material.dart';
183. *import* 'package:flutter/services.dart';
184. *import* 'package:material\_design\_icons\_flutter/material\_design\_icons\_flutter.dart';
185. *import* 'package:drawnig\_flutter/app\_colors.dart';
186. *import* 'package:drawnig\_flutter/globalvariable.dart';
187. *import* 'package:drawnig\_flutter/model/drawing\_board\_page.dart';
188. *import* 'package:drawnig\_flutter/model/image\_save.dart';
189. *import* 'package:drawnig\_flutter/model/tool\_drawing.dart';
190. *import* 'package:zoom\_tap\_animation/zoom\_tap\_animation.dart';
191. *void* main() {
192. runApp(EditDrawingApp());
193. }
194. *class* EditDrawingApp *extends* StatelessWidget {
195. @override
196. Widget build(BuildContext *context*) {
197. *return* MaterialApp(
198. *home*: DrawingAppPage(), *// เปลี่ยนชื่อคลาสเป็น DrawingAppPage*
199. );
200. }
201. }
202. Future<*void*> \_getJsonList() *async* {
203. print((*await* drawingController.getImageData())?.buffer.asInt8List());
204. }
205. *class* DrawingAppPage *extends* StatefulWidget {
206. @override
207. \_DrawingAppPageState createState() => \_DrawingAppPageState();
208. }
209. *class* \_DrawingAppPageState *extends* State<DrawingAppPage> {
210. @override
211. *void* initState() {
212. *//ฟั่งที่จะทำงานเมื่อเริ่มโปรแกรมหรือเริ่มหน้า*
213. *super*.initState();
214. \_resetState();
215. GetJsonDrawing(editFileName); *//ส่งชื่อไฟล์สำหรับการแก้ไขรูป*
216. }
217. *void* \_resetState() {
218. *//รีเช็ต drawingController*
219. drawingController.setStyle(
220. *color*: selectedColor,
221. *strokeWidth*: strokeWidth,
222. *isAntiAlias*: true,
223. *style*: PaintingStyle.stroke,
224. );
225. }
226. *void* \_saveDrawing() {
227. *// เพิ่ม drawingController ลงในลิสต์*
228. savedControllers.add(drawingController);
229. *// บันทึกภาพตามที่คุณต้องการ*
230. saveDrawing(
231. *context*: context,
232. *globalKey*: globalKey,
233. *newDrawing*: false); *//ส่งข้อมูลต่างๆไปยังการ save*
234. }
235. @override
236. Widget build(BuildContext *context*) {
237. *return* Scaffold(
238. *appBar*: AppBar(
239. *leading*: IconButton(
240. *icon*: Icon(
241. MdiIcons.arrowLeftCircle,
242. *color*: ColorPalette.white,
243. *size*: 40,
244. ),
245. *onPressed*: () {
246. Navigator.pushReplacement(
247. *context*,
248. MaterialPageRoute(
249. *builder*: (*context*) => MainPage(),
250. ),
251. );
252. },
253. ),
254. *title*: *const* Text(
255. 'Drawing Board Edit',
256. *style*:
257. TextStyle(*color*: ColorPalette.white, *fontWeight*: FontWeight.bold),
258. ),
259. *backgroundColor*: ColorPalette.black,
260. *actions*: [
261. ZoomTapAnimation(
262. *//กำหนดอนิเมชั่นการกด*
263. *end*: 0.8,
264. *begin*: 1,
265. *child*: IconButton(
266. *icon*: Icon(
267. MdiIcons.contentSaveAll,
268. *color*: ColorPalette.grey,
269. ),
270. *onPressed*: \_saveDrawing,
271. ),
272. )
273. ],
274. ),
275. *body*: Stack(
276. *children*: [
277. Positioned(
278. *child*:
279. DrawingBoardPage(), *// เรียกใช้ DrawingBoardPage จากที่กำหนดไว้ จาก drawing\_board\_page.dart*
280. ),
281. Positioned(
282. *//ตำแหน่งเครื่องเมื่อ*
283. *bottom*: 0,
284. *child*: Container(
285. *padding*: *const* EdgeInsets.all(2),
286. *width*: MediaQuery.of(*context*).size.width,
287. *color*: ColorPalette.toolBg,
288. *child*: ToolDrawing(),
289. ),
290. ),
291. ],
292. ),
293. );
294. }
295. }
296. *import* 'package:drawnig\_flutter/main.dart';
297. *import* 'package:drawnig\_flutter/model/image\_save.dart';
298. *import* 'package:flutter/cupertino.dart';
299. *import* 'package:flutter/material.dart';
300. *import* 'package:flutter/widgets.dart';
301. *import* 'package:flutter\_drawing\_board/flutter\_drawing\_board.dart';
302. *import* 'package:material\_design\_icons\_flutter/material\_design\_icons\_flutter.dart';
303. *import* 'package:drawnig\_flutter/app\_colors.dart';
304. *import* 'package:drawnig\_flutter/globalvariable.dart';
305. *import* 'package:drawnig\_flutter/model/drawing\_board\_page.dart';
306. *import* 'package:drawnig\_flutter/model/tool\_drawing.dart';
307. *import* 'package:zoom\_tap\_animation/zoom\_tap\_animation.dart';
308. *void* main() {
309. runApp(DrawingApp());
310. }
311. *class* DrawingApp *extends* StatelessWidget {
312. @override
313. Widget build(BuildContext *context*) {
314. *return* MaterialApp(
315. *home*:
316. DrawingAppPage(), *// เปลี่ยนชื่อคลาสให้ตรงกับ StatefulWidget ที่ถูกต้อง*
317. );
318. }
319. }
320. *class* DrawingAppPage *extends* StatefulWidget {
321. *// เปลี่ยนชื่อคลาส StatefulWidget เป็น DrawingAppPage*
322. @override
323. \_DrawingAppPageState createState() => \_DrawingAppPageState();
324. }
325. *class* \_DrawingAppPageState *extends* State<DrawingAppPage> {
326. @override
327. *void* initState() {
328. *super*.initState();
329. drawingController = DrawingController(); *//กำหนด drawingController เป็นค่าเริ่มต้น*
330. \_resetState();
331. }
332. *void* \_resetState() { *//รีเช็ต การกระดานวาด*
333. drawingController.setStyle(
334. *color*: selectedColor,
335. *strokeWidth*: strokeWidth,
336. *isAntiAlias*: true,
337. *style*: PaintingStyle.stroke,
338. );
339. }
340. *void* \_saveDrawing() {
341. *// เพิ่ม drawingController ลงในลิสต์*
342. savedControllers.add(drawingController);
343. *// บันทึกภาพตามที่คุณต้องการ*
344. saveDrawing(*context*: context, *globalKey*: globalKey, *newDrawing*: true); *//ส่งข้อมูลต่างๆไปยังการ save*
345. }
346. @override
347. Widget build(BuildContext *context*) {
348. *return* Scaffold(
349. *appBar*: AppBar( *//appbar*
350. *leading*: IconButton(
351. *icon*: Icon(
352. MdiIcons.arrowLeftCircle,
353. *color*: ColorPalette.white,
354. *size*: 40,
355. ),
356. *onPressed*: () { *//ปุ่มย้อนกลับ*
357. Navigator.pushReplacement(
358. *context*,
359. MaterialPageRoute(
360. *builder*: (*context*) => MainPage(),
361. ),
362. );
363. },
364. ),
365. *title*: *const* Text(
366. 'Drawing Board',
367. *style*:
368. TextStyle(*color*: ColorPalette.white, *fontWeight*: FontWeight.bold),
369. ),
370. *backgroundColor*: ColorPalette.black,
371. *actions*: [
372. ZoomTapAnimation(*//กำหนดอนิเมชั่นการกดเป็นการซูมเมื่อกด จาก แพกเก็จ ZoomTapAnimation*
373. *end*: 0.8, *// เมื่อกดย่อ icon = 0.8*
374. *begin*: 1,*//เริ่มต้นของขนาด icon*
375. *child*: IconButton( *//ปุ่ม save*
376. *icon*: Icon(
377. MdiIcons.contentSaveAll,
378. *color*: ColorPalette.grey, *// ใช้สีพื้นหลังจาก ColorPalette*
379. ),
380. *onPressed*: \_saveDrawing, *//เรียกใช้ฟั่งชั่น save*
381. ),
382. )
383. ],
384. ),
385. *body*: Stack( *//Stack ใช้ในการจัด  Positioned ของหน้าจอต่างๆ เช่น ตำแหน่งเครื่องมือ*
386. *children*: [
387. Positioned(
388. *child*: DrawingBoardPage(),*//เรียกใช้ กระดานวาด จาก  drawing\_board\_page.dart ที่สร้างไว้*
389. ),
390. Positioned( *//กำหนดตำแหนน่งของเครื่องมือ*
391. *bottom*: 0,
392. *child*: Container(
393. *padding*: *const* EdgeInsets.all(2),
394. *width*: MediaQuery.of(*context*).size.width,
395. *color*: ColorPalette.toolBg, *// ใช้สีพื้นหลังจาก ColorPalette*
396. *child*: ToolDrawing(),*//เรียกใช้เครื่องมือ tool\_drawing.dart ที่เตรียมไว้*
397. ),
398. ),
399. ],
400. ),
401. );
402. }
403. }

*import* 'package:flutter/material.dart';

*// Define color constants*

*class* ColorPalette { *//กำหนดตัวแปรสำหรับสีที่ใช้ในแอพ*

*static* *const* Color black = Colors.black;

*static* *const* Color white = Colors.white;

*static* *const* Color red = Colors.red;

*static* *const* Color green = Colors.green;

*static* *const* Color blue = Colors.blue;

*static* *const* Color yellow = Colors.yellow;

*static* *const* Color pink = Colors.pink;

*static* *const* Color purple = Colors.purple;

*static* *const* Color cyanAccent = Colors.cyanAccent;

*static* *const* Color brown = Colors.brown;

*static* *const* Color orange = Colors.orange;

*static* *const* Color grey = Colors.grey;

*static* *const* Color darkGrey = Color.fromARGB(255, 133, 133, 133);

*static* *const* Color primary = Color.fromARGB(255, 219, 219, 219);

*static* *const* Color toolBg = Color.fromARGB(255, 32, 32, 32);

}

*import* 'package:flutter/material.dart';

*import* 'package:flutter\_drawing\_board/flutter\_drawing\_board.dart';

List<DrawingController> savedControllers =

    []; *//ตัวแปรที่ไว้เก็บการ save drawingControllers*

*final* GlobalKey globalKey = GlobalKey(); *//key ในการบันทึกรูป*

DrawingController drawingController =

    DrawingController(); *//ตัวแปร drawingController*

DrawingController jsonCon =

    DrawingController(); *//ตัวแปรที่ไว้เก็บ json ของ drawingController ที่โหลดมาจากการบันทึก*

String editFileName =

    ""; *//ตัวแปรที่ไว้เก็บชื่อ ไฟล์ json ที่จะใช้ในการแก้ไขรูปวาด*

*// bool isActive = false; //ไว้ ในการเช็คว่าเลือก เครื่องมือในการวาดตัวไหน*

Color selectedColor = Colors.blue; *//ตัวแปรในการเลือกสีในการวาด*

double strokeWidth = 2.0; *//ตัวแปรขนาดของเส้นในการวาด*

*enum* ToolType {

*//ตัวแปรที่ไว้เก็บประเภทของเครื่องมือ*

  none,

  eraser,

  strokeWidth,

  pencil

}

*import* 'package:drawnig\_flutter/globalvariable.dart';

*import* 'package:flutter/material.dart';

*import* 'package:flutter\_drawing\_board/flutter\_drawing\_board.dart';

*void* main() {

  runApp(DrawingBoardPage());

}

*class* DrawingBoardPage *extends* StatefulWidget {

  @override

  \_DrawingBoardPageState createState() => \_DrawingBoardPageState();

}

*class* \_DrawingBoardPageState *extends* State<DrawingBoardPage> {

  @override

  Widget build(BuildContext *context*) {

*return* Scaffold(

*body*: LayoutBuilder(

*builder*: (*context*, *constraints*) {

*return* RepaintBoundary(

*//ใช้ในการบันทึกรูปที่เขียน*

*key*: globalKey, *//key ที่ใช้ในการบันทึกรูป*

*child*: DrawingBoard(

*controller*: drawingController, *//ตัว controller สำหรับการวาด*

*background*: Container(

*width*: *constraints*.maxWidth, *//กำหนดขนาด กระดานวาด*

*height*: *constraints*.maxHeight,

*color*: Colors.white, *//กำหนดสีพื้นหลังกระดานวาด*

              ),

*showDefaultTools*: false, *//ปิดการใช้ เครื่องมือพื้นฐาน ต่างๆ*

*showDefaultActions*: false,

            ),

          );

        },

      ),

    );

  }

}

*import* 'package:json\_annotation/json\_annotation.dart';

*part* 'drawing\_model.g.dart';

*//หน้าในการ แปลงไฟล์ json เป็น obj ของ dart โดยใช้ json\_serializable*

@JsonSerializable()

*class* PaintData { *//ค่าต่างๆในไฟล์ json ที่วาด เช่น สี ขนาดเส้น*

*final* int blendMode;

*final* int color;

*final* int filterQuality;

*final* bool invertColors;

*final* bool isAntiAlias;

*final* int strokeCap;

*final* int strokeJoin;

*final* double strokeWidth;

*final* int style;

  PaintData({

*required* *this*.blendMode,

*required* *this*.color,

*required* *this*.filterQuality,

*required* *this*.invertColors,

*required* *this*.isAntiAlias,

*required* *this*.strokeCap,

*required* *this*.strokeJoin,

*required* *this*.strokeWidth,

*required* *this*.style,

  });

*factory* PaintData.fromJson(Map<String, dynamic> *json*) => \_$PaintDataFromJson(*json*);

  Map<String, dynamic> toJson() => \_$PaintDataToJson(*this*);

}

@JsonSerializable()

*class* StepData { *//ประเภท //ตำแหน่ง x y*

*final* String type;

*final* double x;

*final* double y;

  StepData({

*required* *this*.type,

*required* *this*.x,

*required* *this*.y,

  });

*factory* StepData.fromJson(Map<String, dynamic> *json*) => \_$StepDataFromJson(*json*);

  Map<String, dynamic> toJson() => \_$StepDataToJson(*this*);

}

@JsonSerializable()

*class* PathData {

*final* int fillType;

*final* List<StepData> steps;

  PathData({

*required* *this*.fillType,

*required* *this*.steps,

  });

*factory* PathData.fromJson(Map<String, dynamic> *json*) => \_$PathDataFromJson(*json*); *//แปลงข้อมูลต่างๆ*

  Map<String, dynamic> toJson() => \_$PathDataToJson(*this*);

}

@JsonSerializable()

*class* Simple {

*final* String type;

*final* PathData path;

*final* PaintData paint;

  Simple({

*required* *this*.type,

*required* *this*.path,

*required* *this*.paint,

  });

*factory* Simple.fromJson(Map<String, dynamic> *json*) => \_$SimpleFromJson(*json*);

  Map<String, dynamic> toJson() => \_$SimpleToJson(*this*);

}

*// GENERATED CODE - DO NOT MODIFY BY HAND*

*part of* 'drawing\_model.dart';

*// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*// JsonSerializableGenerator*

*// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

PaintData \_$PaintDataFromJson(Map<String, dynamic> *json*) => PaintData(

*blendMode*: (*json*['blendMode'] *as* num).toInt(),

*color*: (*json*['color'] *as* num).toInt(),

*filterQuality*: (*json*['filterQuality'] *as* num).toInt(),

*invertColors*: *json*['invertColors'] *as* bool,

*isAntiAlias*: *json*['isAntiAlias'] *as* bool,

*strokeCap*: (*json*['strokeCap'] *as* num).toInt(),

*strokeJoin*: (*json*['strokeJoin'] *as* num).toInt(),

*strokeWidth*: (*json*['strokeWidth'] *as* num).toDouble(),

*style*: (*json*['style'] *as* num).toInt(),

    );

Map<String, dynamic> \_$PaintDataToJson(PaintData *instance*) => <String, dynamic>{

      'blendMode': *instance*.blendMode,

      'color': *instance*.color,

      'filterQuality': *instance*.filterQuality,

      'invertColors': *instance*.invertColors,

      'isAntiAlias': *instance*.isAntiAlias,

      'strokeCap': *instance*.strokeCap,

      'strokeJoin': *instance*.strokeJoin,

      'strokeWidth': *instance*.strokeWidth,

      'style': *instance*.style,

    };

StepData \_$StepDataFromJson(Map<String, dynamic> *json*) => StepData(

*type*: *json*['type'] *as* String,

*x*: (*json*['x'] *as* num).toDouble(),

*y*: (*json*['y'] *as* num).toDouble(),

    );

Map<String, dynamic> \_$StepDataToJson(StepData *instance*) => <String, dynamic>{

      'type': *instance*.type,

      'x': *instance*.x,

      'y': *instance*.y,

    };

PathData \_$PathDataFromJson(Map<String, dynamic> *json*) => PathData(

*fillType*: (*json*['fillType'] *as* num).toInt(),

*steps*: (*json*['steps'] *as* List<dynamic>)

          .map((*e*) => StepData.fromJson(*e* *as* Map<String, dynamic>))

          .toList(),

    );

Map<String, dynamic> \_$PathDataToJson(PathData *instance*) => <String, dynamic>{

      'fillType': *instance*.fillType,

      'steps': *instance*.steps,

    };

Simple \_$SimpleFromJson(Map<String, dynamic> *json*) => Simple(

*type*: *json*['type'] *as* String,

*path*: PathData.fromJson(*json*['path'] *as* Map<String, dynamic>),

*paint*: PaintData.fromJson(*json*['paint'] *as* Map<String, dynamic>),

    );

Map<String, dynamic> \_$SimpleToJson(Simple *instance*) => <String, dynamic>{

      'type': *instance*.type,

      'path': *instance*.path,

      'paint': *instance*.paint,

    };

*import* 'dart:convert';

*import* 'dart:io';

*import* 'dart:ui' *as* ui;

*import* 'package:drawnig\_flutter/globalvariable.dart';

*import* 'package:drawnig\_flutter/main.dart';

*import* 'package:flutter/rendering.dart';

*import* 'package:path\_provider/path\_provider.dart';

*import* 'package:photo\_manager/photo\_manager.dart';

*import* 'package:flutter/material.dart';

*import* 'package:intl/intl.dart';

Future<*void*> saveDrawing({

*required* BuildContext *context*,

*required* bool *newDrawing*,

}) *async* {

*try* {

*// รับข้อมูลภาพจาก DrawingController*

*final* byteData = *await* drawingController.getImageData();

*// ตรวจสอบว่า byteData เป็น null หรือไม่*

*if* (byteData != *null*) {

*final* buffer =

          byteData.buffer.asUint8List(); *// แปลง ByteData เป็น Uint8List*

*// ดึงที่อยู่ของ directory ที่ใช้ในการบันทึกรูป*

*final* directory = *await* getTemporaryDirectory();

*final* now = DateTime.now(); *// รับวันที่และเวลา*

*final* timestamp =

          '${*now*.*year*}-${*now*.*month*}-${*now*.*day*}\_${*now*.*hour*}-${*now*.*minute*}-${*now*.*second*}';

*final* path =

          '${*directory*.*path*}/drawing\_image\_$*timestamp*.png'; *// กำหนดเส้นทางไฟล์*

*final* file = File(path); *// สร้างไฟล์ที่เส้นทางกำหนด*

*await* file.writeAsBytes(buffer); *// เขียนข้อมูลลงในไฟล์*

*// ใช้ PhotoManager ในการบันทึกรูปลงในอัลบั้มภาพ*

*final* result = *await* PhotoManager.editor.saveImage(

        buffer, *// ใช้ buffer ที่ได้จาก getImageData*

*filename*: '/drawing\_image\_$*timestamp*.png', *// ตั้งชื่อไฟล์*

      );

*if* (result != *null*) {

*// เช็คว่าการบันทึกสำเร็จหรือไม่*

        ScaffoldMessenger.of(*context*).showSnackBar(

          SnackBar(*content*: Text('Save image')),

        );

*// ตรวจสอบว่าเป็นรูปใหม่หรือไม่*

*if* (*newDrawing*) {

*// สร้างวันที่และเวลาปัจจุบัน*

*final* formatter = DateFormat('yyyyMMdd\_HHmmss');

*final* dateTimeString = formatter.format(DateTime.now());

          \_SaveJsonListNew(dateTimeString); *// บันทึก JSON พร้อมชื่อใหม่*

        } *else* {

          \_SaveJsonListNew(editFileName); *// บันทึก JSON ด้วยชื่อเดิม*

        }

        Future.delayed(Duration(*seconds*: 2), () {

          Navigator.pushReplacement(

*context*,

            MaterialPageRoute(

*builder*: (*context*) => Main()), *// เปลี่ยนหน้าหลังจากบันทึก*

          );

        });

      } *else* {

        ScaffoldMessenger.of(*context*).showSnackBar(

          SnackBar(*content*: Text('Failed to save image')),

        );

      }

    } *else* {

      ScaffoldMessenger.of(*context*).showSnackBar(

        SnackBar(*content*: Text('Failed to retrieve image data')),

      );

    }

  } *catch* (e) {

    ScaffoldMessenger.of(*context*).showSnackBar(

      SnackBar(*content*: Text('Failed to save image: $*e*')),

    );

  }

}

*//ฟังชั่นในการ save json*

Future<*void*> \_SaveJsonListNew(String *name*) *async* {

*try* {

*// รับข้อมูล JSON จาก DrawingController*

    String jsonString = *const* JsonEncoder.withIndent('  ')

        .convert(drawingController.getJsonList());

*// กำหนดพาธสำหรับบันทึกไฟล์ (เปลี่ยนพาธตามต้องการ)*

*final* directory = Directory.systemTemp; *// ใช้ไดเรกทอรี temp ของระบบ*

*final* filePath = '${*directory*.*path*}/$*name*.json';

*// สร้างไฟล์และเขียนข้อมูล JSON ลงในไฟล์*

*final* file = File(filePath);

*await* file.writeAsString(jsonString);

    print('ข้อมูล JSON ถูกบันทึกลงในไฟล์ที่ $*filePath*');

  } *catch* (e) {

    print('เกิดข้อผิดพลาดในการบันทึกข้อมูล JSON: $*e*');

  }

}

*import* 'dart:convert';

*import* 'dart:io';

*import* 'package:drawnig\_flutter/globalvariable.dart';

*import* 'package:drawnig\_flutter/model/drawing\_model.dart';

*import* 'package:flutter\_drawing\_board/flutter\_drawing\_board.dart';

*import* 'package:flutter\_drawing\_board/paint\_contents.dart';

Future<*void*> GetJsonDrawing(String *name*) *async* {

*// กำหนดพาธของไฟล์ JSON ที่จะอ่าน (เปลี่ยนพาธตามต้องการ)*

*final* directory = Directory.systemTemp; *// ใช้ไดเรกทอรี temp ของระบบ*

*final* filePath = '${*directory*.*path*}/${*name*}.json';

*final* file = File(filePath);

  drawingController = DrawingController();

*// ตรวจสอบว่าไฟล์มีอยู่หรือไม่*

*if* (*await* file.exists()) {

*// อ่านข้อมูลจากไฟล์*

    String jsonString = *await* file.readAsString();

*// แปลงข้อมูล JSON เป็น Dart object*

*final* List<dynamic> jsonData = jsonDecode(jsonString);

*// แสดงข้อมูล JSON*

    print(jsonData);

*// final List<dynamic> jsonData = jsonDecode(jsonString);*

*final* List<Simple> drawings =

        jsonData.map((*item*) => Simple.fromJson(*item*)).toList();

*//แปลงข้อมูลจาก json ที่ได้มาเพื่อทำให้ข้อมูลที่ได้ ตรงกันกับ ฟังชั่น addContent ของ flutter\_drawing\_board*

*for* (*var* drawing *in* drawings) {

*final* Map<String, dynamic> \_testLineSimple = {

        "type": drawing.type,

        "path": {

          "fillType": drawing.path.fillType,

          "steps": drawing.path.steps

              .map((*step*) => {

                    "type": *step*.type,

                    "x": *step*.x,

                    "y": *step*.y,

                  })

              .toList(),

        },

        "paint": {

          "blendMode": drawing.paint.blendMode,

          "color": drawing.paint.color,

          "filterQuality": drawing.paint.filterQuality,

          "invertColors": drawing.paint.invertColors,

          "isAntiAlias": drawing.paint.isAntiAlias,

          "strokeCap": drawing.paint.strokeCap,

          "strokeJoin": drawing.paint.strokeJoin,

          "strokeWidth": drawing.paint.strokeWidth,

          "style": drawing.paint.style,

        }

      };

*// print(\_testLineSimple);*

*// ตรวจสอบประเภทและแปลงเป็นวัตถุที่เหมาะสม*

*if* (\_testLineSimple["type"] == "SimpleLine") {

        drawingController.addContent(SimpleLine.fromJson(\_testLineSimple));

      } *else* *if* (\_testLineSimple["type"] == "Eraser") {

        drawingController.addContent(Eraser.fromJson(\_testLineSimple));

      }

    }

  } *else* {

    print('ไฟล์ JSON ไม่พบที่ $*filePath*');

  }

}

*import* 'package:drawnig\_flutter/app\_colors.dart';

*import* 'package:flutter/material.dart';

*import* 'package:drawnig\_flutter/globalvariable.dart';

*import* 'package:flutter\_drawing\_board/paint\_contents.dart';

*import* 'package:material\_design\_icons\_flutter/material\_design\_icons\_flutter.dart';

*import* 'package:flutter\_colorpicker/flutter\_colorpicker.dart';

*class* ToolDrawing *extends* StatefulWidget {

  @override

  \_ToolDrawingState createState() => \_ToolDrawingState();

}

*class* \_ToolDrawingState *extends* State<ToolDrawing> {

  ToolType \_activeTool = ToolType.pencil; *//กำหนดตัวแปร ToolType ให้เป็นปากกา*

  @override

  Widget build(BuildContext *context*) {

*return* Column(

*children*: [

        Row(

*mainAxisAlignment*: MainAxisAlignment.spaceBetween,

*children*: [

            Row(

*children*: [

*//สร้างเครื่องมือ ปากกา*

                Tooltip(

*message*: "Pencil",

*child*: IconButton(

*icon*: Icon(MdiIcons.pencil),

*color*: \_activeTool ==

                            ToolType

                                .pencil *//เช็คว่าเลือกไหมถ้าใช่ ให้เป็นสีที่เลือก*

                        ? selectedColor

                        : ColorPalette.darkGrey,

*onPressed*: () {

                      setState(() {

*//ใช้  setState เพื่อ อัปเดตการเลือกการแสดงผล*

                        drawingController.setPaintContent(

                            SimpleLine()); *//กำหนดให้เครื่องมือเป็น SimpleLine คือเส้นการวาดทั่วไป*

                        \_activeTool = ToolType.pencil;

                        drawingController.setStyle(

*//กำหนดสไตล์ ของเส้น*

*color*:

                              selectedColor, *//กำหนดสี ให้เป็นสีที่เลือกจาก selectedColor*

*strokeWidth*:

                              strokeWidth, *//กำหนดขนาดเส้นให้เป็นการเลือกจาก strokeWidth*

*isAntiAlias*: true, *//โหมดป้องกันรอยหยัก*

*style*: PaintingStyle

                              .fill, *//มีแบบ PaintingStyle.fill ลงสีทึบ*

                        );

                      });

                    },

                  ),

                ),

                Tooltip(

*message*: "Eraser",

*child*: IconButton(

*icon*: Icon(MdiIcons.eraserVariant),

*color*: \_activeTool ==

                            ToolType

                                .eraser *//เช็คว่าเลือกไหมถ้าใช่ ให้เป็นสีที่เลือก*

                        ? ColorPalette.primary

                        : ColorPalette.darkGrey,

*onPressed*: () {

                      setState(() {

                        \_activeTool = ToolType.eraser;

                        setState(() {

                          drawingController.setPaintContent(

                              Eraser()); *//กำหนดให้เครื่องมือเป็น ยางลบ*

                        });

                      });

                    },

                  ),

                ),

                Tooltip(

*message*: "StrokeWidth",

*child*: Container(

*width*: 100,

*child*: Slider(

*//สไลล์*

*value*: strokeWidth,

*min*: 0, *//ค่าของสไลล์ต่ำสุด*

*max*: 20, *//สูงสุด*

*divisions*:

                            20, *//การช่วง เช่น ในค่ามากสุด 0- 20 แบ่งเป็น 20 จากตัวค่า 1,2,3,4...20 แต่ถ้าเป็น 40 จะได้ 0.5,1,1.5,2...20*

*onChanged*: (*value*) {

                          setState(() {

                            strokeWidth =

*value*; *//กำหนด strokeWidth ให้ทำกับค่าที่เลือกจาก  สไลล์*

                            drawingController.setStyle(

*color*: selectedColor,

*strokeWidth*:

                                  strokeWidth, *//กำนาดขนาดเส้นเป็น strokeWidth*

*isAntiAlias*: true,

*style*: PaintingStyle.stroke,

                            );

                          });

                        },

*label*: strokeWidth.toStringAsFixed(2),

                      ),

                    )),

                SizedBox(

*width*: 50,

*child*: Tooltip(

*message*: "Selete Color",

*child*: ElevatedButton(

*style*: ElevatedButton.styleFrom(

*//กำสไตล์ให้ Tooltip ElevatedButton*

*backgroundColor*: Colors.transparent,

*shadowColor*: Colors.transparent,

                      ),

*onPressed*:

                          \_openColorPicker, *//เรียกใช้ฟังชั่น \_openColorPicker*

*child*: Icon(MdiIcons.palette, *//กำสไตล์ให้ icon*

*size*: 30,

*color*: selectedColor),

                    ),

                  ),

                ),

              ],

            ),

            Row(

*children*: [

                SizedBox(

*width*: 50,

*child*: Tooltip(

*message*: "Undo",

*child*: IconButton(

*icon*: Icon(MdiIcons.undo),

*color*: ColorPalette.darkGrey,

*onPressed*: () {

                        setState(() {

                          drawingController.undo(); *//กำหนดเครื่องมือให้เป็น undo*

                        });

                      },

                    ),

                  ),

                ),

                SizedBox(

*width*: 50,

*child*: Tooltip(

*message*: "Redo",

*child*: IconButton(

*icon*: Icon(MdiIcons.redo),

*color*: ColorPalette.darkGrey,

*onPressed*: () {

                        setState(() {

                          drawingController.redo();*//กำหนดเครื่องมือให้เป็น redo*

                        });

                      },

                    ),

                  ),

                ),

              ],

            )

          ],

        ),

      ],

    );

  }

*void* \_openColorPicker() {*//ฟังชั่นในการเลือกสี*

    showDialog(

*context*: context,

*builder*: (BuildContext *context*) {

*return* AlertDialog( *//เป็น popup เด้นขึ้นมา*

*title*: Text('Select a color'),

*content*: SingleChildScrollView( *//ทำให้เลือนได้*

*child*: BlockPicker(*//ถาดสีจาก package flutter\_colorpicker*

*pickerColor*: selectedColor, *// สีที่แสดงตอนเริ่ม*

*onColorChanged*: (Color *color*) {

                setState(() { *//กำหนด setState*

                  selectedColor = *color*; *//กำหนดสีให้เป็นสีที่เลือก*

                  drawingController.setStyle(

*color*: selectedColor, *//กำหนดสี*

*strokeWidth*: strokeWidth,

*isAntiAlias*: true,

*style*: PaintingStyle.stroke,

                  );

                });

              },

            ),

          ),

*actions*: <Widget>[ *// actions กำหนด ข้อความ Done เป็น TextButton เมือกด ปิดหน้า selectedColor หรือ ย้อนกลับมาหน้าวาดรูป*

            TextButton(

*child*: Text('Done'),

*onPressed*: () {

                Navigator.of(*context*).pop();

              },

            ),

          ],

        );

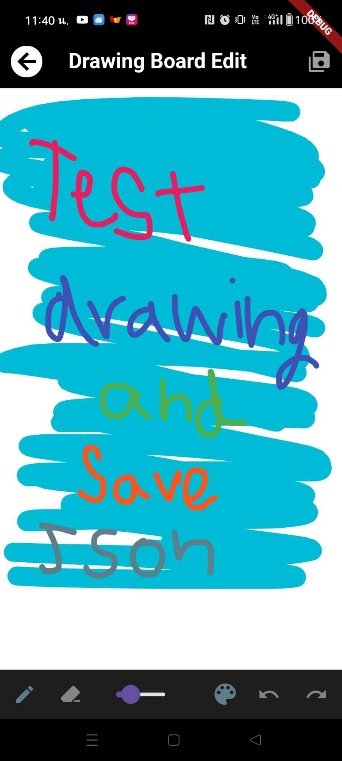
      },

    );

  }

}

12.บันทึกภาพผลการทำงานของ Application

การทำงานของโปรแกรม  ถูกต้อง  ไม่ถูกต้อง

**คำถามในแง่จริยธรรมในวิชาชีพ**

นักเรียนมีความซื่อสัตย์ ต่อตนเอง และผู้อื่น พยายามทำด้วยตนเอง และไม่ส่งคำตอบให้ผู้อื่น เพื่อลดความพยายามศึกษาของผู้อื่น ไม่ขอ หรือรับคำตอบจากผู้อื่น เพื่อด้อยค่าลดความสามารถของตนเอง แม้แต่ข้อเดียว

มีความซื่อสัตย์  ไม่มีความซื่อสัตย์

นักเรียนสามารถทำใบงานฉบับนี้ได้ด้วยตนเอง โดยไม่ต้องถามผู้อื่น หรือคัดลอกใช่หรือไม่

สามารถ  ไม่สามารถ

หากนักเรียนไม่สามารถทำได้ด้วยตนเอง ให้เขียนสาเหตุ หรือเหตุผลที่ไม่สามารถทำได้