ใบงานการทดลองที่ 8

เรื่อง Form Widgets

1. จุดประสงค์

1. สร้าง Form พื้นฐานได้
2. สร้าง Validate ข้อมูลใน Form ได้
3. สร้าง Submit Form ได้
4. ใช้ Widgets ที่เกี่ยวข้องกับ Form ได้

2. ทฤษฎี

บทที่ 4.1 Form

4.1.1 Forms Widget คืออะไร

ใน Flutter, Widget ที่เกี่ยวข้องกับการสร้างและจัดการฟอร์มมีชื่อว่า Form Widget ซึ่งเป็น Widget ที่มีความสำคัญในการสร้างแอปพลิเคชันที่มีการป้อนข้อมูลจากผู้ใช้ เช่น แบบฟอร์มสมัครสมาชิก แบบฟอร์มเข้าสู่ระบบ หรือแบบฟอร์มสำหรับกรอกข้อมูลต่าง ๆ นอกเหนือจากการเป็นตัวควบคุมสำหรับการป้อนข้อมูล Form Widget ยังเป็นตัวควบคุมสำหรับการจัดการกับสถานะของข้อมูลที่ป้อนเข้ามา และใช้สำหรับ Validate ข้อมูลก่อนการส่งหรือประมวลผล นอกจากนี้ยังสามารถใช้สำหรับเรียกใช้งานฟังก์ชันหรือการประมวลผลอื่น ๆ เมื่อผู้ใช้ทำการส่งข้อมูลหรือกดปุ่มต่าง ๆ ในฟอร์ม

4.1.2 ลักษณะสำคัญของ Form Widget

ลักษณะสำคัญของ Form Widget ได้แก่

- **การจัดรูปแบบและโครงสร้างของฟอร์ม**: Form Widget ช่วยให้คุณสามารถจัดการโครงสร้างของฟอร์มได้อย่างอิสระ โดยคุณสามารถเพิ่มและจัดวาง Widget ต่าง ๆ เช่น TextFormField, Checkbox, DropdownButton เป็นต้น ลงใน Form ในลักษณะที่คุณต้องการ
- **การจัดการสถานะของฟอร์มและข้อมูล**: Form Widget มีฟังก์ชันสำหรับจัดการสถานะของฟอร์ม ซึ่งสามารถใช้ในการ Validate ข้อมูล จัดเก็บค่าข้อมูล และดึงค่าข้อมูลที่ผู้ใช้ป้อนเข้ามาได้
- **การ Validate ข้อมูล**: Form Widget มีฟังก์ชัน validate() ซึ่งสามารถใช้ในการตรวจสอบความถูกต้องของข้อมูลที่ผู้ใช้ป้อนเข้ามาในฟอร์ม โดยสามารถกำหนดเงื่อนไขต่าง ๆ ให้กับข้อมูล เช่น ความยาวของข้อมูล รูปแบบของข้อมูล เป็นต้น
- **การ Submit ข้อมูล**: เมื่อข้อมูลในฟอร์มถูกต้องตามที่กำหนด คุณสามารถใช้ฟังก์ชัน save() เพื่อดึงข้อมูลที่ผู้ใช้ป้อนเข้ามาออกจากฟอร์มเพื่อนำไปใช้งานต่อไป หรือประมวลผลข้อมูลตามที่ต้องการ เช่น บันทึกข้อมูล ส่งข้อมูลไปยังเซิร์ฟเวอร์ เป็นต้น

การใช้ Form Widget ใน Flutter ช่วยให้การจัดการกับฟอร์มและข้อมูลที่ผู้ใช้ป้อนเข้ามาเป็นไปอย่างราบรื่นและมีประสิทธิภาพ นอกจากนี้ยังช่วยลดความซับซ้อนในการจัดการสถานะของข้อมูลและการ Validate ข้อมูลในแอปพลิเคชันด้วย

4.1.3 ประโยชน์ของการใช้ Forms

- ช่วยให้ผู้ใช้สามารถกรอกข้อมูลลงในระบบได้อย่างสะดวกความง่ายในการเรียนรู้
- ช่วยให้ตรวจสอบความถูกต้องของข้อมูลก่อนส่ง
- ช่วยให้จัดการข้อมูลได้ง่าย

4.1.4 TextFormField

TextFormField เป็น widget พื้นฐานที่ใช้สร้างช่องกรอกข้อมูลใน Flutter ทำงานร่วมกับระบบ Form ของ Flutter ช่วยให้ผู้ใช้สามารถป้อนข้อความสั้นๆ หรือตัวเลขได้ ตัวอย่างการใช้งานทั่วไป เช่น

- ช่องกรอกชื่อผู้ใช้
- ช่องกรอกรหัสผ่าน
- ช่องกรอกอีเมล
- ช่องกรอกเบอร์โทรศัพท์

คุณสมบัติหลักของ TextFormField

- **รองรับ Material Design**: แสดงผลตามแนวทาง Material Design ของ Google
- **ปรับแต่งได้หลากหลาย**: สามารถกำหนดขนาด สี รูปแบบตัวอักษร ไอคอน ฯลฯ
- **รองรับการตรวจสอบข้อมูล**: ตรวจสอบความถูกต้องของข้อมูลที่ป้อน เช่น รูปแบบอีเมล ความยาวของรหัสผ่าน
- **ทำงานร่วมกับ Form**: ผสานรวมกับระบบ Form ของ Flutter ตรวจสอบความถูกต้องของข้อมูลทั้งหมดใน Form

บทที่ 4.2 การใช้งาน Form Widget

4.2.1 Form Widget

Form เป็น Widget หลักสำหรับสร้าง Form โดยใช้ร่วมกับ Key เพื่อระบุ Form นั้นๆ

```
Form(
  key: _formKey,
  child: ... //widget ต่าง ๆ,
)
```

ภายใน Form Widget คุณสามารถเพิ่ม Widget อื่นๆ ได้ เช่น TextField, DropdownButton, Checkbox ฯลฯ Widget เหล่านี้จะใช้เพื่อแสดงฟิลด์ป้อนข้อมูลต่างๆ ให้กับผู้ใช้

```
Form(
child: Column(
 children: [
  TextFormField(
   decoration: InputDecoration(
    labelText: "ชื่อเล่น",
   ),),
  ElevatedButton(
   onPressed: () {},
   child: Text("Submit"),),],),)
```

ตัวอย่างนี้แสดงฟอร์มง่ายๆ ที่มีฟิลด์ชื่อผู้ใช้ และปุ่ม Submit

4.2.2 การใช้ Key กับ Form

Key ใน Form ของ Flutter เปรียบเสมือนรหัสประจำตัวของฟอร์ม ช่วยให้ Flutter สามารถระบุตัวตนของฟอร์มได้ Key นี้มีประโยชน์ในหลายกรณี เช่น

- การเข้าถึงสถานะของฟอร์ม: Key ช่วยให้เราสามารถเข้าถึงสถานะของฟอร์ม เช่น ข้อมูลที่ผู้ใช้กรอกเข้ามา ข้อผิดพลาด ฯลฯ
- การรีเซ็ตฟอร์ม: Key ช่วยให้เราสามารถรีเซ็ตฟอร์มได้โดยไม่ต้องสร้างฟอร์มใหม่

```
final _formKey = GlobalKey<FormState>();

Form(
  key: _formKey,
  child: ... //Widget ต่าง ๆ,
)
```

ตัวอย่าง การใช้ Key กับ Form

```
final formName = GlobalKey<FormState>();
                    …
        Form(
       Key: formName,
child: Column(
 children: [
  TexFormtField(
   decoration: InputDecoration(
    labelText: "ชื่อเล่น",   ),),
  ElevatedButton(
   onPressed: () {
    formName.currentState!.save();},
   child: Text("Submit"),),],),)
```

4.2.3 Validate Form

เป็นกระบวนการที่ช่วยตรวจสอบความถูกต้องของข้อมูลที่ผู้ใช้ป้อนเข้ามาในฟอร์ม โดยการตรวจสอบว่าข้อมูลที่ผู้ใช้ป้อนเข้ามาตรงกับเงื่อนไขที่กำหนดหรือไม่ การทำ Validate Form ช่วยให้แอปพลิเคชันสามารถป้องกันการป้อนข้อมูลที่ไม่ถูกต้องเข้าสู่ระบบหรือส่งข้อมูลที่ไม่สมบูรณ์ไปยังเซิร์ฟเวอร์ได้ โดยมักจะทำ Validate Form ในขั้นตอนต่อจากที่ผู้ใช้กดปุ่มส่งข้อมูลฟิลด์ป้อนข้อมูลไม่ว่าง

ลักษณะการ Validate Form อาจแตกต่างกันไปตามความต้องการของแอปพลิเคชัน แต่โดยทั่วไปแล้วมักจะมีกระบวนการอย่างนี้

1. **รับข้อมูลจากผู้ใช้**: ในขั้นตอนแรก รับข้อมูลที่ผู้ใช้ป้อนเข้ามาผ่านฟอร์ม โดยมักใช้ TextFormField หรือ TextField เพื่อรับข้อมูลที่ผู้ใช้ป้อนเข้ามา เช่น ชื่อผู้ใช้ รหัสผ่าน หรือข้อมูลอื่น ๆ
2. **ตรวจสอบความถูกต้องของข้อมูล**: หลังจากที่ผู้ใช้ป้อนข้อมูลเสร็จสิ้น ในขั้นตอนนี้จะทำการตรวจสอบว่าข้อมูลที่ผู้ใช้ป้อนเข้ามาตรงกับเงื่อนไขที่กำหนดหรือไม่ เช่น ต้องมีความยาวไม่น้อยกว่า 6 ตัวอักษร ต้องประกอบด้วยตัวอักษรและตัวเลข ต้องมีรูปแบบของอีเมลที่ถูกต้อง เป็นต้น
3. **แสดงข้อความเตือนหรือข้อผิดพลาด**: หากข้อมูลที่ผู้ใช้ป้อนไม่ถูกต้อง ให้แสดงข้อความเตือนหรือข้อผิดพลาดเพื่อแจ้งให้ผู้ใช้ทราบว่าจะต้องป้อนข้อมูลใหม่อย่างถูกต้อง

```
TextFormField(
  validator: (value) {
    if (value.isEmpty) {
      return 'กรุณากรอกข้อมูล';
    }
    return null;
  },
),
```

RegExp ใน Validator

RegExp ย่อมาจาก Regular Expression เป็นรูปแบบข้อความที่ใช้จับคู่กับรูปแบบข้อความอื่นๆ ตัวอย่างเช่น คุณสามารถใช้ RegExp เพื่อตรวจสอบว่า อีเมลมีรูปแบบที่ถูกต้องหรือไม่, รหัสผ่านมีความยาวอย่างน้อย 8 ตัวอักษร, เบอร์โทรศัพท์มีรูปแบบที่ถูกต้องหรือไม่

รูปแบบพื้นฐานของ RegExp

ตัวอักษร	ความสามารถ
\d	ข้อมูลที่เป็นตัวเลข 0-9
\w	ข้อมูลที่เป็นตัวอักษรภาษาอังกฤษ และตัวเลข 0-9 รวมถึงขีดล่าง (_)
[...]	ข้อมูลตรงกับตัวอักษรในวงเล็บ ใช้เพื่อระบุช่วงของตัวอักษร เช่น [a-z], [0-9]
[^...]	ข้อมูลตรงกับตัวอักษรที่ไม่อยู่ในวงเล็บ

การกำหนดจำนวนของตัวอักษร

ตัวอักษร	ความสามารถ
*	ตรงกับ 0 หรือมากกว่าตัวอักษรนั้น ๆ
+	ตรงกับ 1 หรือมากกว่าตัวอักษรนั้น ๆ
{n}	ตรงกับจำนวนที่แน่นอนของตัวอักษร เช่น {10} , {13}
{n, }	ตรงกับอย่างน้อย n ตัวอักษร
{n,m}	ตรงกับอย่างน้อย n และไม่เกิน m ตัวอักษร

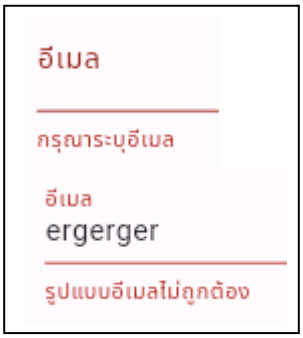
ตัวอย่าง การใช้ Validate Form

```
final formName = GlobalKey<FormState>();
      Form(
     Key: formName,
child: Column(
 children: [
  TextFormField(
   validator: (value){
    if (value == null || value.isEmpty){
     return 'กรุณาระบุชื่อเล่น';
    }
    return null;
   },
```

ตัวอย่างอื่นๆของ Validate Form ที่เป็นแนวสามารถนำไปต่อยอดใช้งานได้

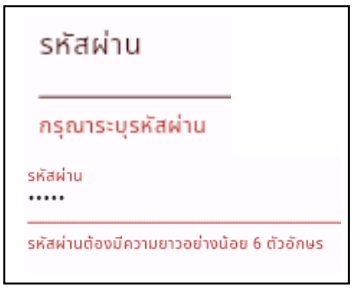
1. ตรวจสอบการกรอก Email และรูปแบบของ Email

```
validator: (value) {
            if (value!.isEmpty) {
              return 'กรุณาระบุอีเมล';
            } else if (!RegExp(r"^[a-zA-Z0-
9.a-zA-Z0-9.!#$%&'*+-/=?^_`{|}~]+@[a-
zA-Z0-9]+\.[a-zA-Z]+").hasMatch(value)) {
              return 'รูปแบบอีเมลไม่ถูกต้อง';
            }
            return null;
          },
```



2. ตรวจสอบการกรอก password และ จำนวนตัวอักษร

```
TextFormField(
        controller: _passwordController,
        decoration: InputDecoration(
          labelText: 'รหัสผ่าน',
        ),
        obscureText: true,
        validator: (value) {
          if (value!.isEmpty) {
            return 'กรุณาระบุรหัสผ่าน';
          } else if (value.length < 6) {
            return 'รหัสผ่านต้องมีความยาวอย่างน้อย 6 ตัวอักษร';
          }
          return null;
        },
      ),
```



4.2.4 Submit Form

การ Submit Form ใน Flutter เป็นกระบวนการที่เกิดขึ้นเมื่อผู้ใช้กดปุ่มส่งหรือทำการส่งข้อมูลจากฟอร์มไปยังตำแหน่งปลายทาง โดยมักจะทำการ Validate ข้อมูลที่ผู้ใช้ป้อนเข้ามาก่อนเพื่อให้แน่ใจว่าข้อมูลถูกต้องตามที่ต้องการก่อนที่จะทำการส่ง หากข้อมูลไม่ถูกต้อง แอปพลิเคชันอาจจะแสดงข้อความเตือนหรือแจ้งให้ผู้ใช้ทราบเพื่อแก้ไขข้อมูลก่อนที่จะส่ง

ขั้นตอนการ Submit Form ใน Flutter สามารถอธิบายได้ดังนี้

1. **รับข้อมูลจากผู้ใช้**: ในขั้นตอนแรก รับข้อมูลที่ผู้ใช้ป้อนเข้ามาผ่านฟอร์ม โดยมักใช้ TextFormField หรือ TextField เพื่อรับข้อมูลที่ผู้ใช้ป้อนเข้ามา เช่น ชื่อผู้ใช้ รหัสผ่าน หรือข้อมูลอื่น ๆ
2. **ตรวจสอบความถูกต้องของข้อมูล**: หลังจากที่ผู้ใช้ป้อนข้อมูลเสร็จสิ้น ในขั้นตอนนี้จะทำการตรวจสอบว่าข้อมูลที่ผู้ใช้ป้อนเข้ามาตรงกับเงื่อนไขที่กำหนดหรือไม่ เช่น ต้องมีความยาวไม่น้อยกว่า 6 ตัวอักษร ต้องประกอบด้วยตัวอักษรและตัวเลข ต้องมีรูปแบบของอีเมลที่ถูกต้อง เป็นต้น
3. **Submit ข้อมูล**: หากข้อมูลที่ผู้ใช้ป้อนถูกต้อง ให้ดำเนินการส่งข้อมูลหรือบันทึกข้อมูลตามที่กำหนด อาจเป็นการส่งข้อมูลไปยังเซิร์ฟเวอร์หรือบันทึกข้อมูลลงในฐานข้อมูล โดยการ Submit ข้อมูลอาจมีการใช้งาน API เพื่อส่งข้อมูลไปยังเซิร์ฟเวอร์หรือใช้งานหน้าที่เกี่ยวข้องเพื่อบันทึกข้อมูล

การใช้งาน Submit Form ใน Flutter จึงเป็นกระบวนการที่สำคัญที่ต้องดำเนินการอย่างระมัดระวัง เพื่อให้ผู้ใช้สามารถส่งข้อมูลได้อย่างถูกต้องและมีประสิทธิภาพในแอปพลิเคชันของคุณ

3. เครื่องมือและอุปกรณ์การทดลอง

3.1 คอมพิวเตอร์ 1 เครื่อง

3.2 ใบงานที่ 8 เรื่อง Form

4. ลำดับขั้นการทดลอง

4.1 นักเรียนศึกษาเนื้อหา เรื่อง Form

4.2 ให้นักเรียนตอบคำถาม ลงในใบงานที่ 8

4.3 ส่งงานครูหลังจากเสร็จเรียบร้อย

คำสั่ง ให้นักศึกษาเขียนคำตอบตามที่โจทย์กำหนดให้ถูกต้อง (สามารถแนบรูปโค้ดและผลลัพธ์คำตอบของโปรแกรมได้)

1. สร้าง Form (ต้องใช้ Form Widget ในการสร้าง) สำหรับเก็บข้อมูลที่ประกอบไปด้วยสิ่งที่ต้องทำดังนี้

    1.1 ต้องประกอบไปด้วยข้อมูลขั้นต่ำ 4 รายการ สามารถเลือกรายการว่าจะใส่ข้อมูลอะไรเองได้เลย พร้อมกับใส่ validate ข้อมูลทุกตัวตามความเหมาะสม **** บังคับข้อมูลที่ต้องมีคือ ข้อมูลใส่ email พร้อม validate ****

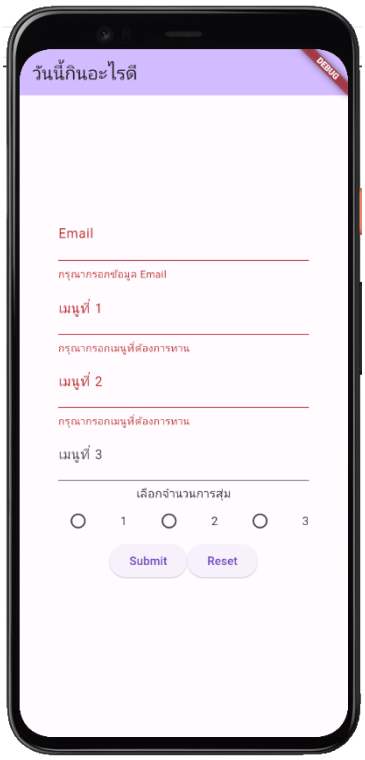
    1.2 สามารถเลือกใช้ Widgets ที่มีความหลากหลายมาใช้ในการทำรายการได้ เช่น TextFormField, Radio Button, Checkbox เป็นต้น

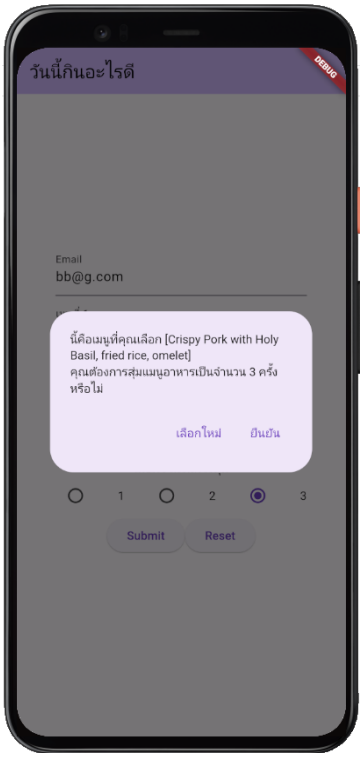
    1.3 ให้สร้างปุ่มในการกดใช้งาน 2 ปุ่ม คือ

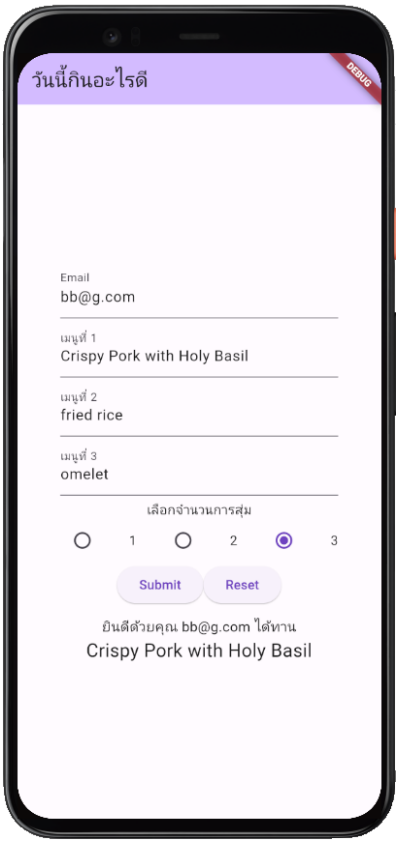
        1.3.1 ปุ่มสำหรับยืนยันข้อมูล – เมื่อกดปุ่มจะแสดงข้อมูลตามที่เราได้ทำการกรอกไป

        1.3.2 ปุ่ม reset – เมื่อกดปุ่ม ข้อมูลที่กรอกใน Form จะถูก reset

    1.4 แสดงข้อมูลตามที่ได้กรอกไป


วันนี้กินอะไรดี
DEBUG
Email
กรุณากรอกข้อมูล Email
เมนูที่ 1
กรุณากรอกเมนูที่ต้องการทาน
เมนูที่ 2
กรุณากรอกเมนูที่ต้องการทาน
เมนูที่ 3
เลือกจำนวนการสุ่ม
1
2
3
Submit
Reset


วันนี้กินอะไรดี
DEBUG
Email
bb@g.com
นี่คือเมนูที่คุณเลือก [Crispy Pork with Holy Basil, fried rice, omelet]
คุณต้องการสุ่มเมนูอาหารเป็นจำนวน 3 ครั้ง หรือไม่
เลือกใหม่
ยืนยัน
1
2
3
Submit
Reset


วันนี้กินอะไรดี
DEBUG
Email
bb@g.com
เมนูที่ 1
Crispy Pork with Holy Basil
เมนูที่ 2
fried rice
เมนูที่ 3
omelet
เลือกจำนวนการสุ่ม
1
2
3
Submit
Reset
ยินดีด้วยคุณ bb@g.com ได้ทาน
Crispy Pork with Holy Basil

```dart
import 'package:flutter/material.dart';
import 'dart:math';

Run | Debug | Profile
void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ), // ThemeData
      home: const MyHomePage(title: 'วันนี้กินอะไรดี'),
    ); // MaterialApp
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});
  final String title;

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  final formName = GlobalKey<FormState>();
  TextEditingController nemu1 = TextEditingController();
  TextEditingController nemu2 = TextEditingController();
  TextEditingController nemu3 = TextEditingController();
  TextEditingController email = TextEditingController();
  List<String> nemu = [''];
  List<int> numMenu = [1, 2, 3];
  int numMenuC = 1;
  Random rng = Random();
  int randomNumber = 0;
  String menu = "";
  String _email = "";

  @override
  Widget build(BuildContext context) {
    void add() {
      if (nemu1 != null) {
        nemu.add(nemu1.text);
      }
      if (nemu2 != null) {
        nemu.add(nemu2.text);
      }
```

```
    if (nemu3 != null) {
      nemu.add(nemu3.text);
    }
  }

  void Ra() {
    int i = 0;
    while (i <= numMenuC) {
      randomNumber = rng.nextInt(nemu.length);
      i++;
    }
    menu = nemu[randomNumber];
  }

  void Re() {
    setState(() {
      nemu1.text = '';
      nemu2.text = '';
      nemu3.text = '';
      email.text = '';
      numMenuC = 1;
      menu = '';
      nemu.clear();
    });
  }

  return Scaffold(
    appBar: AppBar(
      backgroundColor: Theme.of(context).colorScheme.inversePrimary,
      title: Text(widget.title),
    ), // AppBar
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Container(
            width: 300,
            child: Column(
              children: [
                Form(
                  key: formName,
                  child: Column(
                    children: [
                      TextFormField(
                        controller: email,
                        validator: (value) {
                          if (value!.isEmpty) {
                            return 'กรุณากรอกข้อมูล Email';
                          } else if (!RegExp(
                                  r"^[a-zA-Z0-9.a-zA-Z0-9.!#$%&'*+-/=?^_`{|}~]+@[a-zA-Z0-9]+\.[a-zA-Z]+") // RegExp
                              .hasMatch(value)) {
                            return 'รูปแบบอีเมลไม่ถูกต้อง';
                          }
                          return null;
```

```
109                     },
110                     decoration: const InputDecoration(
111                       labelText: "Email",
112                     ), // InputDecoration
113                   ), // TextFormField
114                   TextFormField(
115                     controller: nemu1,
116                     validator: (value) {
117                       if (value!.isEmpty) {
118                         return 'กรุณากรอกเมนูที่ต้องการทาน';
119                       }
120                       return null;
121                     },
122                     decoration: const InputDecoration(
123                       labelText: "เมนูที่ 1",
124                     ), // InputDecoration
125                   ), // TextFormField
126                   TextFormField(
127                     controller: nemu2,
128                     validator: (value) {
129                       if (value!.isEmpty) {
130                         return 'กรุณากรอกเมนูที่ต้องการทาน';
131                       }
132                       return null;
133                     },
134                     decoration: const InputDecoration(
135                       labelText: "เมนูที่ 2",
136                     ), // InputDecoration
137                   ), // TextFormField
138                   TextFormField(
139                     controller: nemu3,
140                     decoration: const InputDecoration(
141                       labelText: "เมนูที่ 3",
142                     ), // InputDecoration
143                   ), // TextFormField
144                   const SizedBox(
145                     height: 5.0,
146                   ), // SizedBox
147                   const Text('เลือกจำนวนการสุ่ม'),
148                   Container(
149                     child: Row(
150                       mainAxisAlignment: MainAxisAlignment.spaceBetween,
151                       children: [
152                         Radio(
153                             value: numMenu[0],
154                             groupValue: numMenuC,
155                             onChanged: (value) {
156                               numMenuC = value as int;
157                             }), // Radio
158                         const Text('1'),
159                         Radio(
160                             value: numMenu[1],
161                             groupValue: numMenuC,
162                             onChanged: (value) {
```

```
                                numMenuC = value as int;
                              }), // Radio
                            const Text('2'),
                            Radio(
                                value: numMenu[2],
                                groupValue: numMenuC,
                                onChanged: (value) {
                                  numMenuC = value as int;
                                }), // Radio
                            const Text('3'),
                          ],
                        ), // Row
                      ), // Container
                    ],
                  ), // Column
                ), // Form
              ],
            ), // Column
          ), // Container
          Container(
            child:
                Row(mainAxisAlignment: MainAxisAlignment.center, children: [
              ElevatedButton(
                onPressed: () {
                  if (formName.currentState!.validate()) {
                    setState(() {
                      _email = email.text;
                    });
                    showDialog(
                        context: context,
                        builder: (BuildContext context) {
                          return AlertDialog(
                            title: Column(
                              crossAxisAlignment: CrossAxisAlignment.start,
                              children: [
                                Text(
                                  'นี่คือเมนูที่คุณเลือก ${nemu1.text} ${nemu2.text} ${nemu3.text}',
                                  style: const TextStyle(fontSize: 14),
                                ), // Text
                                Text(
                                    'คุณต้องการสุ่มแมนูอาหารเป็นจำนวน $numMenuC ครั้งหรือไม่',
                                    style: const TextStyle(fontSize: 14)), // Text
                              ],
                            ), // Column
                            actions: <Widget>[
                              TextButton(
                                  onPressed: () {
                                    Navigator.pop(context, 'OK');
                                  },
                                  child: const Text('เลือกใหม่')), // TextButton
                              TextButton(
                                  onPressed: () {
                                    formName.currentState!.save();
                                    setState(() {
```

```
                                    add();
                                    Ra();
                                  });
                                  Navigator.pop(context, 'OK');
                                  nemu.clear();
                                },
                                child: const Text('ยืนยัน')), // TextButton
                            ], // <Widget>[]
                          ); // AlertDialog
                        });
                    }
                  },
                  child: Text('Submit'),
                ), // ElevatedButton
                ElevatedButton(
                  onPressed: () {
                    if (formName.currentState != null) {
                      formName.currentState!.reset();
                      Re();
                    }
                  },
                  child: Text('Reset'),
                ), // ElevatedButton
              ]), // Row
            ), // Container
            const SizedBox(
              height: 10.0,
            ), // SizedBox
            Container(
              width: 300,
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  Text(
                    '${menu == "" ? "" : "ยินดีด้วยคุณ $_email ได้ทาน"}',
                    style: const TextStyle(fontSize: 15),
                  ), // Text
                  Text(
                    '${menu == "" ? "" : "$menu"}',
                    style: const TextStyle(fontSize: 20),
                  ), // Text
                ],
              ), // Column
            ), // Container
          ],
        ), // Column
      ), // Center
    ); // Scaffold
  }
}

```

5. สรุปผลการทดลอง

การใช้ Form widget ทำให้กระบวนการจัดการและการตรวจสอบข้อมูลในแอปของคุณเป็นไปอย่างมีระบบและยืดหยุ่น

ทำให้ผู้พัฒนาสามารถสร้างแอปพลิเคชันที่มีประสิทธิภาพการจัดการ Input Validation: Form ช่วยให้คุณสามารถสร้างกฎการตรวจสอบข้อมูล (validation rules) สำหรับข้อมูลที่ผู้ใช้ป้อนเข้ามา เช่น ตรวจสอบว่าข้อมูลต้องไม่ว่างเปล่า, ต้องมีรูปแบบที่ถูกต้อง, หรือต้องมีความยาวที่ต้องการ เป็นต้น การจัดการ Submission: Form มี onSubmit callback ที่ช่วยในการจัดการข้อมูลเมื่อผู้ใช้ทำการส่งฟอร์ม (submit) ซึ่งทำให้คุณสามารถทำงานที่เกี่ยวข้องหลังจากการ submit เช่น ส่งข้อมูลไปยังเซิร์ฟเวอร์, บันทึกข้อมูลลงในฐานข้อมูล, หรือประมวลผลข้อมูลอื่นๆ การจัดการ State: Form ช่วยในการจัดการ state ของข้อมูลที่ผู้ใช้ป้อนลงในฟอร์ม ทำให้คุณสามารถเข้าถึงและใช้ข้อมูลนั้นได้อย่างสะดวก

6. คำถามหลังการทดลอง

6.1 ข้อดีของ Form Widgets กับไม่ใช่ Form Widgets ต่างกันอย่างไร

Form Widgets ใน Flutter มีความสามารถในการจัดการและตรวจสอบข้อมูลที่ผู้ใช้ป้อนลงในแอปพลิเคชันของคุณ ด้วยคุณสมบัติและข้อดีที่มีอยู่, Form Widgets มีบางคุณสมบัติที่ทำให้เป็นทางเลือกที่ดีเมื่อต้องการจัดการข้อมูลฟอร์ม. นี่คือข้อดีของ Form Widgets และวิธีที่มันแตกต่างจากการไม่ใช้ Form Widgets:

Input Validation: Form Widgets มีความสามารถในการจัดการ input validation โดยการให้บล็อกตัวยึด (form fields)ที่ช่วยในการตรวจสอบข้อมูลที่ผู้ใช้ป้อน, เช่น ตรวจสอบว่าข้อมูลว่างเปล่า, ตรวจสอบรูปแบบที่ถูกต้อง, หรือตรวจสอบความยาวของข้อมูล

State Management: Form Widgets ช่วยในการจัดการ state ของข้อมูลที่ผู้ใช้ป้อนลงในฟอร์ม นี้ช่วยในการเข้าถึงข้อมูลนั้น ๆ และใช้ในการแสดงผลหรือประมวลผล.การจัดการ Submission: Form Widgets มี callback ที่เรียกว่า onSubmit ซึ่งทำหน้าที่จัดการข้อมูลเมื่อผู้ใช้ submit ฟอร์ม.นี้ช่วยในการดำเนินการที่ต้องการหลังจากการส่งฟอร์ม

Input Decoration: Form Widgets ช่วยในการกำหนดรูปแบบของ input fields ที่แตกต่างกันไป, เช่น การกำหนดรูปแบบของ label, border, หรือ style

การจัดการ Focus: Form Widgets ช่วยในการจัดการ focus ของ input fields, ทำให้เป็นไปตามการกระทำของผู้ใช้

การจัดการ Error Handling: Form Widgets มีการจัด การข้อผิดพลาดที่มีความสามารถในการแสด งข้อผิดพลาดและการจัดการกับข้อผิดพลาดที่เกิดขึ้นในการตรวจสอบข้อมูล

การไม่ใช้ Form Widgets:คือจะไม่มีหัวข้อข้างต้นที่กล่าวมา ในทางทั่วไป, Form Widgets มีความสามารถมากมายที่ช่วยในการจัดการฟอร์มและข้อมูลที่ผู้ใช้ป้อน, ทำให้โค้ดมีความอ่านง่ายและง่ายต่อการบำรุงรักษา