

## BÀI 4

# THIẾT KẾ GIAO DIỆN NGƯỜI DÙNG (NC)

GV: ThS. BÙI PHÚ KHUYÊN



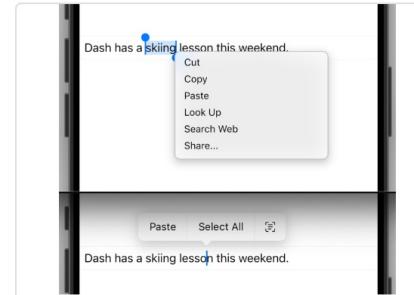
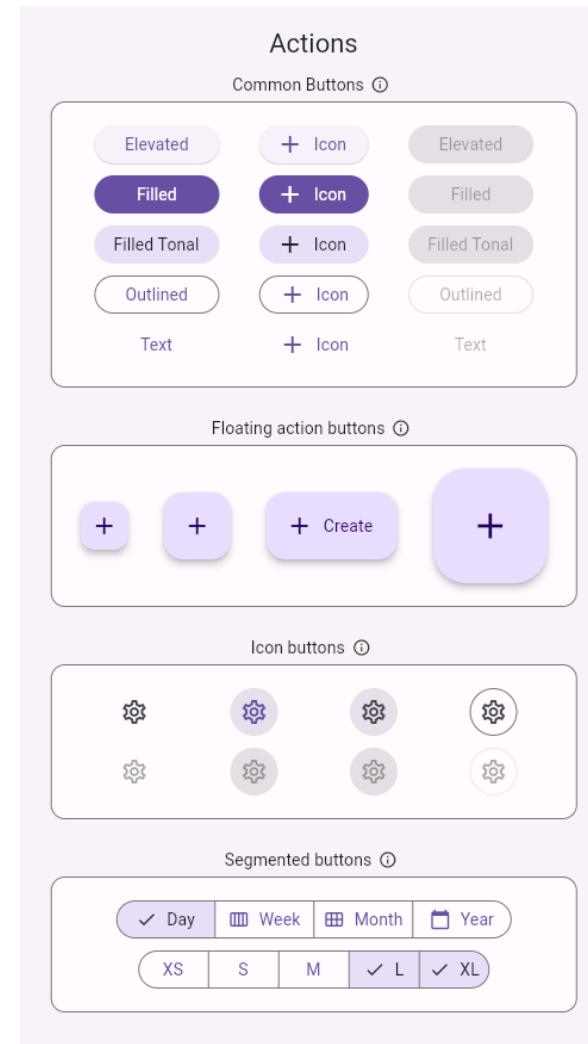
\* Cập nhật: 26.11.2024

# NỘI DUNG BÀI HỌC

- Design & Theming (Material & Cupertino Design)
- Navigation & Layout Control
- Interactivity
- Accessibility & Internationalization
- Animations & Transitions

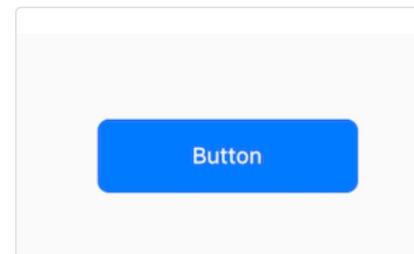


# MATERIAL & CUPERTINO DESIGN



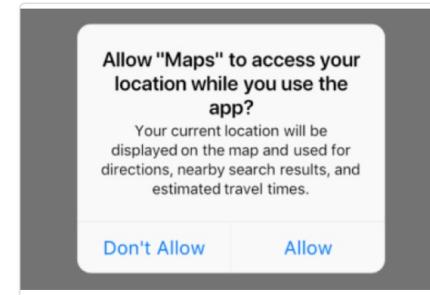
**CupertinoAdaptiveTextSelectionToolbar**

The default Cupertino context menu for text selection for the current platform with the given children.



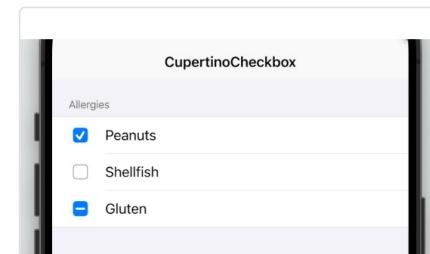
**CupertinoButton**

An iOS-style button.



**CupertinoAlertDialog**

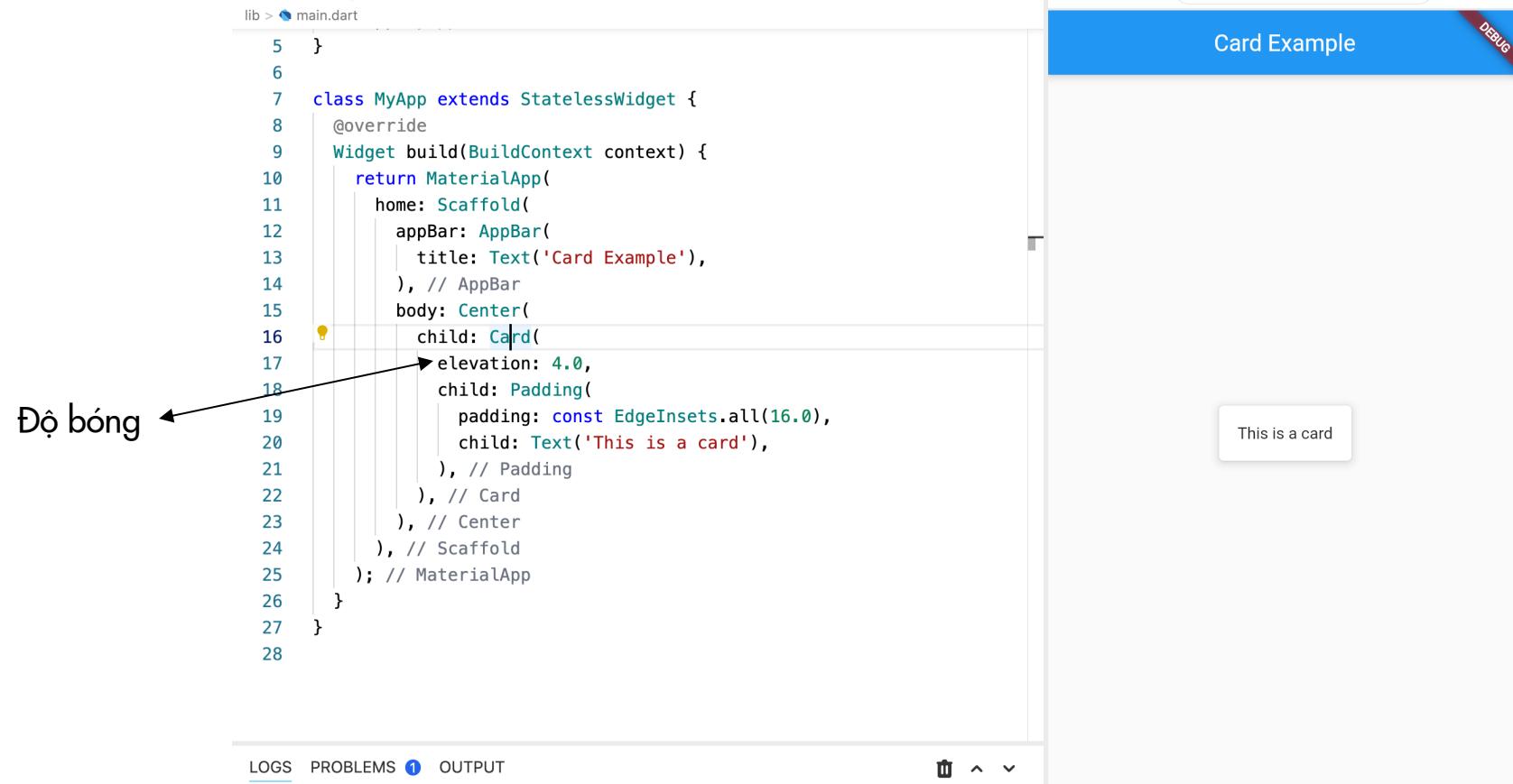
An iOS-style alert dialog.



**CupertinoCheckBox**

A macOS-style checkbox.

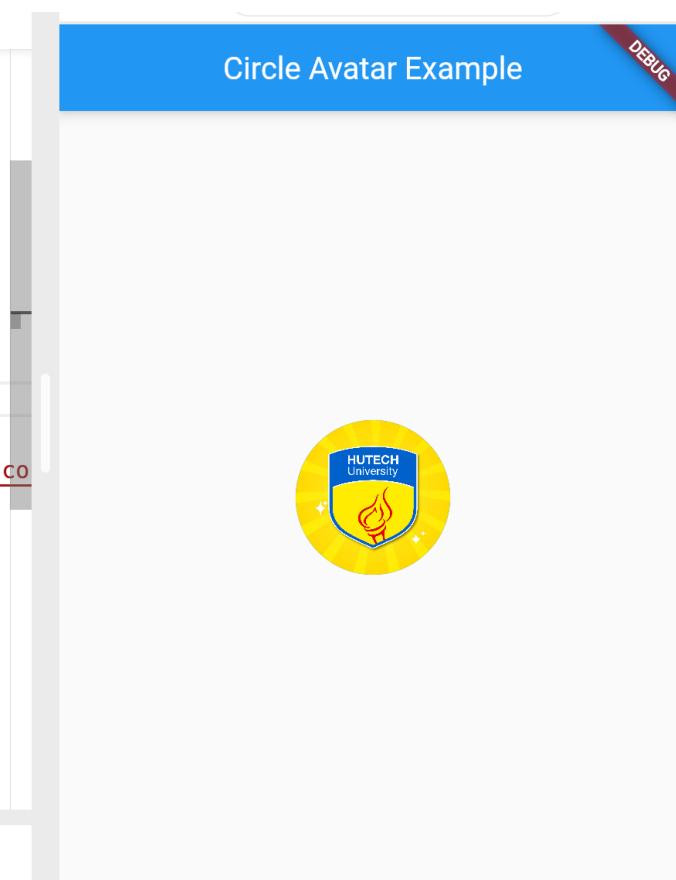
Card là một widget Material Design dùng để hiển thị nội dung và hành động trong một khung có thể tùy chỉnh



```
lib > main.dart
5 }
6
7 class MyApp extends StatelessWidget {
8   @override
9   Widget build(BuildContext context) {
10   return MaterialApp(
11     home: Scaffold(
12       appBar: AppBar(
13         title: Text('Card Example'),
14       ), // AppBar
15       body: Center(
16         child: Card(
17           elevation: 4.0,
18           child: Padding(
19             padding: const EdgeInsets.all(16.0),
20             child: Text('This is a card'),
21           ), // Padding
22         ), // Card
23       ), // Center
24     ), // Scaffold
25   ); // MaterialApp
26 }
27
28 }
```

Độ bóng

CircleAvatar là một widget để hiển thị hình ảnh hoặc chữ cái trong một vòng tròn, thường được sử dụng cho ảnh đại diện



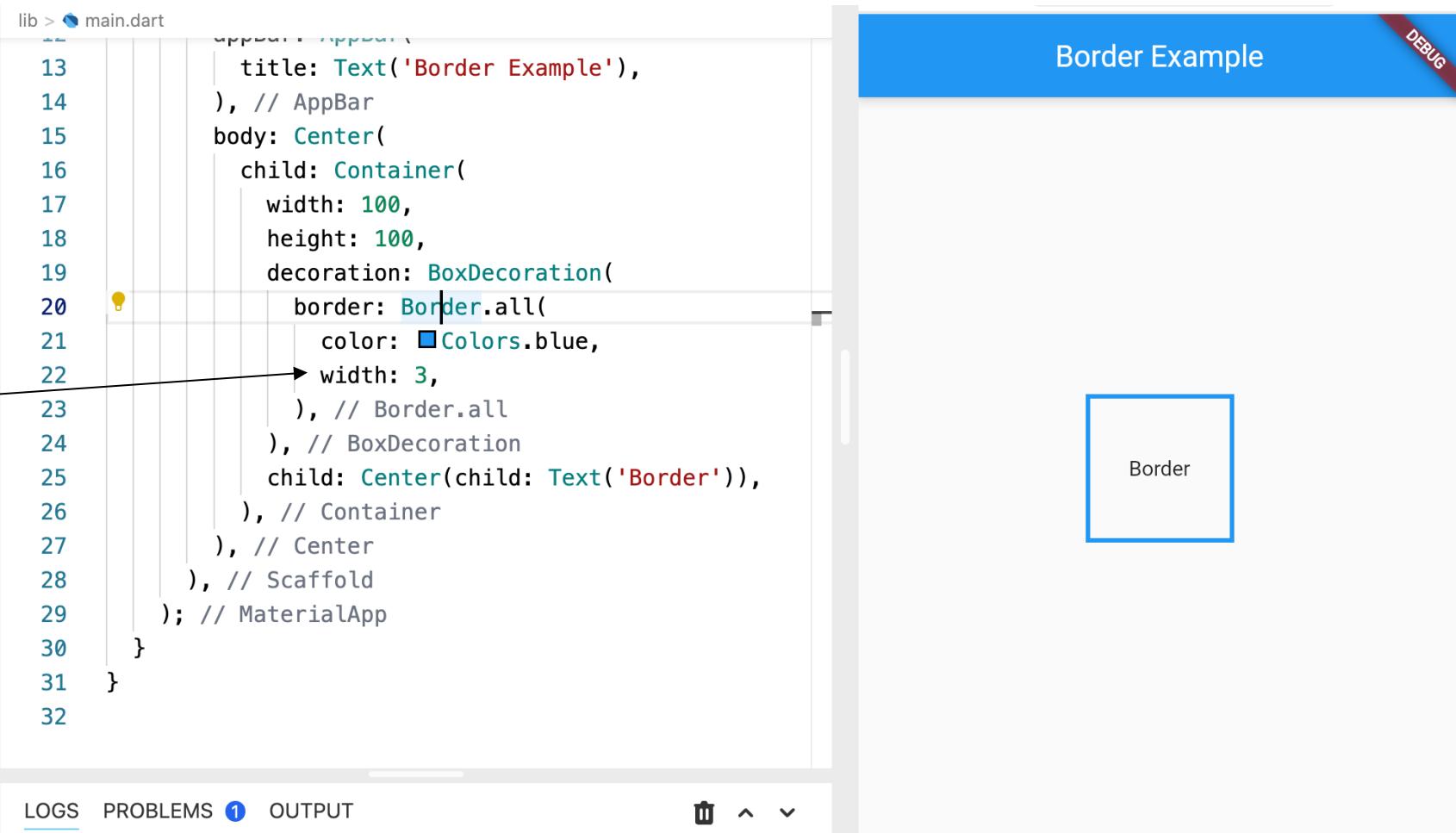
The screenshot shows a Flutter application titled "Circle Avatar Example". On the right, there is a circular image of the HUTECH University logo. On the left, the code for this widget is displayed in a code editor:

```
lib > main.dart
1 class MyApp extends StatelessWidget {
2   @override
3   Widget build(BuildContext context) {
4     return MaterialApp(
5       home: Scaffold(
6         appBar: AppBar(
7           title: Text('Circle Avatar Example'),
8         ), // AppBar
9         body: Center(
10           child: CircleAvatar(
11             radius: 50,
12             backgroundImage: NetworkImage('https://sco...'),
13           ), // CircleAvatar
14         ), // Center
15       ), // Scaffold
16     ); // MaterialApp
17 }
```

Annotations in Vietnamese point to specific parts of the code and the resulting UI:

- A line labeled "Bán kính" points to the `radius: 50,` line in the code.
- A line labeled "Chứa ảnh nền" points to the `backgroundImage: NetworkImage('https://sco...',)` line in the code.

Border là một thuộc tính dùng để tạo viền cho các widget



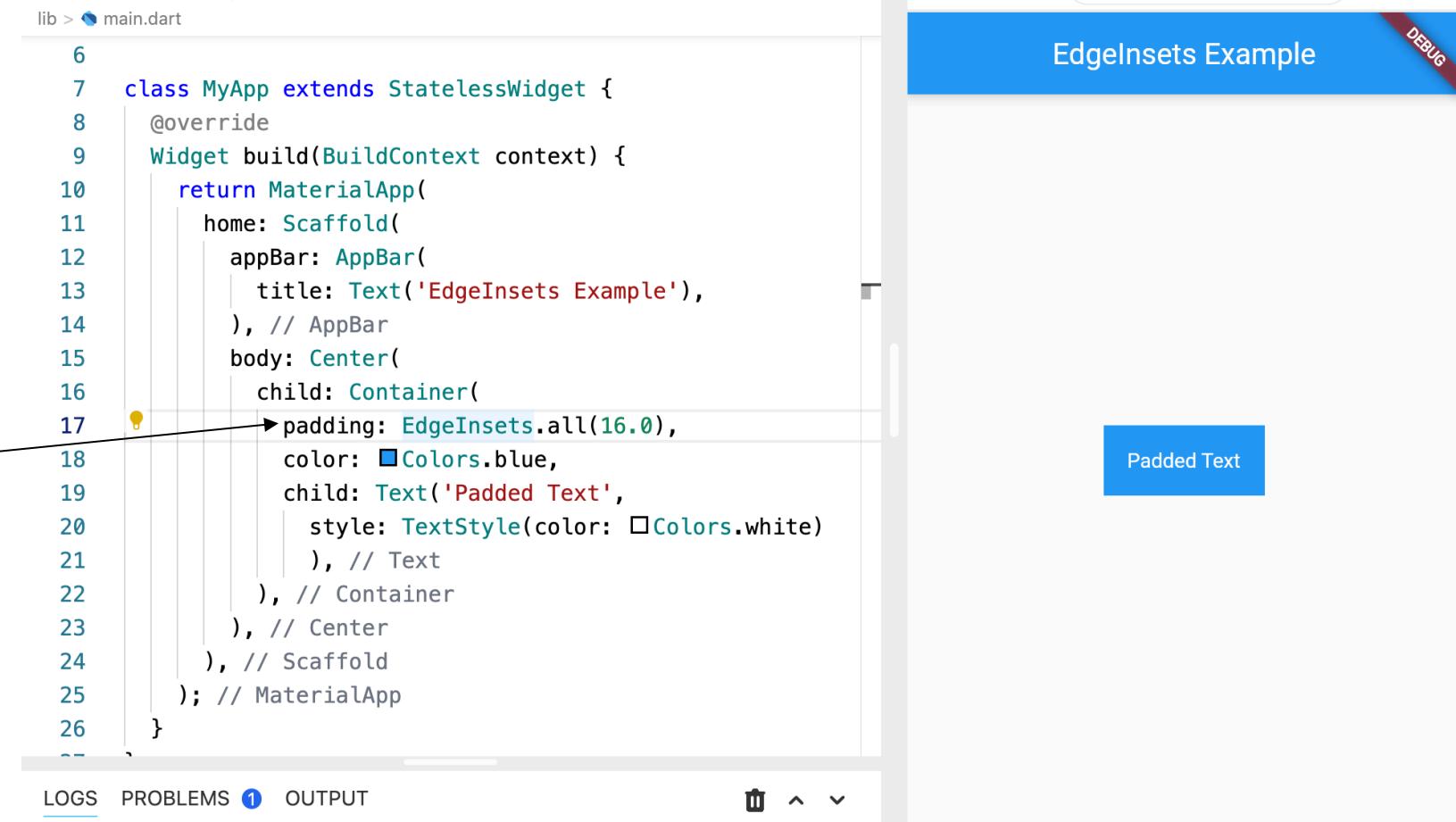
The screenshot shows a Flutter application running in the Android Studio emulator. The title bar says "Border Example" and has a "DEBUG" indicator. The main content area displays a single blue-bordered container with the text "Border" inside. On the left side of the code editor, there is a vertical annotation with the text "Độ dày viền" (Border thickness) and an arrow pointing to the line of code where the border width is specified.

```
lib > main.dart
13     title: Text('Border Example'),
14   ), // AppBar
15   body: Center(
16     child: Container(
17       width: 100,
18       height: 100,
19       decoration: BoxDecoration(
20         border: Border.all(
21           color: Colors.blue,
22           width: 3,
23         ), // Border.all
24       ), // BoxDecoration
25       child: Center(child: Text('Border')),
26     ), // Container
27   ), // Center
28   ), // Scaffold
29 ); // MaterialApp
30 }
31 }
```

LOGS PROBLEMS 1 OUTPUT

EdgeInsets là một lớp dùng để xác định khoảng cách đệm (padding) và khoảng cách viền (margin)

Khoảng cách đệm  
(padding)  
cho toàn bộ cạnh



The screenshot shows the Dart code for an application named 'EdgeInsets Example'. The code defines a class 'MyApp' that extends 'StatelessWidget'. It sets up a Material App with a blue AppBar titled 'EdgeInsets Example'. The main body contains a Center widget with a Container. The Container has a padding of 16.0 on all four sides ( EdgeInsets.all(16.0) ). Inside the container is a Text widget with the content 'Padded Text' and a white color. The UI preview on the right shows a blue header bar with the title 'EdgeInsets Example' and a red 'DEBUG' indicator. Below the header is a white area containing a blue button labeled 'Padded Text' with white text.

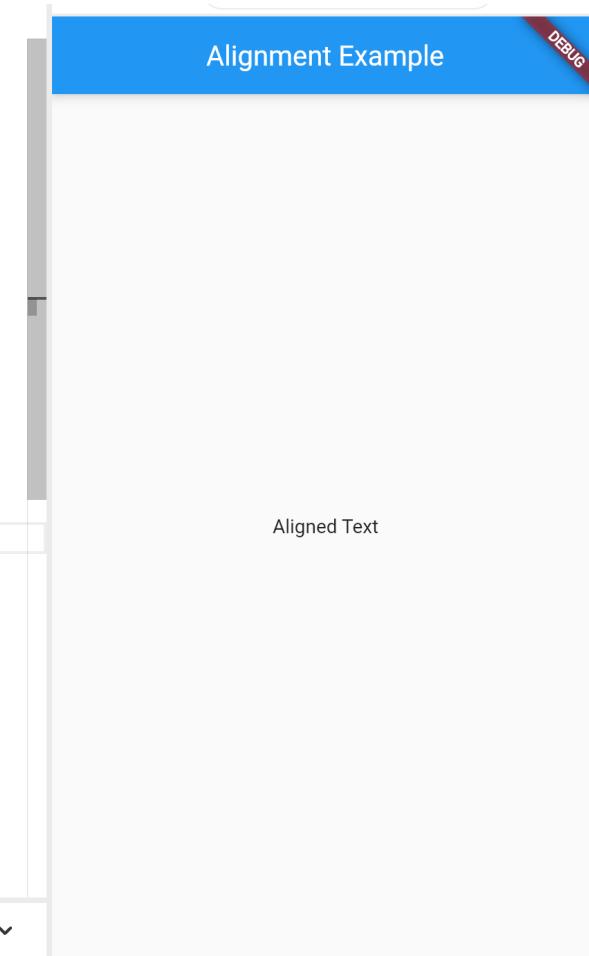
```
lib > main.dart
6
7 class MyApp extends StatelessWidget {
8   @override
9   Widget build(BuildContext context) {
10     return MaterialApp(
11       home: Scaffold(
12         appBar: AppBar(
13           title: Text('EdgeInsets Example'),
14         ), // AppBar
15         body: Center(
16           child: Container(
17             padding: EdgeInsets.all(16.0),
18             color: Colors.blue,
19             child: Text('Padded Text',
20               style: TextStyle(color: Colors.white)
21             ), // Text
22           ), // Container
23         ), // Center
24       ), // Scaffold
25     ); // MaterialApp
26 }
```

LOGS PROBLEMS 1 OUTPUT

EdgeInsets là một lớp dùng để xác định khoảng cách đệm (padding) và khoảng cách viền (margin)

Căn chỉnh widget con  
(Text)  
về giữa

```
lib > main.dart
1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(MyApp());
5 }
6
7 class MyApp extends StatelessWidget {
8   @override
9   Widget build(BuildContext context) {
10     return MaterialApp(
11       home: Scaffold(
12         appBar: AppBar(
13           title: Text('Alignment Example'),
14         ), // AppBar
15         body: Align(
16           alignment: Alignment.center,
17           child: Text('Aligned Text'),
18         ), // Align
19       ), // Scaffold
20     ); // MaterialApp
21   }
22 }
23
```



12:52

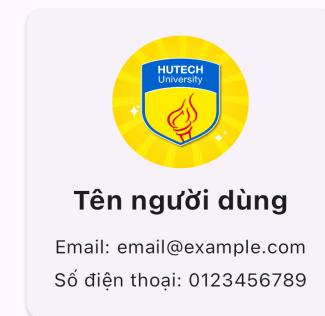


Hồ sơ

## Thiết kế giao diện cho màn hình Hồ sơ. Yêu cầu

- Tạo một màn hình Hồ sơ hiển thị thông tin cá nhân của người dùng.
- Sử dụng CircleAvatar để hiển thị hình đại diện của người dùng.
- Hiển thị thông tin cá nhân như tên, email, số điện thoại trong một Card.

Lưu ý: Cần tạo 1 class riêng (Profile) cho màn hình này.



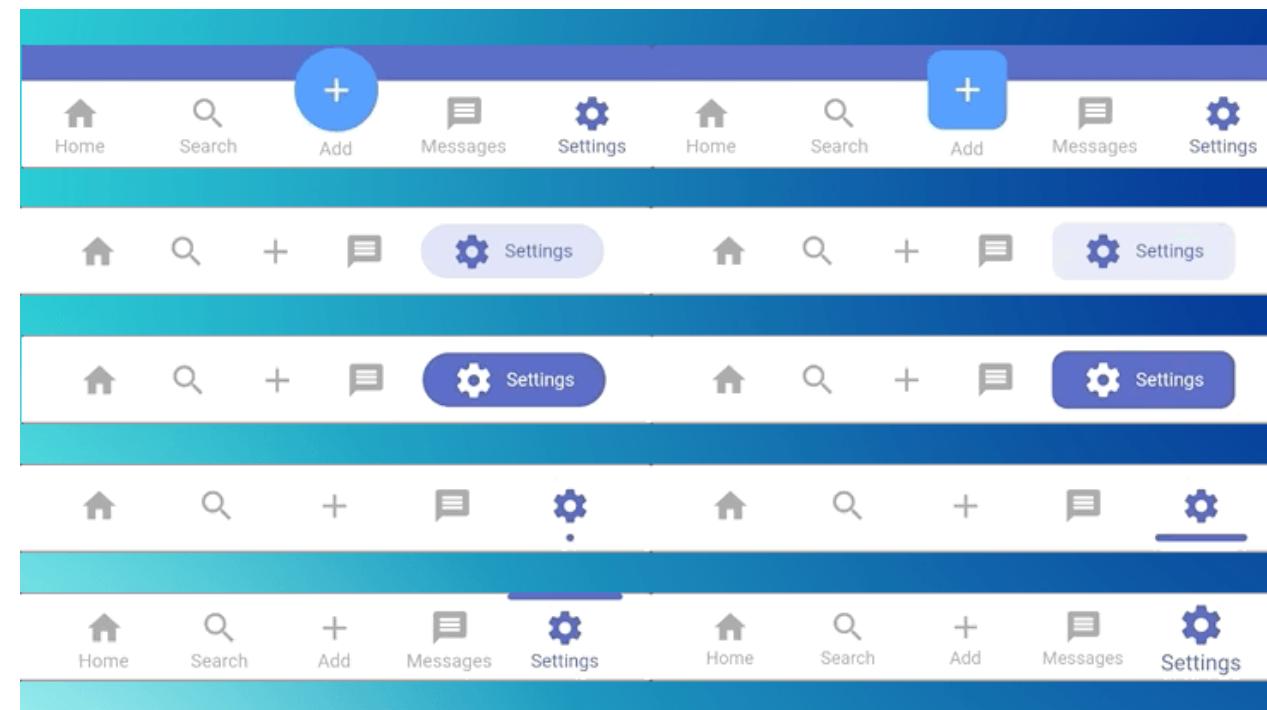


**TÌM HIỂU CƠ BẢN**

**NAVIGATION &  
LAYOUT CONTROL**

## BottomNavigationBar {1}

**BottomNavigationBar** là một widget phổ biến, giúp người dùng điều hướng giữa các màn hình khác nhau bằng cách sử dụng một thanh điều hướng ở cuối màn hình nằm trong **Scaffold**



Thuộc tính  
bottomNavigationBar nằm  
trong Scaffold widget

## BottomNavigationBar {2}

Danh sách các trang, mỗi trang  
tương ứng 1  
BottomNavigationBarItem

Chỉ mục của trang được chọn

Hàm xử lý khi được nhấn

```

lib > main.dart
31   appBar: AppBar(
32     title: const Text('BottomNavigationBar Example'),
33   ), // AppBar
34   body: Center(
35     child: _widgetOptions.elementAt(_selectedIndex),
36   ), // Center
37   bottomNavigationBar: BottomNavigationBar(
38     items: const <BottomNavigationBarItem>[
39       BottomNavigationBarItem(
40         icon: Icon(Icons.home),
41         label: 'Home',
42       ), // BottomNavigationBarItem
43       BottomNavigationBarItem(
44         icon: Icon(Icons.search),
45         label: 'Search',
46       ), // BottomNavigationBarItem
47       BottomNavigationBarItem(
48         icon: Icon(Icons.person),
49         label: 'Profile',
50       ), // BottomNavigationBarItem
51     ], // <BottomNavigationBarItem>[]
52     currentIndex: _selectedIndex,
53     selectedItemColor: Colors.amber[800],
54     onTap: _onItemTapped,
55   ), // BottomNavigationBar
56   ), // Scaffold
57 ); // MaterialApp
  
```

Biểu tượng

Nhãn

Màu của trang  
được chọn

Home Page

Home

Search

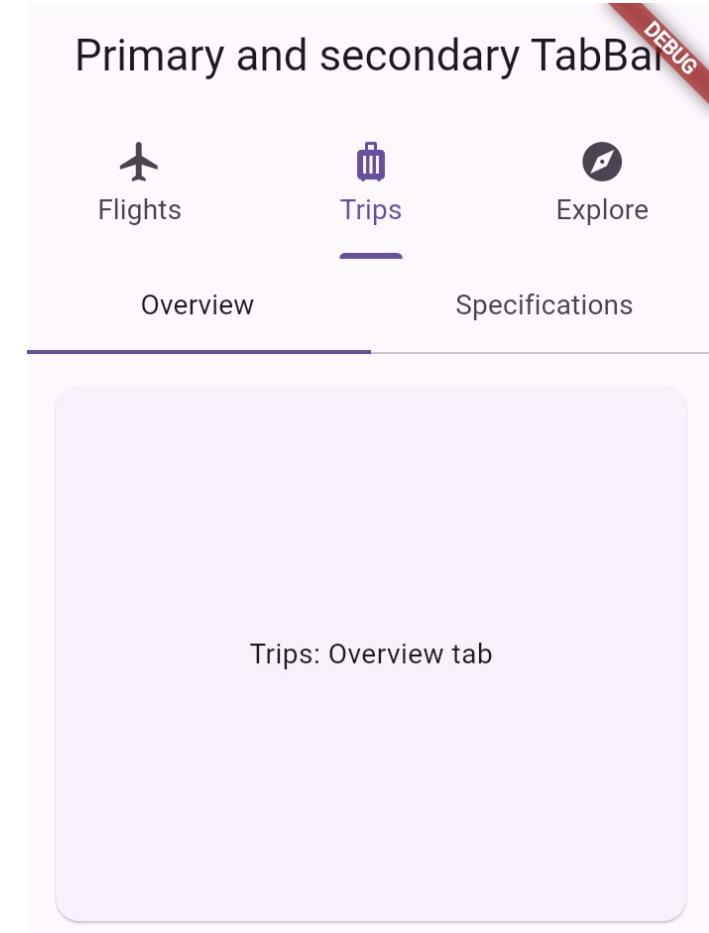
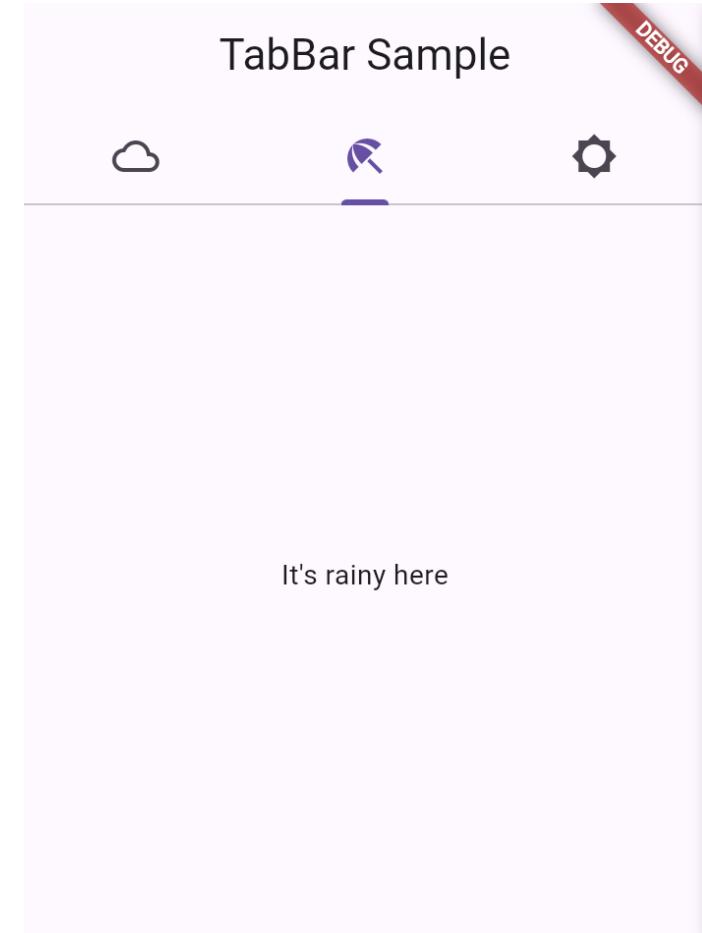
Profile

LOGS PROBLEMS 1 OUTPUT

## TabBar {1}

TabBar là một widget trong Flutter giúp hiển thị thanh điều hướng dạng Tab (Tabs Navigation).

Thường được sử dụng trong các ứng dụng cần **chuyển đổi nhanh giữa các màn hình** hoặc nội dung khác nhau bằng cách bấm vào các Tab.



<https://api.flutter.dev/flutter/material/TabBar-class.html>

**DefaultTabController:**  
Quản lý trạng thái của TabBar  
và TabBarView.

Danh sách các tab  
được hiển thị

**TabBar {2}**

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: DefaultTabController(
        length: 3, // Số lượng tab
        child: Scaffold(
          appBar: AppBar(
            title: Text('TabBar Example'),
            bottom: TabBar( // Danh sách các tab được hiển thị
              tabs: [
                Tab(icon: Icon(Icons.home), text: 'Home'),
                Tab(icon: Icon(Icons.search), text: 'Search'),
                Tab(icon: Icon(Icons.settings), text: 'Settings'),
              ],
            ),
            body: TabBarView( // TabBarView: Chứa nội dung tương ứng với từng tab.
              children: [
                Center(child: Text('Home Screen')),
                Center(child: Text('Search Screen')),
                Center(child: Text('Settings Screen')),
              ],
            ),
          ),
        );
    );
  }
}
```

**TabBar:** Hiển thị các tab với biểu tượng và văn bản.

**TabBarView:** Chứa nội dung  
tương ứng với từng tab.

## Drawer

Drawer là một widget giúp hiển thị menu điều hướng trượt ngang từ cạnh màn hình.

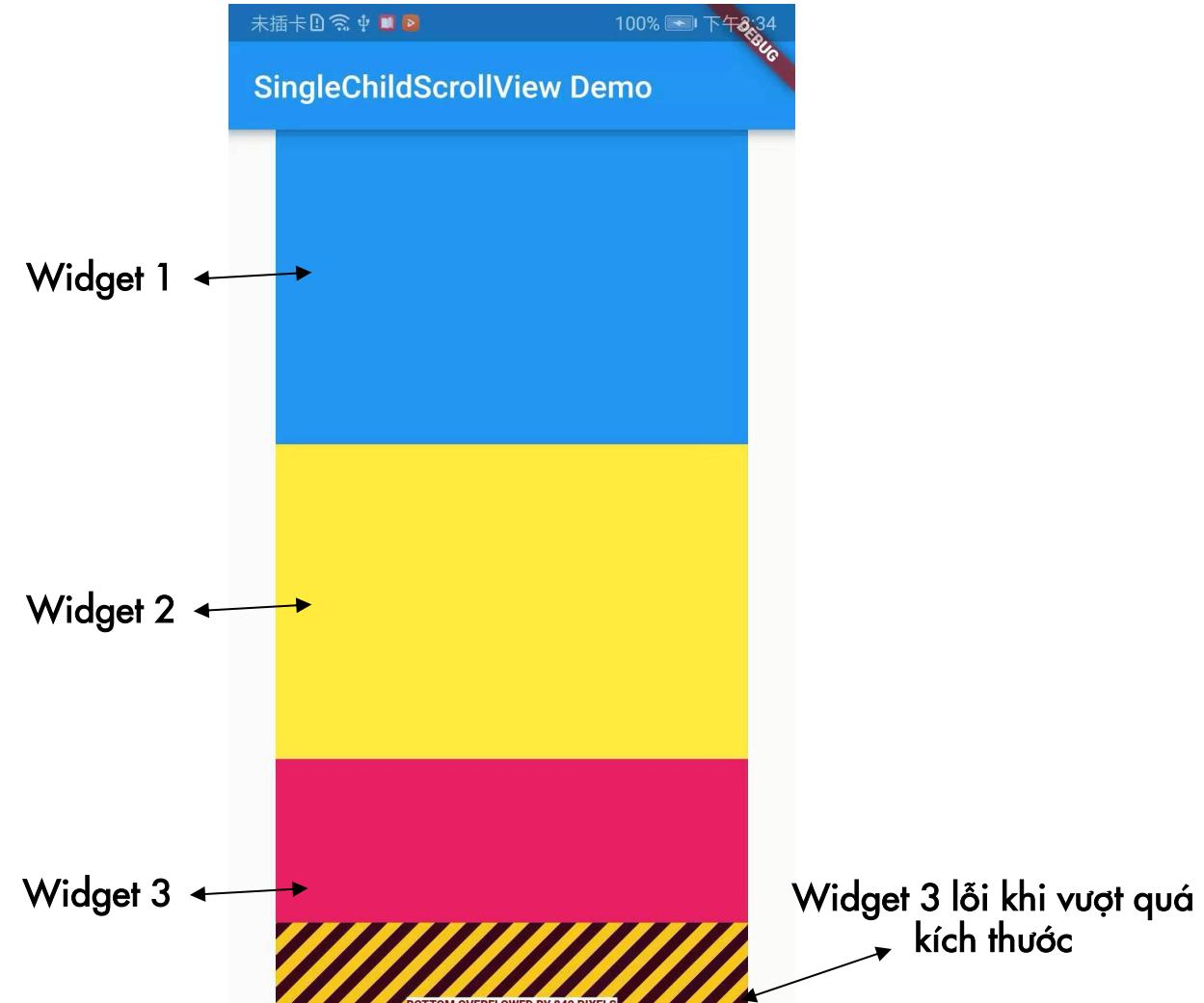
Thường được sử dụng trong các ứng dụng có nhiều mục điều hướng nhưng không muốn chiếm nhiều không gian hiển thị.

≡ Drawer Demo

<https://api.flutter.dev/flutter/material/Drawer-class.html>

## SingleChildScrollView {1}

**SingleChildScrollView** là một widget hỗ trợ cuộn nội dung khi nội dung vượt quá kích thước của màn hình. Nó rất hữu ích khi ta có một danh sách dài các mục mà không thể hiển thị hết trên một màn hình duy nhất.

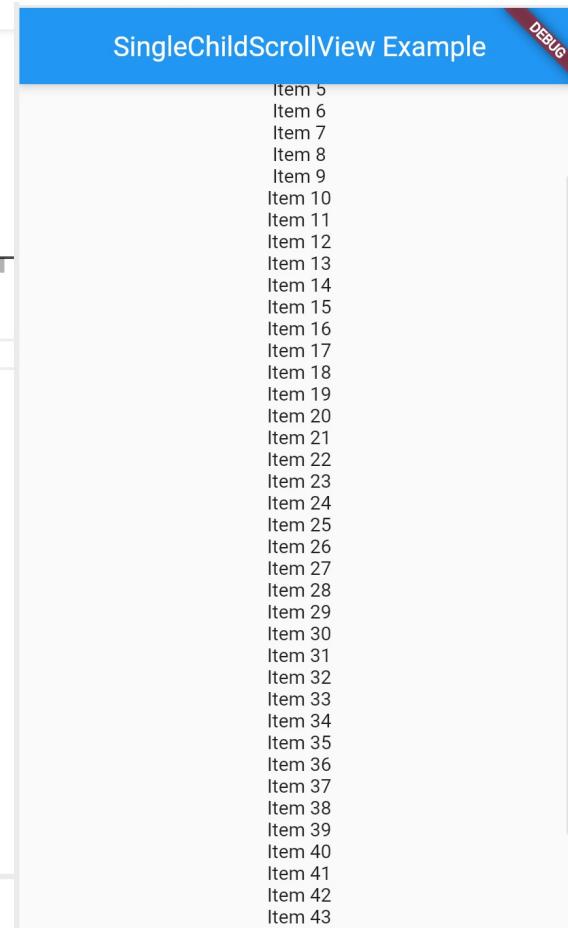


## SingleChildScrollView {2}

SingleChildScrollView cho phép cuộn nội dung bên trong nó (tránh bị lỗi quá kích cỡ)

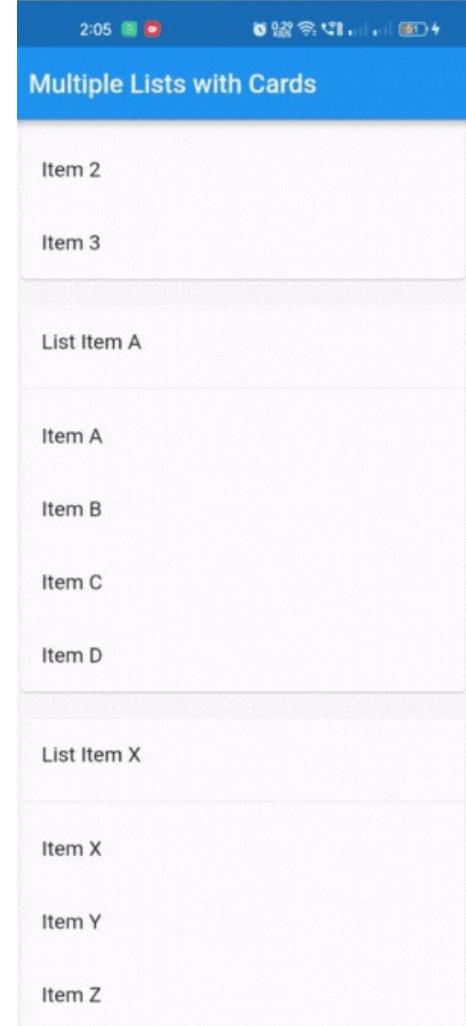
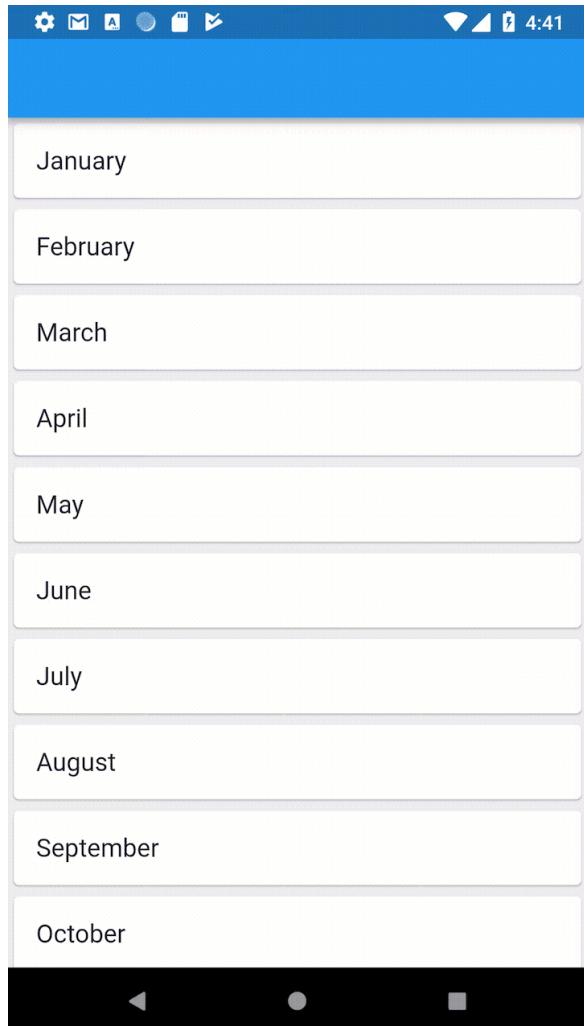
```
lib > main.dart
5 }
6
7 class MyApp extends StatelessWidget {
8   @override
9   Widget build(BuildContext context) {
10     return MaterialApp(
11       home: Scaffold(
12         appBar: AppBar(
13           title: Text('SingleChildScrollView Example'),
14         ), // AppBar
15         body: SingleChildScrollView(
16           child: Column(
17             children: <Widget>[
18               for (int i = 0; i < 50; i++)
19                 Center(
20                   child: Text('Item $i'),
21                 ), // Center
22               ], // <Widget>[]
23             ), // Column
24           ), // SingleChildScrollView
25         ), // Scaffold
26       ); // MaterialApp
27     }
28   }
29 }
```

LOGS PROBLEMS 1 OUTPUT



## ListView {1}

**ListView** là một widget mạnh mẽ giúp hiển thị danh sách các mục theo chiều dọc. Nó tự động cuộn khi danh sách dài hơn chiều cao của màn hình

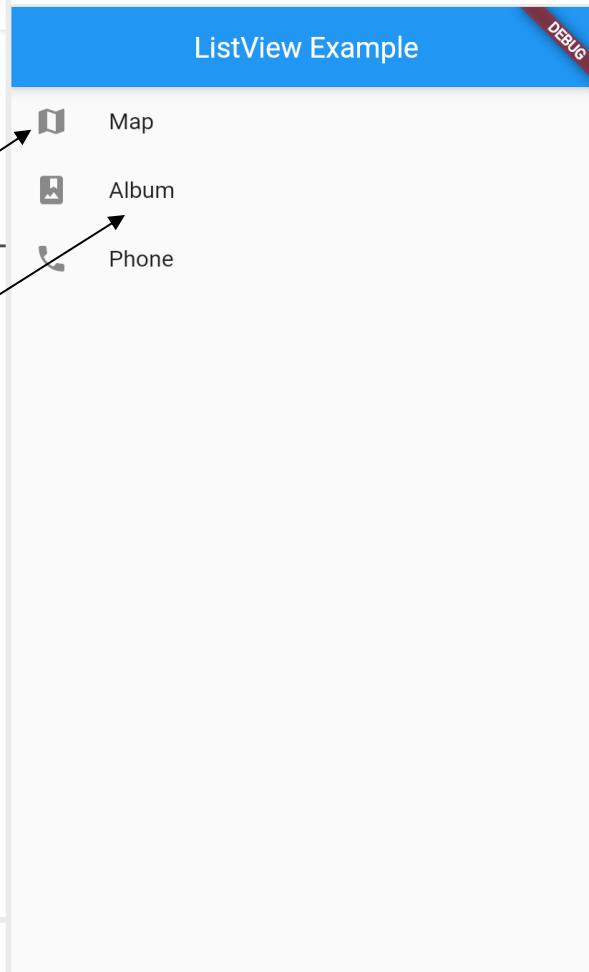


## ListView {2}

lib &gt; main.dart

```
7  class MyApp extends StatelessWidget {
8    @override
9    Widget build(BuildContext context) {
10      return MaterialApp(
11        home: Scaffold(
12          appBar: AppBar(
13            title: Text('ListView Example'),
14          ), // AppBar
15          body: ListView(
16            children: <Widget>[
17              ListTile(
18                leading: Icon(Icons.map),
19                title: Text('Map'),
20              ), // ListTile
21              ListTile(
22                leading: Icon(Icons.photo_album),
23                title: Text('Album'),
24              ), // ListTile
25              ListTile(
26                leading: Icon(Icons.phone),
27                title: Text('Phone'),
28              ), // ListTile
29            ], // <Widget>[]
30          ), // ListView
31        ), // Scaffold
32      ); // MaterialApp
```

ListView với con ListTile  
(có thể thay widgets khác)

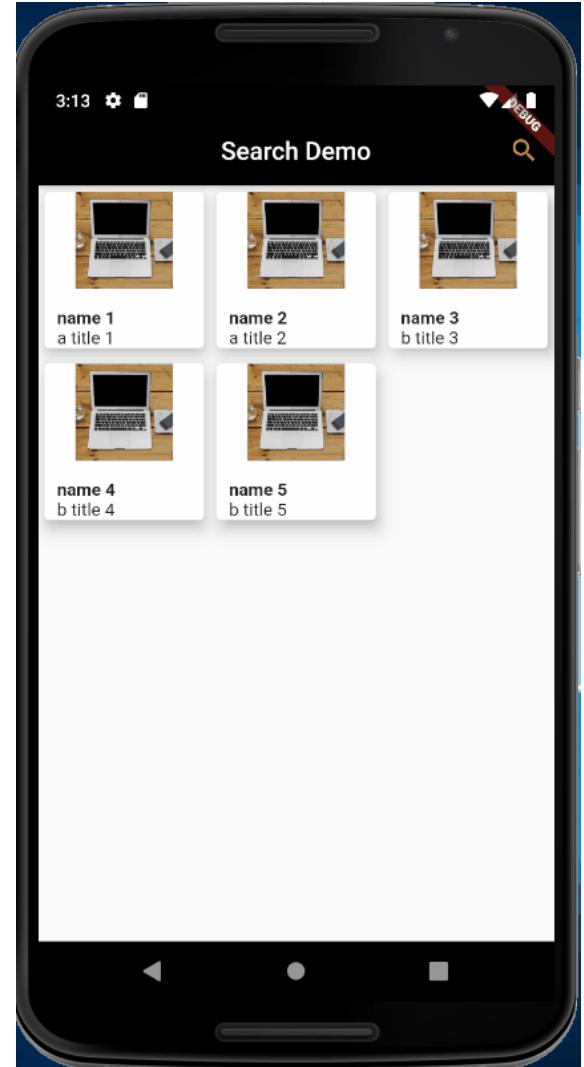


LOGS PROBLEMS 1 OUTPUT

^ v

## GridView {1}

GridView tương tự như ListView nhưng hiển thị các mục theo dạng lưới (grid). Đây là một cách tuyệt vời để hiển thị các mục với bố cục lưới, đặc biệt là trong các ứng dụng hiển thị hình ảnh hoặc sản phẩm

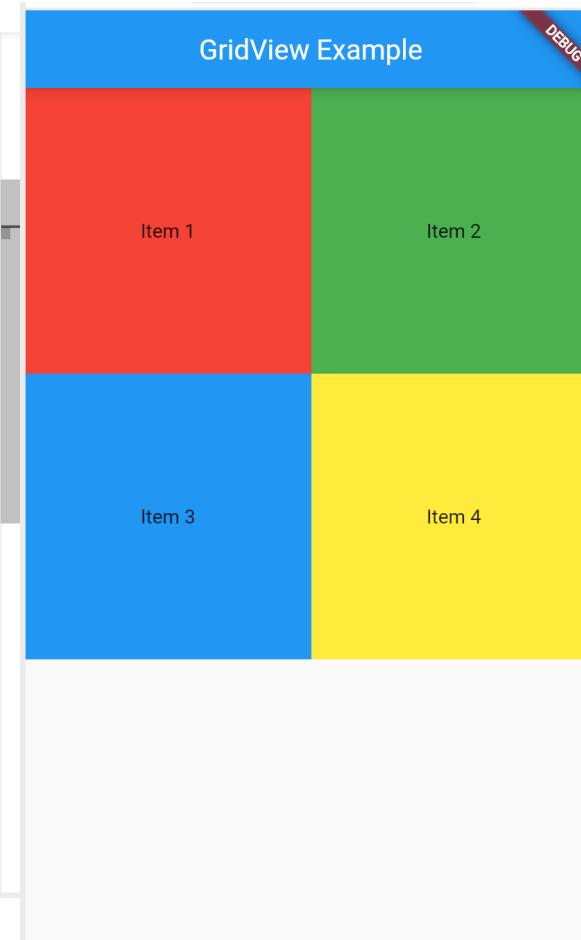


## GridView {2}

Tạo lưới với số cột được xác định bởi **crossAxisCount**

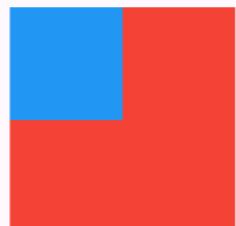
```
lib > main.dart
11   home: Scaffold(
12     appBar: AppBar(
13       title: Text('GridView Example'),
14     ), // AppBar
15     body: GridView.count(
16       crossAxisCount: 2, // Số cột trong GridView
17       children: <Widget>[
18         Container(
19           color: Colors.red,
20           child: Center(child: Text('Item 1')),
21         ), // Container
22         Container(
23           color: Colors.green,
24           child: Center(child: Text('Item 2')),
25         ), // Container
26         Container(
27           color: Colors.blue,
28           child: Center(child: Text('Item 3')),
29         ), // Container
30         Container(
31           color: Colors.yellow,
32           child: Center(child: Text('Item 4')),
33         ), // Container
34       ], // <Widget>[]
35     ), // GridView.count
36   ), // Scaffold
37 ); // MaterialApp
```

LOGS PROBLEMS ① OUTPUT

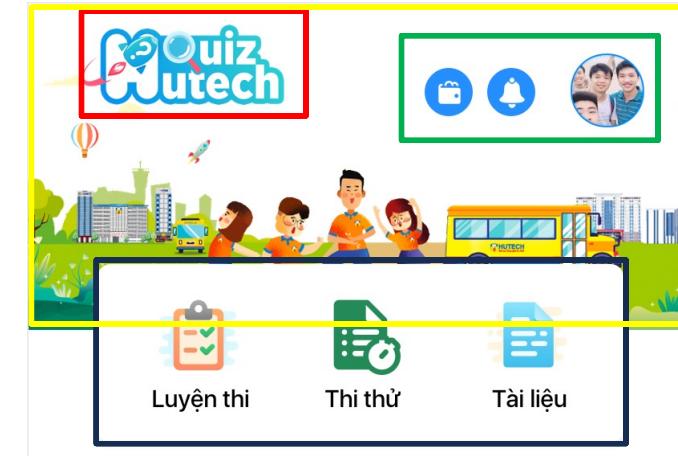


## Positioned {1}

**Positioned** là một widget được sử dụng bên trong **Stack** để định vị các widget con của nó tương đối so với các cạnh của **stack**. Đây là một widget hữu ích khi ta cần định vị chính xác các widget con



### Basic Widgets Example



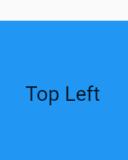
## Positioned {2}

Positioned sử dụng bên trong Stack để định vị widget con.

```
lib > main.dart
14     ), // AppBar
15     body: Stack(
16         children: <Widget>[
17             Positioned(
18                 top: 50, // Khoảng cách từ cạnh trên của Stack
19                 left: 50, // Khoảng cách từ cạnh trái của Stack
20                 child: Container(
21                     width: 100,
22                     height: 100,
23                     color: Colors.blue,
24                     child: Center(child: Text('Top Left')),
25                 ), // Container
26             ), // Positioned
27             Positioned(
28                 bottom: 50, // Khoảng cách từ cạnh dưới của Stack
29                 right: 50, // Khoảng cách từ cạnh phải của Stack
30                 child: Container(
31                     width: 100,
32                     height: 100,
33                     color: Colors.red,
34                     child: Center(child: Text('Bottom Right')),
35                 ), // Container
36             ), // Positioned
37         ], // <Widget>[]
38     ), // Stack
39 ), // Scaffold
40 ); // MaterialApp
```

LOGS PROBLEMS ① OUTPUT

Positioned Example



## Expanded {1}

- **Expanded** là một widget giúp **mở rộng** **một widget** con trong một **Row**, **Column** hoặc **Flex** để **chiếm không gian** còn lại.
- Hữu ích khi ta muốn một widget con chiếm không gian nhiều nhất có thể trong một hướng cụ thể.



## Expanded {2}

Sử dụng widget Column với  
widget con Expanded

2 widget Container đã thiết lập  
chiều cao cố định

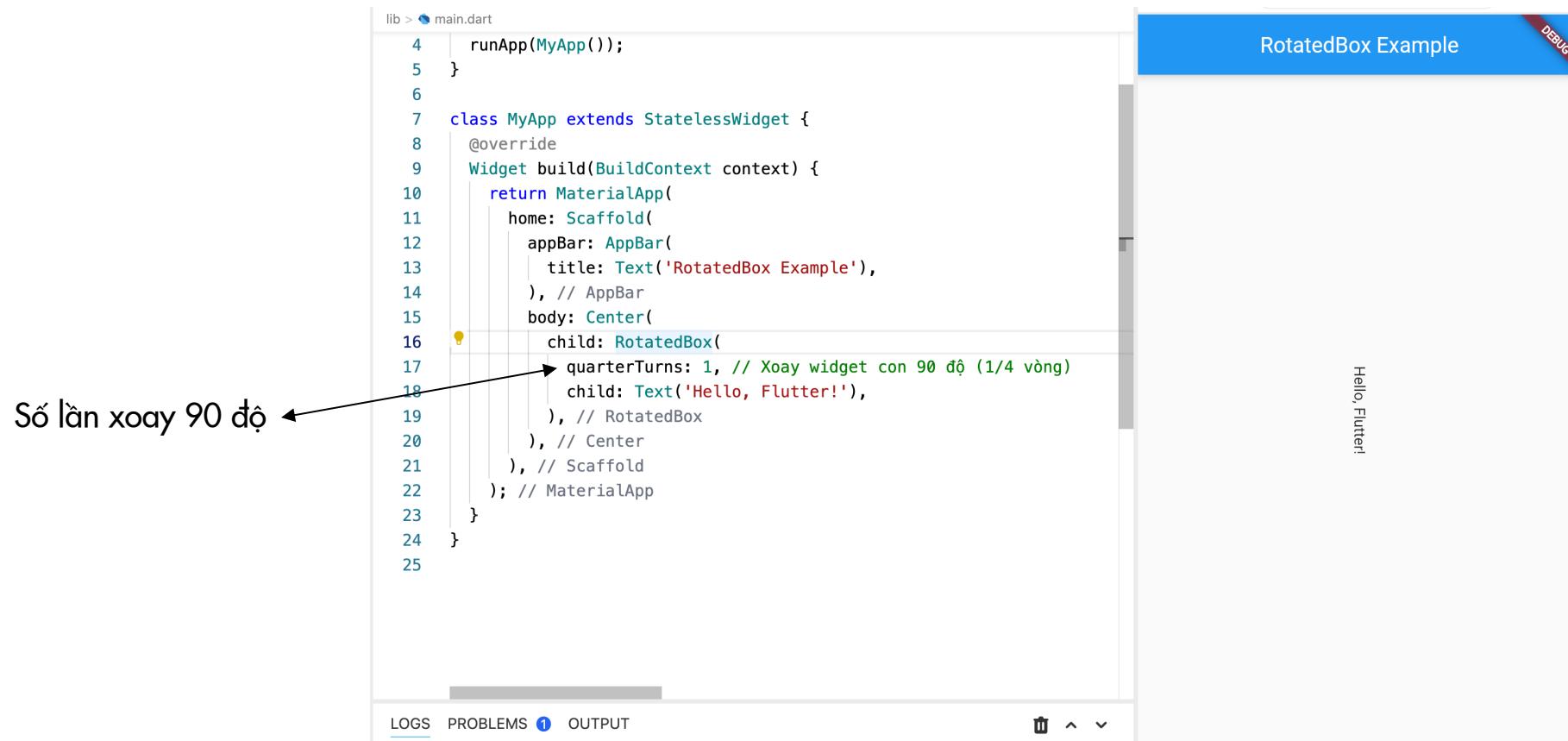
```
lib > main.dart
12     appBar: AppBar(
13       title: Text('Expanded Example'),
14     ), // AppBar
15     body: Column(
16       children: <Widget>[
17         Container(
18           height: 100,
19           color: Colors.red,
20           child: Center(child: Text('Top Container')),
21         ), // Container
22         Expanded(
23           child: Container(
24             color: Colors.green,
25             child: Center(child: Text('Expanded Container')),
26           ), // Container
27         ), // Expanded
28         Container(
29           height: 100,
30           color: Colors.blue,
31           child: Center(child: Text('Bottom Container')),
32         ), // Container
33       ], // <Widget>[]
34     ), // Column
35   ), // Scaffold
36 ); // MaterialApp
37 }
38 }
```

LOGS PROBLEMS 1 OUTPUT



## RotatedBox

**RotatedBox** là một widget xoay một widget con theo bội số của 90 độ.



The screenshot shows the Flutter development environment. On the left, the code editor displays `main.dart` with the following code:

```
lib > main.dart
4   runApp(MyApp());
5 }
6
7 class MyApp extends StatelessWidget {
8   @override
9   Widget build(BuildContext context) {
10     return MaterialApp(
11       home: Scaffold(
12         appBar: AppBar(
13           title: Text('RotatedBox Example'),
14         ), // AppBar
15         body: Center(
16           child: RotatedBox(
17             quarterTurns: 1, // Xoay widget con 90 độ (1/4 vòng)
18             child: Text('Hello, Flutter!'),
19           ), // RotatedBox
20         ), // Center
21       ), // Scaffold
22     ); // MaterialApp
23 }
24
25
```

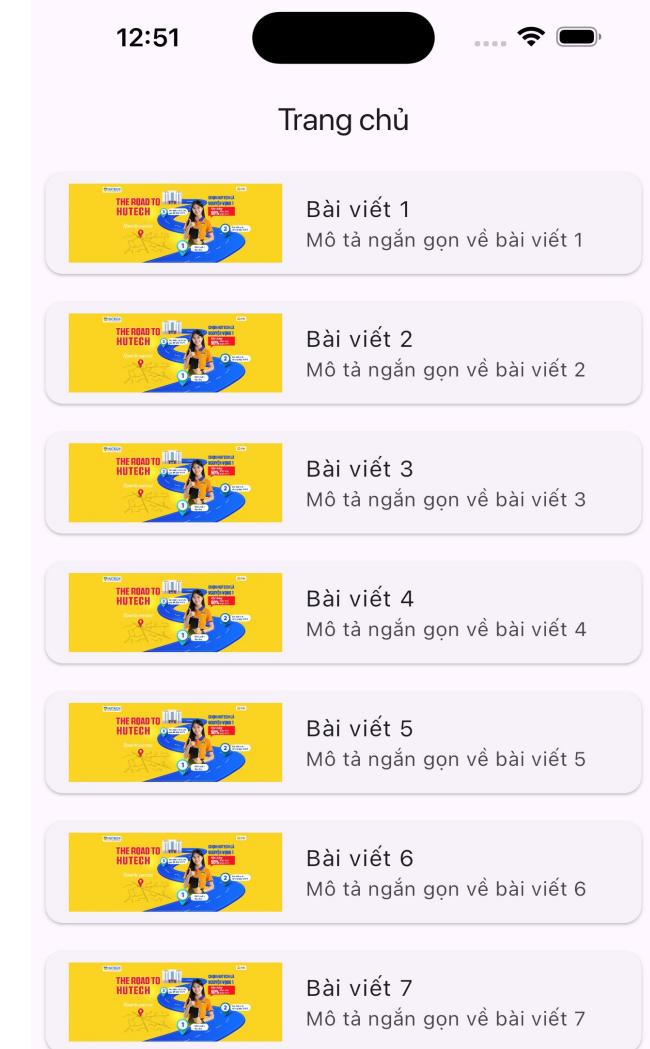
A callout arrow points from the text "Số lần xoay 90 độ" to the line `quarterTurns: 1`. To the right, the emulator window shows the app running with the title "RotatedBox Example" and the text "Hello, Flutter!" rotated 90 degrees counter-clockwise.

Trang chủ

## Thiết kế giao diện cho màn hình Trang chủ. Yêu cầu:

- Tạo một màn hình Trang chủ hiển thị danh sách các mục tin tức hoặc bài viết.
- Sử dụng ListView (hoặc GridView) để hiển thị danh sách.
- Mỗi mục trong danh sách bao gồm một hình ảnh, tiêu đề và mô tả ngắn gọn, được bao quanh bởi một Card (hoặc Container).

Lưu ý: Cần tạo 1 class riêng (HomeScreen) cho màn hình này.





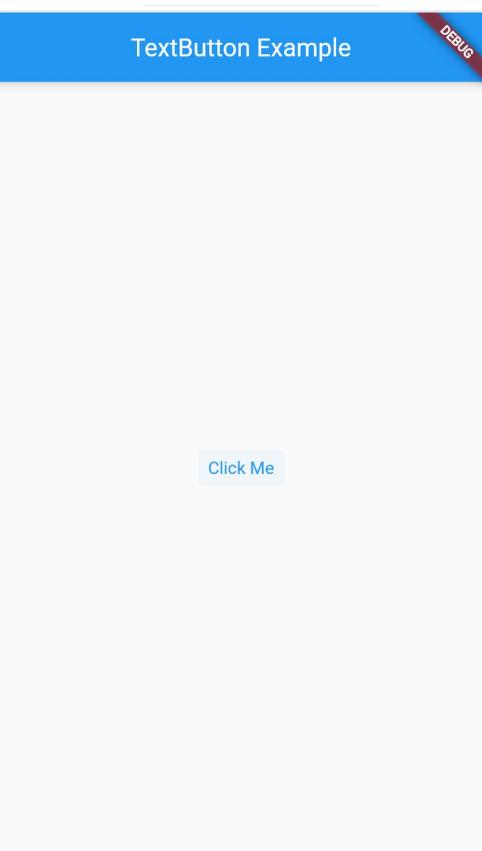
**TÌM HIỂU CƠ BẢN**

**INTERACTIVITY**

## Text Button

**TextButton** là một loại nút bấm đơn giản, thường được sử dụng cho các hành động không quá quan trọng

Hàm xử lý các lệnh khi nút  
được nhấn



The screenshot shows a Flutter application running in the Android Studio emulator. The title bar says "TextButton Example". The main screen has a blue header with the title "TextButton Example". Below it is a white body containing a single TextButton with the text "Click Me". A cursor arrow is pointing at the button. In the bottom right corner of the slide, there is a large watermark-like image of a hand cursor pointing towards the text "Click Me! 0".

```
lib > main.dart
7 class MyApp extends StatelessWidget {
8   @override
9   Widget build(BuildContext context) {
10     return MaterialApp(
11       home: Scaffold(
12         appBar: AppBar(
13           title: Text('TextButton Example'),
14         ), // AppBar
15         body: Center(
16           child: TextButton(
17             onPressed: () {
18               // Hành động khi nhấn nút
19               print('TextButton pressed');
20             },
21             child: Text('Click Me'),
22           ), // TextButton
23         ), // Center
24       ), // Scaffold
25     ); // MaterialApp
26   }
27 }
```

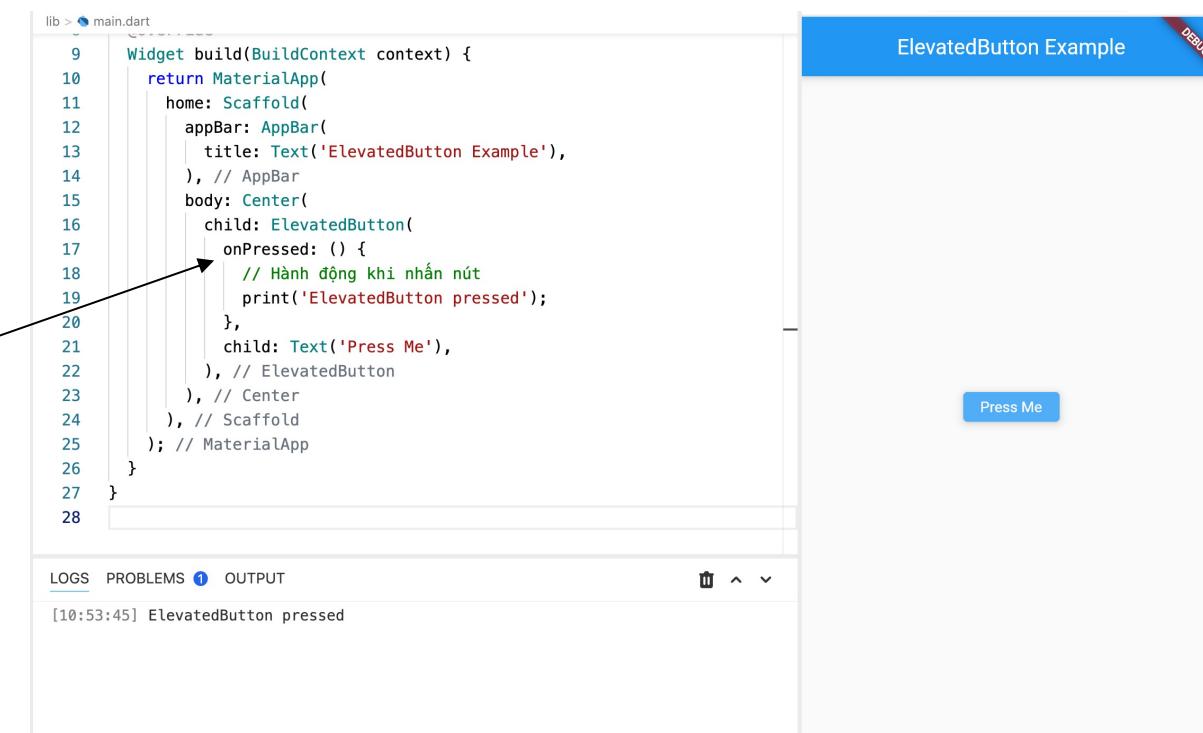
LOGS PROBLEMS OUTPUT

```
[10:48:24] TextButton pressed
[10:48:24] TextButton pressed
[10:48:24] TextButton pressed
[10:48:24] TextButton pressed
[10:48:30] TextButton pressed
```

## Elevated Button

**ElevatedButton** là một nút bấm nâng cao với hiệu ứng nẩy lên đẹp mắt khi được nhấn, thường được sử dụng cho các hành động chính trong ứng dụng

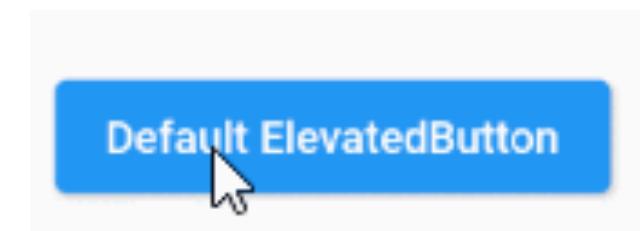
Hàm xử lý các lệnh khi nút  
được nhấn



The screenshot shows the Android Studio interface. On the left, the code for `main.dart` is displayed:

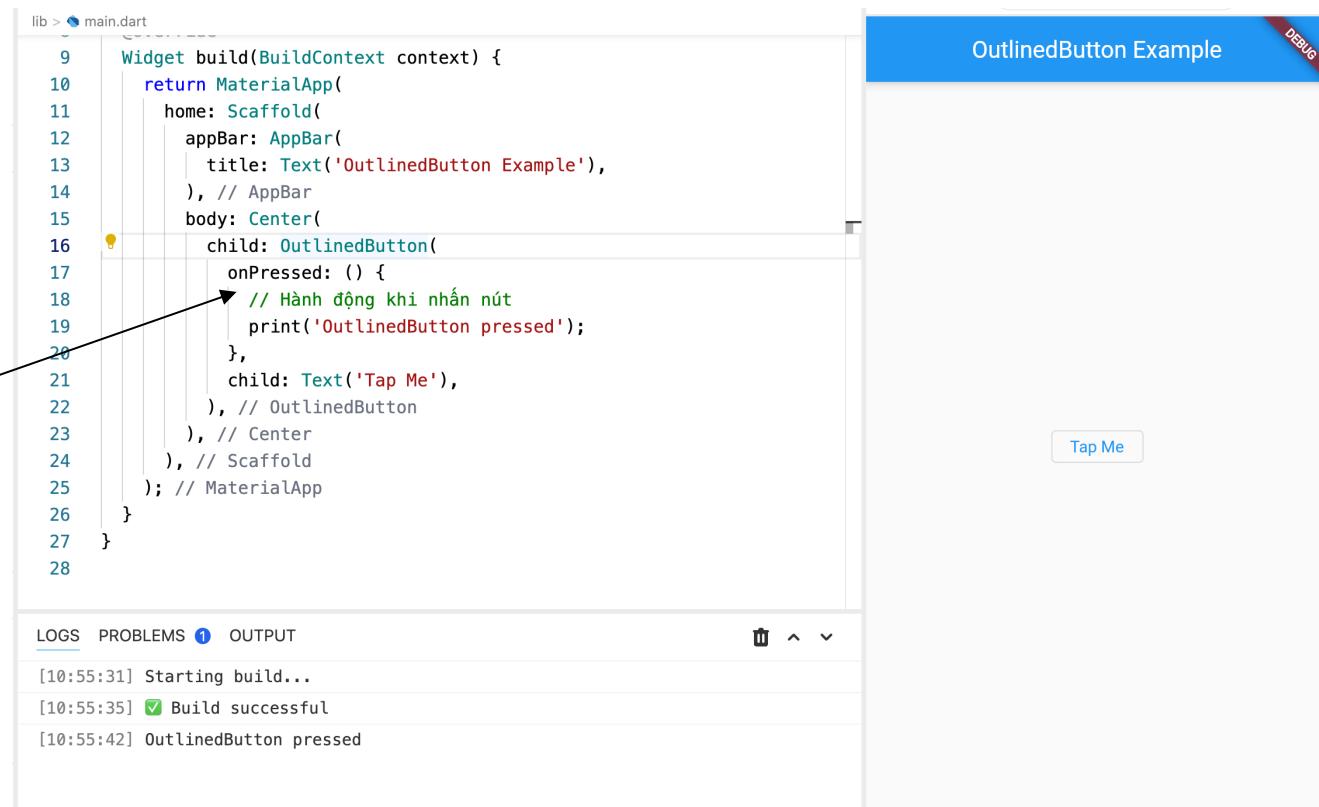
```
lib > main.dart
9  Widget build(BuildContext context) {
10    return MaterialApp(
11      home: Scaffold(
12        appBar: AppBar(
13          title: Text('ElevatedButton Example'),
14        ), // AppBar
15        body: Center(
16          child: ElevatedButton(
17            onPressed: () {
18              // Hành động khi nhấn nút
19              print('ElevatedButton pressed');
20            },
21            child: Text('Press Me'),
22          ), // ElevatedButton
23        ), // Center
24      ), // Scaffold
25    ); // MaterialApp
26  }
27 }
```

An arrow points from the text "Hàm xử lý các lệnh khi nút  
được nhấn" to the `onPressed` callback in the code. On the right, the application window titled "ElevatedButton Example" shows a blue button labeled "Press Me". Below the code editor, the log output shows the message "[10:53:45] ElevatedButton pressed".



## Outline Button

**OutlinedButton** là một nút bấm với viền, thường được sử dụng cho các hành động phụ



Hàm xử lý các lệnh khi nút được nhấn

lib > main.dart

```
9  Widget build(BuildContext context) {
10    return MaterialApp(
11      home: Scaffold(
12        appBar: AppBar(
13          title: Text('OutlinedButton Example'),
14        ), // AppBar
15        body: Center(
16          child: OutlinedButton(
17            onPressed: () {
18              // Hành động khi nhấn nút
19              print('OutlinedButton pressed');
20            },
21            child: Text('Tap Me'),
22          ), // OutlinedButton
23        ), // Center
24      ), // Scaffold
25    ); // MaterialApp
26  }
27}
28
```

LOGS PROBLEMS 1 OUTPUT

[10:55:31] Starting build...
[10:55:35] ✓ Build successful
[10:55:42] OutlinedButton pressed

OutlinedButton Example DEBUG

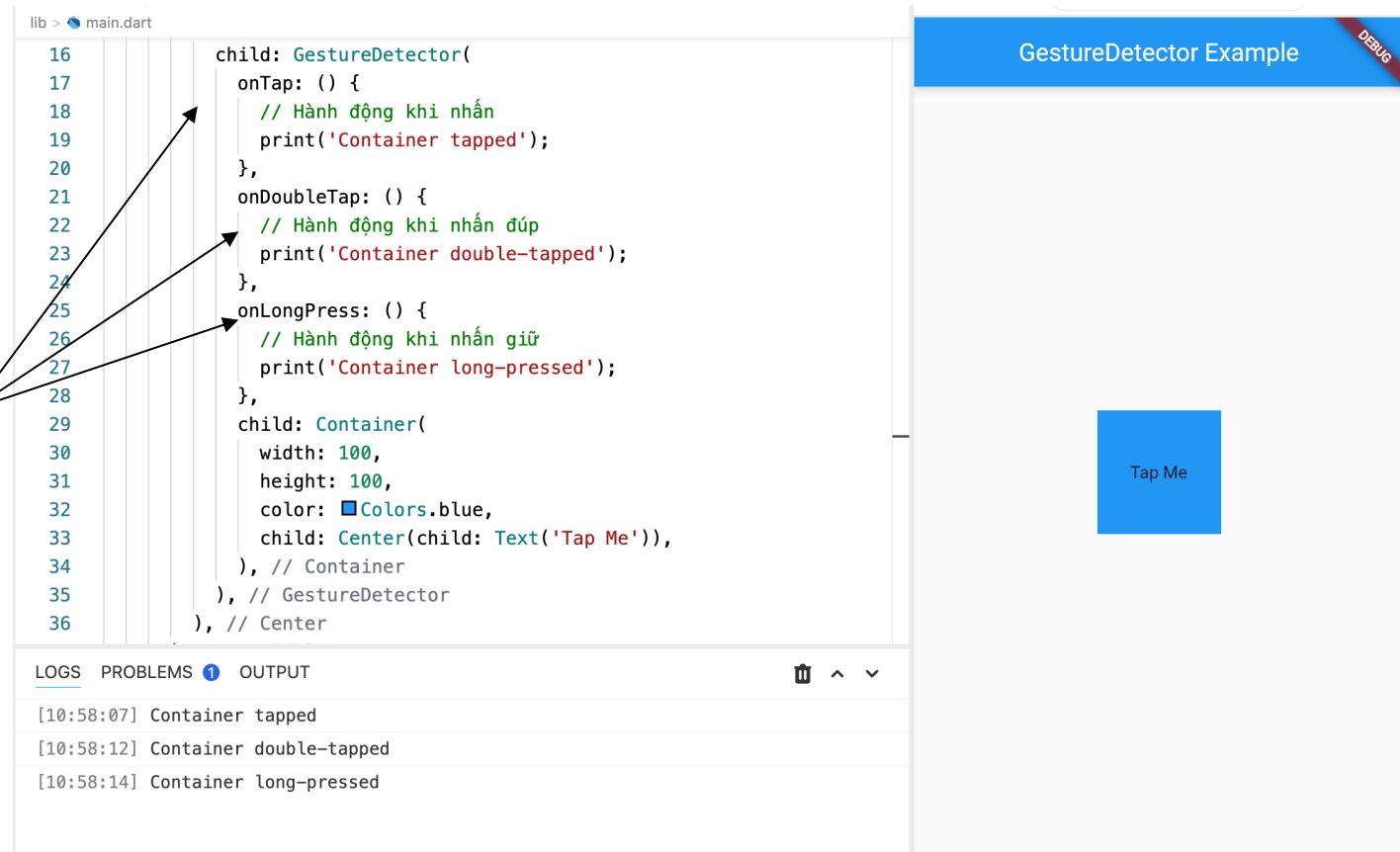
Tap Me

Click OutlinedButton!

## Gestures

Gestures là các hành động của người dùng như chạm, kéo, vuốt mà ứng dụng có thể phản hồi

Hàm xử lý các lệnh tương ứng  
với hành động



```
lib > main.dart
16     child: GestureDetector(
17       onTap: () {
18         // Hành động khi nhấn
19         print('Container tapped');
20       },
21       onDoubleTap: () {
22         // Hành động khi nhấn đúp
23         print('Container double-tapped');
24       },
25       onLongPress: () {
26         // Hành động khi nhấn giữ
27         print('Container long-pressed');
28       },
29       child: Container(
30         width: 100,
31         height: 100,
32         color: Colors.blue,
33         child: Center(child: Text('Tap Me')),
34       ), // Container
35     ), // GestureDetector
36   ), // Center
```

The screenshot shows the Android Studio interface with the code in main.dart. Three arrows point from the text "Hàm xử lý các lệnh tương ứng với hành động" to the event handlers (onTap, onDoubleTap, onLongPress) in the code. On the right, a preview window titled "GestureDetector Example" shows a blue square with the text "Tap Me". Below the code editor, the log output shows three printed messages: "[10:58:07] Container tapped", "[10:58:12] Container double-tapped", and "[10:58:14] Container long-pressed".

LOGS PROBLEMS 1 OUTPUT

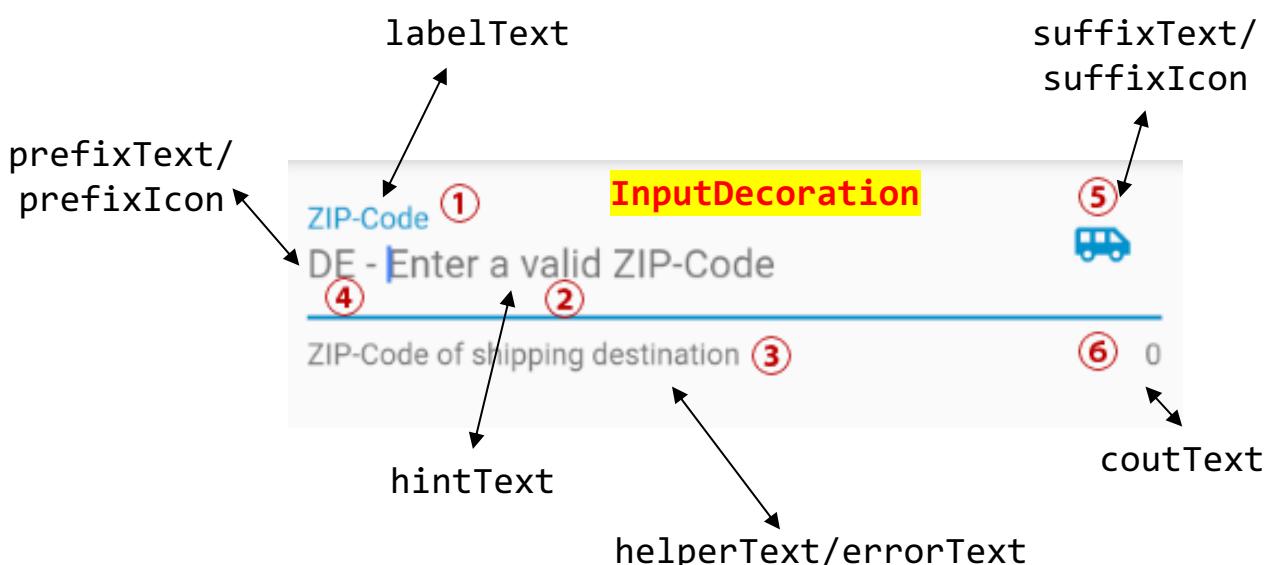
[10:58:07] Container tapped  
[10:58:12] Container double-tapped  
[10:58:14] Container long-pressed

## TextField

**TextField** là một widget cho phép người dùng nhập và chỉnh sửa văn bản. Nó thường được sử dụng để thu thập dữ liệu như tên, email, mật khẩu, v.v.

```
lib > main.dart
13 |   title: Text('TextField Example'),
14 | ), // AppBar
15 | body: Padding(
16 |   padding: const EdgeInsets.all(16.0),
17 |   child: Column(
18 |     children: <Widget>[
19 |       TextField(
20 |         decoration: InputDecoration(
21 |           labelText: 'Enter your name',
22 |           border: OutlineInputBorder(),
23 |         ), // InputDecoration
24 |         // TextField
25 |         SizedBox(height: 16),
26 |         TextField(
27 |           decoration: InputDecoration(
28 |             labelText: 'Enter your email',
29 |             border: OutlineInputBorder(),
30 |           ), // InputDecoration
31 |           keyboardType: TextInputType.emailAddress,
32 |           ), // TextField
33 |         ], // <Widget>[]
34 |       ), // Column
35 |       // Padding
36 |     ), // Scaffold
37 |   ); // MaterialApp
38 |
39 }
```

LOGS PROBLEMS 1 OUTPUT



## Form

- Form là một widget chứa các Input Field và cung cấp cơ chế xác thực và lưu trữ dữ liệu.
- Giúp ta quản lý các Input Field và kiểm tra tính hợp lệ của dữ liệu người dùng.

```
class LoginScreen extends StatefulWidget {
  @override
  _LoginScreenState createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  // GlobalKey để xác định form và quản lý trạng thái của nó
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();

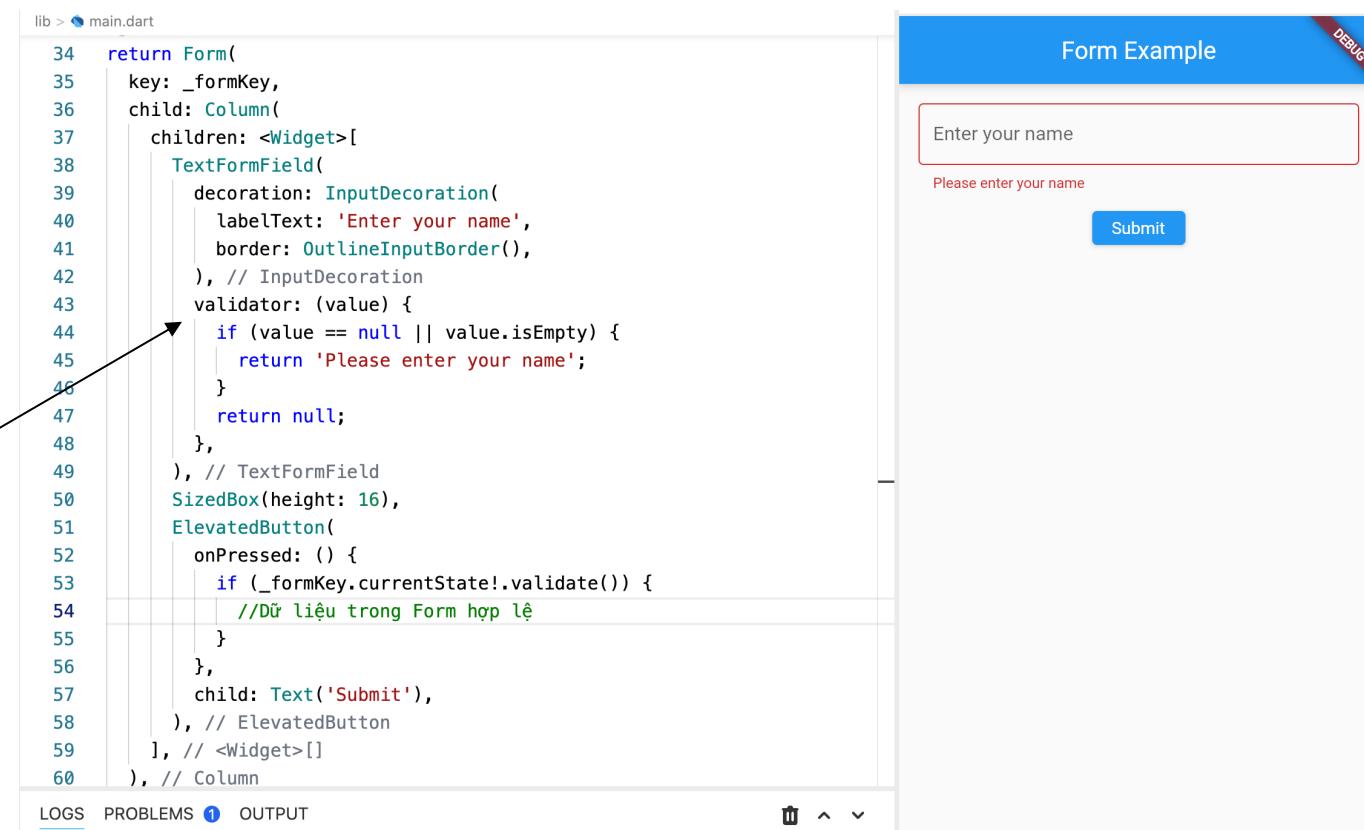
  // Các Controller để quản lý các Input Field
  final TextEditingController _emailController;
  final TextEditingController _passwordController;

  void _login() {
    // Kiểm tra ràng buộc các Input Field có trong Form
    if (_formKey.currentState!.validate()) {
    }
  }
}

Form structure diagram (right side):
child: Form(
  key: _formKey,
  child: Column(
    children: <Widget>[
      TextFormField(
        controller: _emailController,
        decoration: InputDecoration(labelText: 'Email'),
        validator: (value) {
          if (value == null || value.isEmpty) {
            return 'Vui lòng nhập Email';
          }
          return null;
        },
      ),
      // Tương tự với Password
      TextFormField(...),
      SizedBox(height: 20),
      ElevatedButton(
        onPressed: _login,
        child: Text('Đăng nhập'),
      ),
    ],
  ),
)
```

## TextField

**TextField** là 1 widget tạo ra một **TextField** và có thể hiển thị lỗi khi dữ liệu sai. (thường đi kèm với **Form** widget)



The screenshot shows the Flutter development environment. On the left, the code for `main.dart` is displayed:

```
lib > main.dart
34   return Form(
35     key: _formKey,
36     child: Column(
37       children: <Widget>[
38         TextFormField(
39           decoration: InputDecoration(
40             labelText: 'Enter your name',
41             border: OutlineInputBorder(),
42           ), // InputDecoration
43           validator: (value) {
44             if (value == null || value.isEmpty) {
45               return 'Please enter your name';
46             }
47             return null;
48           },
49         ), // TextFormField
50         SizedBox(height: 16),
51         ElevatedButton(
52           onPressed: () {
53             if (_formKey.currentState!.validate()) {
54               //Dữ liệu trong Form hợp lệ
55             }
56           },
57           child: Text('Submit'),
58         ), // ElevatedButton
59       ], // <Widget>[]
60     ), // Column
```

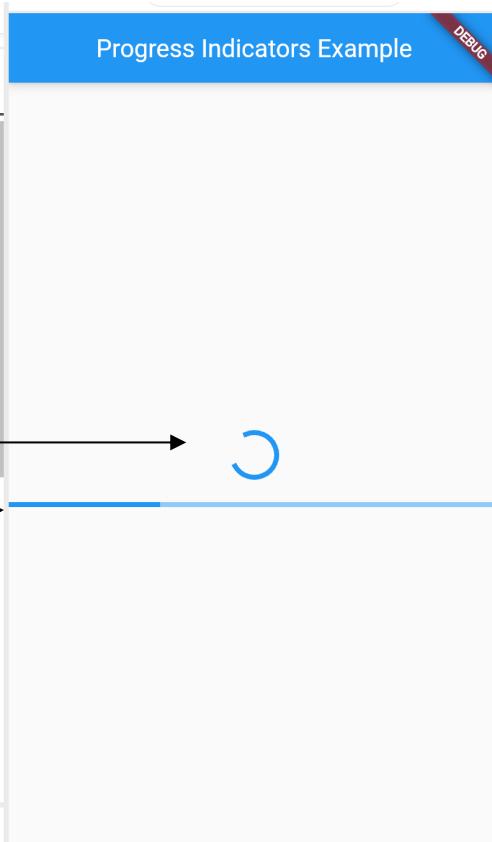
An arrow points from the text "Thuộc tính **validator** chưa hàm xác thực giá trị của **TextField**" to the line of code where the `validator` property is defined.

On the right, the running application titled "Form Example" is shown. It features a single `TextFormField` with the placeholder "Enter your name". Below it, an error message "Please enter your name" is displayed. A blue "Submit" button is at the bottom right.

Thuộc tính **validator** chưa  
hàm xác thực giá trị của  
**TextField**

## Progress Indicator

**Progress Indicators** là các widget dùng để hiển thị trạng thái đang xử lý của một tác vụ.



The screenshot shows a Flutter application titled "Progress Indicators Example". On the left, the code for "main.dart" is displayed:

```
lib > main.dart
1
2 class MyApp extends StatelessWidget {
3   @override
4   Widget build(BuildContext context) {
5     return MaterialApp(
6       home: Scaffold(
7         appBar: AppBar(
8           title: Text('Progress Indicators Example'),
9         ), // AppBar
10        body: Center(
11          child: Column(
12            mainAxisAlignment: MainAxisAlignment.center,
13            children: <Widget>[
14              // Hiển thị vòng tròn quay
15              CircularProgressIndicator(),
16              SizedBox(height: 20),
17              // Hiển thị thanh tiến trình
18              LinearProgressIndicator(),
19            ], // <Widget>[]
20          ), // Column
21        ), // Center
22      ), // Scaffold
23    ); // MaterialApp
24 }
```

The application interface features a blue header bar with the title "Progress Indicators Example" and a "DEBUG" button. Below the header, there is a vertical grey bar. At the top of this bar is a circular progress indicator (CircularProgressIndicator) with a blue loading ring. Below it is a horizontal blue progress bar (LinearProgressIndicator) with a blue progress segment extending across its width.

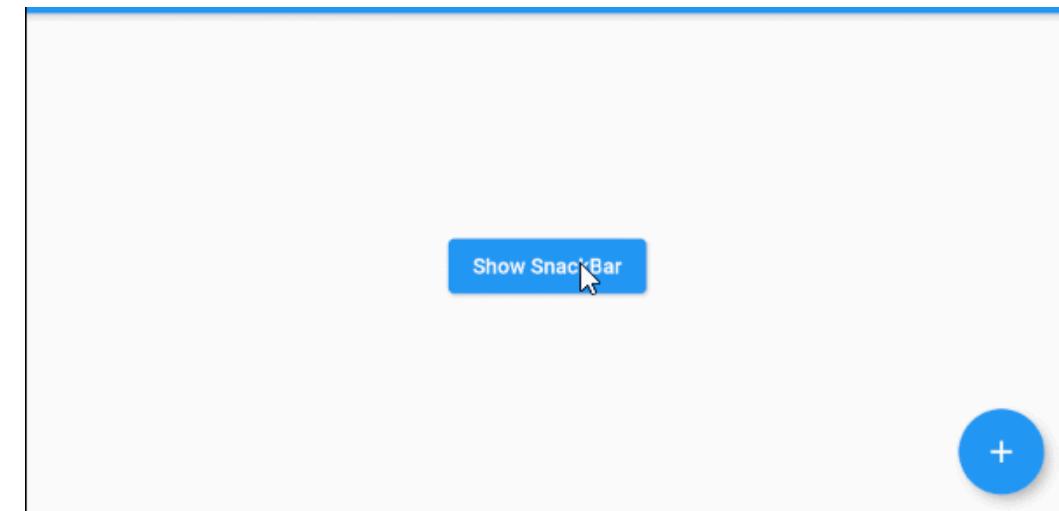
## SnackBar

**SnackBar** là một widget hiển thị thông báo ngắn gọn ở dưới cùng của màn hình.

Phương thức để hiển thị  
Snackbar

Hành động bổ sung (option)  
cho Snackbar

```
lib > main.dart
20
21 class SnackBarExample extends StatelessWidget {
22   @override
23   Widget build(BuildContext context) {
24     return Center(
25       child: ElevatedButton(
26         onPressed: () {
27           ScaffoldMessenger.of(context).showSnackBar(
28             SnackBar(
29               content: Text('This is a SnackBar'),
30               action: SnackBarAction(
31                 label: 'Undo',
32                 onPressed: () {
33                   // Some code to undo the change.
34                 },
35               ), // SnackBarAction
36             ), // SnackBar
37           );
38         },
39         child: Text('Show SnackBar'),
40       ), // ElevatedButton
41     ); // Center
42   }
43 }
44
```

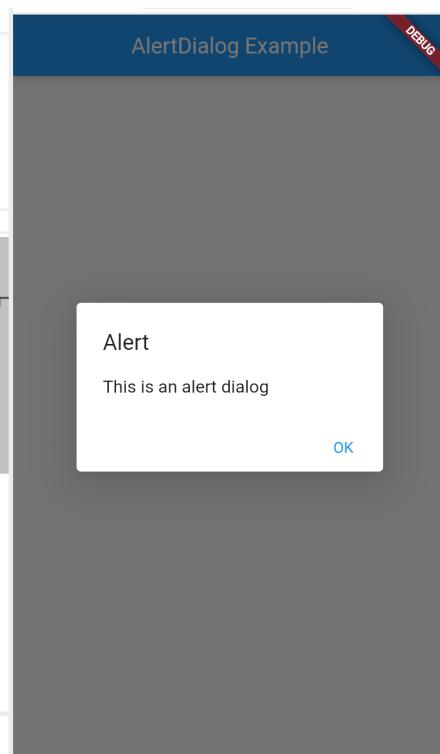


## Alert Dialog

**AlertDialog** là một widget hiển thị hộp thoại cảnh báo, thường được sử dụng để yêu cầu người dùng xác nhận hoặc thực hiện một hành động nào đó.

Phương thức hiển thị  
**AlertDialog**

Tạo nút hành động thoát  
**AlertDialog**



```
lib > main.dart
23  Widget build(BuildContext context) {
24    return Center(
25      child: ElevatedButton(
26        onPressed: () {
27          showDialog(
28            context: context,
29            builder: (context) {
30              return AlertDialog(
31                title: Text('Alert'),
32                content: Text('This is an alert dialog'),
33                actions: <Widget>[
34                  TextButton(
35                    onPressed: () {
36                      Navigator.of(context).pop();
37                    },
38                    child: Text('OK'),
39                  ), // TextButton
40                  ], // <Widget>[]
41                ); // AlertDialog
42              },
43            );
44          },
45          child: Text('Show AlertDialog'),
46        ), // ElevatedButton
47      ); // Center
48    }
49 }
```

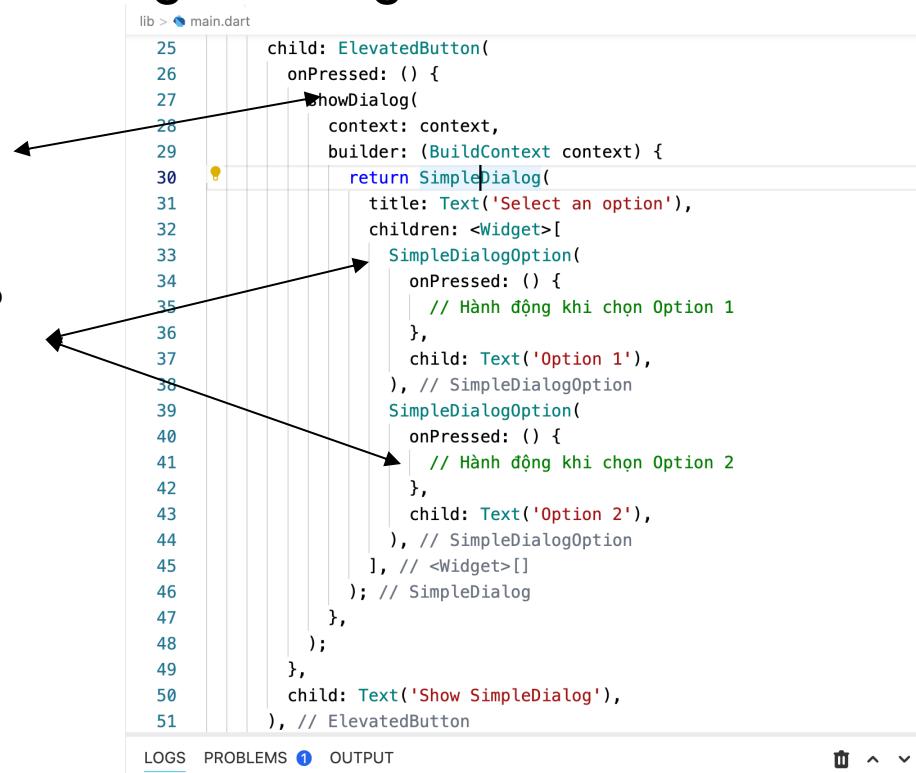
LOG PROBLEMS OUTPUT

## Simple Dialog

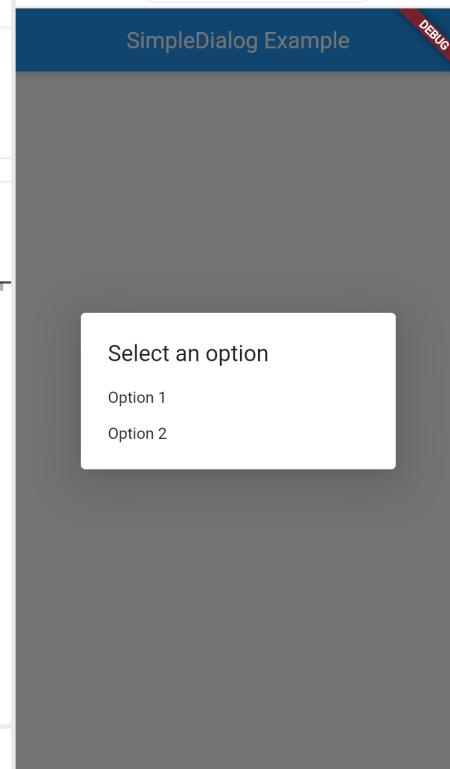
**SimpleDialog** là một widget hiển thị hộp thoại đơn giản, thường được sử dụng để cung cấp một danh sách các lựa chọn cho người dùng.

Phương thức hiển thị  
**SimpleDialog**

**SimpleDialogOption** cho  
các tùy chọn trong  
**SimpleDialog**

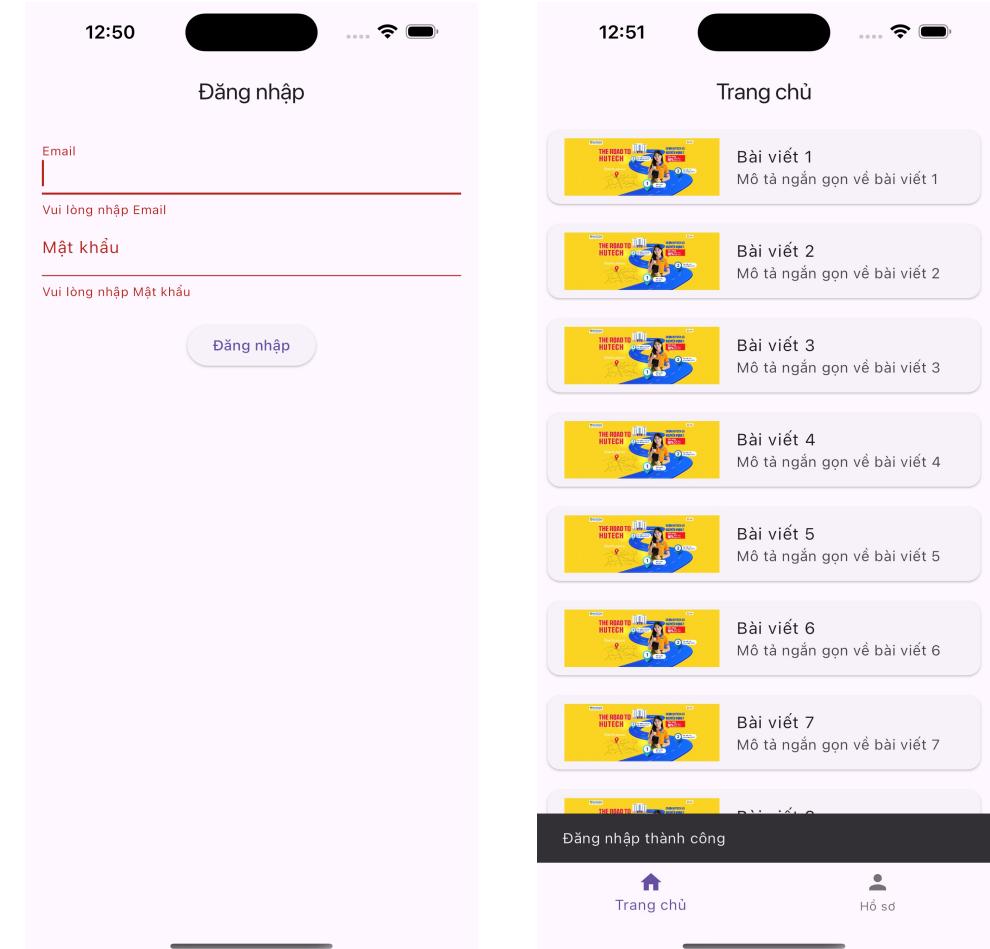


```
lib > main.dart
25     child: ElevatedButton(
26       onPressed: () {
27         showDialog(
28           context: context,
29           builder: (BuildContext context) {
30             return SimpleDialog(
31               title: Text('Select an option'),
32               children: <Widget>[
33                 SimpleDialogOption(
34                   onPressed: () {
35                     // Hành động khi chọn Option 1
36                   },
37                   child: Text('Option 1'),
38                 ), // SimpleDialogOption
39                 SimpleDialogOption(
40                   onPressed: () {
41                     // Hành động khi chọn Option 2
42                   },
43                   child: Text('Option 2'),
44                 ), // SimpleDialogOption
45               ],
46             );
47           );
48         },
49       ),
50       child: Text('Show SimpleDialog'),
51     ), // ElevatedButton
```



## Xây dựng giao diện đăng nhập. Yêu cầu:

- Tạo một giao diện Login với hai trường nhập liệu (email và mật khẩu) và một nút đăng nhập (Login Button).
- Hiển thị thông báo lỗi nếu người dùng không nhập đủ thông tin.
- Nếu nhập đầy đủ thông tin, hãy chuyển sang màn hình DashBoard, sử dụng BottomNavigationBar có 2 màn hình: Trang chủ (HomePage) và Hồ sơ (Profile). Đồng thời, hiển thị Snackbar thông báo đăng nhập thành công.



# Cảm ơn

## Các bạn đã chú ý lắng nghe



# HỎI ĐÁP



# Q&A