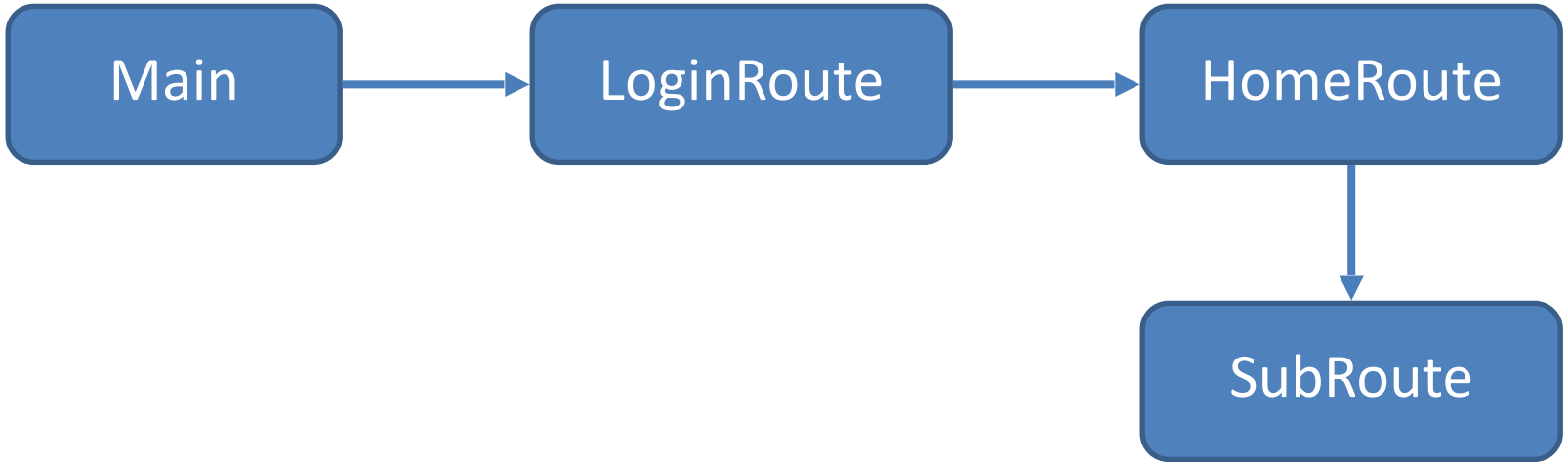
Resource Software Solution

Navigation and Dialogs

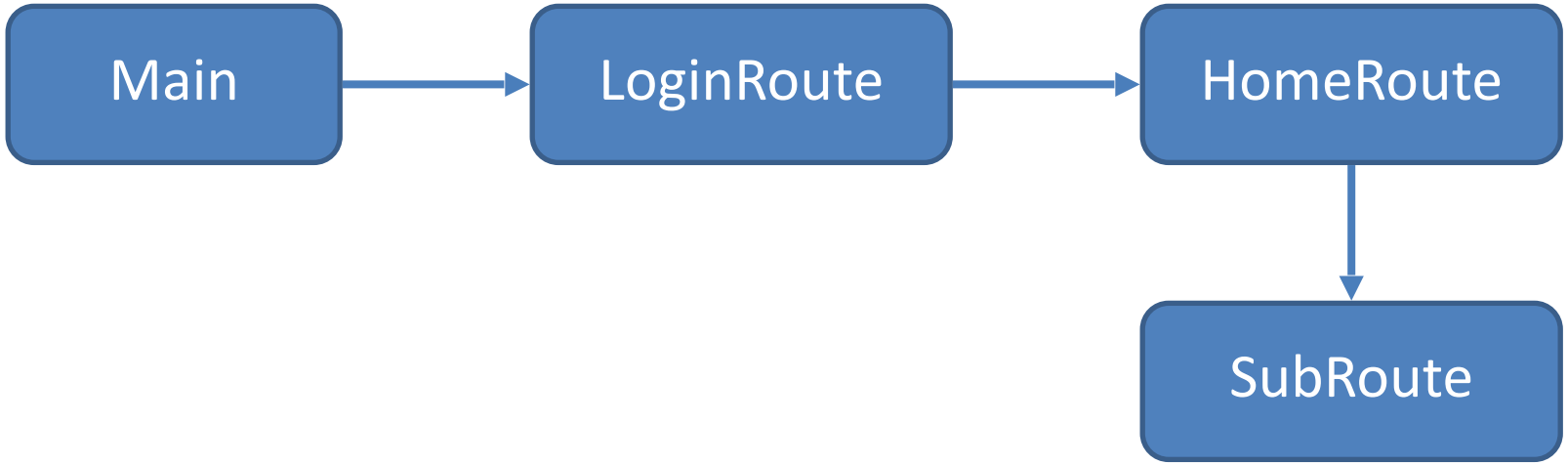
Navigation

1. `Navigator.`**`push`**`()`
2. `Navigator.`**`pop`**`()`
3. `Navigator.`**`pushAndRemoveUntil`**`()`
4. **`RouteSettings`**



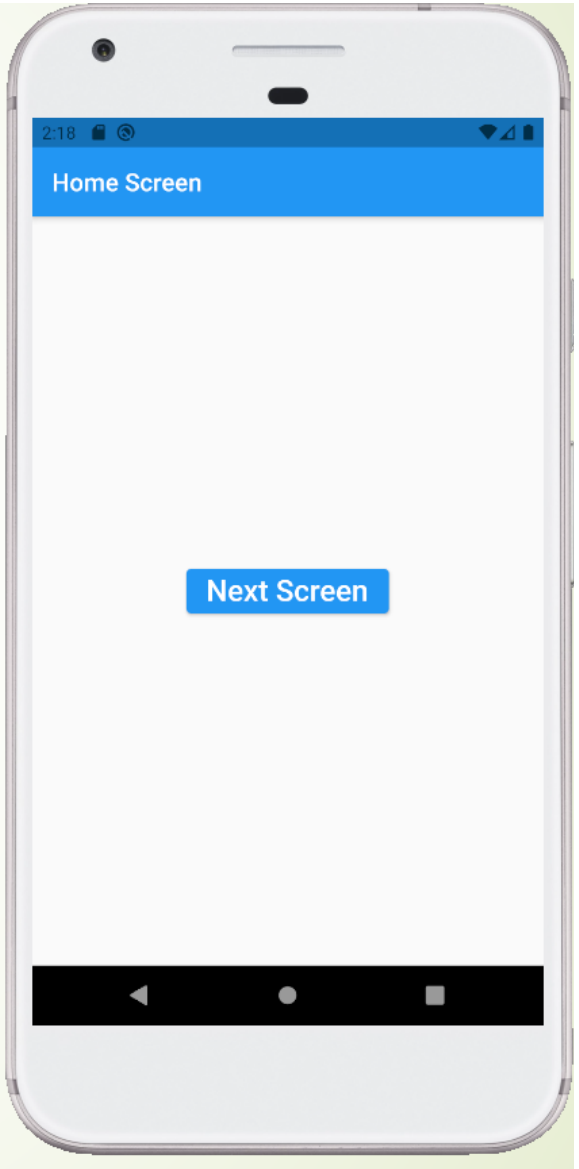
Navigation

- In Flutter, *screens* and *pages* are called *routes*
- *Open a new page using* `Navigator.push()`
- Return to the previous route using `Navigator.pop()`
- Open new page and clear all previous routes using `Navigator.pushAndRemoveUntil()`



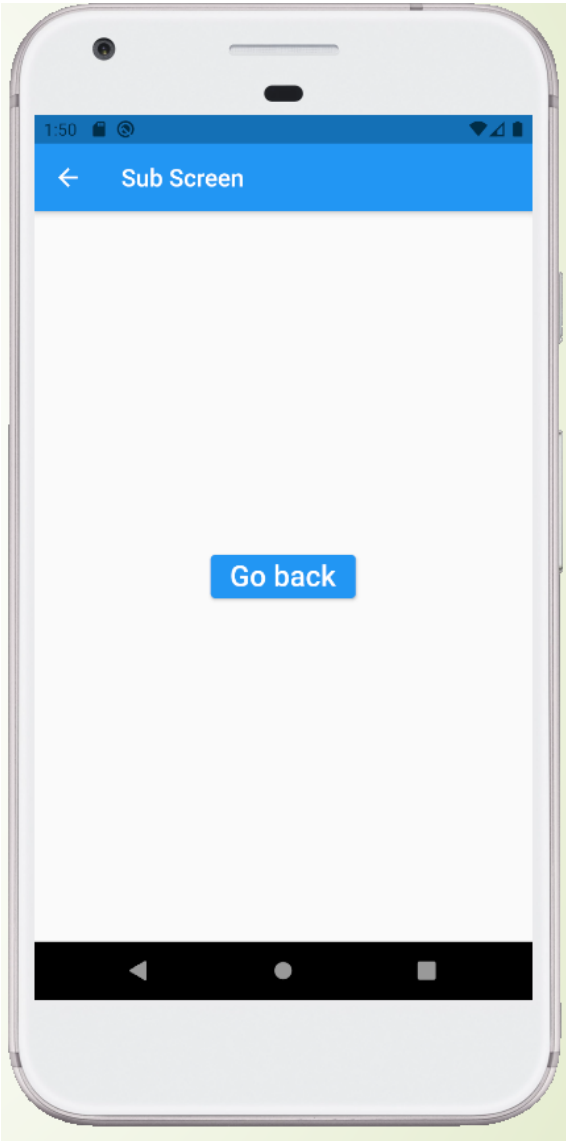
Home Screen

```dart
import 'package:demo_app/sub.dart';
import 'package:flutter/material.dart';

class HomeRoute extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Home Screen"),
      ), // AppBar
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => SubRoute()),
            );
          },
          child: Text('Next Screen',style: TextStyle(fontSize: 24)),
        ), // ElevatedButton
      ), // Center
    ); // Scaffold
  }
}
```



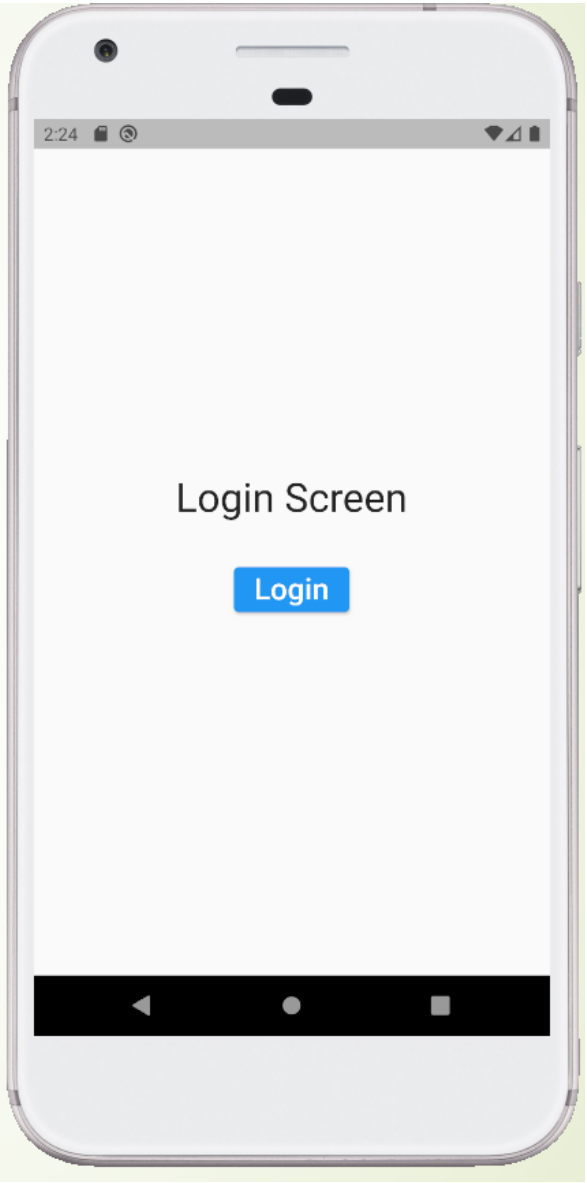
Sub Screen

```dart
import 'package:flutter/material.dart';
class SubRoute extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Sub Screen"),
      ), // AppBar
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            Navigator.pop(context);
          },
          child: Text('Go back', style: TextStyle(fontSize: 24)),
        ), // ElevatedButton
      ), // Center
    ); // Scaffold
  }
}
```



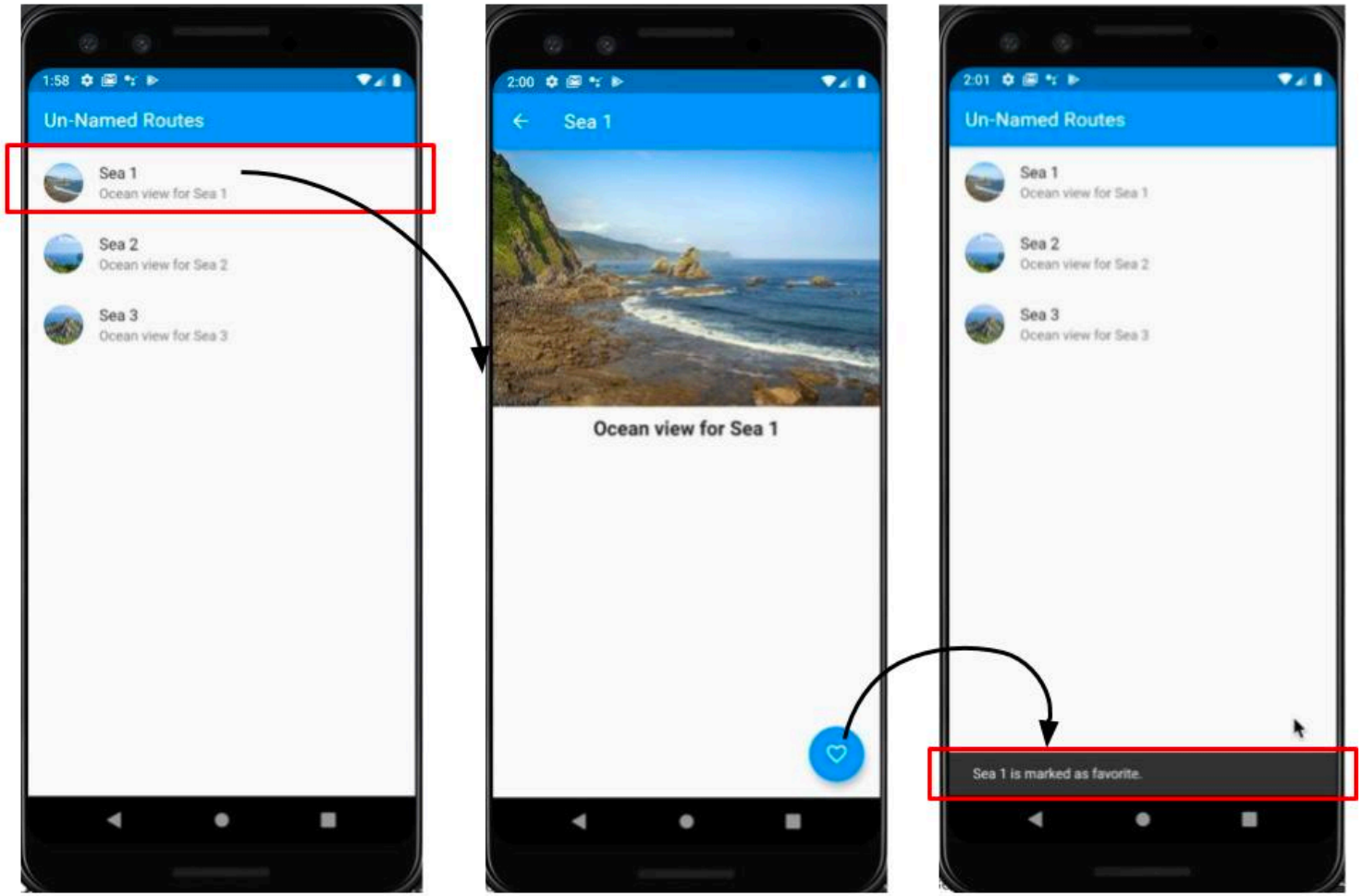
Login Screen

```dart
import 'package:flutter/material.dart';
import 'home.dart';
class LoginRoute extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text('Login Screen', style: TextStyle(fontSize: 32)),
            SizedBox(height: 30),
            ElevatedButton(
              onPressed: () {
                Navigator.pushAndRemoveUntil(context,
                  MaterialPageRoute(builder: (context) => HomeRoute()),
                  (route) => false);
              },
              child: Text('Login', style: TextStyle(fontSize: 24)),
            ), // ElevatedButton
          ],
        )), // Column // Center
    ); // Scaffold
  }
}
```



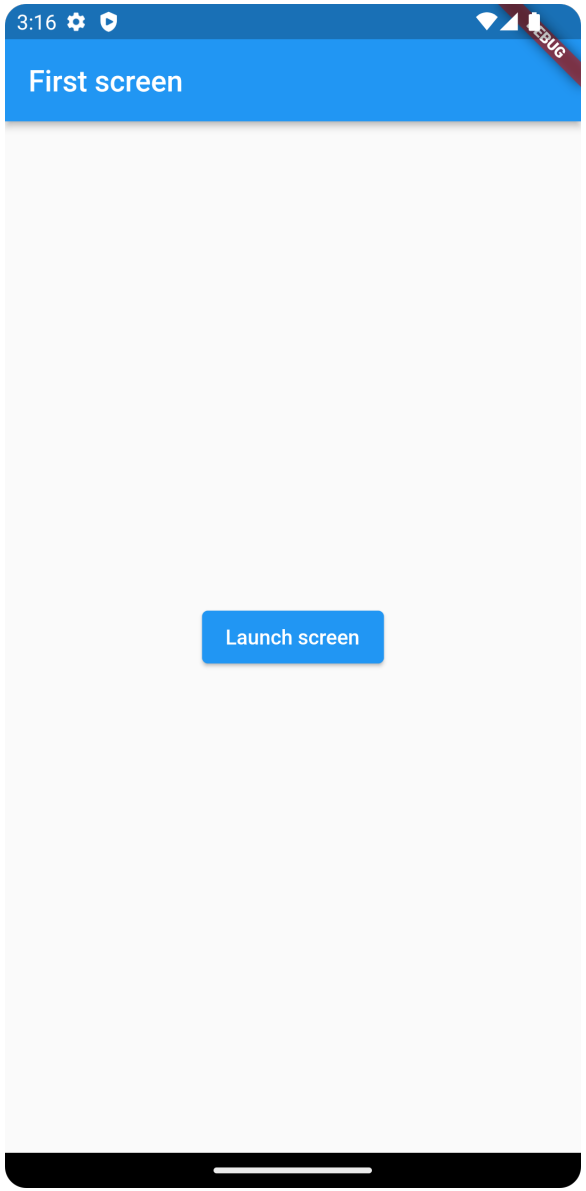
RouteSettings

- **Interaction** with the UI is an integral **part of any application**. But more often than not, the **information needs to be sent from one screen to another**.
- Pass the arguments using RouteSettings



RouteSettings

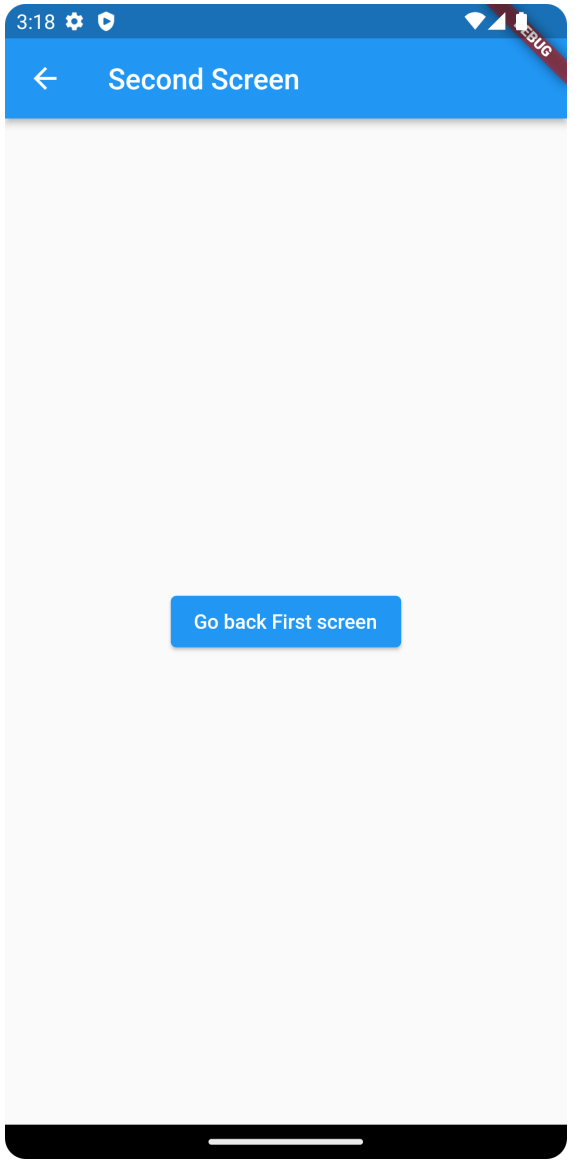
```dart
class FirstScreen extends StatelessWidget {
  const FirstScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('First screen'),
      ),  // AppBar
      body: Center(
        child: ElevatedButton(
          onPressed: () => {
            Navigator.push(
                context,
                MaterialPageRoute(
                    builder: (context) => const SecondScreen(),
                    settings: const RouteSettings(arguments: 'First screen')))
          },
          child: const Text('Launch screen'),
        ),  // ElevatedButton
      ),  // Center
    );  // Scaffold
  }
}
```



RouteSettings

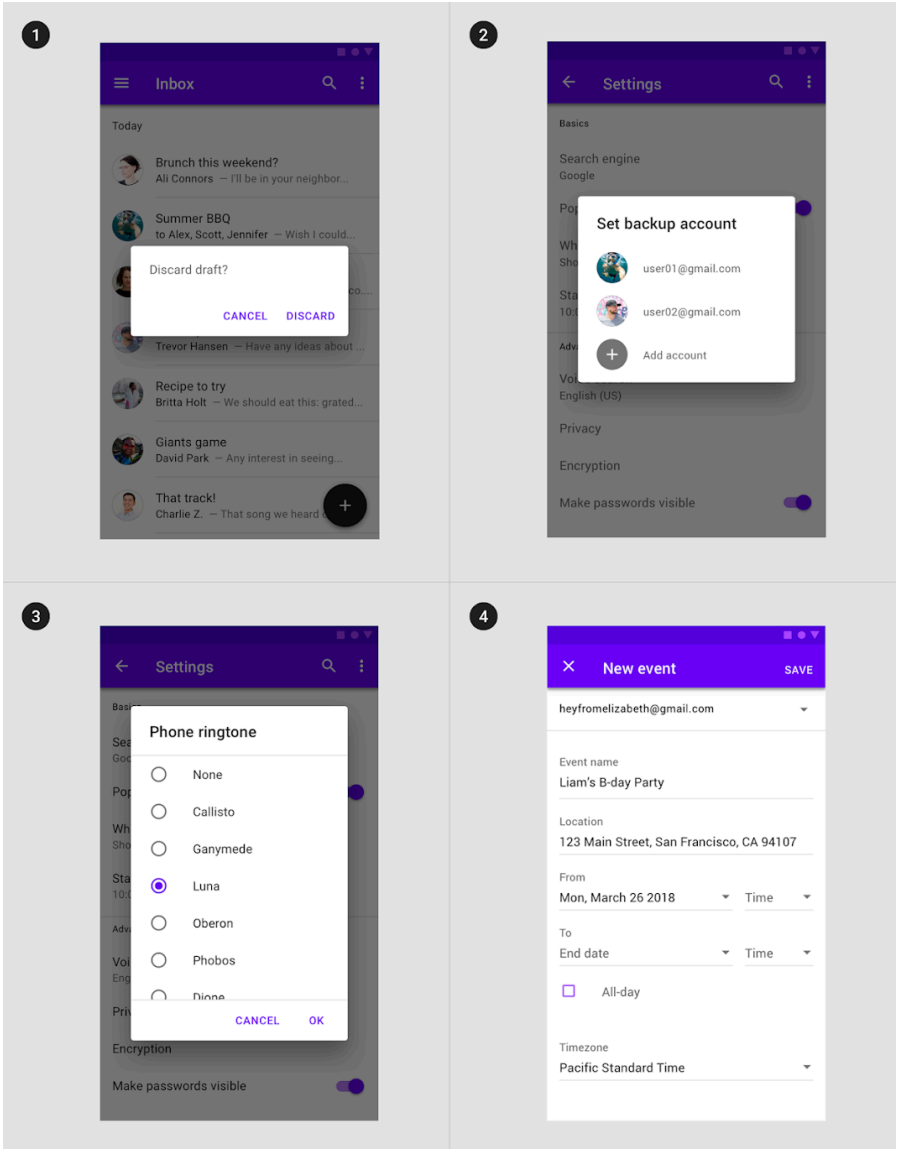
Resource Software Solution

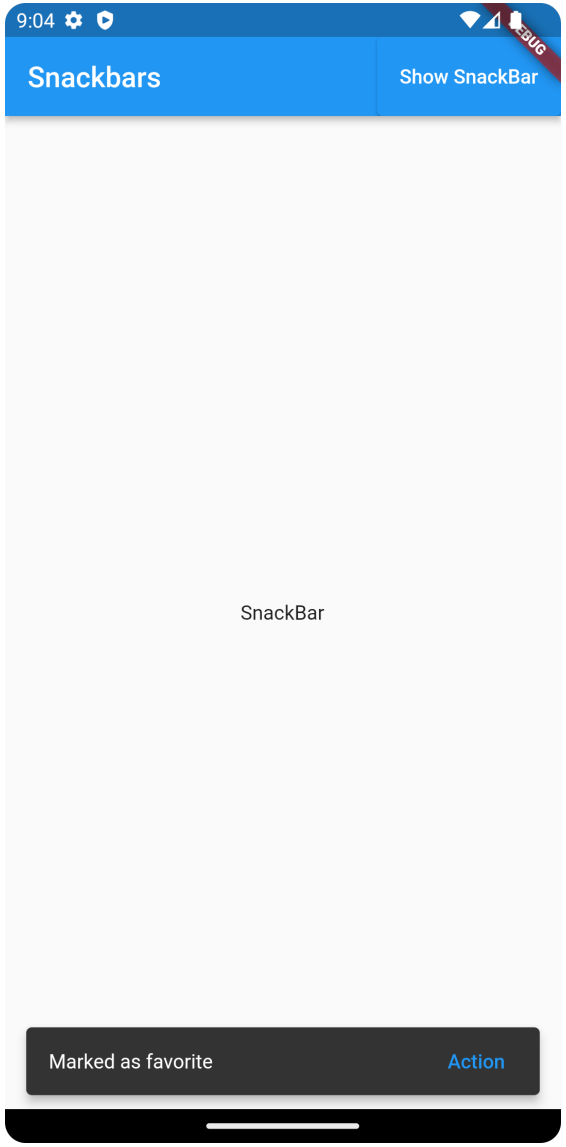
```dart
class SecondScreen extends StatelessWidget {
  const SecondScreen({super.key});

  @override
  Widget build(BuildContext context) {
    final data = ModalRoute.of(context)?.settings.arguments as String;

    return Scaffold(
      appBar: AppBar(
        title: const Text('Second Screen'),
      ),  // AppBar
      body: Center(
        child: ElevatedButton(
          onPressed: () => {Navigator.pop(context)},
          child: Text('Go back $data'),
        ),  // ElevatedButton
      ),  // Center
    );  // Scaffold
  }
}
```



Dialogs

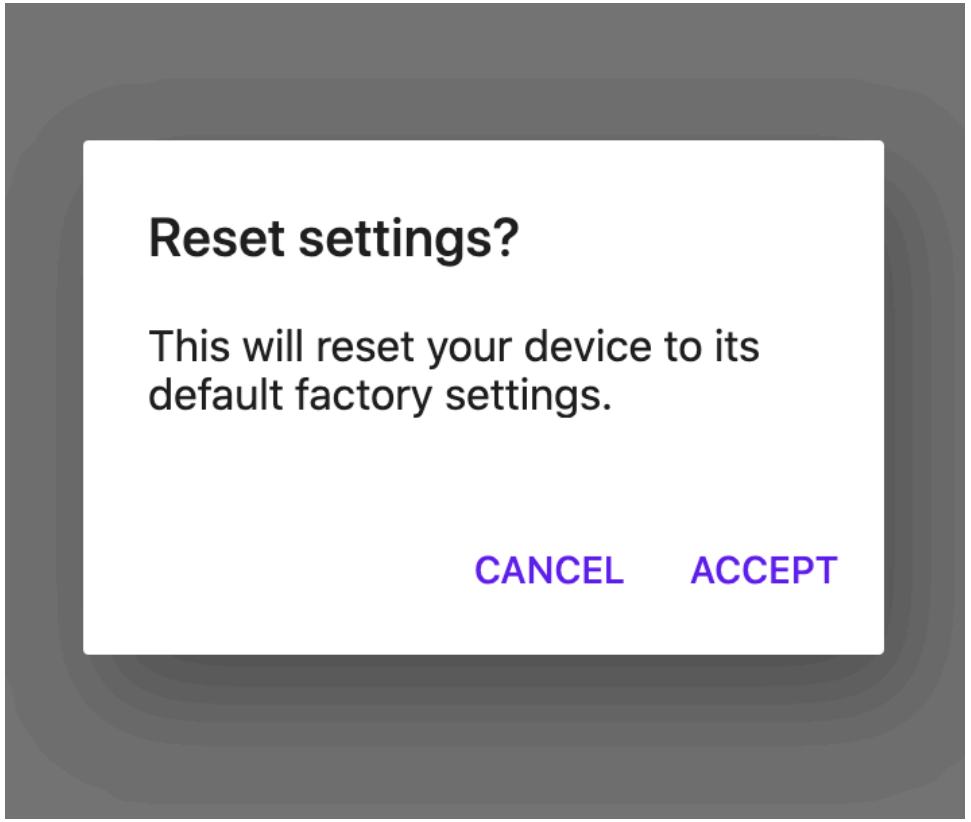
1. Alert
2. Simple
3. Full-screen
4. Snackbars





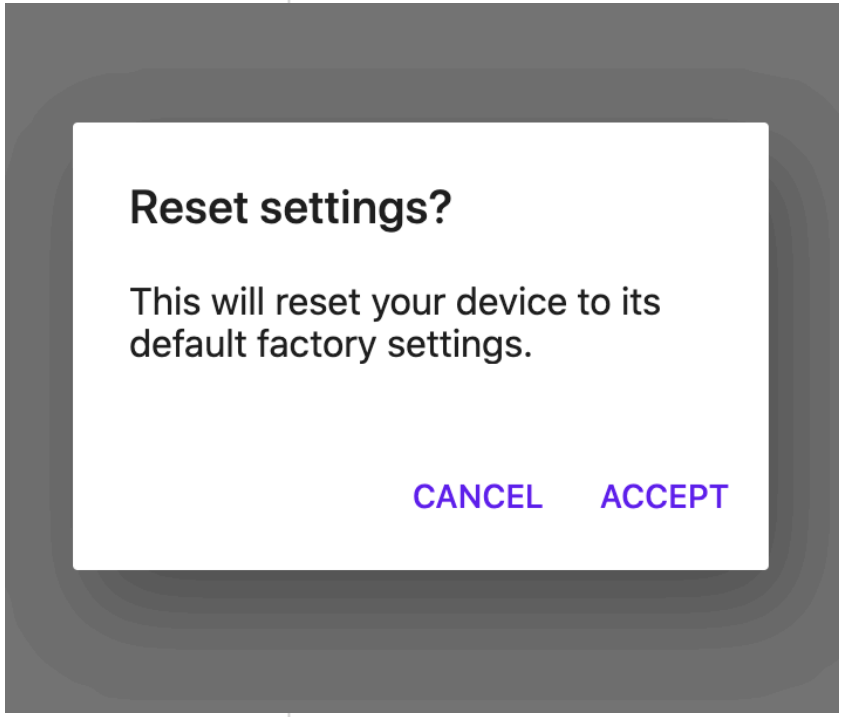
Alert Dialog

- Alert dialogs **interrupt users** with urgent information, details, or actions.
- The following example shows an alert dialog.



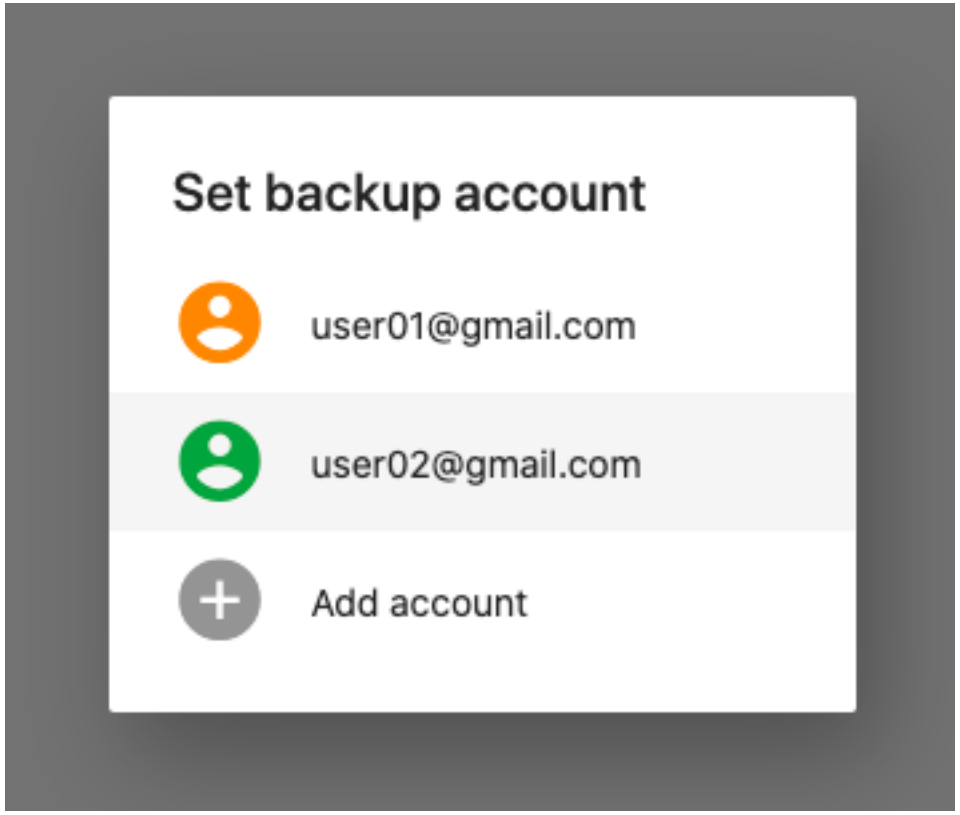
Alert Dialog

```dart
final AlertDialog dialog = AlertDialog(
  title: const Text('Reset settings?'),
  content:
  const Text('This will reset your device to its default factory settings.'),
  actions: [
    ElevatedButton(
      onPressed: () => Navigator.pop(context),
      child: const Text('CANCEL'),
    ),  // ElevatedButton
    ElevatedButton(
      onPressed: () => Navigator.pop(context),
      child: const Text('ACCEPT'),
    ),  // ElevatedButton
  ],
);  // AlertDialog
```



Simple Dialog

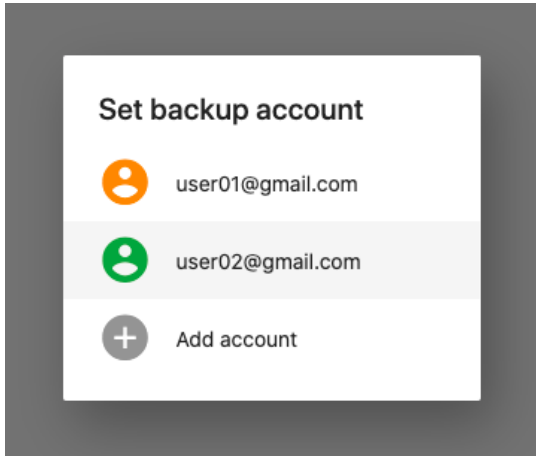
- Simple dialogs can display items that are immediately actionable when selected. They don’t have text buttons.
- The following example shows a simple dialog.



Simple Dialog

```dart
class SimpleDialogItem extends StatelessWidget {
  const SimpleDialogItem(
      {super.key, this.icon,  this.color,  this.text,  this.onPressed});

  final IconData? icon;
  final Color? color;
  final String? text;
  final VoidCallback? onPressed;

  @override
  Widget build(BuildContext context) {
    return SimpleDialogOption(
      onPressed: onPressed,
      child: Row(
        mainAxisAlignment: MainAxisAlignment.start,
        crossAxisAlignment: CrossAxisAlignment.center,
        children: [
          Icon(icon, size: 36.0, color: color),
          Padding(
            padding: const EdgeInsetsDirectional.only(start: 16.0),
            child: Text(text!),
          ),  // Padding
        ],
      ),  // Row
    );  // SimpleDialogOption
  }
}
```

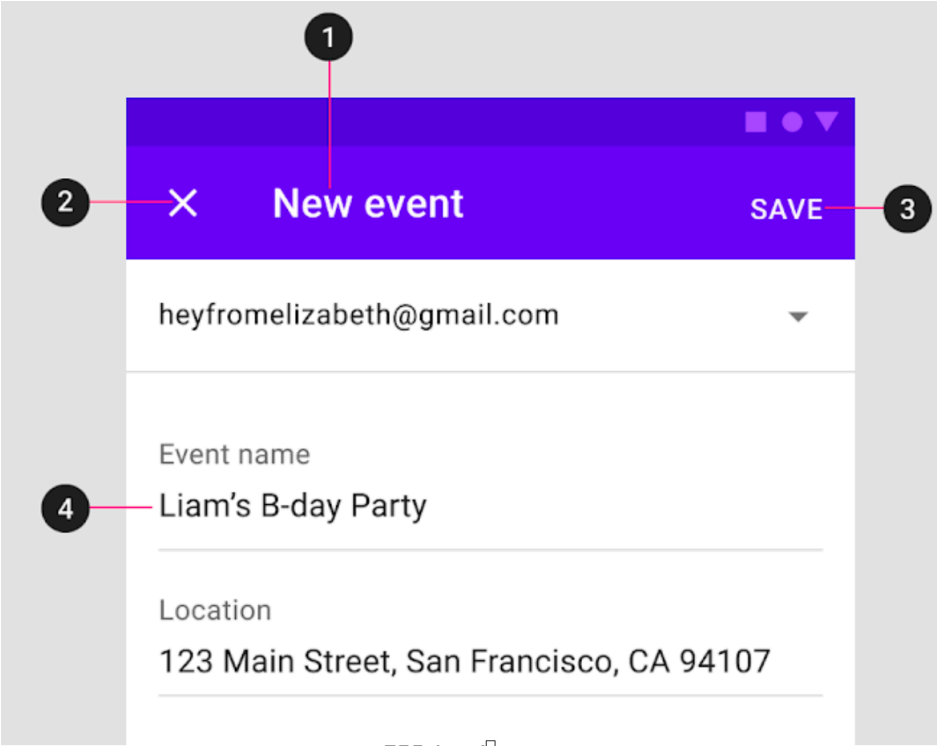
```dart
SimpleDialog createSimpleDialog(BuildContext context) {
  return SimpleDialog(
    title: const Text('Set backup account'),
    children: [
      SimpleDialogItem(
        icon: Icons.account_circle,
        color: Colors.orange,
        text: 'user01@gmail.com',
        onPressed: () {
          Navigator.pop(context, 'user01@gmail.com');
        },
      ),  // SimpleDialogItem
      SimpleDialogItem(
        icon: Icons.account_circle,
        color: Colors.green,
        text: 'user02@gmail.com',
        onPressed: () {
          Navigator.pop(context, 'user02@gmail.com');
        },
      ),  // SimpleDialogItem
      SimpleDialogItem(
        icon: Icons.add_circle,
        color: Colors.grey,
        text: 'Add account',
        onPressed: () {
          Navigator.pop(context, 'Add account');
        },
      ),  // SimpleDialogItem
    ],
  );  // SimpleDialog
}
```




```dart
final SimpleDialog dialog = createSimpleDialog(context);

return Scaffold(
  body: Center(
    child: ElevatedButton(
      onPressed: () {
        showDialog<void>(context: context, builder: (context) => dialog);
      },
      child: const Text("SHOW DIALOG"),
    ),  // ElevatedButton
  ),  // Center
);  // Scaffold
```

Full-screen Dialog

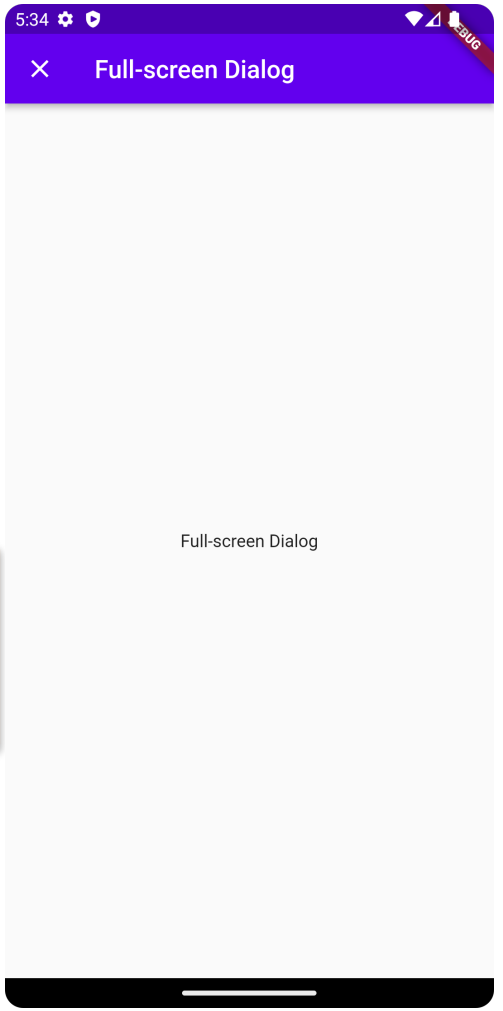
- Full-screen dialogs group a series of tasks, such as creating a calendar entry with the event title, date, location, and time.
- To use a full-screen dialog, simply set the **fullscreenDialog** to true when pushing a new **MaterialPageRoute**.



Full-screen Dialog

```dart
class FullScreenDialog extends StatelessWidget {
  const FullScreenDialog({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: const Color(0xFF6200EE),
        title: const Text('Full-screen Dialog'),
      ),  // AppBar
      body: const Center(
        child: Text('Full-screen Dialog'),
      ),  // Center
    );  // Scaffold
  }
}
```

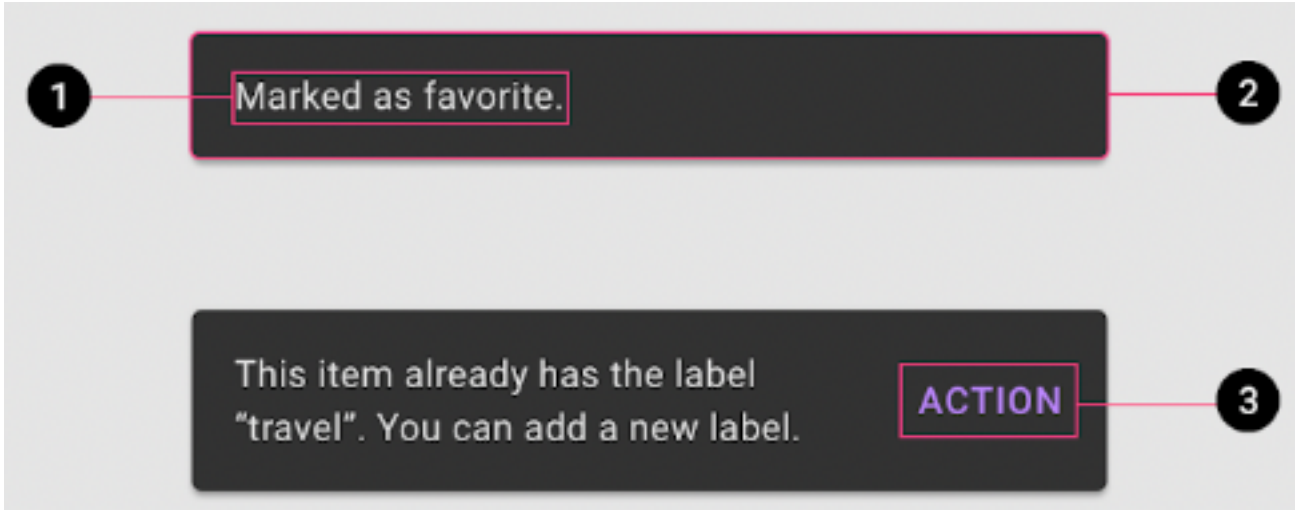
```dart
class MyFullScreenPage extends StatelessWidget {
  const MyFullScreenPage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Using Dialogs'),
      ),  // AppBar
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            Navigator.push(
                context,
                MaterialPageRoute(
                  builder: (context) => const FullScreenDialog(),
                  fullscreenDialog: true,
                ));  // MaterialPageRoute
          },
          child: const Text('Show Full-screen Dialog'),
        ),  // ElevatedButton
      ),  // Center
    );  // Scaffold
  }
}
```



Snackbars

- Snackbars inform users of a **process** that an app has **performed** or **will perform**. They **appear temporarily**, towards the **bottom** of the screen. They shouldn't interrupt the user experience, and they don't require user input to disappear.
- The following is an anatomy diagram of a snackbar

1. Text label
2. Container
3. Action (optional)



Snackbars

```dart
class SnackBarButton extends StatelessWidget {
  const SnackBarButton({super.key});

  @override
  Widget build(BuildContext context) {
    return ElevatedButton(
        onPressed: () {
          final snackBar = SnackBar(
            content: const Text('Marked as favorite'),
            behavior: SnackBarBehavior.floating,
            action: SnackBarAction(
                label: 'Action',
                onPressed: () {}),  // SnackBarAction
          );  // SnackBar

          ScaffoldMessenger.of(context).showSnackBar(snackBar);
        },
        child: const Text('Show SnackBar'));  // ElevatedButton
  }
}
```

```dart
class SnackBarsDemo extends StatelessWidget {
  const SnackBarsDemo({Key? key}) : super(key:

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Snackbars'),
        actions: const [SnackBarButton()],
      ),  // AppBar
      body: const Center(
        child: Text('SnackBar'),
      ),  // Center
    );  // Scaffold
  }
}
```