EXPERIMENT NO. 2

Aim: To design Flutter UI by including common widgets.

Theory:
Flutter is a cross-platform framework developed by Google for building natively compiled applications for mobile, web, and desktop from a single codebase. It employs a reactive-style programming paradigm, where widgets are the basic building blocks for constructing user interfaces. Widgets in Flutter are lightweight, customizable elements that can be composed and nested to create complex UI layouts.

Key features of Flutter include:

1.Fast Development: Flutter's hot reload feature allows developers to quickly see the effects of code changes without losing the app's state. This speeds up the development process and facilitates iterative testing and debugging.

2. Expressive UI: Flutter offers a rich set of customizable widgets that enable developers to create visually appealing and interactive user interfaces. These widgets range from basic elements like buttons and text fields to more complex components like lists, grids, and animations.

3. Native Performance: Flutter apps are compiled directly to native machine code, resulting in high performance and smooth animations across different platforms. By leveraging the Skia graphics engine, Flutter achieves consistent performance on iOS, Android, and other platforms.

4. Single Codebase: With Flutter, developers can write a single codebase for multiple platforms, reducing development time and effort. This not only streamlines the development process but also ensures consistency in functionality and design across different devices.

5. Platform Integration: Flutter provides plugins and packages for integrating with native platform features such as camera, geolocation, and sensors. This allows developers to access device-specific functionalities without compromising performance or user experience.

6. Community and Ecosystem: Flutter has a vibrant community of developers and contributors who actively contribute to its growth and evolution. This ecosystem includes libraries, tools, and resources that extend Flutter's capabilities and support developers in building diverse types of applications.

Designing a Flutter UI involves leveraging these features to create a visually appealing and functional user interface. Common widgets such as Container, Row, Column, Text, Image, Button, TextField, ListView, and GridView are frequently used to organize and display content within the app. By combining these widgets and customizing their properties, developers can design intuitive and user-friendly interfaces for their Flutter applications.

Properties and syntax for each widget used :

1. MaterialApp:
   - Properties:
     - title: A string representing the title of the application.
     - theme: An instance of ThemeData defining the overall theme of the application.
     - home: A widget representing the home screen of the application.
   - Syntax:
     dart
     MaterialApp(
       title: 'Flutter UI Example',
       theme: ThemeData(
         primarySwatch: Colors.purple,
       ),
       home: const MyHomePage(),
)

2. Scaffold:
   - Properties:
     - appBar: An instance of AppBar representing the app bar at the top of the screen.
     - body: A widget representing the main content of the screen.
     - floatingActionButton: An instance of FloatingActionButton representing the floating action button.
   - Syntax:
     dart
     Scaffold(
       appBar: AppBar(
         title: const Text('Experiment-2 by Ajinkya'),
       ),
       body: SingleChildScrollView(
         // Body content here
       ),
       floatingActionButton: FloatingActionButton(
         onPressed: () {

```
        // onPressed logic here
      },
      tooltip: 'Add Task',
      child: const Icon(Icons.add),
    ),
  )
```

3. AppBar:
   - Properties:
     - title: A widget representing the title displayed in the app bar.
     - leading: A widget displayed before the title in the app bar (e.g., a back button).
     - actions: A list of widgets displayed after the title in the app bar (e.g., action buttons).
   - Syntax:
     dart
     AppBar(
       title: const Text('Experiment-2 by Ajinkya'),
       // Additional properties like leading and actions can be added here
)

4. SingleChildScrollView:
   - Properties:
     - child: A widget that needs to be scrolled if it's larger than the viewport.
   - Syntax:
     dart
     SingleChildScrollView(
       // Child widget here
)

5. Column:
   - Properties:
     - children: A list of widgets displayed in a vertical arrangement.
     - crossAxisAlignment: Defines how children are aligned in the cross axis (e.g., CrossAxisAlignment.start).
     - mainAxisAlignment: Defines how children are aligned in the main axis (e.g., MainAxisAlignment.center).
   - Syntax:
     dart
     Column(
       children: <Widget>[

```
      // List of children widgets here
    ],
    // Additional properties like crossAxisAlignment and mainAxisAlignment can be added here
  )
```

(Continued in the next message)

6. Text:
- Properties:
  - data: A string representing the text content.
  - style: An instance of TextStyle defining the text's appearance (e.g., font size, color).
- Syntax:

```
dart
Text(
  'Today\'s Tasks',
  style: TextStyle(
    fontSize: 24,
    fontWeight: FontWeight.bold,
  ),
)
```

7. SizedBox:
- Properties:
  - width: A double representing the width of the box.
  - height: A double representing the height of the box.
- Syntax:

```
dart
SizedBox(
  height: 20,
)
```

8. TextFormField:
- Properties:
  - decoration: An instance of InputDecoration defining the appearance of the text input field.
  - onChanged: A callback function triggered when the text input changes.
  - onFieldSubmitted: A callback function triggered when the user submits the text input field.

- Syntax:
  dart
  TextFormField(
    decoration: const InputDecoration(
      labelText: 'Enter task name',
      border: OutlineInputBorder(),
    ),
    onChanged: (value) {
      // onChanged logic here
    },
    onFieldSubmitted: (value) {
      // onFieldSubmitted logic here
    },
)

9. ElevatedButton:
   - Properties:
     - onPressed: A callback function triggered when the button is pressed.
     - child: A widget representing the button's content (e.g., text, icon).
     - style: An instance of ButtonStyle defining the button's appearance.
   - Syntax:
     dart
     ElevatedButton(
       onPressed: () {
         // onPressed logic here
       },
       child: const Text('Add Task'),
)

(Continued in the next message)

10. ListTile:
    - Properties:
      - leading: A widget displayed before the title in the list tile (e.g., an icon).
      - title: A widget representing the title of the list tile.
      - subtitle: A widget representing the subtitle of the list tile.
      - trailing: A widget displayed after the title in the list tile (e.g., an icon or a switch).
    - Syntax:
      dart
      ListTile(

```
    leading: Icon(Icons.notifications),
    title: Text('Notifications'),
    trailing: Switch(
      value: true,
      onChanged: (bool value) {
        // onChanged logic here
      },
    ),
  )
```

11. Switch:
- Properties:
  - value: A boolean representing the current state of the switch (on/off).
  - onChanged: A callback function triggered when the switch's state changes.
- Syntax:
  dart
```
  Switch(
    value: true,
    onChanged: (bool value) {
      // onChanged logic here
    },
  )
```

12. AlertDialog:
- Properties:
  - title: A widget representing the title of the alert dialog.
  - content: A widget representing the content of the alert dialog.
  - actions: A list of widgets representing the action buttons in the alert dialog.
- Syntax:
  dart
```
  AlertDialog(
    title: Text('Theme Settings'),
    content: Text('Navigate to theme settings page.'),
    actions: <Widget>[
      // List of action buttons here
    ],
```

)

13. SnackBar:
- Properties:
  - content: A widget representing the content of the snackbar.
  - action: An instance of SnackBarAction representing the action button in the snackbar.
- Syntax:
  dart
  SnackBar(
    content: Text('Navigate to about page.'),
    action: SnackBarAction(
      label: 'Undo',
      onPressed: () {
        // onPressed logic here
      },
    ),
)

14. FloatingActionButton:
- Properties:
  - onPressed: A callback function triggered when the button is pressed.
  - tooltip: A string representing the tooltip message displayed when the button is long-pressed.
  - child: A widget representing the content of the floating action button (e.g., an icon).
- Syntax:
  dart
  FloatingActionButton(
    onPressed: () {
      // onPressed logic here
    },
    tooltip: 'Add Task',
    child: Icon(Icons.add),
)

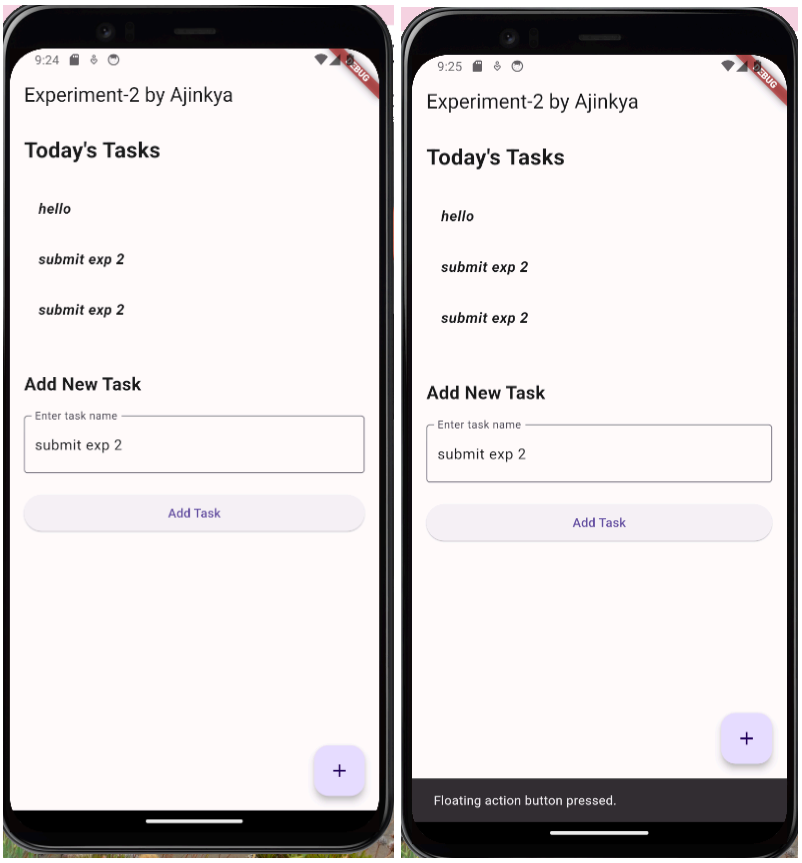
```dart
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}
```

```
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter UI Example',
      theme: ThemeData(
        primarySwatch: Colors.purple,
        fontFamily: 'Roboto', // Set the default font family
      ),
      home: const MyHomePage(),
    );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({Key? key}) : super(key: key);

  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  List<String> tasks = [];
  late String newTask; // Variable to hold the value entered in the
TextFormField

  @override
  void initState() {
    super.initState();
    newTask = ''; // Initialize newTask to empty string
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Experiment-2 by Ajinkya'),
      ),
      body: SingleChildScrollView(
        padding: const EdgeInsets.all(16.0),
```

```
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: <Widget>[
            const Text(
              'Today\'s Tasks',
              style: TextStyle(
                fontSize: 24,
                fontWeight: FontWeight.bold,
              ),
            ),
            const SizedBox(height: 20),
            for (var task in tasks) TaskTile(title: task),
            const SizedBox(height: 40),
            const Text(
              'Add New Task',
              style: TextStyle(
                fontSize: 20,
                fontWeight: FontWeight.bold,
              ),
            ),
            const SizedBox(height: 20),
            TextFormField(
              decoration: const InputDecoration(
                labelText: 'Enter task name',
                border: OutlineInputBorder(),
              ),
              onChanged: (value) {
                setState(() {
                  newTask = value;
                });
              },
              onFieldSubmitted: (value) {
                setState(() {
                  tasks.add(value);
                });
                // Clear the TextFormField after adding task
                newTask = '';
              },
            ),
            const SizedBox(height: 20),
            ElevatedButton(
              onPressed: () {
                setState(() {
```

```
                if (newTask.isNotEmpty) {
                  tasks.add(newTask);
                  // Clear the TextFormField after adding task
                  newTask = '';
                }
              });
            },
            child: const Text('Add Task'),
          ),
        ],
      ),
    ),
    floatingActionButton: FloatingActionButton(
      onPressed: () {
        // Add onPressed logic for floating action button
        // For demo, we'll just show a snackbar
        ScaffoldMessenger.of(context).showSnackBar(
          const SnackBar(
            content: Text('Floating action button pressed.'),
          ),
        );
      },
      tooltip: 'Add Task',
      child: const Icon(Icons.add),
    ),
  );
 }
}

class TaskTile extends StatelessWidget {
 final String title;

 const TaskTile({Key? key, required this.title}) : super(key: key);

 @override
 Widget build(BuildContext context) {
   return ListTile(
     title: Text(
       title,
       style: const TextStyle(
         // Set a custom font for the task title
         fontFamily: 'Roboto',
         fontStyle: FontStyle.italic,
```

```
            fontWeight: FontWeight.bold,
          ),
        ),

      );
    }
}
```



Conclusion: We have Studied the Basic and common widgets of flutter and implemented them in an application. The syntax is a bit unfamiliar but it is not very difficult to learn