EXPERIMENT 2

Aim:To design Flutter UI by including common widgets.

Theory:

Flutter, Google's UI toolkit for building natively compiled applications across various platforms, owes much of its flexibility and efficiency to its extensive collection of widgets. In the Flutter framework, a widget is a fundamental building block, representing everything from structural elements like containers to interactive components like buttons. Widgets are designed to be composable, enabling developers to create complex and dynamic user interfaces by combining smaller, reusable pieces. This widget-centric approach not only facilitates code organization but also ensures a consistent and visually appealing user experience across different platforms.

1. Basic Structural Widgets:
   - Container: A versatile box model for layout and styling.
   - Row and Column: Widgets for arranging child widgets horizontally and vertically.
   - Stack and Positioned: Used for overlaying widgets, creating complex UI designs.

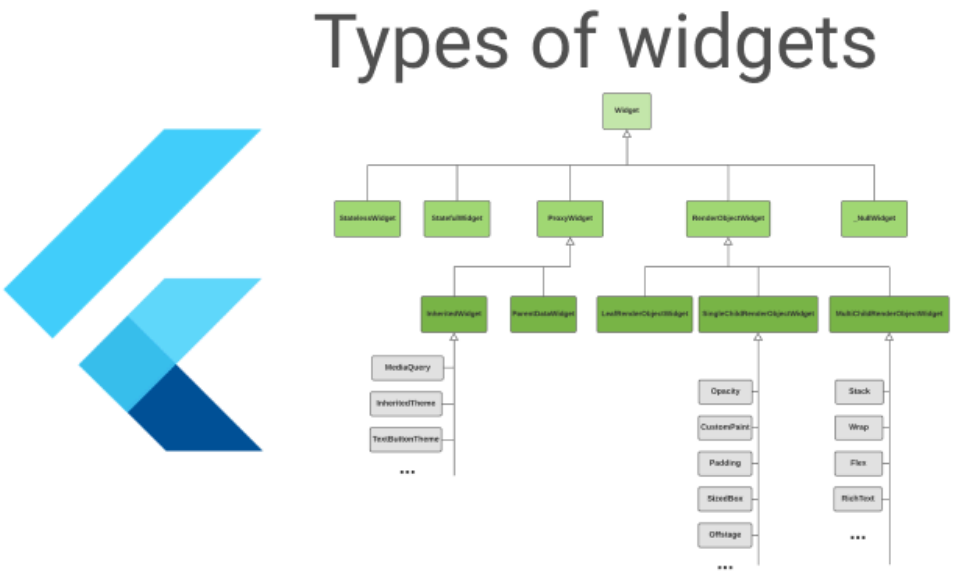
2. Text and Typography Widgets:
   - Text: For displaying simple text with styling options.
   - RichText: Allows for more complex text styling and formatting.
   - DefaultTextStyle: Sets the default text styling for a subtree of widgets.

3. Interactive Widgets:
   - GestureDetector: Detects gestures like taps and swipes.
   - FlatButton, RaisedButton, and IconButton: Interactive buttons with different styles.
   - TextField: For accepting user input through text.

4. List and Grid Widgets:
   - ListView: A scrollable list of widgets.
   - GridView: A scrollable grid of widgets.
   - ExpansionTile: Creates an expandable/collapsible list item.

5. Material Design Widgets:
   - AppBar and BottomNavigationBar: Common components for Material Design.
   - Card: A material design card with a shadow.
   - Drawer: A slide-in menu panel typically used for navigation.

6. Custom and Modifying Widgets:
   - CustomPaint: Custom drawing on the screen.
   - Transform: Applying transformations like rotation and scaling to child widgets.
   - ClipRRect: Clipping child widgets with rounded corners.



Types of Widget We can split the Flutter widget into two categories:

- Visible (Output and Input)
- Invisible (Layout and Control)

Visible Widgets:

1. Output Widget - `Text`:
   - The `Text` widget is used to display text on the screen.

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

void main() {
  runApp(
    MaterialApp(
      home: Scaffold(
        body: Center(
          child: Text(
            'Hello, Flutter!',
            style: TextStyle(fontSize: 24),
          ),
        ),
      ),
    ),
  );
}
```

2. Input Widget - `TextField`:
   - The `TextField` widget allows users to input text.

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

void main() {
  runApp(
    MaterialApp(
      home: Scaffold(
        body: Center(
```

```
          child: TextField(
            decoration: InputDecoration(
              hintText: 'Enter your text here',
            ),
          ),
        ),
      ),
    ),
  );
}
```

Invisible Widgets:

1. Layout Widget - `Column`:
   - The `Column` widget arranges its children vertically.

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

void main() {
  runApp(
    MaterialApp(
      home: Scaffold(
        body: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text('Item 1'),
            Text('Item 2'),
            Text('Item 3'),
          ],
        ),
      ),
    ),
  );
}
```

2. Control Widget - `GestureDetector`:
   - The `GestureDetector` widget allows you to handle gestures on its child.

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

void main() {
  runApp(
    MaterialApp(
      home: Scaffold(
        body: GestureDetector(
          onTap: () {
            print('Container tapped!');
          },
          child: Container(
            width: 200,
            height: 200,
            color: Colors.blue,
            child: Center(
              child: Text('Tap me!'),
            ),
          ),
        ),
      ),
    ),
  );
}
```

Code:
(main.dart)

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

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

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Library App',
      theme: ThemeData(
        primarySwatch: Colors.pink,
        hintColor: Colors.black,
      ),
      home: MyLibraryPage(),
    );
  }
}

class MyLibraryPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Library App'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              'Welcome to the Library App!',
              style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
            ),
            SizedBox(height: 20),
            Text(
              'List of Books:',
              style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold),
            ),
```

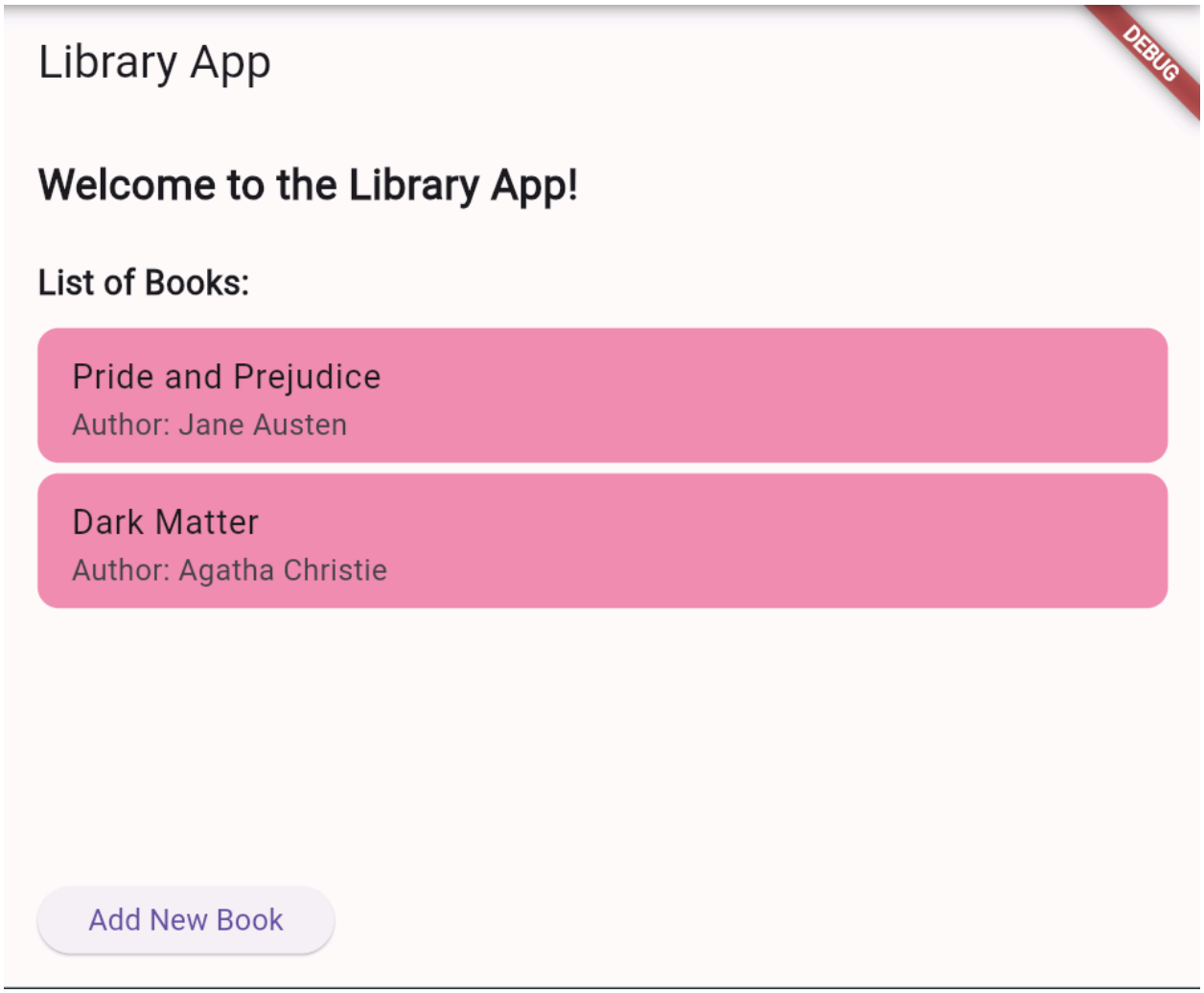
```
SizedBox(height: 10),
Expanded(
  child: ListView(
    children: [
      ListTile(
        title: Text('Pride and Prejudice'),
        subtitle: Text('Author: Jane Austen'),
        tileColor: Colors.pink[200],
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(10),
        ),
        onTap: () {
          // Handle book tap
        },
      ),
      SizedBox(height: 5),
      ListTile(
        title: Text('Dark Matter'),
        subtitle: Text('Author: Agatha Christie'),
        tileColor: Colors.pink[200],
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(10),
        ),
        onTap: () {

        },
      ),
    ],
  ),
),
SizedBox(height: 20),
ElevatedButton(
  onPressed: () {

  },
```

```
              child: Text('Add New Book'),
            ),
          ],
        ),
      ),
    );
  }
}
```

Output:



Step 1: Create a new Flutter project.

flutter create library_app
cd library_app

Step 2: Open the project in your code editor.

code .

Step 3: Replace the contents of `lib/main.dart` with the following code:

```
// lib/main.dart
import 'package:flutter/material.dart';
import 'library_page.dart'; // Import the library page file

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Library App',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        hintColor: Colors.black,
      ),
      home: MyLibraryPage(), // Use MyLibraryPage as the home page
    );
  }
}
```

Step 4: Create a new Dart file for the library page (`library_page.dart`).

touch lib/library_page.dart

Step 5: Import Flutter material package.

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

Step 6: Create a class named `MyLibraryPage` that extends `StatelessWidget`.

```
class MyLibraryPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      // Scaffold widget for the overall page structure
      // ...
    );
  }
}
```

Step 7: Add an `AppBar` to the `Scaffold` with the title "Library App".

```
appBar: AppBar(
  title: Text('Library App'),
),
```

Step 8: Add padding to the body of the `Scaffold`.

```
body: Padding(
```

```
  padding: const EdgeInsets.all(16.0),
  child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      // Content of the library page will go here
      // ...
    ],
  ),
),
```

Step 9: Display a welcome message with bold styling and add spacing using `SizedBox`

```
Text(
  'Welcome to the Library App!',
  style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
),
SizedBox(height: 20),
```

Step 10: Create a section header for the list of books and implement a scrollable list of books using `ListView`

```
Text(
  'List of Books:',
  style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold),
),
SizedBox(height: 10),
Expanded(
  child: ListView(
    children: [
      // List items for books will go here
      // ...
```

```
    ],
  ),
),
```

Step 11: Create a `ListTile` for each book in the list and implement an `ElevatedButton` for adding a new book.

```
ListTile(
  title: Text('Book 1'),
  subtitle: Text('Author: Author Name'),
  tileColor: Colors.lightBlue,
  shape: RoundedRectangleBorder(
    borderRadius: BorderRadius.circular(10),
  ),
  onTap: () {
    // Handle book tap
  },
),
SizedBox(height: 5),
ListTile(
  title: Text('Book 2'),
  subtitle: Text('Author: Another Author'),
  tileColor: Colors.lightBlue,
  shape: RoundedRectangleBorder(
    borderRadius: BorderRadius.circular(10),
  ),
  onTap: () {

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

```
ElevatedButton(
  onPressed: () {
    // Handle button press to add a new book
  },
  child: Text('Add New Book'),
),
```

Step 12: Run the app.

flutter run

Conclusion:
In conclusion, we successfully implemented a Flutter library page utilizing various Flutter widgets, showcasing the flexibility and ease of building a dynamic user interface for a library app.