Experiment No. 2

AIM : To design Flutter UI by including common widgets

Theory:
In Flutter, designing UIs involves combining various widgets to build interactive and visually appealing applications. Here's a more detailed overview of key concepts:

1. **Widgets in Flutter**: Everything in Flutter is a widget. Widgets are the building blocks of the UI. There are two main types:
    - Stateless Widgets: These are immutable and don't change over time. They are responsible for displaying UI based on fixed data or input.
    - Stateful Widgets: These can change their state over time. They are dynamic and are used when the UI needs to update in response to user interaction or other factors.

2. **Layout Widgets**: The layout of your UI is primarily constructed using widgets like:
    - **Container**: A versatile widget used to hold other widgets and apply styling such as padding, margin, colors, and shapes.
    - **Column**: A widget that arranges its children vertically. It's useful for stacking widgets in a vertical list.
    - **Row**: A widget that arranges its children horizontally. It's useful for placing widgets side by side.
    - **Expanded**: A widget that can be used inside a Column, Row, or Flex to make child widgets flexible and fill available space.

3. **Text and Icons**

- **Text**
  Text: The Text widget is used to display static or dynamic text. It can be styled with custom fonts, sizes, colors, and more.
- **Icon**
  Icon: Flutter provides a large set of material design icons, and the Icon widget lets you display them in various sizes and colors.

4. **Buttons and User Interactions**

- **Button Widgets**
  Flutter provides multiple button widgets like ElevatedButton, TextButton, and IconButton to handle user interaction. These widgets can trigger actions when tapped.

- **TextField**
  TextField: Used for user input. You can configure it to accept different types of text, such as email or password.
- **Checkbox, Radio, and Switch**
  Checkbox, Radio, and Switch: Used for boolean selections, allowing users to choose options in forms or settings.

5. **Navigation**
- **Navigator**
  Flutter's Navigator widget is responsible for managing routes or screens. You use Navigator.push to navigate to a new screen, and Navigator.pop to return to the previous one.
- **Routes**
  Routes define the pages of an app, and you can pass data between them using arguments.

6. **Displaying Lists and Grids**
- **ListView**
  ListView: The ListView widget is used to display a list of items that can scroll. It's perfect for long lists that need to be dynamically generated.
- **GridView**
  GridView: This widget allows you to display items in a grid format, with configurable row and column layouts.

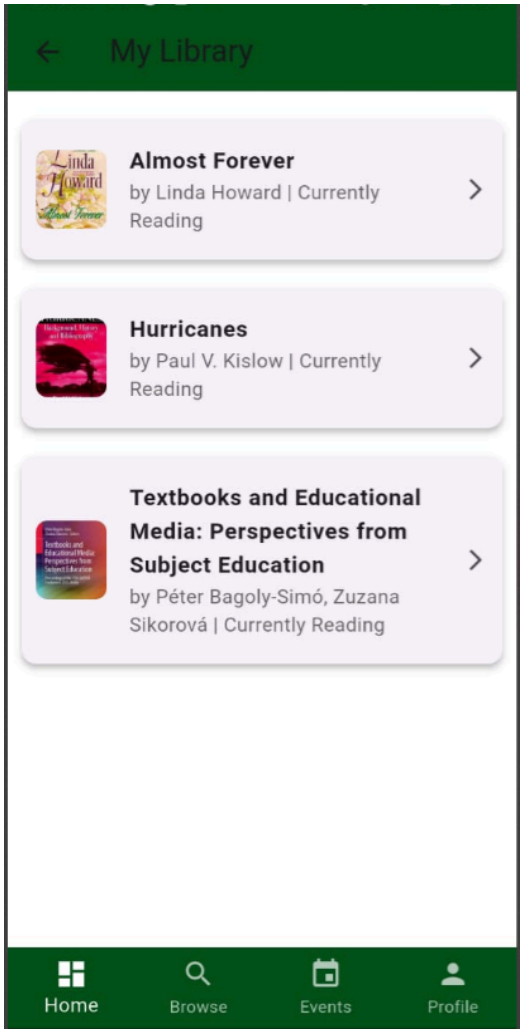
7. **State Management**
- **setState and Advanced Solutions**
  Flutter provides a variety of ways to manage state. The simplest approach is using setState() to update the UI. For more complex apps, you can use state management solutions like Provider, Riverpod, or Bloc to separate business logic from UI code.
- **Why State Management Matters**
  Proper state management ensures your UI stays in sync with the underlying data, especially in interactive or dynamic applications.

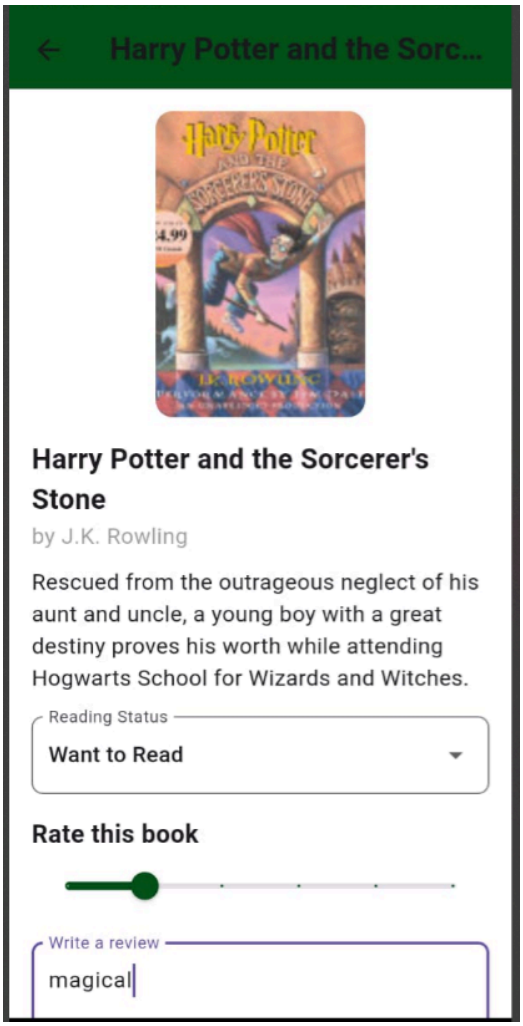
8. **Theming and Styling**
- **ThemeData**
  Flutter allows you to define a global Theme for your app using ThemeData, which ensures consistent styling across the entire app. You can customize colors, typography, and button styles.

9. **Animations and Transitions**

- **AnimatedContainer**
  Flutter provides powerful animation support to create smooth and visually appealing transitions between UI states.
  AnimatedContainer: A widget that animates changes in properties like width, height, or color over a given duration.
- **Custom Animations**
  You can also create custom animations using AnimationController and Tween.

Screenshots:





Code Snippets:

AppBar

```
appBar: AppBar(
  title: Text(
    widget.bookTitle,
    style: const TextStyle(fontSize: 22, fontWeight: FontWeight.bold),
  ),
  backgroundColor: const Color(0xFF04511A),
),
```

SingleChildScrollView and Column

```
body: SingleChildScrollView(
  padding: const EdgeInsets.all(16),
  child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      // children widgets here...
    ],
  ),
),
```

DropdownButtonFormField

```
DropdownButtonFormField<String>(
  value: selectedStatus,
  decoration: InputDecoration(
    labelText: "Reading Status",
    border: OutlineInputBorder(borderRadius: BorderRadius.circular(8)),
  ),
  items: ["Reading", "Read", "Want to Read"].map((status) {
    return DropdownMenuItem(value: status, child: Text(status));
  }).toList(),
  onChanged: updateReadingStatus,
),
```

Slider

```
Slider(
  value: _rating,
  onChanged: (newRating) {
    setState(() {
      _rating = newRating;
    });
  },
  min: 0,
  max: 5,
  divisions: 5,
  label: _rating.toString(),
  activeColor: const Color(0xFF04511A),
),
```

TextField

```
TextField(
  controller: _reviewController,
  decoration: InputDecoration(
    labelText: "Write a review",
    border: OutlineInputBorder(borderRadius: BorderRadius.circular(8)),
  ),
  maxLines: 3,
),
```

ElevatedButton

```
ElevatedButton(
  onPressed: submitReview,
  style: ElevatedButton.styleFrom(
    backgroundColor: const Color(0xFF04511A),
    padding: const EdgeInsets.symmetric(horizontal: 24, vertical: 12),
    shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(8)),
  ),
  child: const Text("Submit Review", style: TextStyle(fontSize: 16, color: Colors.white)),
),
```

Conclusion:

In conclusion, Flutter offers a comprehensive set of widgets that are crucial for building intuitive and responsive user interfaces. Common widgets such as TextField, IconButton, ElevatedButton, Card, and Checkbox play a vital role in shaping the user interface of mobile applications. These widgets enable users to input data, perform actions, and view content in a structured way, all while improving the overall user experience. By understanding and effectively utilizing these widgets, developers can create mobile applications that are clean, efficient, and user-friendly.