Assignment

MODULE: 3

Introduction to Flutter Widgets and UI Components

Q.1 Explain the difference between Stateless and Stateful widgets with examples.

- **Ans .**

Stateless vs Stateful Widgets in Flutter :

Stateless Widget:

- Does not store state/data that changes over time.

- UI is **static** after it's built.

Example: `Text`, `Icon`, `RaisedButton` (when no change in appearance/data is needed).

```
class MyStatelessWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Text('Hello');
  }
}
```

- **Stateful Widget:**

- Stores state/data that can change over time.

- UI can **rebuild** when state changes.

Example: `Checkbox`, `TextField`, `Slider`.

```
class MyStatefulWidget extends StatefulWidget {
  @override
  _MyStatefulWidgetState createState() =>
_MyStatefulWidgetState();
}

class _MyStatefulWidgetState extends
State<MyStatefulWidget> {
  int counter = 0;

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        Text(’$counter’),
        ElevatedButton(
          onPressed: () => setState(() => counter++),
          child: Text(’Increment’),
        ),
      ],
    );
  }
}
```

In short:
Stateless = UI doesn't change,
Stateful = UI can update with data/state changes.

Q.2 Describe the widget lifecycle and how state is managed in Stateful widgets.

- ➢ Ans.

Stateful Widget Lifecycle :

1. **`createState()`** – Called once when the widget is created.

2. **`initState()`** – Called once when the state is initialized; use for setup.

3. **`build()`** – Called every time the UI needs to update.

4. **`setState()`** – Triggers `build()` to redraw UI with new state.

`5.didUpdateWidget()` – Called if the widget is rebuilt with a new config.

`6.dispose()` – Called when the widget is removed; clean up resources here.

State Management:

- The `State` object holds the **mutable state**.

- Use `setState()` to **update state and rebuild the UI**.

Q.3 List and describe five common Flutter layout widgets (e.g., Container, Column, Row).

➢ **Ans.**

5 Common Flutter Layout Widgets (Short Description):

1 - Container

- A box model widget used for **styling**, **padding**, **margin**, **alignment**, and **decoration**.

- Example: `Container(color: Colors.red, padding: EdgeInsets.all(10))`

2- **Column**

- Arranges children **vertically**.

- Useful for stacking widgets top to bottom.

- Example: `Column(children: [Text('A'), Text('B')])`

3 - Row

- Arranges children **horizontally**.

- Useful for left-to-right layouts.

- Example: `Row(children: [Icon(Icons.star), Text('Rate')])`

4-Expanded

- Expands a child of Row, Column, or Flex to **fill available space**.

- Example: `Expanded(child: Text('Stretch me'))`

5- **Stack**

- **Overlays widgets** on top of each other.

- Great for positioning widgets **freely**.

- Example: `Stack(children: [Image.asset('bg.png'), Positioned(...)])`