Name: Aakanksha Suvarna Class: D15B Roll_No: 65

Experiment 2

AIM: To design Flutter UI by including common widgets.

THEORY:

What is a Widget?
In Flutter, everything is a widget. Widgets are the basic building blocks of a Flutter application. They are elements of the user interface, such as buttons, text, images, layouts, and more. Widgets describe how elements should be painted on the screen and how they should respond to user interactions.

Types of Widgets:
StatelessWidget: A stateless widget is immutable and does not have any internal state. It defines its configuration using constructor arguments and does not change over time. Examples include Text, Icon, and Container.

StatefulWidget: A stateful widget can maintain internal state that can change over time. It consists of two classes: the widget itself and a corresponding state class. Examples include TextField, Checkbox, and ListView.

InheritedWidget: An inherited widget is used to propagate information down the widget tree. It allows widgets to access shared data without needing to pass it explicitly as constructor arguments. Examples include Theme, MediaQuery, and Provider (from provider package).

Layout Widgets: Layout widgets are used to arrange other widgets on the screen. They control the size and position of their children. Examples include Row, Column, Stack, and GridView.

Material Components: Flutter provides a set of widgets that implement the Material Design guidelines. These widgets offer consistent and visually appealing UI elements for Android apps. Examples include AppBar, Card, Button, and TextField.

Cupertino Components: Flutter also provides widgets that implement the iOS design language, known as Cupertino. These widgets offer iOS-style UI components for iOS apps. Examples include CupertinoNavigationBar, CupertinoButton, and CupertinoTextField.

Widget Lifecycle:
Creation: Widgets are instantiated by calling their constructors.

Initialization: Widgets initialize their state and configure themselves based on constructor arguments.

Building: Widgets build their subtree by creating and configuring their children.

Updating: Widgets can be rebuilt in response to changes in application state or user interactions.

Disposal: Stateful widgets dispose of their state when they are no longer needed.

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

void main() {
  runApp(const WeddingPlannerApp());
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Wedding Planner Dashboard',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: const Dashboard(),
    );
  }
}

class Dashboard extends StatelessWidget {
  const Dashboard({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Wedding Planner Dashboard'),
        backgroundColor: Colors.purple[100],
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
```

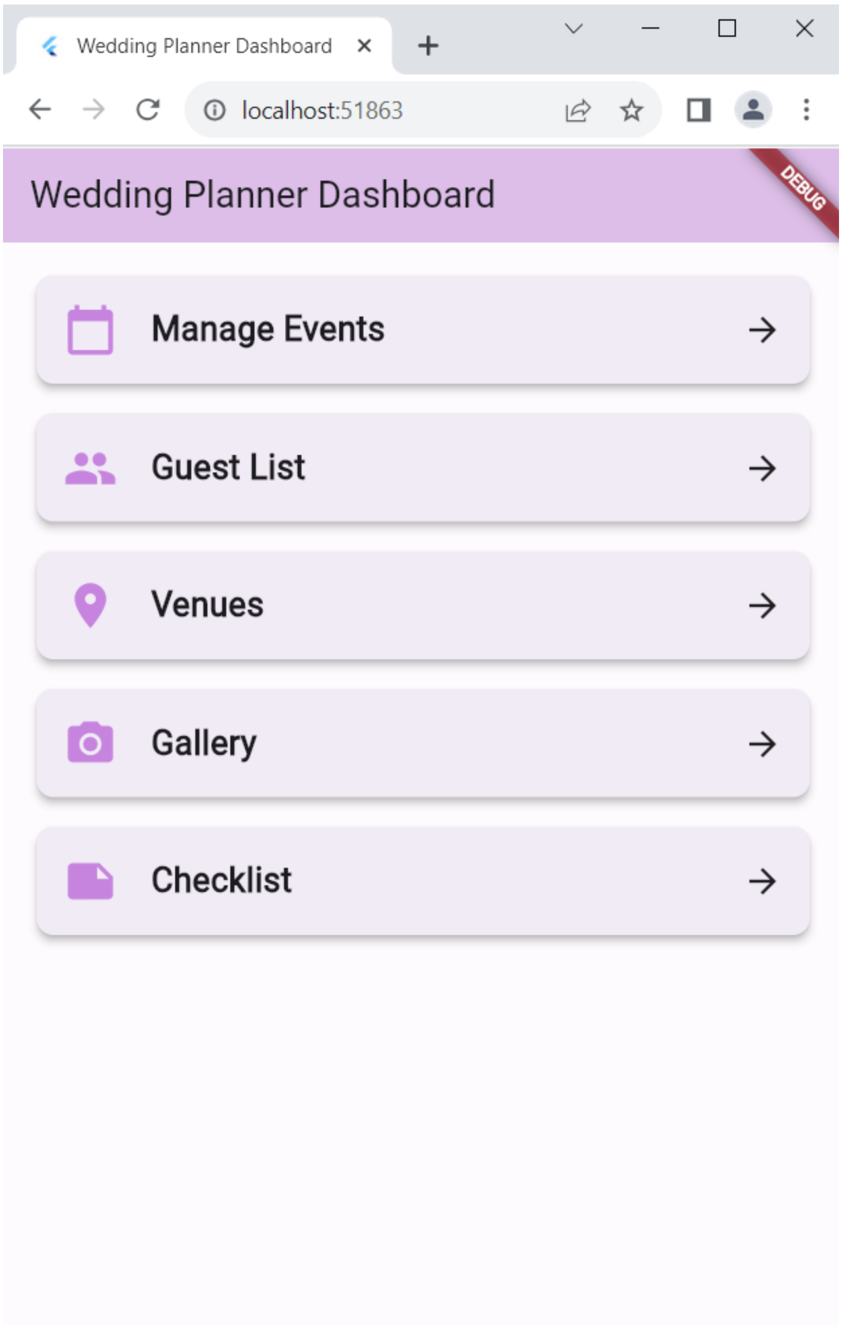
Name: Aakanksha Suvarna Class: D15B Roll_No: 65

```
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: <Widget>[
            buildCard(
              context,
              title: 'Manage Events',
              icon: Icons.calendar_today,
              onTap: () {
                // Navigate to event management page
              },
            ),
            const SizedBox(height: 10),
            buildCard(
              context,
              title: 'Guest List',
              icon: Icons.people,
              onTap: () {
                // Navigate to guest list page
              },
            ),
            const SizedBox(height: 10),
            buildCard(
              context,
              title: 'Venues',
              icon: Icons.location_on,
              onTap: () {
                // Navigate to venue management page
              },
            ),
            const SizedBox(height: 10),
            buildCard(
              context,
              title: 'Gallery',
              icon: Icons.photo_camera,
              onTap: () {
                // Navigate to gallery page
              },
            ),
            const SizedBox(height: 10),
            buildCard(
              context,
              title: 'Checklist',
              icon: Icons.note,
              onTap: () {
                // Navigate to checklist page
              },
            ),
          ],
        ),
      ),
    );
  }

  Widget buildCard(BuildContext context,
      {required String title,
      required IconData icon,
      required VoidCallback onTap}) {
    return InkWell(
      onTap: onTap,
      child: Card(
        elevation: 4,
        shape: RoundedRectangleBorder(borderRadius:
BorderRadius.circular(10)),
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Row(
            children: [
              Icon(icon,
                  size: 32, color: const Color.fromARGB(255, 207, 133,
220)),
              const SizedBox(width: 20),
              Text(
                title,
                style:
                    const TextStyle(fontSize: 20, fontWeight:
FontWeight.bold),
              ),
              const Spacer(),
              const Icon(Icons.arrow_forward),
            ],
          ),
```

Name: Aakanksha Suvarna Class: D15B Roll_No: 65

```
        ),
      ),
    );
  }
}
```


Wedding Planner Dashboard

localhost:51863

Wedding Planner Dashboard

Name: Aakanksha Suvarna Class: D15B Roll_No: 65

1. We create a WeddingPlannerApp class that serves as the entry point for our application.
2. Inside WeddingPlannerApp, we define a Dashboard class which is a stateless widget representing the dashboard screen.
3. The dashboard screen consists of a Scaffold with an AppBar and a Column containing multiple Card widgets.
4. Each Card represents a different feature of the wedding planning system, such as event management, guest list, venues, gallery, and checklist.
5. Within each Card, we use a ListTile to display the feature title and an icon. We also define an onTap callback for each ListTile to navigate to the corresponding feature page (currently empty).