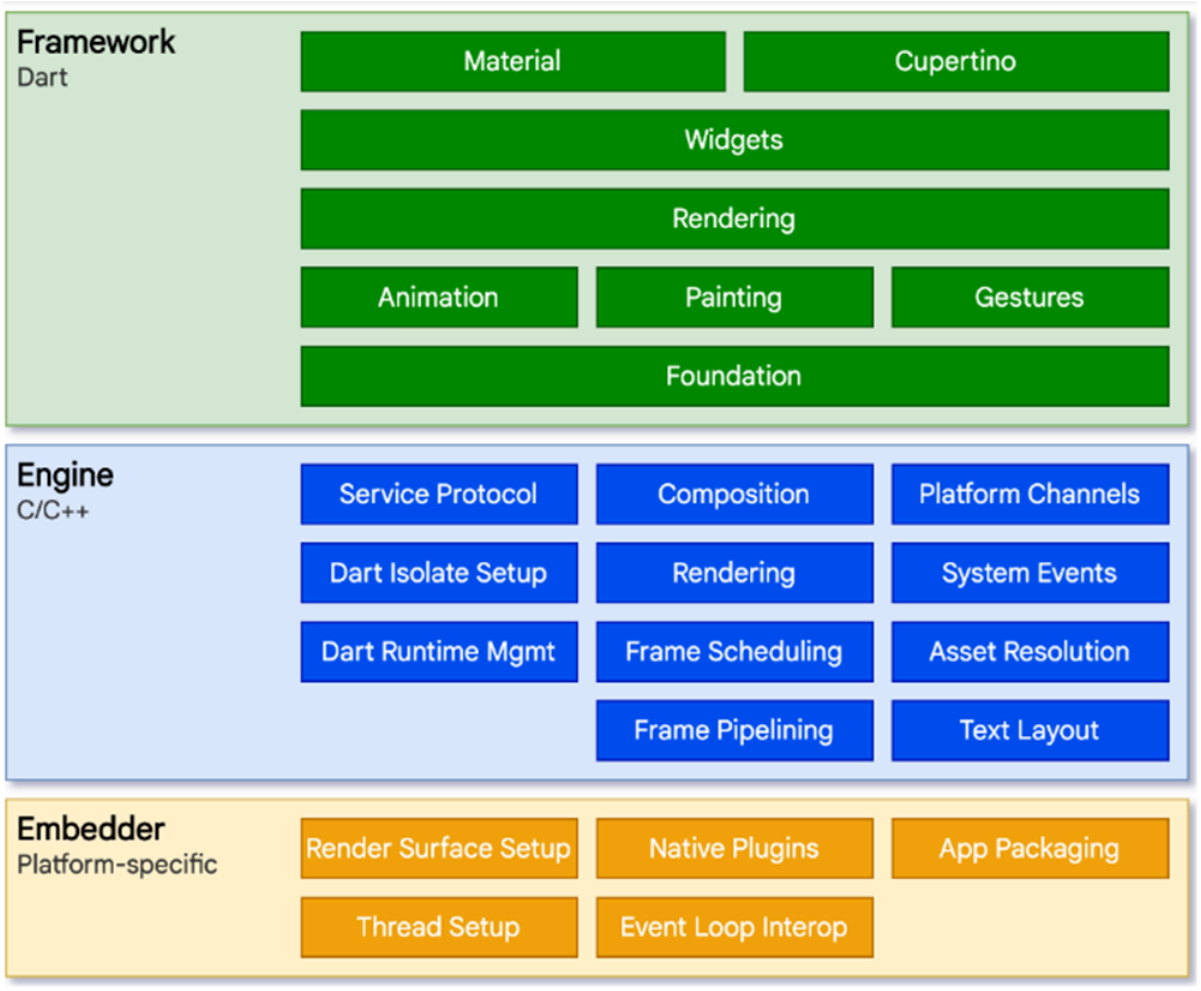
Name of Student: Pushkar Sane		
Roll Number: 45	**Lab Assignment Number: 10**	
Title of Lab Assignment: Flutter program using layout, widget and state management.		
DOP: 26-10-2024	**DOS: 26-10-2024**	
CO Mapped: **CO3, CO4**	**PO Mapped:** **PO2, PO3,PO5, PSO1, PSO2**	**Signature:**

Practical No. 10

Aim: Flutter program using layout, widget and state management.

Theory:

Flutter: Flutter is an open-source UI software development toolkit created by Google for building natively compiled applications for mobile, web, and desktop from a single codebase. Flutter provides a rich set of widgets and tools to create high-performance, expressive, and flexible user interfaces. It also follows a specific architectural pattern for structuring Flutter applications. There are two primary architectural patterns used in Flutter: Single Widget and Reactive Framework and BLoC (Business Logic Component) Architecture.



Layout in Flutter:

1. Widget Hierarchy: In Flutter, everything is a widget. Understanding how to create a hierarchy of widgets is crucial for building user interfaces. Widgets are composed into a tree structure, with a single root widget, which is typically a MaterialApp or CupertinoApp, and many child widgets that make up the user interface. Widgets can have children, and these children can be any type of widget.
2. Box Constraints: Flutter uses box constraints to determine the size of widgets. Widgets can have minimum and maximum dimensions, and their sizes are calculated based on available space and constraints. Widgets can overflow or be clipped if their content exceeds the constraints.
3. Rows and Columns: Flutter provides Row and Column widgets for arranging widgets horizontally and vertically, respectively. These widgets are commonly used to create structured layouts.
4. Containers: The Container widget is a versatile widget for styling and positioning child widgets. You can use it to set padding, margins, alignment, and background color for its children.
5. Expanded Widget: The Expanded widget is often used in combination with Row and Column widgets. It allows a child widget to take up available space within the parent widget, distributing space proportionally among multiple Expanded widgets.
6. Card Widget: The Card widget is a material design card that can be used to group related information. It provides an elegant and consistent way to present content in a structured manner.
7. Padding and Margins: Padding is used to add space inside a widget, while margins add space outside a widget. Both are important for controlling the spacing between widgets in your layout.

Layout Challenges: Designing responsive and adaptive layouts is essential. This involves handling screen sizes, orientations, and ensuring that your app looks and functions well on a variety of devices.

Widgets in Flutter:

1. **Introduction to Widgets:** Widgets are the building blocks of a Flutter app. Every visual element, from text and buttons to layout structures, is a widget. Understanding their role is fundamental.
2. **Stateless Widgets:** Stateless widgets represent components that don't change after they are initially created. They are used for displaying static content.

3. **Stateful Widgets:** Stateful widgets are used when the content needs to change over time, such as user interactions or data updates. They have an associated state object that can change during the widget's lifetime.
4. **Widget Building:** Building custom widgets is essential for creating reusable components and maintaining a clean and organized codebase. You can compose widgets by nesting them inside one another.
5. **Widget Lifecycle:** Widgets go through a life cycle of creation, updating, and potentially disposal. Understanding when these lifecycle methods (e.g., build, initState, dispose) are called is crucial for managing your widgets' behavior.
6. **Widget Libraries:** Flutter provides a rich set of built-in widgets, including those for common UI elements, navigation, and gestures. Additionally, you can use external packages to expand the set of available widgets.
7. **Custom Painting:** For advanced customization, you can use the CustomPaint widget to create custom-drawn widgets. This allows you to paint on the canvas with full control over the rendering process.

State Management in Flutter:

1. **Introduction to State Management:** State management is essential for handling and maintaining the state of your app. It involves keeping track of data, user interactions, and UI updates.
2. **Local State:** You can manage state within a widget using setState. This method is suitable for simple scenarios where a widget's state needs to be updated.
3. **Provider Package:** The provider package is a popular choice for state management in Flutter. It allows you to create and access data models throughout your app, making state management more organized and efficient.

Create a widget Product Box that contains the details of the product, such as image, name, price, and description. In the product Box widget, we use the following child widgets: Container, Row, Column, Expanded, Card, Text, Image, etc.

Code:

main.dart

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

void main() {
    runApp(
        ChangeNotifierProvider(
                create: (context) => ProductList(),
                child: MyApp(),
            ),
    );
}

class Product {
    final String imageUrl;
    final String name;
    final double price;
    final String description;

    Product({
            required this.imageUrl,
            required this.name,
            required this.price,
            required this.description,
    });
}

class ProductList with ChangeNotifier {
    final List<Product> _products = [
            Product(
```

```
                                                                    imageUrl:
'https://images.unsplash.com/photo-1505740420928-5e560c06d30e?auto=format&fit=crop&
q=80&w=2070&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx
8fA%3D%3D',
            name: 'Boat Headphone',
            price: 19.99,
                description: 'Wireless audio with Boat: clear, stylish, and convenient
headphones.',
        ),
        Product(
                                                                    imageUrl:
'https://images.unsplash.com/photo-1523275335684-37898b6baf30?auto=format&fit=crop&q
=80&w=1999&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8f
A%3D%3D',
            name: 'Apple Smart Watch',
            price: 29.99,
            description: 'Wristwear for smart living: time, apps, fitness, and notifications',
        ),
  ];

  List<Product> get products => _products;
  void addProduct(Product product) {
        _products.add(product);
        notifyListeners();
  }
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
      return MaterialApp(
          home: Scaffold(
          appBar: AppBar(
              title: Text('Product Catalog'),
          ),
```

```
            body: ProductListWidget(),
          ),
        );
    }
}

class ProductListWidget extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
            final productProvider = Provider.of<ProductList>(context);
            return ListView.builder(
                itemCount: productProvider.products.length,
                itemBuilder: (context, index) {
                return ProductBox(product: productProvider.products[index]);
                },
            );
    }
}

class ProductBox extends StatelessWidget {
    final Product product;
    ProductBox({required this.product});

    @override
    Widget build(BuildContext context) {
            return Card(
                margin: EdgeInsets.all(10.0),
                child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                    Image.network(product.imageUrl),
                    Padding(
                    padding: const EdgeInsets.all(8.0),
                    child: Text(
```

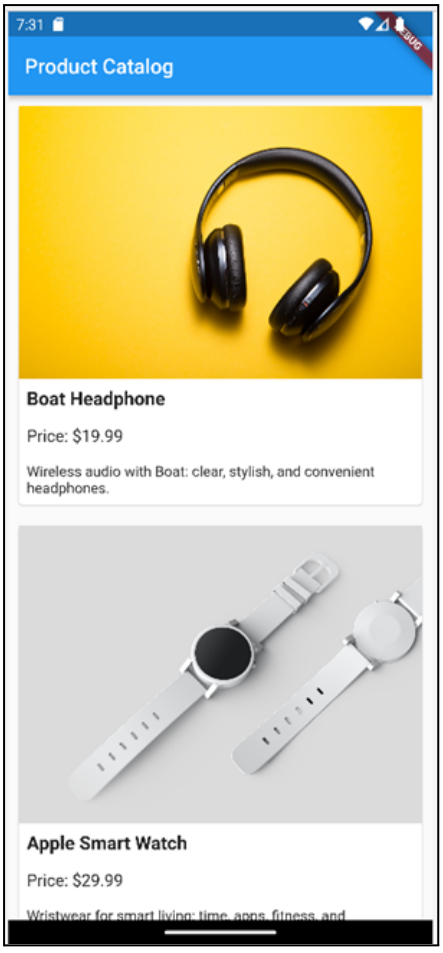
```
                    product.name,
                    style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
                ),
                ),
                Padding(
                padding: const EdgeInsets.all(8.0),
                child: Text(
                    'Price: \$${product.price.toStringAsFixed(2)}',
                    style: TextStyle(fontSize: 16),
                ),
                ),
                Padding(
                padding: const EdgeInsets.all(8.0),
                child: Text(
                    product.description,
                    style: TextStyle(fontSize: 14),
                ),
                ),
            ],
            ),
        );
    }
}
```

pubspec.yml

```
name: p_10
description: A new Flutter project.
publish_to: 'none'
version: 1.0.0+1

environment:
  sdk: '>=3.1.2 <4.0.0'

dependencies:
```

```
  flutter:
    sdk: flutter
  provider: ^5.0.0
  cupertino_icons: ^1.0.2

dev_dependencies:
  flutter_test:
            sdk: flutter
  flutter_lints: ^2.0.0

flutter:
  uses-material-design: true
```

Output:



Conclusion: I have successfully implemented a flutter application in android.