Lab Session 1:

a) Install flutter and Dart sdk

To install Flutter and the Dart SDK, you can follow these steps:

a) **Download Flutter:** Visit the Flutter website's Get Started page and download the Flutter SDK for your operating system (Windows, macOS, or Linux).

b) **Extract the Flutter SDK:** After downloading, extract the contents of the compressed file to a location on your computer where you want to store the Flutter SDK. For example, you can extract it to C:\flutter on Windows, /Users/<your-username>/flutter on macOS, or ~/flutter on Linux.

c) **Add Flutter to your PATH:** Update your system's PATH variable to include the Flutter bin directory. This step allows you to execute Flutter commands from any directory in your terminal or command prompt. The precise steps for updating the PATH vary depending on your operating system.

Windows:

From the Start search bar, type 'env' and select 'Edit the system environment variables'. Click on 'Environment Variables'.
Under 'System Variables', find the 'Path' variable, select it, and click 'Edit'.
Click 'New' and add the path to the bin directory inside the Flutter directory (e.g., C:\flutter\bin).
Click 'OK' on all open dialogs to save your changes.

Verify the Flutter installation: Open a new terminal window, and run the following command to verify that Flutter is properly installed:
flutter --version
This command should display the Flutter version and other relevant information if the installation was successful.

Install Flutter dependencies: Depending on your development environment, you may need to install additional dependencies, such as Android Studio to fully set up your Flutter development environment.

Download Dart SDK (if not bundled with Flutter): Flutter comes with the Dart SDK bundled, so if you've installed Flutter, you should have the Dart SDK as well. However, if you need to install Dart separately; you can download it from the Dart "SDK archive".

b) Write a simple dart program to understand the language basics

```
// Define a main function, which is the entry point of a Dart program.
void main() {
// Variables and data
types int myNumber = 10;
double myDouble = 3.14;
String myString = 'Hello
World'; bool myBool = true;
// Printing variables
 print('My number is: $myNumber');
print('My double is: $myDouble');
print('My string is: $myString');
print('My boolean is: $myBool');
// Basic arithmetic
operations int result =
myNumber + 5;
print('Result of addition: $result');
// Conditional statements
if (myBool)
{
print('myBool is true');
}
else
{
print('myBool is false');
}
// Loops
for (int i = 0; i < 5; i++)
{
 print('Iteration $i');
}
// Lists
List<int> numbers = [1, 2, 3, 4, 5];
print('First element of the list: ${numbers[0]}');
print('Length of the list: ${numbers.length}');
// Maps
Map<String, int> ages = { 'Kiran': 30,'Raj': 25,'Alekya': 35,};
print('Kiran\'s age: ${ages['Kiran']}' );
}
```

OUTPUT:

```
My number is: 10
My double is: 3.14
My string is: Hello World
My boolean is: true
Result of addition: 15
myBool is true
Iteration 0
Iteration 1
Iteration 2
Iteration 3
Iteration 4
```

Lab Session 2:

Explore various flutter widgets

Flutter provides a rich set of widgets to build user interfaces for mobile, web, and desktop applications. These widgets help in creating visually appealing and interactive UIs. Here are some of the commonly used Flutter widgets categorized by their functionalities:

Layout Widgets:

Container: A versatile widget that can contain other widgets and provides options for alignment, padding, margin, and decoration.

Row and Column: Widgets that arrange their children in a horizontal or vertical line respectively.

Stack: Allows widgets to be stacked on top of each other, enabling complex layouts.

List View and Grid View: Widgets for displaying a scrollable list or grid of children, with support for various layouts and scrolling directions.

Scaffold: Implements the basic material design layout structure, providing app bars, drawers, and floatingaction buttons.

Text and Styling Widgets:

Text: Displays a string of text with options for styling such as font size, color, and alignment.

Rich Text: Allows for more complex text styling and formatting, including different styles within the same text span.

Text Style: A class for defining text styles that can be applied to Text widgets.

Input Widgets:

Text Field: A widget for accepting user input as text, with options for customization and validation.

Checkbox and Radio: Widgets for selecting from a list of options, either through checkboxes or radio buttons.

Dropdown Button: Provides a dropdown menu for selecting from a list of options.

Button Widgets:

Elevated Button and Text Button: Widgets for displaying buttons with different styles and customization options.

Icon Button: A button widget that displays an icon and responds to user taps.

Gesture Detector: A versatile widget that detects gestures such as taps, swipes, and drags, allowing for custom interactions

Image and Icon Widgets:

Image: Widget for displaying images from various sources, including assets, network URLs, and memory.

Icon: Displays a Material Design icon.

Navigation Widgets:

Navigator: Manages a stack of route objects and transitions between different screens or pages in the app.

Page Route Builder: A customizable widget for building page transitions and animations.

Animation Widgets:

Animated Container: An animated version of the Container widget, with support for transitioning properties over a specified duration.

Animated Opacity, Animated Positioned, Animated Builder: Widgets for animating opacity, position, and custom properties respectively.

Material Design Widgets:

App Bar: A material design app bar that typically contains a title, leading and trailing widgets, and actions.

Bottom Navigation Bar: Provides a navigation bar at the bottom of the screen for switching betwee different screens or tabs.

Card: Displays content organized in a card-like structure with optional elevation and padding.

Cupertino (iOS-style) Widgets:

Cupertino Navigation Bar: A navigation bar in the iOS style.

Cupertino Button: A button widget with the iOS style.

Cupertino Text Field: A text field widget with the iOS style.

These are just a few examples of the many widgets available in Flutter. Each widget comes with its set of properties and customization options, allowing developers to create highly customizable and responsive user interfaces.

b) User implement different layout structures using Row, Column, and Stack widgets

Row widgets:

```
import 'package:flutter/material.dart';
void main() { runApp(MyApp()); }
class MyApp extends StatelessWidget {
@override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: MyHomePage()
    );
  }
}

class  MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  @override
  Widget build(BuildContext context) {
    return
      Scaffold( appBar
      : AppBar(
        title: Text("Flutter Row Example"),
      ),
      body: Row(
          mainAxisAlignment: MainAxisAlignment.spaceEvenly,
          children:<Widget>[
            Container(
              margin:
              EdgeInsets.all(12.0),
              padding: EdgeInsets.all(8.0),
              decoration:BoxDecoration(
                  borderRadius:BorderRadius.circular(8),
                  color:Colors.green
              ),
              child: Text("React.js",style: TextStyle(color:Colors.yellowAccent,fontSize:25),),
            ),
            Container(
              margin:
              EdgeInsets.all(15.0),
              padding: EdgeInsets.all(8.0),
              decoration:BoxDecoration(
                  borderRadius:BorderRadius.circular(8),
                  color:Colors.green
              ),
              child: Text("Flutter",style: TextStyle(color:Colors.yellowAccent,fontSize:25),),
            ),
            Container(
```

```
            margin:
            EdgeInsets.all(12.0),
            padding: EdgeInsets.all(8.0),
            decoration:BoxDecoration(
               borderRadius:BorderRadius.circular(8),
               color:Colors.green
            ),
            child: Text("MySQL",style: TextStyle(color:Colors.yellowAccent,fontSize:25),),
          )
        ]
    ),
  );
 }
}
```

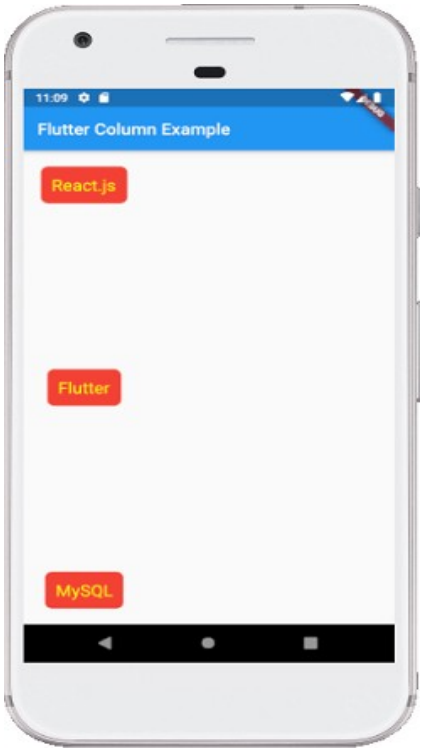
Output:

Column Widget

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

void main() { runApp(MyApp()); }
class MyApp extends StatelessWidget {
@override
  Widget build(BuildContext context) {
    return MaterialApp(
       home: MyHomePage()
    );
  }
}

class MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  @override
  Widget build(BuildContext context) {
    return
      Scaffold( appBar
      : AppBar(
        title: Text("Flutter Column Example"),
      ),
      body: Column(
         mainAxisAlignment: MainAxisAlignment.spaceBetween,
         children:<Widget>[
          Container(
            margin: EdgeInsets.all(20.0),
            padding: EdgeInsets.all(12.0),
            decoration:BoxDecoration(
               borderRadius:BorderRadius.circular(8),
               color:Colors.red
            ),
            child: Text("React.js",style: TextStyle(color:Colors.yellowAccent,fontSize:20),),
          ),
          Container(
            margin: EdgeInsets.all(20.0),
            padding: EdgeInsets.all(12.0),
            decoration:BoxDecoration(
               borderRadius:BorderRadius.circular(8),
               color:Colors.red
            ),
            child: Text("Flutter",style: TextStyle(color:Colors.yellowAccent,fontSize:20),),
          ),
          Container(
            margin: EdgeInsets.all(20.0),
            padding: EdgeInsets.all(12.0),
```

```
            decoration:BoxDecoration( borderRadius:BorderRadius.
               circular(8), color:Colors.red
            ),
            child: Text("MySQL",style: TextStyle(color:Colors.yellowAccent,fontSize:20),),
          )
        ]
    ),
  );
 }
}
```

Output:



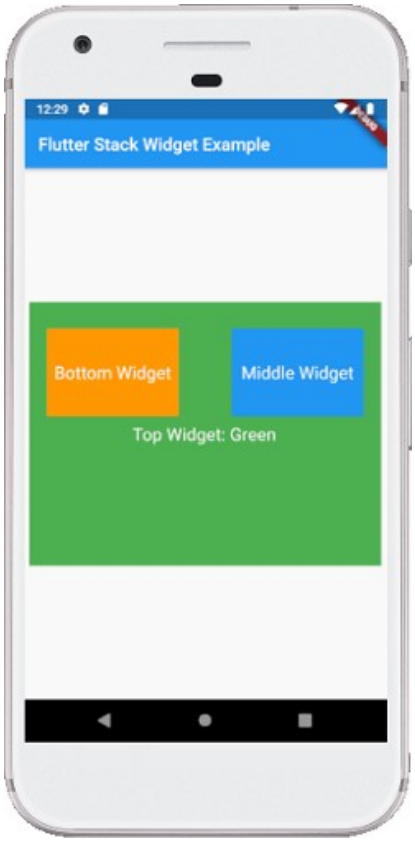
Stack Widget

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

void main() => runApp(MyApp());

/// This Widget is the main application
widget. class MyApp extends
StatelessWidget { @override
  Widget build(BuildContext context) {
```

```
    return
      MaterialApp( home:
      MyStackWidget(),
    );
  }
}

class MyStackWidget extends
  StatelessWidget { @override
  Widget build(BuildContext context) {
    return
      MaterialApp( ho
      me:
      Scaffold( appBar
      : AppBar(
          title: Text("Flutter Stack Widget Example"),
        ),
        body:
          Center( ch
          ild: Stack(
            fit:
            StackFit.passthrough,
            overflow:
            Overflow.visible,
            children: <Widget>[
              // Max Size Widget
              Container(
                height: 300,
                width: 400,
                color:
                Colors.green,
                child: Center(
                  child: Text(
                    'Top Widget: Green',
                    style: TextStyle(color: Colors.white, fontSize: 20),
                  ),
                ),
              ),
              Positioned
                ( top: 30,
                right: 20,
                child:
                  Container( h
                  eight: 100,
                  width: 150,
                  color:
                  Colors.blue,
                  child:
                  Center( child:
                  Text(
                      'Middle Widget',
                      style: TextStyle(color: Colors.white, fontSize: 20),
                    ),
                  ),
                ),
              ),
              Positioned
                ( top: 30,
```

```
            left: 20,
            child:
              Container( h
              eight: 100,
              width: 150,
              color:
              Colors.orange,
              child: Center(
                child: Text(
                  'Bottom
                  Widget',
                  style: TextStyle(color: Colors.white, fontSize: 20),
                ),
              ),
            )
          ),
        ],
      ),
    )
  ),
 );
 }
}
```

Output:



Lab Session 5:

a) Learn about stateful and stateless widgets

In Flutter, widgets can be categorized into two main types based on their behavior regarding state management: stateful widgets and stateless widgets.

Stateless Widgets:
Definition: Stateless widgets are widgets that do not have any mutable state. Once created, their properties (configuration) cannot change.

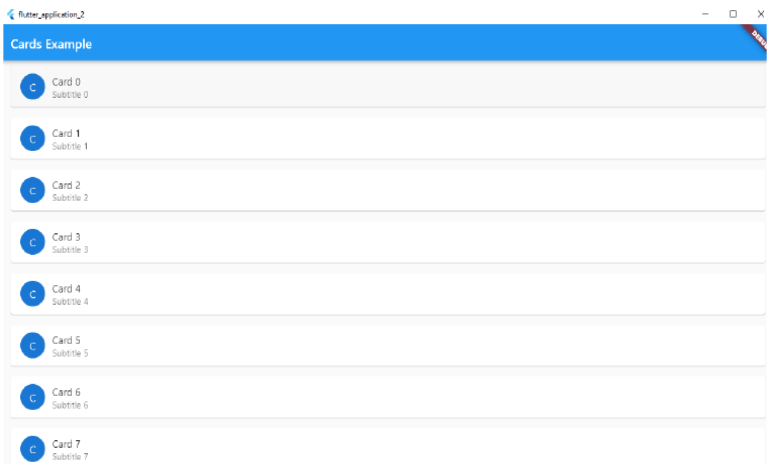
Characteristics:

They are immutable and lightweight.
They only depend on their configuration and the build context provided during construction. Their appearance (UI) is purely a function of their configuration.
They are ideal for UI elements that do not change over time, such as static text labels, icons, or simple buttons.

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

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

class MyApp extends StatelessWidget {
@override
 Widget build(BuildContext context)
 { return MaterialApp(
   home: Scaffold(
   appBar: AppBar(

     title: Text('Cards Example'),
    ),
    body: CardList(),
   ),
  );
 }
}

class CardList extends StatelessWidget {
 @override
  Widget build(BuildContext context)
    { return ListView.builder(
   itemCount: 10,
   itemBuilder: (context, index)
    { return CardItem(
     title: 'Card $index',
      subtitle: 'Subtitle $index',
```

```
      );
    },
  );
 }
}


class CardItem extends StatelessWidget {
 final String title;
 final String subtitle;

 const
  CardItem({ Key
  key, @required
  this.title,
  @required this.subtitle,
 }) : super(key: key);

@override
 Widget build(BuildContext context)
 { return Card(
   margin: EdgeInsets.symmetric(horizontal: 16, vertical: 8),
   child:
   ListTile( title:
   Text(title),
    subtitle: Text(subtitle),
    leading: CircleAvatar(
     child: Text('${title.substring(0, 1)}'),
    ),
    onTap: () {
     // Handle card tap
    },
   ),
  );
 }
```

Output::



Stateful Widgets:

Definition: Stateful widgets are widgets that maintain state, allowing them to change and update over time inresponse to user actions, network events, or other factors.

Characteristics:

They have an associated mutable state that can change during the widget's lifetime. The state is stored in a separate class that extends State and is associated with the stateful widget. Changes to the state trigger a rebuild of the widget's UI, allowing dynamic updates.

They are ideal for UI elements that need to change or react to user interactions, such as input forms, animations, or scrollable lists.

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

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

class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) { return
  MaterialApp(
   home: CounterApp(),
  );
 }
}

class CounterApp extends StatefulWidget {
 @override
 _CounterAppState createState() => _CounterAppState();
}
class _CounterAppState extends State<CounterApp> { int
 _counter = 0;

void _incrementCounter() { setState(() {
    _counter++;
  });
 }

 @override
 Widget build(BuildContext context) { return
  Scaffold(
   appBar: AppBar(
```

```
      title: Text('Counter App'),
    ),
    body: Center( child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: <Widget>[ Text(
        'Counter:',
        style: TextStyle(fontSize: 24),
      ),
      Text( '$_counter',
        style: TextStyle(fontSize: 36, fontWeight: FontWeight.bold),
      ),
    ],
    ),
    ),
    floatingActionButton: FloatingActionButton(
     onPressed: _incrementCounter,
     tooltip: 'Increment', child:
     Icon(Icons.add),
    ),
   );
  }
}
```

Output::

b) Implement state management using set State and Provider

Stateful widgets are composed of two classes: the stateful widget itself (which extends s StatefulWidget) and its corresponding state class (which extends State). The state class is responsible for maintaining the widget'smutable state and updating the UI accordingly via the setState() method.

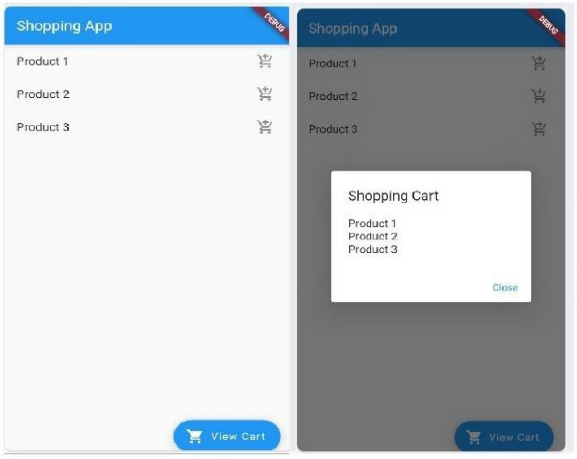
stateless widgets are static and immutable, while stateful widgets are dynamic and can change over time by managing their internal state. Understanding the difference between these two types of widgets is essential for designing and building efficient and responsive Flutter UIs.

State Management using setState():

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

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

class MyApp extends StatelessWidget {
@override
 Widget build(BuildContext context) { return
  MaterialApp(
   home: CounterPage(),
  );
 }
}

class CounterPage extends StatefulWidget {
 @override
 _CounterPageState createState() => _CounterPageState();
}

class _CounterPageState extends State<CounterPage> { int
 _counter = 0;

 void _incrementCounter() { setState(() {
   _counter++;
  });
 }

 @override
 Widget build(BuildContext context) { return
  Scaffold(
   appBar: AppBar(
    title: Text('Counter Example (setState)'),
```

```
    ),
    body: Center( child: Column(
      mainAxisAlignment: MainAxisAlignment.center, children:
      <Widget>[ T
       ext(
        'Counter Value:',
       ),
       Text( '$_counter',
        style: Theme.of(context).textTheme.headline4,
       ),
      ],
     ),
    ),
    floatingActionButton: FloatingActionButton(
     onPressed: _incrementCounter,
     tooltip: 'Increment', child:
     Icon(Icons.add),
    ),
   );
  }
}
```

Output::



State Management using provider package:

```
 // main.dart
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'provider/movie_provider.dart';
import 'screens/home_screen.dart';
void main() {
 runApp(ChangeNotifierProvider<MovieProvider>(
  child: const MyApp(),
  create: (_) => MovieProvider(), // Create a new ChangeNotifier object
 ));
}

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

 @override
 Widget build(BuildContext context) {

  return MaterialApp(
   // Remove the debug banner
   debugShowCheckedModeBanner: false,
   title: 'State Management using provider',
   theme: ThemeData(
    primarySwatch: Colors.indigo,
   ),

   home: const HomeScreen(),
  );
}
}
```

create a model folder for models and create file movie.dart

```
class Movie {
 final String title;
 final String? runtime; // how long this movie is (in minute)

 Movie({required this.title, this.runtime});
}
```

Create a provider folder and create movie_provider.dart inside the provider folder

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

```
import '../models/movie.dart';

// A list of movies
final List<Movie> initialData = List.generate( 50,
  (index) => Movie(
    title: "Moview $index",
    runtime: "${Random().nextInt(100) + 60} minutes"));

class MovieProvider with ChangeNotifier {
 // All movies (that will be displayed on the Home screen)
 final List<Movie> _movies = initialData;

 // Retrieve all movies
 List<Movie> get movies => _movies;

 // Favorite movies (that will be shown on the MyList screen)
 final List<Movie> _myList = [];

 // Retrieve favorite movies List<Movie> get
 myList => _myList;

 // Adding a movie to the favorites list void
 addToList(Movie movie) {
  _myList.add(movie); notifyListeners();
 }

 // Removing a movie from the favorites list void
 removeFromList(Movie movie) {
  _myList.remove(movie); notifyListeners();
 }
}
```

Create a screens folder for screens

```
// screens/home_screen.dart
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../provider/movie_provider.dart';
import 'my_list_screen.dart';
class HomeScreen extends StatefulWidget {
 const HomeScreen({Key? key}) : super(key: key);
 @override
 State<HomeScreen> createState() => _HomeScreenState();
}
class _HomeScreenState extends State<HomeScreen> {
```

```
@override
Widget build(BuildContext context) {
 var movies = context.watch<MovieProvider>().movies; var
 myList = context.watch<MovieProvider>().myList;
 return Scaffold( appBar: AppBar(
   title: const Text('State Management using provider'),
  ),
  body: Padding(
   padding: const EdgeInsets.all(15), child:
   Column(
    crossAxisAlignment: CrossAxisAlignment.stretch,
    children: [ ElevatedButton.icon(
     onPressed: () {
       Navigator.of(context).push(
        MaterialPageRoute(
         builder: (context) => const MyListScreen(),
        ),
       );
      },
      icon: const Icon(Icons.favorite), label:
       Text( "Go to my list (${myList.length})",
       style: const TextStyle(fontSize: 24),
      ),
     style:
        ElevatedButton.styleFrom( primar
        y:Colors.red,
        padding: const EdgeInsets.symmetric(vertical: 20)),
  ),
     const SizedBox( height: 15,
     ),
     Expanded(
      child: ListView.builder( itemCount:
        movies.length, itemBuilder: (_, index)
        { final currentMovie = movies[index];
        return Card(
          key: ValueKey(currentMovie.title), color:
          Colors.amberAccent.shade100, elevation: 4,
          child: ListTile(
           title: Text(currentMovie.title), subtitle:
             Text(currentMovie.runtime ?? 'No
             information'),
           trailing: IconButton(
            icon: Icon( Icons.favorite,
             color: myList.contains(currentMovie)
               ? Colors.red
               : Colors.white,
              size: 30,
            ),
            onPressed: () {
             if (!myList.contains(currentMovie)) { context
```

```
                .read<MovieProvider>()
                .addToList(currentMovie);
            } else { context
                .read<MovieProvider>()
                .removeFromList(currentMovie);
            }
          },
        ),
      ),
    );
  }),
 ),
 ],
 ),
 ),
 );
 }
}
```

create my_list_screen.dart inside the screens folder

```
// screens/my_list_screen.dart
import 'package:flutter/material.dart'; import
'package:provider/provider.dart';
import '../provider/movie_provider.dart'; class
MyListScreen extends StatefulWidget {
 const MyListScreen({Key? key}) : super(key: key);

 @override
 State<MyListScreen> createState() => _MyListScreenState();
}

class _MyListScreenState extends State<MyListScreen> {
 @override
 Widget build(BuildContext context) {
  final myList =
  context.watch<MovieProvider>().myList; return
  Scaffold(
   appBar: AppBar(
    title: Text("My List (${myList.length})"),
   ),
   body: ListView.builder( itemCount:
     myList.length, itemBuilder: (_, index)
     { final currentMovie = myList[index];

       return Card(
       key: ValueKey(currentMovie.title), elevation: 4,

       child: ListTile(
```

```
          title: Text(currentMovie.title),
          subtitle: Text(currentMovie.runtime ?? ''), trailing:
          TextButton(
           child: const Text( 'Remove',
            style: TextStyle(color: Colors.red),
           ),
           onPressed: () {
            context.read<MovieProvider>().removeFromList(currentMovie);
           },
          ),
         ),
        );
       }),
    });
}
```

Output::

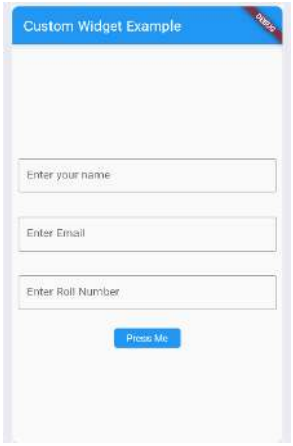
Lab Session 6:

a) Create custom widgets for specific UI elements

```
import 'package:flutter/material.dart'; void
main() {
 runApp(MyApp());
}

class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) { return
  MaterialApp(
   home: Scaffold( appBar: AppBar(
     title: Text('Custom Widget Example'),
    ),
    body: Column(
     mainAxisAlignment: MainAxisAlignment.center, children:
     <Widget>[ P
      adding(
       padding: const EdgeInsets.all(8.0), child:
       CustomTextField(
        hintText: 'Enter your name', onChanged:
        (value) { print('Name changed: $value');
        },
       ),
      ),
      SizedBox(height: 20), Padding(
       padding: const EdgeInsets.all(8.0), child:
       CustomTextField(
        hintText: 'Enter Email', onChanged: (value)
        { print('Name changed: $value');
        },
       ),
      ),
      SizedBox(height: 20), Padding(
       padding: const EdgeInsets.all(8.0), child:
       CustomTextField(
        hintText: 'Enter Roll Number', onChanged:
        (value) { print('Name changed: $value');
        },
       ),
      ),
      SizedBox(height: 20), CustomButton(
      text: 'Press Me',
      onPressed: () {
```

```
          print('Button pressed!');
        },
      ),
    ],
   ),
  ),
 );
 }
}

class CustomButton extends StatelessWidget { final
 String? text;
 final VoidCallback? onPressed;

 const CustomButton({ Key? key,
  @required this.text,
  @required this.onPressed,
 }) : super(key: key);

 @override
 Widget build(BuildContext context) { return
  ElevatedButton(
   onPressed: onPressed, child:
   Text(text!),
  );
 }
}

class CustomTextField extends StatelessWidget { final
 String hintText;
 final ValueChanged<String> onChanged;

 const CustomTextField({ Key? key,
  required this.hintText,
  required this.onChanged,
 }) : super(key: key);

 @override
 Widget build(BuildContext context) { return
  TextField(
   onChanged: onChanged, decoration:
   InputDecoration( hintText: hintText,
    border: OutlineInputBorder(),
   ),
  );
 }
  }
```

Output::



b) Apply styling using themes and custom styles

In Flutter, you can apply styling to your widgets using themes and custom styles to maintain consistency and make your UI morevisually appealing.

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

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

class MyApp extendsStatelessWidget {
 @override
 Widget build(BuildContext context)
 {
  return MaterialApp( theme:
   ThemeData(
    // Define the overall theme of the app
    primaryColor: Colors.blue, accentColor:
    Colors.orange, fontFamily: 'Roboto',
    textTheme: TextTheme(
     headline1: TextStyle(fontSize: 24, fontWeight:
     FontWeight.bold), bodyText1: TextStyle(fontSize: 16),
    ),
    elevatedButtonTheme: ElevatedButtonThemeData( style:
     ElevatedButton.styleFrom(
      primary: Colors.blue,
      textStyle: TextStyle(fontSize: 18),
```

```
        padding: EdgeInsets.symmetric(horizontal: 20, vertical: 15),
        shape: RoundedRectangleBorder(
         borderRadius: BorderRadius.circular(10),
        ),

      ),
     ),
     ),
   home: HomePage(),
  );
 }
}

class HomePage extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Scaffold( appBar:
    AppBar( title: Text('Styling
    Example'),
   ),
   body: Center( child: Column(
     mainAxisAlignment: MainAxisAlignment.center, children:
     <Widget>[ T
      ext(
       'Welcome to MyApp',
       style: Theme.of(context).textTheme.headline1,
      ),
      SizedBox(height: 20), ElevatedButton(
      onPressed: () {},
       child: Text('Get Started'),
      ),
     ],
    ),
   ),
  );
 }
   }
```

Output: