**4.b) Implement navigation with named routes.**

import 'package:flutter/material.dart';

void main() {

  runApp(MyApp());

}

class MyApp extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      title: 'Named Routes Demo',

      initialRoute: '/',

      routes: {

        '/': (context) => HomeScreen(),

        '/second': (context) => SecondScreen(),

        '/third': (context) => ThirdScreen(),

      },

    );

  }

}

class HomeScreen extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Home Screen'),

      ),

      body: Center(

        child: ElevatedButton(

          onPressed: () {

            Navigator.pushNamed(context, '/second');

          },

          child: Text('Go to Second Screen'),

        ),

      ),

    );

  }

}

class SecondScreen extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Second Screen'),

      ),

      body: Center(

        child: ElevatedButton(

          onPressed: () {

            Navigator.pushNamed(context, '/third');

          },

          child: Text('Go to Third Screen'),

        ),

      ),

    );

  }

}

class ThirdScreen extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Third Screen'),

      ),

      body: Center(

        child: ElevatedButton(

          onPressed: () {

            Navigator.popUntil(context, ModalRoute.withName('/'));

          },

          child: Text('Go Back to Home'),

        ),

      ),

    );

  }

}

**5 b) Implement state management using set state and provider.**

A simple counter app using setState to manage the state.

**Using setState**: It's suitable for simple state management in small widgets or when you have straightforward state needs. The state is local to the widget and requires rebuilding the widget tree.

The counter state is managed locally within the CounterScreen widget. When the button is pressed, setState is called to update the \_counter variable, and the widget rebuilds to reflect the new count.

import 'package:flutter/material.dart'

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'SetState Example',

home: CounterScreen(),

);

}

}

class CounterScreen extends StatefulWidget {

@override

\_CounterScreenState createState() => \_CounterScreenState();

}

class \_CounterScreenState extends State<CounterScreen> {

int \_counter = 0;

void \_incrementCounter() {

setState(() {

\_counter++;

});

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text('Counter')),

body: Center(

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Text('Counter: $\_counter'),

ElevatedButton(

onPressed: \_incrementCounter,

child: Text('Increment'),

),

],

),

),

);

}

}

**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>[

// Custom text field widgets

CustomTextField('Enter your name'),

SizedBox(height: 20),

CustomTextField('Enter your email'),

SizedBox(height: 20),

CustomTextField('Enter your roll number'),

SizedBox(height: 20),

// Custom button widget

CustomButton('Press Me', () {

print('Button pressed!');

}),

],

),

),

);

}

}

class CustomTextField extends StatelessWidget {

final String hintText;

CustomTextField(this.hintText); // Constructor to accept hint text

@override

Widget build(BuildContext context) {

return Padding(

padding: const EdgeInsets.all(8.0),

child: TextField(

decoration: InputDecoration(

hintText: hintText,

border: OutlineInputBorder(),

),

),

);

}

}

class CustomButton extends StatelessWidget {

final String text;

final VoidCallback onPressed;

CustomButton(this.text, this.onPressed); // Constructor for button text and callback

@override

Widget build(BuildContext context) {

return ElevatedButton(

onPressed: onPressed,

child: Text(text),

);

}

}

**8 a) Add animations to UI elements using Flutter's animation framework.**

import 'package:flutter/material.dart';

void main() {

  runApp(MyApp());

}

class MyApp extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      title: 'Simple Animation',

      home: SimpleAnimation(),

    );

  }

class SimpleAnimation extends StatefulWidget {

  @override

  \_SimpleAnimationState createState() => \_SimpleAnimationState();

}

class \_SimpleAnimationState extends State<SimpleAnimation> with TickerProviderStateMixin {

  late AnimationController \_controller;

  late Animation<double> \_animation;

  @override

  void initState() {

    super.initState();

    \_controller = AnimationController(duration: Duration(seconds: 2), vsync: this);

    \_animation = Tween(begin: 0.0, end: 100.0).animate(\_controller);

  }

  @override

  void dispose() {

    \_controller.dispose();

    super.dispose();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Animation'),

      ),

      body: Center(

        child: AnimatedBuilder(

          animation: \_animation,

          builder: (context, child) {

            return Container(

              width: \_animation.value,

              height: \_animation.value,

              child: FlutterLogo(),

            );

          },

        ),

      ),

      floatingActionButton: FloatingActionButton(

        onPressed: () {

          \_controller.forward();

        },

        child: Icon(Icons.play\_arrow),

      ),

    );

  }

}

**8 b) Experiment with different types of animations (fade, slide, etc.).**

**Fade Animation**

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Fade Animation Example',

theme: ThemeData(

primarySwatch: Colors.blue,

),

home: FadeAnimationWidget(),

);

}

}

class FadeAnimationWidget extends StatefulWidget {

@override

\_FadeAnimationWidgetState createState() => \_FadeAnimationWidgetState();

}

class \_FadeAnimationWidgetState extends State<FadeAnimationWidget>

with SingleTickerProviderStateMixin {

late AnimationController \_animationController;

late Animation<double> \_opacityAnimation;

@override

void initState() {

super.initState();

\_animationController = AnimationController(

vsync: this,

duration: Duration(seconds: 10),

);

\_opacityAnimation = Tween<double>(begin: 0.0, end: 1.0).animate(

CurvedAnimation(

parent: \_animationController,

curve: Curves.easeInOut,

),

);

\_animationController.forward();

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Fade Animation Example'),

),

body: Center(

child: FadeTransition(

opacity: \_opacityAnimation,

child: Container(

width: 200,

height: 200,

color: Colors.blue,

child: Center(

child: Text(

'Fade Animation',

style: TextStyle(

color: Colors.white,

fontSize: 20,

),

),

),

),

),

),

);

}

@override

void dispose() {

\_animationController.dispose();

super.dispose();

}}

**Slide Animation:**

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Slide Animation Example',

theme: ThemeData(

primarySwatch: Colors.blue,

),

home: SlideAnimationWidget(),

);

}

}

class SlideAnimationWidget extends StatefulWidget {

@override

\_SlideAnimationWidgetState createState() => \_SlideAnimationWidgetState();

}

class \_SlideAnimationWidgetState extends State<SlideAnimationWidget>

with SingleTickerProviderStateMixin {

late AnimationController \_animationController;

late Animation<Offset> \_slideAnimation;

@override

void initState() {

super.initState();

\_animationController = AnimationController(

vsync: this,

duration: Duration(seconds: 2),

);

\_slideAnimation = Tween<Offset>(

begin: Offset(-1.0, 0.0),

end: Offset(0.0, 0.0),

).animate(

CurvedAnimation(

parent: \_animationController,

curve: Curves.easeInOut,

),

);

\_animationController.forward();

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Slide Animation Example'),

),

body: SlideTransition(

position: \_slideAnimation,

child: Container(

width: 200,

height: 200,

color: Colors.blue,

child: Center(

child: Text(

'Slide Animation',

style: TextStyle(

color: Colors.white,

fontSize: 20,

),

),

),

),

),

);

}

@override

void dispose() {

\_animationController.dispose();

super.dispose();

}

}

**9 b) Display the fetched data in a meaningful way in the UI.**

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'dart:convert';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'API Fetch Example',

theme: ThemeData(

primarySwatch: Colors.blue,

),

home: MyApiFetchWidget(),

);

}

}

class MyApiFetchWidget extends StatefulWidget {

@override

\_MyApiFetchWidgetState createState() => \_MyApiFetchWidgetState();

}

class \_MyApiFetchWidgetState extends State<MyApiFetchWidget> {

late Future<List<Post>> \_posts;

@override

void initState() {

super.initState();

\_posts = fetchPosts();

}

Future<List<Post>> fetchPosts() async {

final response =

await http.get(Uri.parse('https://jsonplaceholder.typicode.com/posts'));

if (response.statusCode == 200) {

List<dynamic> data = json.decode(response.body);

List<Post> posts = data.map((post) => Post.fromJson(post)).toList();

return posts;

} else {

throw Exception('Failed to load posts');

}

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('API Fetch Example'),

),

body: FutureBuilder<List<Post>>(

future: \_posts,

builder: (context, snapshot) {

if (snapshot.connectionState == ConnectionState.waiting) {

return Center(child: CircularProgressIndicator());

} else if (snapshot.hasError) {

return Center(child: Text('Error: ${snapshot.error}'));

} else {

return PostList(posts: snapshot.data!);

}

},

),

);

}

}

class PostList extends StatelessWidget {

final List<Post> posts;

PostList({required this.posts});

@override

Widget build(BuildContext context) {

return ListView.builder(

itemCount: posts.length,

itemBuilder: (context, index) {

return PostItem(post: posts[index]);

},

);

}

}

class PostItem extends StatelessWidget {

final Post post;

PostItem({required this.post});

@override

Widget build(BuildContext context) {

return Card(

margin: EdgeInsets.all(10),

elevation: 3,

child: Padding(

padding: EdgeInsets.all(15),

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

Text(

post.title,

style: TextStyle(

fontSize: 18,

fontWeight: FontWeight.bold,

),

),

SizedBox(height: 10),

Text(

post.body,

style: TextStyle(fontSize: 16),

),

],

),

),

);

}

}

class Post {

final int userId;

final int id;

final String title;

final String body;

Post({

required this.userId,

required this.id,

required this.title,

required this.body,

});

factory Post.fromJson(Map<String, dynamic> json) {

return Post(

userId: json['userId'],

id: json['id'],

title: json['title'],

body: json['body'],

);

}

}