**Modal lab for UIT1711- Mobile application development**

**Questions:**

1. Suppose you're designing a meditation app. How would you change the app's background color to a calming tone that suits the app's purpose using Flutter?

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

void main() {

runApp(MyMeditationApp());

}

class MyMeditationApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: Scaffold(

body: Container(

color: Colors.blueGrey[300], // Change this to the calming color of your choice

child: Center(

child: Text(

'Meditation App',

style: TextStyle(fontSize: 24.0),

),

),

),

),

);

}

}

1. In a fitness app, how could you implement a side menu allowing users to navigate between sections like workouts, progress, and settings using Flutter?

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

void main() {

runApp(MyFitnessApp());

}

class MyFitnessApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: MyHomePage(),

);

}

}

class MyHomePage extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Fitness App'),

),

drawer: Drawer(

child: ListView(

padding: EdgeInsets.zero,

children: <Widget>[

DrawerHeader(

decoration: BoxDecoration(

color: Colors.blue,

),

child: Text(

'Menu',

style: TextStyle(

color: Colors.white,

fontSize: 24,

),

),

),

ListTile(

title: Text('Workouts'),

onTap: () {

// Navigate to the workouts section

Navigator.pop(context); // Close the drawer

// Add your navigation logic here

},

),

ListTile(

title: Text('Progress'),

onTap: () {

// Navigate to the progress section

Navigator.pop(context); // Close the drawer

// Add your navigation logic here

},

),

ListTile(

title: Text('Settings'),

onTap: () {

// Navigate to the settings section

Navigator.pop(context); // Close the drawer

// Add your navigation logic here

},

),

],

),

),

body: Center(

child: Text(

'Welcome to the Fitness App!',

style: TextStyle(fontSize: 24.0),

),

),

);

}

}

1. In a language learning app, how could you create a button that switches the displayed text between English and another language when clicked in Flutter?

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

void main() {

runApp(MyLanguageApp());

}

class MyLanguageApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: MyLanguagePage(),

);

}

}

class MyLanguagePage extends StatefulWidget {

@override

\_MyLanguagePageState createState() => \_MyLanguagePageState();

}

class \_MyLanguagePageState extends State<MyLanguagePage> {

bool isEnglish = true;

void toggleLanguage() {

setState(() {

isEnglish = !isEnglish;

});

}

@override

Widget build(BuildContext context) {

String displayedText = isEnglish ? 'Hello, World!' : 'Hola, Mundo!';

return Scaffold(

appBar: AppBar(

title: Text('Language Learning App'),

),

body: Center(

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

Text(

displayedText,

style: TextStyle(fontSize: 24.0),

),

SizedBox(height: 20.0),

ElevatedButton(

onPressed: () {

toggleLanguage();

},

child: Text('Switch Language'),

),

],

),

),

);

}

}

1. In a game app, create a button that increments a player's score each time it's tapped in Flutter.

Ans:

import 'package:flutter/material.dart';

void main() {

runApp(MyGameApp());

}

class MyGameApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: MyGamePage(),

);

}

}

class MyGamePage extends StatefulWidget {

@override

\_MyGamePageState createState() => \_MyGamePageState();

}

class \_MyGamePageState extends State<MyGamePage> {

int playerScore = 0;

void incrementScore() {

setState(() {

playerScore++;

});

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Game App'),

),

body: Center(

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

Text(

'Player Score: $playerScore',

style: TextStyle(fontSize: 24.0),

),

SizedBox(height: 20.0),

ElevatedButton(

onPressed: () {

incrementScore();

},

child: Text('Increment Score'),

),

],

),

),

);

}

}

1. Design an application to get input ‘n’ from user. Print a pyramid for the given input n.

import 'package:flutter/material.dart';

void main() {

runApp(PyramidApp());

}

class PyramidApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: PyramidPage(),

);

}

}

class PyramidPage extends StatefulWidget {

@override

\_PyramidPageState createState() => \_PyramidPageState();

}

class \_PyramidPageState extends State<PyramidPage> {

TextEditingController \_controller = TextEditingController();

int pyramidHeight = 0;

void buildPyramid() {

setState(() {

// Get the input value from the user

int userInput = int.tryParse(\_controller.text) ?? 0;

pyramidHeight = userInput;

});

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Pyramid App'),

),

body: Padding(

padding: const EdgeInsets.all(16.0),

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

TextField(

controller: \_controller,

keyboardType: TextInputType.number,

decoration: InputDecoration(

labelText: 'Enter a number (n)',

),

),

SizedBox(height: 20.0),

ElevatedButton(

onPressed: () {

buildPyramid();

},

child: Text('Build Pyramid'),

),

SizedBox(height: 20.0),

buildPyramidWidget(),

],

),

),

);

}

Widget buildPyramidWidget() {

List<Widget> rows = [];

for (int i = 1; i <= pyramidHeight; i++) {

rows.add(

Padding(

padding: EdgeInsets.symmetric(horizontal: (pyramidHeight - i) \* 10.0),

child: Text(

'\*' \* (2 \* i - 1),

style: TextStyle(fontSize: 20.0),

),

),

);

}

return Column(

children: rows,

);

}

}

1. In a restaurant app, how could you arrange menu items horizontally and categories vertically using Row and Column widgets in Flutter?

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

void main() {

runApp(RestaurantApp());

}

class RestaurantApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: Scaffold(

appBar: AppBar(

title: Text('Restaurant App'),

),

body: MenuScreen(),

),

);

}

}

class MenuScreen extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Padding(

padding: const EdgeInsets.all(16.0),

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: <Widget>[

Text(

'Categories',

style: TextStyle(fontSize: 24.0, fontWeight: FontWeight.bold),

),

SizedBox(height: 10.0),

buildCategory('Appetizers'),

buildCategory('Main Courses'),

buildCategory('Desserts'),

SizedBox(height: 20.0),

Text(

'Menu Items',

style: TextStyle(fontSize: 24.0, fontWeight: FontWeight.bold),

),

SizedBox(height: 10.0),

buildMenuItem('Garlic Bread'),

buildMenuItem('Pasta Carbonara'),

buildMenuItem('Chocolate Cake'),

],

),

);

}

Widget buildCategory(String categoryName) {

return Padding(

padding: const EdgeInsets.symmetric(vertical: 8.0),

child: Text(

categoryName,

style: TextStyle(fontSize: 18.0),

),

);

}

Widget buildMenuItem(String itemName) {

return Padding(

padding: const EdgeInsets.symmetric(vertical: 8.0),

child: Row(

children: <Widget>[

Icon(Icons.fastfood),

SizedBox(width: 10.0),

Text(

itemName,

style: TextStyle(fontSize: 18.0),

),

],

),

);

}

}

1. You're creating a Journal app. How would you fetch information or news from the user and display it in the UI.

import 'package:flutter/material.dart';

void main() {

runApp(JournalApp());

}

class JournalApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: JournalHomePage(),

);

}

}

class JournalHomePage extends StatefulWidget {

@override

\_JournalHomePageState createState() => \_JournalHomePageState();

}

class \_JournalHomePageState extends State<JournalHomePage> {

TextEditingController \_journalEntryController = TextEditingController();

List<String> journalEntries = [];

void addJournalEntry() {

setState(() {

String entry = \_journalEntryController.text;

if (entry.isNotEmpty) {

journalEntries.add(entry);

\_journalEntryController.clear();

}

});

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Journal App'),

),

body: Padding(

padding: const EdgeInsets.all(16.0),

child: Column(

crossAxisAlignment: CrossAxisAlignment.stretch,

children: <Widget>[

TextField(

controller: \_journalEntryController,

decoration: InputDecoration(

hintText: 'Write your journal entry...',

),

),

SizedBox(height: 10.0),

ElevatedButton(

onPressed: () {

addJournalEntry();

},

child: Text('Add Entry'),

),

SizedBox(height: 20.0),

Text(

'Journal Entries:',

style: TextStyle(fontSize: 20.0, fontWeight: FontWeight.bold),

),

SizedBox(height: 10.0),

Expanded(

child: ListView.builder(

itemCount: journalEntries.length,

itemBuilder: (context, index) {

return Card(

child: ListTile(

title: Text(journalEntries[index]),

),

);

},

),

),

],

),

),

);

}

}

1. Get current location of user and share it using geolocator and geocoding
2. A company has a rectangle logo placed in square box, the default color of the logo is red, design an application such that user can change the color of logo by clicking on the button.

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

void main() {

runApp(LogoColorChangeApp());

}

class LogoColorChangeApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: LogoColorChangePage(),

);

}

}

class LogoColorChangePage extends StatefulWidget {

@override

\_LogoColorChangePageState createState() => \_LogoColorChangePageState();

}

class \_LogoColorChangePageState extends State<LogoColorChangePage> {

Color logoColor = Colors.red; // Default color is red

void changeLogoColor() {

setState(() {

// Change the color to blue when the button is clicked

logoColor = Colors.blue;

});

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Logo Color Change App'),

),

body: Center(

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

Container(

width: 150.0,

height: 150.0,

color: logoColor,

child: Center(

child: Text(

'Logo',

style: TextStyle(

color: Colors.white,

fontSize: 24.0,

),

),

),

),

SizedBox(height: 20.0),

ElevatedButton(

onPressed: () {

changeLogoColor();

},

child: Text('Change Logo Color'),

),

],

),

),

);

}

}

1. Guessing game- your app should fix a random number and the user should guess that number. If the user guesses the correct number then the background color should change to blue else it should be red.

import 'package:flutter/material.dart';

import 'dart:math';

void main() {

runApp(GuessingGameApp());

}

class GuessingGameApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: GuessingGamePage(),

);

}

}

class GuessingGamePage extends StatefulWidget {

@override

\_GuessingGamePageState createState() => \_GuessingGamePageState();

}

class \_GuessingGamePageState extends State<GuessingGamePage> {

late int targetNumber;

TextEditingController \_guessController = TextEditingController();

Color backgroundColor = Colors.red;

@override

void initState() {

super.initState();

generateRandomNumber();

}

void generateRandomNumber() {

setState(() {

targetNumber = Random().nextInt(10) + 1; // Generate a random number between 1 and 10

backgroundColor = Colors.red; // Reset background color to red

});

}

void checkGuess() {

int userGuess = int.tryParse(\_guessController.text) ?? 0;

setState(() {

if (userGuess == targetNumber) {

backgroundColor = Colors.blue; // Change background color to blue for correct guess

} else {

backgroundColor = Colors.red; // Keep background color red for incorrect guess

}

});

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Guessing Game'),

),

body: Container(

color: backgroundColor,

padding: EdgeInsets.all(16.0),

child: Center(

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

Text(

'Guess the Number (1-10)',

style: TextStyle(fontSize: 20.0),

),

SizedBox(height: 10.0),

TextField(

controller: \_guessController,

keyboardType: TextInputType.number,

decoration: InputDecoration(

hintText: 'Enter your guess',

),

),

SizedBox(height: 10.0),

ElevatedButton(

onPressed: () {

checkGuess();

},

child: Text('Check Guess'),

),

SizedBox(height: 10.0),

ElevatedButton(

onPressed: () {

generateRandomNumber();

},

child: Text('New Game'),

),

],

),

),

),

);

}

}

1. Make use of primitive graphical shapes in flutter .

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

void main() {

runApp(PrimitiveShapesApp());

}

class PrimitiveShapesApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: Scaffold(

appBar: AppBar(

title: Text('Primitive Shapes in Flutter'),

),

body: Center(

child: CustomPaint(

painter: MyPainter(),

size: Size(200.0, 200.0), // Adjust the size of the canvas

),

),

),

);

}

}

class MyPainter extends CustomPainter {

@override

void paint(Canvas canvas, Size size) {

Paint paint = Paint()

..color = Colors.blue

..strokeWidth = 2.0

..style = PaintingStyle.stroke;

// Draw a circle

canvas.drawCircle(Offset(100.0, 100.0), 50.0, paint);

// Draw a rectangle

Rect rect = Rect.fromPoints(Offset(50.0, 150.0), Offset(150.0, 200.0));

canvas.drawRect(rect, paint);

// Draw a line

canvas.drawLine(Offset(20.0, 20.0), Offset(180.0, 20.0), paint);

}

@override

bool shouldRepaint(CustomPainter oldDelegate) {

return false;

}

}

1. Build an app that converts units, such as length, weight, or temperature. Users can input a value in one unit and get the converted value in another unit. (°C = (°F - 32) × 5/9)

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

void main() {

runApp(UnitConverterApp());

}

class UnitConverterApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: UnitConverterPage(),

);

}

}

class UnitConverterPage extends StatefulWidget {

@override

\_UnitConverterPageState createState() => \_UnitConverterPageState();

}

class \_UnitConverterPageState extends State<UnitConverterPage> {

TextEditingController \_inputController = TextEditingController();

String \_result = '';

bool \_isFahrenheitToCelsius = true;

void convert() {

double inputValue = double.tryParse(\_inputController.text) ?? 0;

setState(() {

if (\_isFahrenheitToCelsius) {

// Convert Fahrenheit to Celsius

double celsius = (inputValue - 32) \* 5 / 9;

\_result = '$inputValue°F is ${celsius.toStringAsFixed(2)}°C';

} else {

// Convert Celsius to Fahrenheit

double fahrenheit = (inputValue \* 9 / 5) + 32;

\_result = '$inputValue°C is ${fahrenheit.toStringAsFixed(2)}°F';

}

});

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Unit Converter App'),

),

body: Padding(

padding: const EdgeInsets.all(16.0),

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

TextField(

controller: \_inputController,

keyboardType: TextInputType.number,

decoration: InputDecoration(

hintText: 'Enter value to convert',

),

),

SizedBox(height: 20.0),

Row(

mainAxisAlignment: MainAxisAlignment.spaceEvenly,

children: <Widget>[

ElevatedButton(

onPressed: () {

setState(() {

\_isFahrenheitToCelsius = true;

});

},

child: Text('F to C'),

),

ElevatedButton(

onPressed: () {

setState(() {

\_isFahrenheitToCelsius = false;

});

},

child: Text('C to F'),

),

],

),

SizedBox(height: 20.0),

ElevatedButton(

onPressed: () {

convert();

},

child: Text('Convert'),

),

SizedBox(height: 20.0),

Text(

\_result,

style: TextStyle(fontSize: 18.0),

),

],

),

),

);

}

}

1. Build a virtual dice roller app. Users can tap a button to roll the dice, and the app will display a random result.

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

import 'dart:math';

void main() {

runApp(DiceRollerApp());

}

class DiceRollerApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: DiceRollerPage(),

);

}

}

class DiceRollerPage extends StatefulWidget {

@override

\_DiceRollerPageState createState() => \_DiceRollerPageState();

}

class \_DiceRollerPageState extends State<DiceRollerPage> {

int diceResult = 1;

void rollDice() {

setState(() {

diceResult = Random().nextInt(6) + 1; // Generate a random number between 1 and 6

});

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Dice Roller App'),

),

body: Center(

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

Text(

'Roll the Dice!',

style: TextStyle(fontSize: 24.0, fontWeight: FontWeight.bold),

),

SizedBox(height: 20.0),

Image.asset(

'assets/dice\_$diceResult.png', // Dice images named dice\_1.png, dice\_2.png, ..., dice\_6.png

height: 150.0,

width: 150.0,

),

SizedBox(height: 20.0),

ElevatedButton(

onPressed: () {

rollDice();

},

child: Text('Roll'),

),

],

),

),

);

}

}

(add the necessary pictures in asset folder)

1. Create a Flutter app with a list of items displayed using ListView.

import 'package:flutter/material.dart';

void main() {

runApp(ItemListApp());

}

class ItemListApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: ItemListPage(),

);

}

}

class ItemListPage extends StatelessWidget {

final List<String> items = [

'Item 1',

'Item 2',

'Item 3',

'Item 4',

'Item 5',

'Item 6',

'Item 7',

'Item 8',

'Item 9',

'Item 10',

];

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Item List App'),

),

body: ListView.builder(

itemCount: items.length,

itemBuilder: (context, index) {

return ListTile(

title: Text(items[index]),

onTap: () {

// Add your item tap logic here

print('Tapped on ${items[index]}');

},

);

},

),

);

}

}

1. Implement a basic calculator with addition, subtraction, multiplication, and division.

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

void main() {

runApp(CalculatorApp());

}

class CalculatorApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: CalculatorPage(),

);

}

}

class CalculatorPage extends StatefulWidget {

@override

\_CalculatorPageState createState() => \_CalculatorPageState();

}

class \_CalculatorPageState extends State<CalculatorPage> {

String \_input = '';

double \_result = 0;

String \_operation = '';

bool \_isOperationSelected = false;

void \_onDigitPressed(String digit) {

setState(() {

if (\_isOperationSelected) {

\_input = digit;

\_isOperationSelected = false;

} else {

\_input += digit;

}

});

}

void \_onOperationPressed(String operation) {

setState(() {

if (\_input.isNotEmpty) {

if (\_operation.isNotEmpty) {

\_calculateResult();

} else {

\_result = double.parse(\_input);

}

\_operation = operation;

\_input = '';

\_isOperationSelected = true;

}

});

}

void \_onEqualsPressed() {

setState(() {

if (\_input.isNotEmpty && \_operation.isNotEmpty) {

\_calculateResult();

\_operation = '';

\_isOperationSelected = false;

}

});

}

void \_onClearPressed() {

setState(() {

\_input = '';

\_result = 0;

\_operation = '';

\_isOperationSelected = false;

});

}

void \_calculateResult() {

double inputValue = double.parse(\_input);

switch (\_operation) {

case '+':

\_result += inputValue;

break;

case '-':

\_result -= inputValue;

break;

case '×':

\_result \*= inputValue;

break;

case '÷':

if (inputValue != 0) {

\_result /= inputValue;

} else {

// Handle division by zero error

\_input = 'Error';

return;

}

break;

}

\_input = \_result.toString();

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Calculator App'),

),

body: Padding(

padding: const EdgeInsets.all(16.0),

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

Text(

\_input.isEmpty ? '0' : \_input,

style: TextStyle(fontSize: 24.0),

),

SizedBox(height: 20.0),

Row(

mainAxisAlignment: MainAxisAlignment.spaceAround,

children: <Widget>[

ElevatedButton(

onPressed: () => \_onDigitPressed('7'),

child: Text('7'),

),

ElevatedButton(

onPressed: () => \_onDigitPressed('8'),

child: Text('8'),

),

ElevatedButton(

onPressed: () => \_onDigitPressed('9'),

child: Text('9'),

),

ElevatedButton(

onPressed: () => \_onOperationPressed('÷'),

child: Text('÷'),

),

],

),

Row(

mainAxisAlignment: MainAxisAlignment.spaceAround,

children: <Widget>[

ElevatedButton(

onPressed: () => \_onDigitPressed('4'),

child: Text('4'),

),

ElevatedButton(

onPressed: () => \_onDigitPressed('5'),

child: Text('5'),

),

ElevatedButton(

onPressed: () => \_onDigitPressed('6'),

child: Text('6'),

),

ElevatedButton(

onPressed: () => \_onOperationPressed('×'),

child: Text('×'),

),

],

),

Row(

mainAxisAlignment: MainAxisAlignment.spaceAround,

children: <Widget>[

ElevatedButton(

onPressed: () => \_onDigitPressed('1'),

child: Text('1'),

),

ElevatedButton(

onPressed: () => \_onDigitPressed('2'),

child: Text('2'),

),

ElevatedButton(

onPressed: () => \_onDigitPressed('3'),

child: Text('3'),

),

ElevatedButton(

onPressed: () => \_onOperationPressed('-'),

child: Text('-'),

),

],

),

Row(

mainAxisAlignment: MainAxisAlignment.spaceAround,

children: <Widget>[

ElevatedButton(

onPressed: () => \_onDigitPressed('0'),

child: Text('0'),

),

ElevatedButton(

onPressed: () => \_onOperationPressed('.'),

child: Text('.'),

),

ElevatedButton(

onPressed: \_onEqualsPressed,

child: Text('='),

),

ElevatedButton(

onPressed: () => \_onOperationPressed('+'),

child: Text('+'),

),

],

),

SizedBox(height: 20.0),

ElevatedButton(

onPressed: \_onClearPressed,

child: Text('Clear'),

),

],

),

),

);

}

}

1. How can you add padding around a widget in Flutter?

Ans: 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('Padding Example'),

),

body: Padding(

padding: EdgeInsets.all(16.0), // Add padding of 16.0 to all sides

child: Center(

child: Text(

'This is a widget with padding!',

style: TextStyle(fontSize: 20.0),

),

),

),

),

);

}

}

1. Build an app that shows an alert dialog when a button is pressed.

Ans: 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('AlertDialog Example'),

),

body: Center(

child: ElevatedButton(

onPressed: () {

\_showAlertDialog(context);

},

child: Text('Show Alert Dialog'),

),

),

),

);

}

// Function to show the alert dialog

void \_showAlertDialog(BuildContext context) {

showDialog(

context: context,

builder: (BuildContext context) {

return AlertDialog(

title: Text('Alert Dialog'),

content: Text('This is a simple alert dialog.'),

actions: <Widget>[

TextButton(

onPressed: () {

Navigator.of(context).pop(); // Close the alert dialog

},

child: Text('OK'),

),

],

);

},

);

}

}

1. Develop a Flutter app that uses different sized containers on the screen.

Ans: 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('Different Sized Containers'),

),

body: Center(

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

Container(

width: 100.0,

height: 100.0,

color: Colors.blue,

child: Center(

child: Text('Container 1'),

),

),

SizedBox(height: 20.0),

Container(

width: 150.0,

height: 150.0,

color: Colors.green,

child: Center(

child: Text('Container 2'),

),

),

SizedBox(height: 20.0),

Container(

width: 200.0,

height: 200.0,

color: Colors.orange,

child: Center(

child: Text('Container 3'),

),

),

],

),

),

),

);

}

}

1. Create a simple to-do app with the use of database.

import 'package:flutter/material.dart';

import 'package:cloud\_firestore/cloud\_firestore.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'To-Do App with Firebase',

home: ToDoList(),

);

}

}

class ToDoList extends StatefulWidget {

@override

\_ToDoListState createState() => \_ToDoListState();

}

class \_ToDoListState extends State<ToDoList> {

final TextEditingController \_taskController = TextEditingController();

final FirebaseFirestore \_firestore = FirebaseFirestore.instance;

CollectionReference \_tasks = FirebaseFirestore.instance.collection('tasks');

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('To-Do App'),

),

body: Column(

children: <Widget>[

Expanded(

child: StreamBuilder(

stream: \_tasks.snapshots(),

builder: (context, AsyncSnapshot<QuerySnapshot> snapshot) {

if (!snapshot.hasData) {

return Center(

child: CircularProgressIndicator(),

);

}

return ListView.builder(

itemCount: snapshot.data.docs.length,

itemBuilder: (context, index) {

var task = snapshot.data.docs[index];

return ListTile(

title: Text(task['task']),

trailing: IconButton(

icon: Icon(Icons.delete),

onPressed: () {

\_deleteTask(task.id);

},

),

);

},

);

},

),

),

Padding(

padding: const EdgeInsets.all(8.0),

child: Row(

children: <Widget>[

Expanded(

child: TextField(

controller: \_taskController,

decoration: InputDecoration(

hintText: 'Enter a task',

),

),

),

IconButton(

icon: Icon(Icons.add),

onPressed: () {

\_addTask();

},

),

],

),

),

],

),

);

}

void \_addTask() {

String task = \_taskController.text.trim();

if (task.isNotEmpty) {

\_tasks.add({'task': task});

\_taskController.clear();

}

}

void \_deleteTask(String taskId) {

\_tasks.doc(taskId).delete();

}

}

1. Implement a simple animation (e.g., moving a widget across the screen).

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

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Animation Example',

home: AnimatedSquare(),

);

}

}

class AnimatedSquare extends StatefulWidget {

@override

\_AnimatedSquareState createState() => \_AnimatedSquareState();

}

class \_AnimatedSquareState extends State<AnimatedSquare> with SingleTickerProviderStateMixin {

late AnimationController \_controller;

late Animation<Offset> \_animation;

@override

void initState() {

super.initState();

// Create an animation controller with a duration of 2 seconds

\_controller = AnimationController(

duration: Duration(seconds: 2),

vsync: this,

);

// Create a Tween to define the range of values for the animation

Tween<Offset> \_tween = Tween(begin: Offset(0, 0), end: Offset(2, 0));

// Create the animation using the Tween and the controller

\_animation = \_tween.animate(\_controller);

// Add a listener to rebuild the widget when the animation value changes

\_animation.addListener(() {

setState(() {});

});

// Repeat the animation indefinitely

\_controller.repeat(reverse: true);

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Animated Square'),

),

body: Center(

child: SlideTransition(

position: \_animation,

child: Container(

width: 50,

height: 50,

color: Colors.blue,

),

),

),

);

}

@override

void dispose() {

// Dispose of the animation controller when the widget is disposed

\_controller.dispose();

super.dispose();

}

}