a) Install flutter and Dart sdk

**Aim**:To install Flutter and the Dart SDK.

**Procedure:**

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

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,

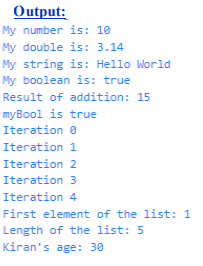
'Raj': 25,

'Alekya': 35,

};

print('Kiran\'s age: ${ages['Kiran']}');

}



**Project 1: Basic Calculator program:**

**Source Code:**

import 'package:flutter/material.dart';

import 'dart:math' as math;

import 'package:math\_expressions/math\_expressions.dart';

//import 'home.dart';

void main() {

runApp(const MyApp());

}

class MyApp extends StatelessWidget {

const MyApp({super.key});

// This widget is the root of your application.

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Flutter Demo',

theme: ThemeData.dark(),

home: MyCalculator(),

);

}

}

class MyCalculator extends StatefulWidget {

const MyCalculator({super.key});

@override

State<MyCalculator> createState() => \_MyCalculatorState();

}

class \_MyCalculatorState extends State<MyCalculator> {

double num1 = 0.0;

double num2 = 0.0;

var input = '';

var output = '';

var operation = '';

//var hideinput=false;

var outputSize=32.0;

\_onButtonClicked(value) {

if (value == "AC") {

input = '';

output = '';

} else if (value == "A") {

if (input.isNotEmpty) {

input = input.substring(0, input.length - 1);

}

} else if (value == "=") {

if (input.isNotEmpty) {

var userinput = input;

userinput = input.replaceAll("", "");

Parser P = Parser();

Expression expression = P.parse(userinput);

ContextModel cm = ContextModel();

var finalvalue = expression.evaluate(EvaluationType.REAL, cm);

output = finalvalue.toString();

if (output.endsWith(".0")) {

output = output.substring(0, output.length - 2);

}

// input=output;

// hideinput=true;

// outputSize=50;

}

} else {

input = input + value;

}

setState(() {});

}

@override

Widget build(BuildContext context) {

return Scaffold(

body: Column(

// ignore: prefer\_const\_literals\_to\_create\_immutables

children: [

Expanded(

child: Container(

height: 200,

width: 500,// double.infinity,

color: Colors.transparent,

child: Column(

crossAxisAlignment: CrossAxisAlignment.end,

mainAxisAlignment: MainAxisAlignment.end,

// ignore: prefer\_const\_literals\_to\_create\_immutables

children: [

Text(

input,

style: TextStyle(fontSize: 48, color: Colors.white),

),

SizedBox(

height: 10,

),

Text(

output,

style: TextStyle(

fontSize: 30,

color: Colors.white.withOpacity(0.7),

),

),

SizedBox(

height: 20,

),

],

),

),

),

Row(

// ignore: prefer\_const\_literals\_to\_create\_immutables

children: [

button("AC", Colors.black, Colors.orangeAccent),

button("A", Colors.black, Colors.orangeAccent),

button("", Colors.transparent, Colors.white),

button("/", Colors.black, Colors.orangeAccent),

],

),

Row(

// ignore: prefer\_const\_literals\_to\_create\_immutables

children: [

button("7", Colors.black, Colors.white),

button("8", Colors.black, Colors.white),

button("9", Colors.black, Colors.white),

button("\*", Colors.black, Colors.orangeAccent),

],

),

Row(

// ignore: prefer\_const\_literals\_to\_create\_immutables

children: [

button("4", Colors.black, Colors.white),

button("5", Colors.black, Colors.white),

button("6", Colors.black, Colors.white),

button("-", Colors.black, Colors.orangeAccent),

],

),

Row(

// ignore: prefer\_const\_literals\_to\_create\_immutables

children: [

button("1", Colors.black, Colors.white),

button("2", Colors.black, Colors.white),

button("3", Colors.black, Colors.white),

button("+", Colors.black, Colors.orangeAccent),

],

),

Row(

// ignore: prefer\_const\_literals\_to\_create\_immutables

children: [

button("%", Colors.black, Colors.orangeAccent),

button("0", Colors.black, Colors.white),

button(".", Colors.black, Colors.white),

button("=", Colors.orangeAccent, Colors.white),

],

),

],

),

);

}

button(text, color, tcolor) {

return Expanded(

child: Container(

padding: EdgeInsets.all(10),

child: ElevatedButton(

style: ElevatedButton.styleFrom(

padding: EdgeInsets.all(10),

shape: RoundedRectangleBorder(

borderRadius: BorderRadius.circular(5),

),

// primary: color,

),

onPressed: () => \_onButtonClicked(text),

child: Text(

text,

style: TextStyle(

fontSize: 15, fontWeight: FontWeight.bold, color: tcolor),

),

),

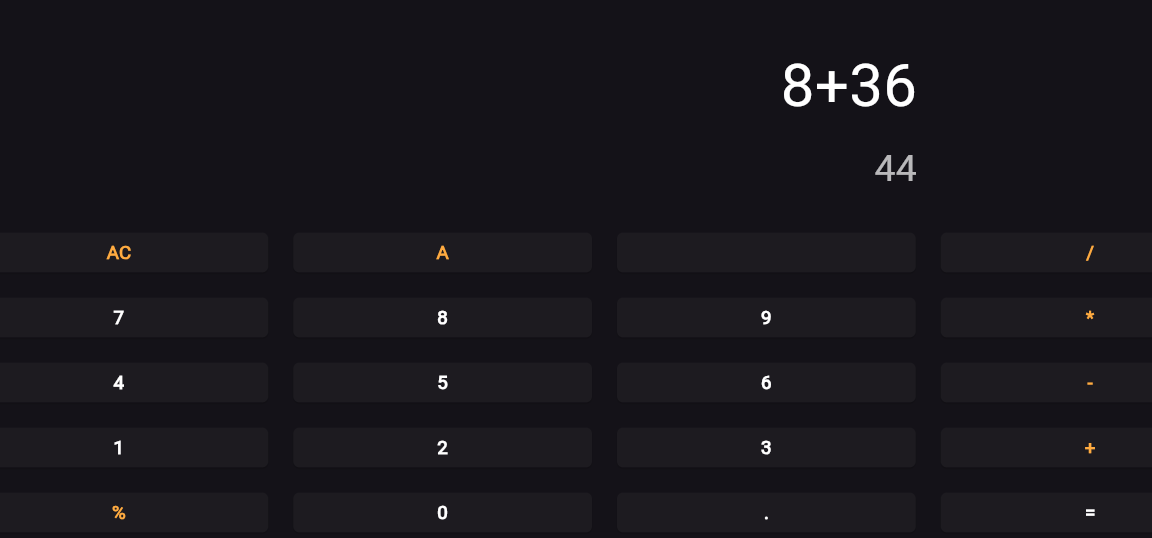
),

);

}

}

**Output:**

****

**a) 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.

ListView and GridView: 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 floating action buttons.

Text and Styling Widgets:

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

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

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

Input Widgets:

TextField: 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.

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

Button Widgets:

ElevatedButton and TextButton: Widgets for displaying buttons with different styles and customization options.

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

GestureDetector: 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.

PageRouteBuilder: A customizable widget for building page transitions and animations.

Animation Widgets:

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

AnimatedOpacity, AnimatedPositioned, AnimatedBuilder: Widgets for animating opacity, position,and custom properties respectively.

Material Design Widgets:

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

BottomNavigationBar: Provides a navigation bar at the bottom of the screen for switching between different screens or tabs.

Cupertino (iOS-style) Widgets:

CupertinoNavigationBar: A navigation bar in the iOS style.

CupertinoButton: A button widget with the iOS style.

CupertinoTextField: 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**

1. Row Layout:

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: Scaffold(

appBar: AppBar(

title: Text('Row Layout'),

),

body: Row(

mainAxisAlignment: MainAxisAlignment.spaceEvenly,

children: <Widget>[

Container(

color: Colors.red,

width: 100,

height: 100,

),

Container(

color: Colors.green,

width: 100,

height: 100,

),

Container(

color: Colors.blue,

width: 100,

height: 100,

),

],

),

),

);

}

}A blue green red and blue rectangles

Description automatically generated

Output:

2. Column Layout:

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('Column Layout'),

),

body: Column(

mainAxisAlignment: MainAxisAlignment.spaceEvenly,

children: <Widget>[

Container(

color: Colors.red,

width: 100,

height: 100,

),

Container(

color: Colors.green,

width: 100,

height: 100,

),

Container(

color: Colors.blue,

width: 100,

height: 100,

),

],

),

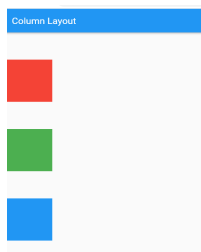
),

);

}

}

Output:



3. Stack Layout:

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('Stack Layout'),

),

body: Stack(

alignment: Alignment.center,

children: <Widget>[

Container(

color: Colors.red,

width: 200,

height: 200,

),

Container(

color: Colors.green,

width: 150,

height: 150,

),

Container(

color: Colors.blue,

width: 100,

height: 100,

),

],

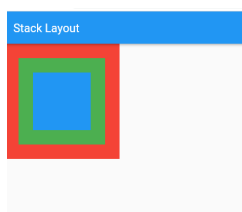
),

),

);

}}

Output:



**Project:Tic Tac Toe**

**Source code:**

import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: HomePage(),

);

}

}

class HomePage extends StatefulWidget {

@override

\_HomePageState createState() => \_HomePageState();

}

class \_HomePageState extends State<HomePage> {

bool oTurn = true;

// 1st player is O

List<String> displayElement = ['', '', '', '', '', '', '', '', ''];

int oScore = 0;

int xScore = 0;

int filledBoxes = 0;

@override

Widget build(BuildContext context) {

return Scaffold(

backgroundColor: Colors.indigo[900],

body: Column(

children: <Widget>[

Expanded(

child: Container(

child: Row(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

Padding(

padding: const EdgeInsets.all(10.0),

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

Text(

'Player X',

style: TextStyle(fontSize: 20,

fontWeight: FontWeight.bold,

color: Colors.white),

),

Text(

xScore.toString(),

style: TextStyle(fontSize: 20,color: Colors.white),

),

],

),

),

Padding(

padding: const EdgeInsets.all(10.0),

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

Text('Player O', style: TextStyle(fontSize: 20,

fontWeight: FontWeight.bold,

color: Colors.white)

),

Text(

oScore.toString(),

style: TextStyle(fontSize: 20,color: Colors.white),

),

],

),

),

],

),

),

),

Expanded(

flex: 5,

child: GridView.builder(

itemCount: 9,

gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(

//childAspectRatio: MediaQuery.of(context).size.width /

//(MediaQuery.of(context).size.height / 4),

crossAxisCount: 3

),

itemBuilder: (BuildContext context, int index) {

return GestureDetector(

onTap: () {

\_tapped(index);

},

child: Container(

decoration: BoxDecoration(

border: Border.all(color: Colors.white)),

child: Center(

child: Text(

displayElement[index],

style: TextStyle(color: Colors.white, fontSize: 35),

),

),

),

);

}),

),

Expanded(

child: Container(

child: Row(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

ElevatedButton(

style: ElevatedButton.styleFrom(

foregroundColor: Colors.white, backgroundColor: Colors.red, // foreground

),

onPressed: () { \_clearScoreBoard(); },

child: Text("Clear Score Board"),

)

],

),

))

],

),

);

}

void \_tapped(int index) {

setState(() {

if (oTurn && displayElement[index] == '') {

displayElement[index] = 'O';

filledBoxes++;

} else if (!oTurn && displayElement[index] == '') {

displayElement[index] = 'X';

filledBoxes++;

}

oTurn = !oTurn;

\_checkWinner();

});

}

void \_checkWinner() {

// Checking rows

if (displayElement[0] == displayElement[1] &&

displayElement[0] == displayElement[2] &&

displayElement[0] != '') {

\_showWinDialog(displayElement[0]);

}

if (displayElement[3] == displayElement[4] &&

displayElement[3] == displayElement[5] &&

displayElement[3] != '') {

\_showWinDialog(displayElement[3]);

}

if (displayElement[6] == displayElement[7] &&

displayElement[6] == displayElement[8] &&

displayElement[6] != '') {

\_showWinDialog(displayElement[6]);

}

// Checking Column

if (displayElement[0] == displayElement[3] &&

displayElement[0] == displayElement[6] &&

displayElement[0] != '') {

\_showWinDialog(displayElement[0]);

}

if (displayElement[1] == displayElement[4] &&

displayElement[1] == displayElement[7] &&

displayElement[1] != '') {

\_showWinDialog(displayElement[1]);

}

if (displayElement[2] == displayElement[5] &&

displayElement[2] == displayElement[8] &&

displayElement[2] != '') {

\_showWinDialog(displayElement[2]);

}

// Checking Diagonal

if (displayElement[0] == displayElement[4] &&

displayElement[0] == displayElement[8] &&

displayElement[0] != '') {

\_showWinDialog(displayElement[0]);

}

if (displayElement[2] == displayElement[4] &&

displayElement[2] == displayElement[6] &&

displayElement[2] != '') {

\_showWinDialog(displayElement[2]);

} else if (filledBoxes == 9) {

\_showDrawDialog();

}

}

void \_showWinDialog(String winner) {

showDialog(

barrierDismissible: false,

context: context,

builder: (BuildContext context) {

return AlertDialog(

title: Text("\" " + winner + " \" is Winner!!!"),

actions: [

TextButton(

style: TextButton.styleFrom(

foregroundColor: Colors.red, // foreground

),

child: Text("Play Again"),

onPressed: () {

\_clearBoard();

Navigator.of(context).pop();

},

)

],

);

});

if (winner == 'O') {

oScore++;

} else if (winner == 'X') {

xScore++;

}

}

void \_showDrawDialog() {

showDialog(

barrierDismissible: false,

context: context,

builder: (BuildContext context) {

return AlertDialog(

title: Text("Draw"),

actions: [

TextButton(

style: TextButton.styleFrom(

foregroundColor: Colors.red, // foreground

),

onPressed: () {

\_clearBoard();

Navigator.of(context).pop();

},

child: Text('Play Again'),

)

],

);

});

}

void \_clearBoard() {

setState(() {

for (int i = 0; i < 9; i++) {

displayElement[i] = '';

}

});

filledBoxes = 0;

}

void \_clearScoreBoard() {

setState(() {

xScore = 0;

oScore = 0;

for (int i = 0; i < 9; i++) {

displayElement[i] = '';

}

});

filledBoxes = 0;

}

}