VIKAS VIDYA EDUCATION TRUST'S

Lords Universal college

Department of Information Technology

CERTIFICATE

This is to certify that Mr./Ms._____of _____ class (_____Semester) has satisfactorily completed _____ Practical, in the subject of_____as a part of B.Sc. in Information Technology Program during the academic year 20__- 20___.

Place:

Date:

Subject In-charge

Co-Ordinator,
Department Computer Science

Signature of Examiner

VIKAS VIDYA EDUCATION TRUST'S

Lords Universal college

INDEX

Practical No	Aim of the Practical	Signature
1.	Program to demonstrate the features of Dart Language.	
2.	Designing the mobile app to implement different Widgets.	
3.	Designing the mobile app to implement different Layouts.	
4.	Designing the mobile app to implement Gestures.	
5.	Designing the mobile app to implement the Theming and Styling.	

Practical 1

1.Program to enter marks of 5 subject and calculate the average and total.

main.dart > main

```dart
import 'dart:io';

Run | Debug
void main() {
  print("Enter the marks of five subjects:");
  var subject1 = double.parse(stdin.readLineSync()!);
  var subject2 = double.parse(stdin.readLineSync()!);
  var subject3 = double.parse(stdin.readLineSync()!);
  var subject4 = double.parse(stdin.readLineSync()!);
  var subject5 = double.parse(stdin.readLineSync()!);

  var total = subject1 + subject2 + subject3 + subject4 + subject5;
  var average = total / 5;
  var percentage = (total / 500) * 100;
 if (average >= 90) {
    print("The Grade is: 'A'");
  } else if (average >= 80 && average < 90) {
    print("The Grade is: 'B'");
  } else if (average >= 70 && average < 80) {
    print("The Grade is: 'C'");
  } else if (average >= 60 && average < 70) {
    print("The Grade is: 'D'");
  } else {
    print("The Grade is: 'E'");
  }

  print("The Total marks are: ${total.toStringAsFixed(2)} / 500.00");
  print("The Average marks are: ${average.toStringAsFixed(2)}");
  print("The Percentage is: ${percentage.toStringAsFixed(2)}%");
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\PC\dartprogramming> dart main.dart
Enter the marks of five subjects:
45
67
89
65
67
The Grade is: 'D'
The Total marks are: 333.00 / 500.00
The Average marks are: 66.60
The Percentage is: 66.60%
PS C:\Users\PC\dartprogramming>
```

2. Program to enter 3 number and find the largest.

main.dart > main

```dart
import 'dart:io';

Run | Debug
void main() {
  // Input three numbers
  print("Enter the first number: ");
  double num1 = double.parse(stdin.readLineSync()!);

  print("Enter the second number: ");
  double num2 = double.parse(stdin.readLineSync()!);

  print("Enter the third number: ");
  double num3 = double.parse(stdin.readLineSync()!);

  // Find the largest number using conditional statements (if-else)
  double largest;

  if (num1 >= num2 && num1 >= num3) {
    largest = num1;
  } else if (num2 >= num1 && num2 >= num3) {
    largest = num2;
  } else {
    largest = num3;
  }

  // Display the largest number
  print("The largest number among $num1, $num2, and $num3 is: $largest");
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\PC\dartprogramming> dart main.dart
Enter the first number:
5
Enter the second number:
7
Enter the third number:
9
The largest number among 5.0, 7.0, and 9.0 is: 9.0
PS C:\Users\PC\dartprogramming>
```

3. Program to find a factorial of number.

main.dart > main

```dart
import 'dart:io';

Run | Debug
void main() {
  print('Enter a number:');
  int n = int.parse(stdin.readLineSync()!); // read the input from console
  int result = 1; // initialize the result variable
  for (int i = 1; i <= n; i++) { // loop from 1 to n
    result *= i; // multiply the result by i
  }
  print('Factorial of $n is $result'); // print the result
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\PC\dartprogramming> dart main.dart
Enter a number:
2
Factorial of 2 is 2
PS C:\Users\PC\dartprogramming>
```

4. Ask 2 number from user and print sum.

Welcome | main.dart

main.dart > main

```dart
import 'dart:io';

Run | Debug
void main() {
  // read number from user
  print('Enter x');
  var x = int.parse(stdin.readLineSync()!);
  print('Enter y');
  var y = int.parse(stdin.readLineSync()!);

  var output = x + y;

  print('x + y = $output');
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\PC\dartprogramming> dart main.dart
Enter x
12
Enter y
30
x + y = 42
PS C:\Users\PC\dartprogramming>
```

Practical 2

```
import 'package:flutter/material.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(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const Home(),
    );
  }
}

class Home extends StatelessWidget {
  const Home({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(title: const Text('Sample Application'),),
        body: const Text('Text1', style: TextStyle(color: Colors.red,
fontSize: 20),),
    );
  }
}
```

Note: Following is the Sample code of individual Widget to try it replace below code with the above highlighted code

1) Text

const Text('Text1', style: TextStyle(color: Colors.red, fontSize: 20),),

2) **Image**

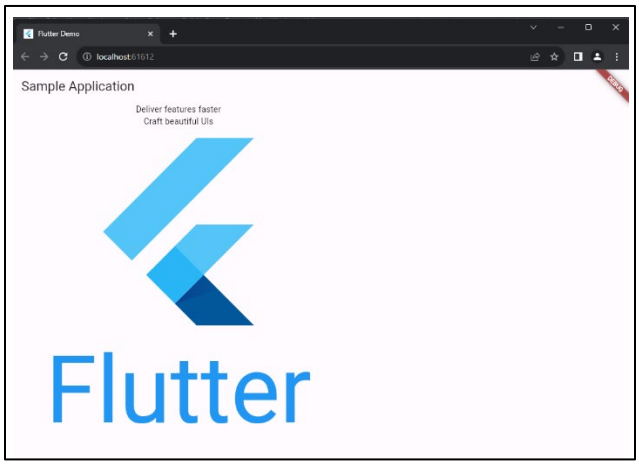
Image.asset("images/prasad.jpg", height: 150, width: 150,) ,

3) ElevatedButton

ElevatedButton(onPressed: (){}, child: Text('Submit')),

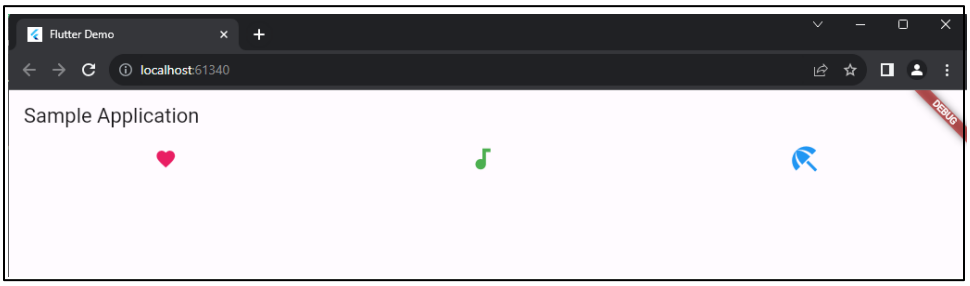
4) FlutterLogo

```
child: FlutterLogo(
  size: 300,
  textColor: Colors.blue,
  style: FlutterLogoStyle.stacked,
), //FlutterLogo
```



5) Icon

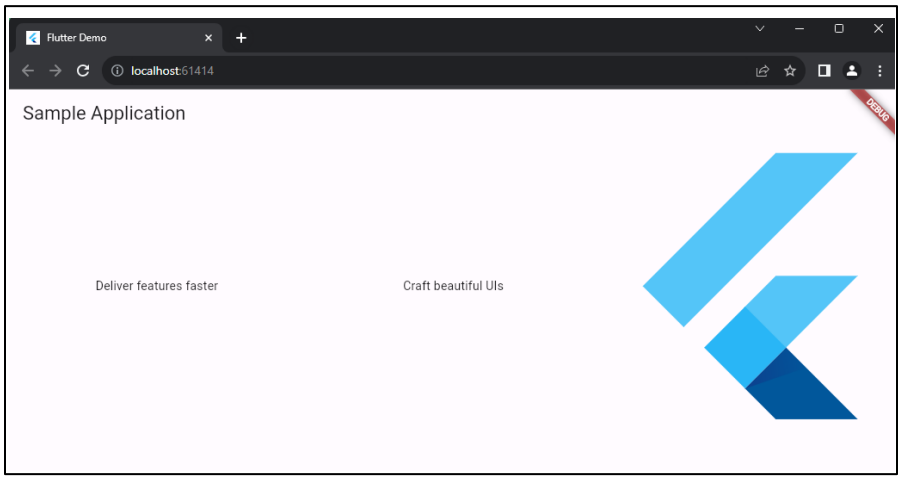
```
const Row(
  mainAxisAlignment: MainAxisAlignment.spaceAround,
  children: <Widget>[
    Icon(
      Icons.favorite,
      color: Colors.pink,
      size: 24.0,
      semanticLabel: 'Text to announce in accessibility modes',
    ),
    Icon(
      Icons.audiotrack,
      color: Colors.green,
      size: 30.0,
    ),
    Icon(
      Icons.beach_access,
      color: Colors.blue,
      size: 36.0,
    ),
  ],
)
```



6) Row

```
const Row(
  children: <Widget>[
    Expanded(
      child: Text('Deliver features faster', textAlign: TextAlign.center),
    ),
    Expanded(
      child: Text('Craft beautiful UIs', textAlign: TextAlign.center),
    ),
    Expanded(
      child: FittedBox(
        child: FlutterLogo(),
```

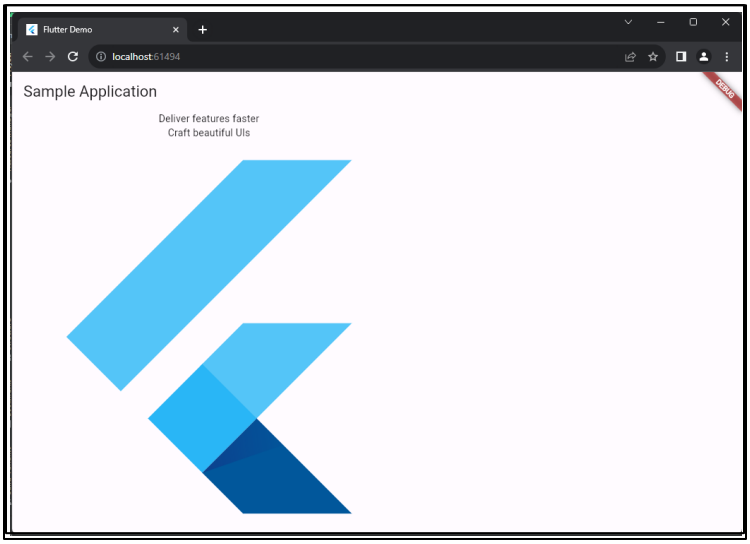
```
    ),
   ),
 ],
```




```
)
```

7) Column

```
const Column(
  children: <Widget>[
    Text('Deliver features faster'),
    Text('Craft beautiful UIs'),
    Expanded(
      child: FittedBox(
        child: FlutterLogo(),
      ),
    ),
  ],
)
```



Practical 3

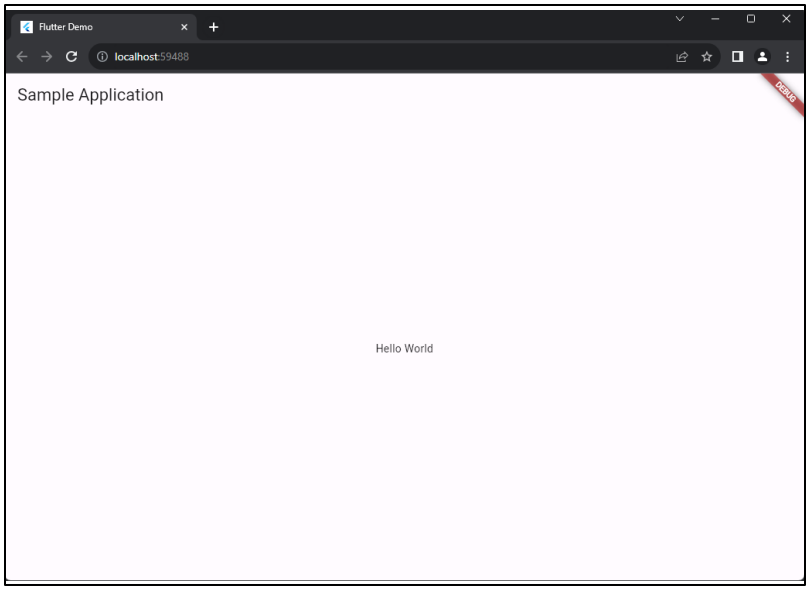
Sample Home Code

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

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override   //indicates method in subclassis intended to override a method with a same signature in its superclass
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const Home(),
    );
  }
}

class Home extends StatelessWidget {
  const Home({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Sample Application'),),
      body: const Center(
        child: Text('Hello World'),
      ),
    );
  }
}
```

Note: Code of individual Layout to try it replace below code with the above highlighted code

1) **Center**

```
const Center(
  child: Text('Hello World'),
),
```

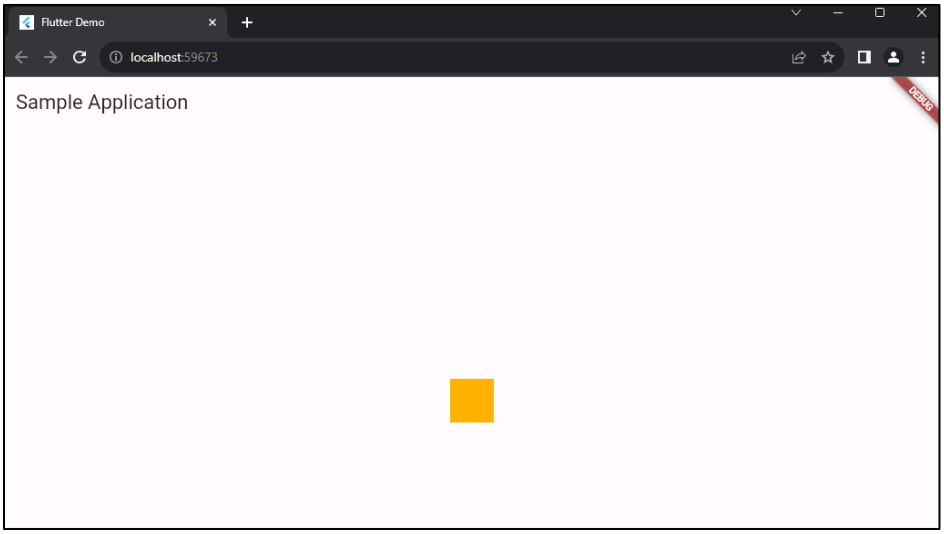


2) Padding

```
const Card(
  child: Padding(
    padding: EdgeInsets.all(16.0),
    child: Text('Hello World!'),
  ),
)
```

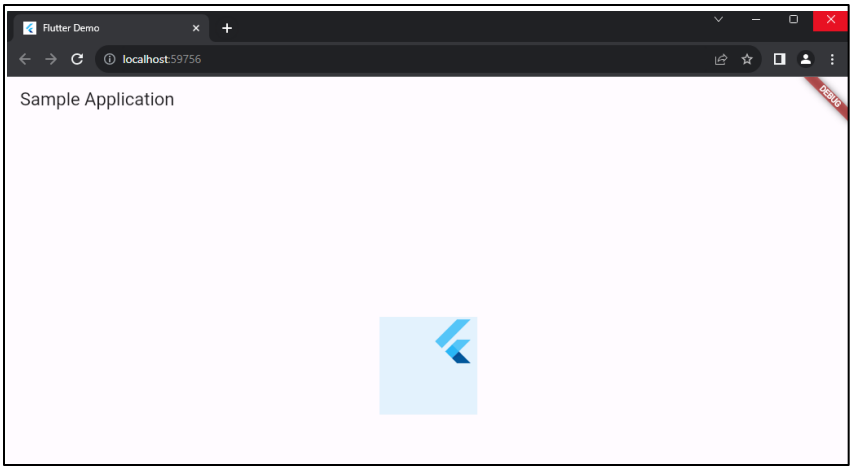
3) Container

```
Center(
  child: Container(
    margin: const EdgeInsets.all(10.0),
    color: Colors.amber[600],
    width: 48.0,
    height: 48.0,
  ),
)
```



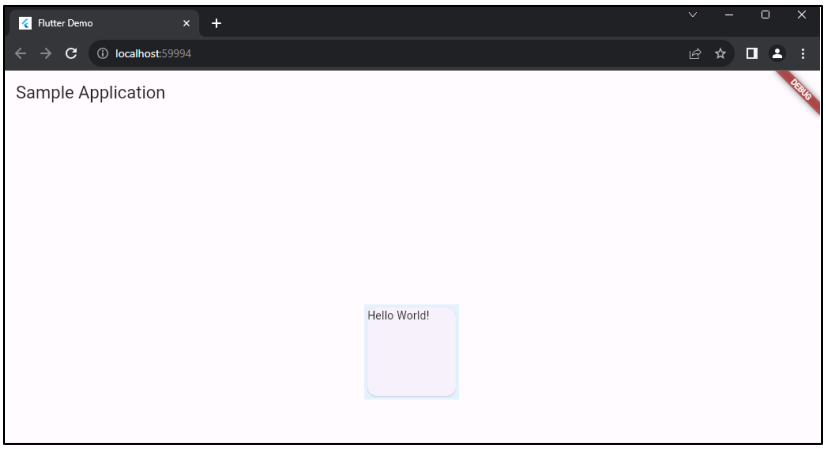
4) Align

```
Center(
  child: Container(
    height: 120.0,
    width: 120.0,
    color: Colors.blue[50],
    child: const Align(
      alignment: Alignment.topRight,
      child: FlutterLogo(
        size: 60,
      ),
    ),
  ),
)
```



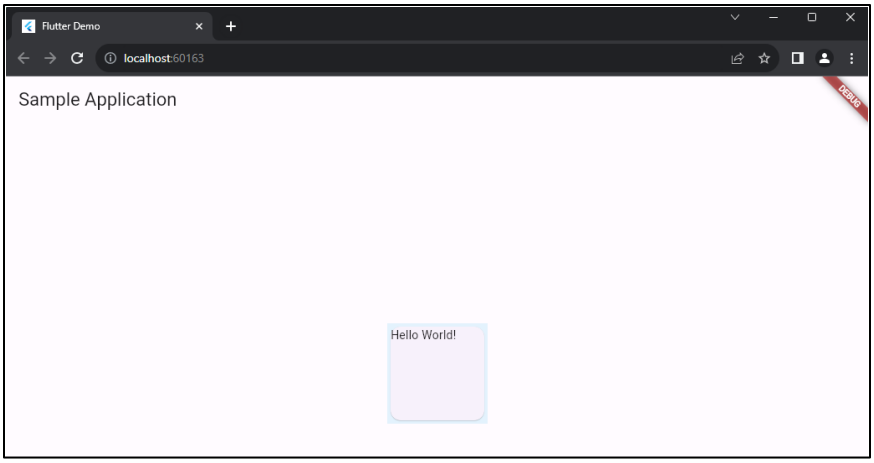
5) SizedBox

```
const SizedBox(
  width: 200.0,
  height: 300.0,
  child: Card(child: Text('Hello World!')),
)
```



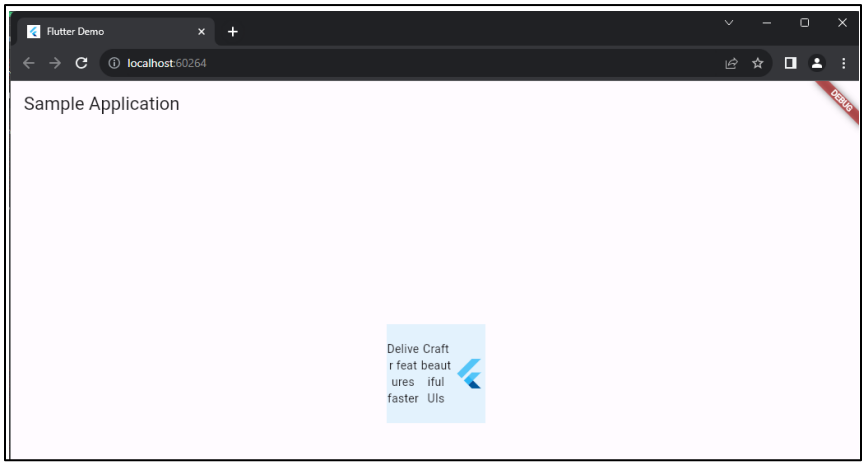
6) ConstrainedBox

```
ConstrainedBox(
  constraints: const BoxConstraints.expand(),
  child: const Card(child: Text('Hello World!')),
)
```



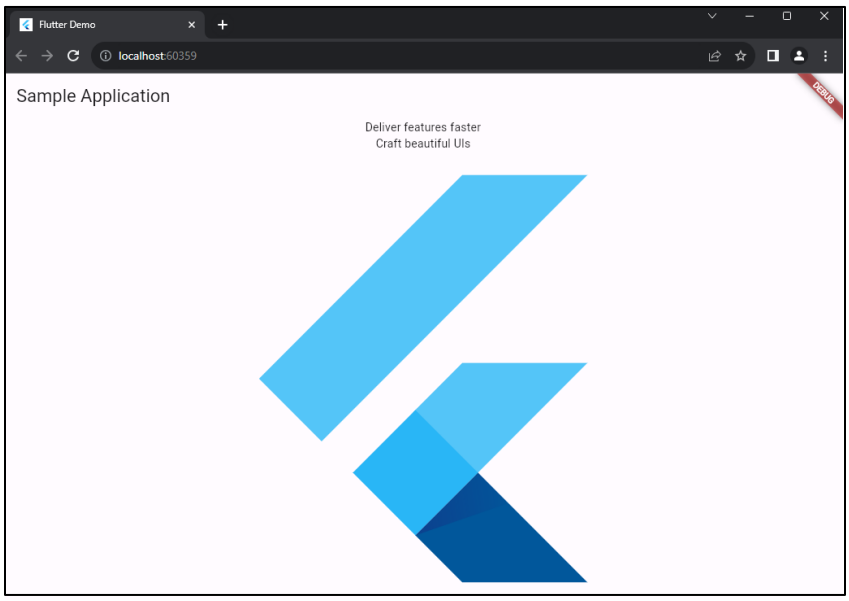
7) Row

```
const Row(
  children: <Widget>[
    Expanded(
      child: Text('Deliver features faster', textAlign: TextAlign.center),
    ),
    Expanded(
      child: Text('Craft beautiful UIs', textAlign: TextAlign.center),
    ),
    Expanded(
      child: FittedBox(
        child: FlutterLogo(),
      ),
    ),
  ],
)
```



8) Column

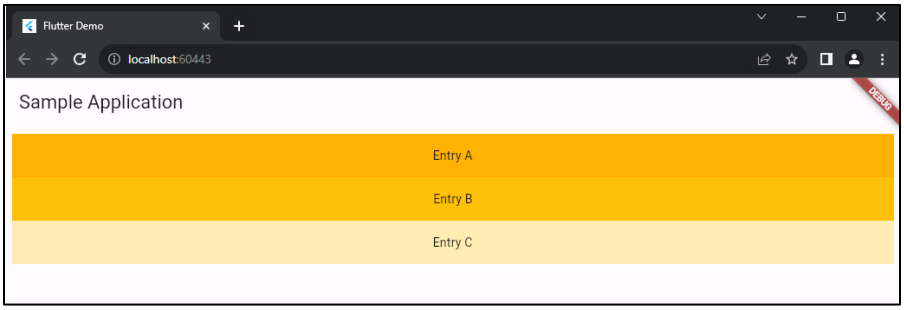
```
const Column(
  children: <Widget>[
    Text('Deliver features faster'),
    Text('Craft beautiful UIs'),
    Expanded(
      child: FittedBox(
        child: FlutterLogo(),
      ),
    ),
  ],
)
```



9) ListView

```
ListView(
  padding: const EdgeInsets.all(8),
  children: <Widget>[
    Container(
      height: 50,
      color: Colors.amber[600],
      child: const Center(child: Text('Entry A')),
    ),
    Container(
```

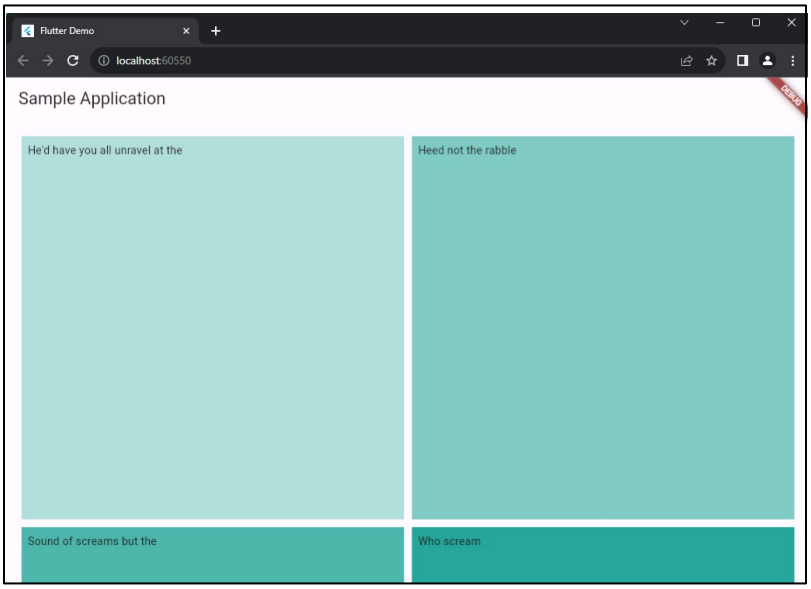
```
      height: 50,
      color: Colors.amber[500],
      child: const Center(child: Text('Entry B')),
    ),
    Container(
      height: 50,
      color: Colors.amber[100],
      child: const Center(child: Text('Entry C')),
    ),
  ],
)
```



10) GridView

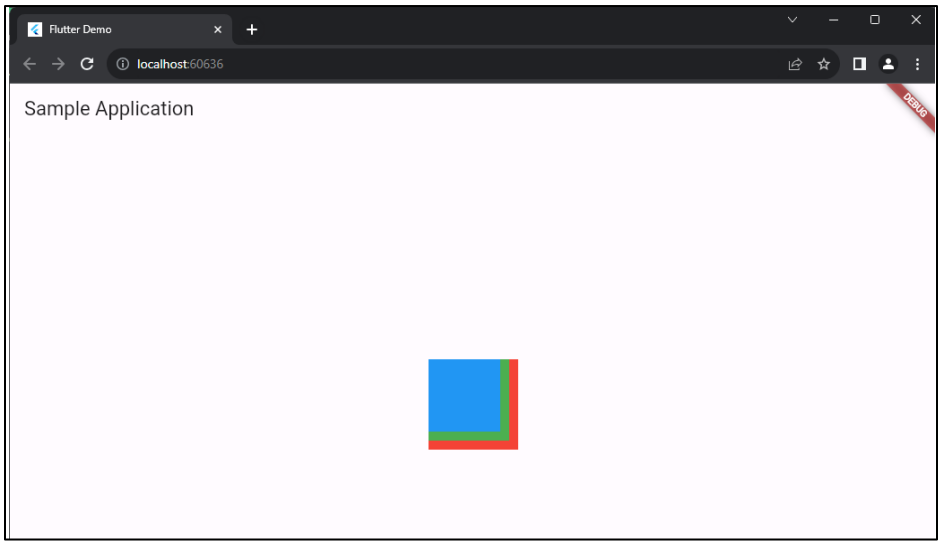
```
GridView.count(
  primary: false,
  padding: const EdgeInsets.all(20),
  crossAxisSpacing: 10,
  mainAxisSpacing: 10,
  crossAxisCount: 2,
  children: <Widget>[
    Container(
      padding: const EdgeInsets.all(8),
      color: Colors.teal[100],
      child: const Text("He'd have you all unravel at the"),
    ),
    Container(
      padding: const EdgeInsets.all(8),
      color: Colors.teal[200],
      child: const Text('Heed not the rabble'),
    ),
    Container(
      padding: const EdgeInsets.all(8),
```

```
    color: Colors.teal[300],
    child: const Text('Sound of screams but the'),
   ),
   Container(
    padding: const EdgeInsets.all(8),
    color: Colors.teal[400],
    child: const Text('Who scream'),
   ),
   Container(
    padding: const EdgeInsets.all(8),
    color: Colors.teal[500],
    child: const Text('Revolution is coming...'),
   ),
   Container(
    padding: const EdgeInsets.all(8),
    color: Colors.teal[600],
    child: const Text('Revolution, they...'),
   ),
  ],
)
```



11) Stack

```
Stack(
  children: <Widget>[
    Container(
      width: 100,
      height: 100,
      color: Colors.red,
    ),
    Container(
      width: 90,
      height: 90,
      color: Colors.green,
    ),
    Container(
      width: 80,
      height: 80,
      color: Colors.blue,
    ),
  ],
)
```



Practical 4

Introduction to Gestures

Introduction

- *Gestures* are primarily a way for a user to interact with a mobile (or any touch based device) application.
- Gestures are generally defined as any physical action / movement of a user in the intention of activating a specific control of the mobile device.
- Gestures are as simple as tapping the screen of the mobile device to more complex actions used in gaming applications.

Some of the widely used gestures

- **Tap** − Touching the surface of the device with fingertip for a short period and then releasing the fingertip.
- **Double Tap** − Tapping twice in a short time.
- **Drag** − Touching the surface of the device with fingertip and then moving the fingertip in a steady manner and then finally releasing the fingertip.
- **Flick** − Similar to dragging, but doing it in a speeder way.
- **Pinch** − Pinching the surface of the device using two fingers.
- **Spread/Zoom** − Opposite of pinching.

Panning − Touching the surface of the device with fingertip and moving it in any direction without releasing

Gesture Detector

- Flutter provides an excellent support for all type of gestures through its exclusive widget, **GestureDetector**.
- GestureDetector is a non-visual widget primarily used for detecting the user's gesture.
- To identify a gesture targeted on a widget, the widget can be placed inside GestureDetector widget.

GestureDetector will capture the gesture and dispatch multiple events based on the gesture

Gestures and the corresponding events

- Tap
  - onTapDown
  - onTapUp
  - onTap
  - onTapCancel
- Double tap
  - onDoubleTap
- Long press
  - onLongPress
- Vertical drag
  - onVerticalDragStart
  - onVerticalDragUpdate
  - onVerticalDragEnd
- Horizontal drag
  - onHorizontalDragStart
  - onHorizontalDragUpdate
  - onHorizontalDragEnd
- Pan
  - onPanStart
  - onPanUpdate

Practical 5

Introduction to Themes

Introduction

- Themes are an integral part of UI for any application.
- Themes are used to design the fonts and colors of an application to make it more presentable.
- In Flutter, the Theme widget is used to add themes to an application.
- One can use it either for a particular part of the application like buttons and navigation bar or define it in the root of the application to use it throughout the entire app.

Creating a Theme

Use the Theme widget to create a theme. In themes some of the properties that can be used are given below:

- TextTheme
- brightness
- primarycolor
- accentColor

fontFamily

Creating a Theme

```
MaterialApp(
title: title,
theme: ThemeData(
    brightness: Brightness.dark,
```

```
primaryColor: Colors.lightBlue[800],
accentColor: Colors.cyan[600],
fontFamily: 'Georgia’,
textTheme: TextTheme(
headline1: TextStyle(fontSize: 72.0, fontWeight: FontWeight.bold),
headline6: TextStyle(fontSize: 36.0, fontStyle: FontStyle.italic),
bodyText2: TextStyle(fontSize: 14.0, fontFamily: 'Hind'),
),
```

How to use it

```
Container(
color: Theme.of(context).accentColor,
child: Text(
'Hello Everyone!',
style: Theme.of(context).textTheme.headline6,
),
);
```

Themes for part of an application

To override the app-wide theme in part of an application, wrap a section of the app in a Theme widget

There are two ways to approach this:

- Creating a unique ThemeData

- Extending the parent theme

Creating unique Theme Data

If you don't want to inherit any application colors or font styles, create a ThemeData() instance and pass that to the Theme widget.

```
Theme(
  // Create a unique theme with `ThemeData`
  data: ThemeData(
    splashColor: Colors.yellow,
  ),
  child: FloatingActionButton(
    onPressed: () {},
    child: const Icon(Icons.add),
  ),
);
```

Extending the parent theme

Rather than overriding everything, it often makes sense to extend the parent theme. You can handle this by using the copyWith() method.

```
Theme(
  // Find and extend the parent theme using `copyWith`. See the next
  // section for more info on `Theme.of`.
  data: Theme.of(context).copyWith(splashColor: Colors.yellow),
  child: const FloatingActionButton(
    onPressed: null,
    child: Icon(Icons.add)  ),);
```