10. Theory:

SafeArea Widget: SafeArea is an important and useful widget in Flutter which makes UI dynamic and adaptive to a wide variety of devices. While designing the layout of widgets, we consider different types of devices and their pre-occupied constraints of screen like status bar, notches, navigation bar, etc. But new devices are being launched with different designs and in certain scenarios, your app might overlay any of those pre-occupied constraints. So, in order to make our UI adaptive and error-free, we use SafeArea widget.

In simple words, SafeArea is basically a padding widget, which adds any necessary padding to your app, based on the device it is running on. If your app's widgets are overlaying any of the system's features like notches, status bar, camera holes, or any other such features, then SafeArea would add padding around the app, as required. Internally SafeArea uses MediaQuery to check the dimensions of the display screen and includes extra padding if needed.

```
const SafeArea({
   Key key,
   bool left: true,
   bool top: true,
   bool right: true,
   bool bottom: true,
   EdgeInsets minimum: EdgeInsets.zero,
   bool maintainBottomViewPadding: false,
   @required Widget child}
)
```

Above shown is the constructor for SafeArea. You can decide whether to avoid intrusions in a particular direction by changing the boolean value to true or false.

Properties :

- **bottom :** This property is of type bool. It is true by default and setting it to false would disable *SafeArea* from adding padding to the bottom of the screen.
- **top :** This property is also of type bool and setting it to false would avoid padding at top of the screen.
- **left :** This property is of type bool and setting it to false would avoid padding at left side of the screen.
- **right** : This property is of type bool and setting it to false would avoid padding at right side of the screen.
- **minimum :** This property is of type *EdgeInsets*. You can specify the minimum padding to be added using this property.
- **maintainBottomViewPadding :** This property is of type bool and it specifies whether *SafeArea* should maintain the *viewPadding* instead of *padding.* For instance, if you are using an on-screen keyboard with *SafeArea* , the the padding can be maintained below the obstruction rather than being consumed.

AppBar:

AppBar is usually the topmost component of the app (or sometimes the bottom-most), it contains the toolbar and some other common action buttons. As all the components in a flutter application is a widget or a combination of widgets. So AppBar is also a built-in class or widget in flutter which gives the functionality of the AppBar out of the box. The AppBar widget is based on Material Design and much of the information is already provided by other classes like MediaQuery, Scaffold as to where the content of the AppBar should be placed. Though the AppBar class is very flexible and can be easily customized, we can also use SilverAppBar widget which gives scrollable functionality to the app bar. Or we can create our own custom app bar from scratch.

Constructor of AppBar class:

```
AppBar({

     Key key,

     Widget leading,

     bool automaticallyImplyLeading: true,

     Widget title,

     List<Widget> actions,

     double elevation,

     Color shadowColor,

     ShapeBorder shape,

     Color backgroundColor,

     Brightness brightness,

     IconThemeData iconTheme,

     IconThemeData actionsIconTheme,

     TextTheme textTheme,

     ...}
```

)

Key Properties of Appbar Widget:

- **actions:** This property takes in a list of widgets as a parameter to be displayed after the title if the *AppBar* is a row.
- **title:** This property usually takes in the main widget as a parameter to be displayed in the AppBar.
- **backgroundColor:** This property is used to add colors to the background of the *Appbar.*
- **elevation:** This property is used to set the z-coordinate at which to place this app bar relative to its parent.
- **shape:** This property is used to give shape to the *Appbar* and manage its shadow.

Row and Column Widget:

Row and Column are the two most important and powerful widgets in Flutter. These widgets let you align children horizontally and vertically as per the requirement. As we know that when we design any UI(User Interface) in flutter, we need to arrange its content in the Row and Column manner so these Row and Column widgets are required when designing UI.

Constructor Of Column Class:

```
Column({
        Key key,
        MainAxisAlignment mainAxisAlignment: MainAxisAlignment.start,
        MainAxisSize mainAxisSize: MainAxisSize.max,
        CrossAxisAlignment crossAxisAlignment: CrossAxisAlignment.center,
        TextDirection textDirection,
        VerticalDirection verticalDirection: VerticalDirection.down,
        TextBaseline textBaseline,
        List<Widget> children: const <Widget>[]}
)
```

Constructor Of Row Class:

```
Row({
        Key key,
        MainAxisAlignment mainAxisAlignment: MainAxisAlignment.start,
        MainAxisSize mainAxisSize: MainAxisSize.max,
        CrossAxisAlignment crossAxisAlignment: CrossAxisAlignment.center,
        TextDirection textDirection,
        VerticalDirection verticalDirection: VerticalDirection.down,
        TextBaseline textBaseline: TextBaseline.alphabetic,
        List<Widget> children: const <Widget>[]}
)
```

Properties of Row and Column Widgets:

- **children:** This property takes in *List<Widget>,* that is a list of widgets to display inside the *Row* or the *Column* widget.
- **clipBehaviour:** This property holds Clip class as the object to decide whether the content on the *Row* or *Column* should be clipped or not.
- **crossAxisAlignment:** The *crossAxisAlignment* takes in *CrossAxisAlignment enum* as the object to how the children's widgets should be places in *crossAxisAlignment.* For *Row* it is vertical and for *Column* it is horizontal.

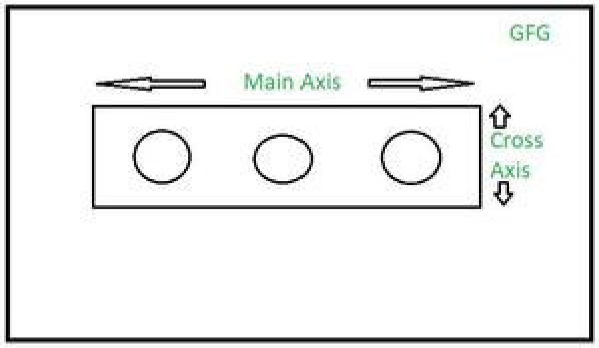
- **direction:** This property holds as the *Axis enum* object to decide the direction used in the main axis. For *Row* and *Column*, it is fixed.
- **mainAxisAlignment:** This property takes in *MainAxisAlignment enum* as the object to decide how the children widgets should be place in *mainAxisAlignment*. For *Row* it is horizontal and for *Column* it is vertical.
- **mainAxisSize:** This property decides the size of main-axis by taking in *MainAxisSize enum* as the object.
- **runtimeType:** This property tells the run-time type of the *Row* or *Column* widget.
- **textBaseline:** This property is responsible for the alignment of the text in the *Row* or *Column* widget with respect to a baseline.
- **textDirection:** This property controls the text direction of the *Row* or *Column* widget, which can either be from left-to-right (by default) or right-to-left.
- **verticalDirection:** This property takes in *VerticalDirection enum* as the object to determine the order in which the children should be layered.

Row:

It creates a horizontal array of children.

Alignment Properties:

We can align content as per our choice by using mainAxisAlignment and crossAxisAlignment. Row's mainAxis is horizontal and cross Axis to Row's main Axis is vertical. We can align children horizontally using MainAxisAlignment and vertically using CrossAxisAlignment in that row. Apart from these mainAxisAlignment and crossAxisAlignment, we also have some other properties like mainAxisSize,textDirection,verticalDirection etc.



We can align the content by using the following properties:

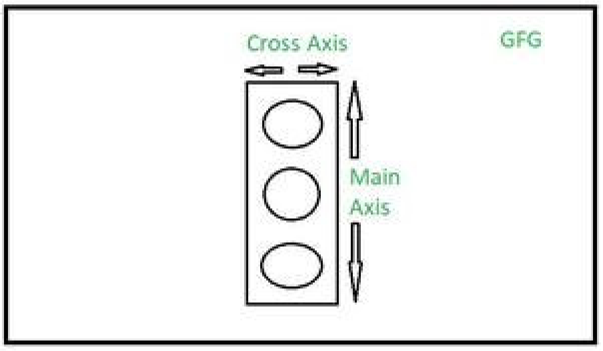
- *start* : Place the children from the starting of the row.
- *end* : Place the children at the end of the row.
- center : Place the children at the center of the row.
- *spaceBetween* : Place the space evenly between the children.
- *spaceAround* : Place the space evenly between the children and also half of that space before and after the first and last child.
- *spaceEvenly* : Place the space evenly between the children and also before and after the first and last child.

Column:

It creates a vertical array of children.

Alignment Properties:

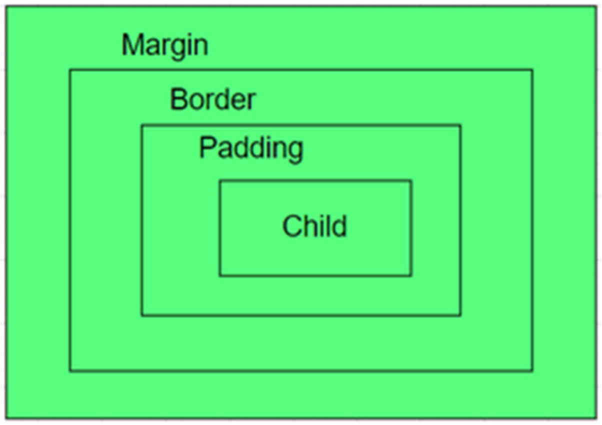
In this also we have mainAxisAlignment and crossAxisAlignment. In column, children are aligned from top to bottom. Main Axis is vertical and the Cross Axis is horizontal. MainAxisAlignment aligns its children vertically and CrossAxisAlignment aligns horizontally in that Column.



We can align the content by using the same properties as discussed above in Row (start, end,spaceBetween,spaceAround,spaceEvenly).

Container Widget:

Container class in flutter is a convenience widget that combines common painting, positioning, and sizing of widgets. A Container class can be used to store one or more widgets and position it on the screen according to our convenience. Basically a container is like a box to store contents. A basic container element that stores a widget has a margin, which separates the present container with other contents. The total container can be given a border of different shapes, for example, rounded rectangles, etc. A container surrounds its child with padding and then applies additional constraints to the padded extent (incorporating the width and height as constraints, if either is non-null).



The constructor of Container Class:

Syntax:

```
Container({
        Key key,
       AlignmentGeometry alignment,
       EdgeInsetsGeometry padding,
       Color color,
       Decoration decoration,
       Decoration foregroundDecoration,
       double width,
       double height,
       BoxConstraints constraints,
       EdgeInsetsGeometry margin,
       Matrix4 transform,
       Widget child,
       Clip clipBehavior: Clip.none}
```

);

Properties of Container Class:

1. **child:** Container widget has a property 'child:' which stores its children. The child class can be any widget.

2. color: The color property sets the background color of the entire container. Now we can visualize the position of the container using a background color.

3. height and width: By default, a container class takes the space that is required by the child. We can also specify height and width to the container based on our requirements.

4. margin: The margin is used to create an empty space around the container. Observe the white space around the container. Here EdgeInsets.geometry is used to set the margin .all() indicates that margin is present in all four directions equally.

5. padding: The padding is used to give space form the border of the container form its children. Observe the space between the border and the text.

6. alignment: The alignment is used to position the child within the container. We can align in different ways: bottom, bottom center, left, right, etc. here the child is aligned to the bottom center.

7. decoration: The decoration property is used to decorate the box(e.g. give a border). This paints behind the child. Whereas foregroundDecoration paints in front of a child. Let us give a border to the container. But, both color and border color cannot be given.

8. transform: This property of container helps us to rotate the container. We can rotate the container in any axis, here we are rotating in the z-axis.

9. constraints: When we want to give additional constraints to the child, we can use this property.

10. clipBehaviour : This property takes in Clip enum as the object. This decides whether the content inside the container will be clipped or not.

11. foregroundDecoration: This parameter holds Decoration class as the object. It controls the decoration in front of the Container widget.

Button Widgets:

a) Flat Button:

This is the most common type. With not many customizations, It has consisted of a simple text. There have two essential properties, child and onPressed(). Then you can decorate it using its attributes(colour, text size etc.).

b) Raised Button:

It's also like the Flat button, but It has an elevation which is changing the size of the button on the z-axis when pressing it. In Flat buttons, there has no elevation feature. Also, there have two callback functions(onPressed() and onLongPress()). We can customize the attributes as in Flat buttons.

c) Floating Action Button:

This is a special kind of button which contains under the Scaffold class. This is placed and fixed at the right bottom corner of the screen. We can use it for sharing, refreshing, adding the content.
There have two types of Floating Action Buttons.

Drop-down Button:

When you want to select an option from a few options, We can use this widget.

Icon Button:

This is another type of button which is consisting of an icon than text. When we touch on the icon it will respond according to its functionality.

RichText Widget:

The RichText widget is used to display text that uses various different styles. The displayed text is described using a tree of TextSpan objects, each of which has its own associated style that is used for that subtree. Depending on the layout constraints the text might break across multiple lines or might all be displayed on the same line.

Constructors:
Syntax:

```
RichText({
        Key key,
        @required InlineSpan text,
        TextAlign textAlign: TextAlign.start,
        TextDirection textDirection,
        bool softWrap: true,
        TextOverflow overflow:
        TextOverflow.clip,
        double textScaleFactor: 1.0,
        int maxLines,
        Locale locale,
        StrutStyle strutStyle,
        TextWidthBasis textWidthBasis: TextWidthBasis.parent,
        TextHeightBehavior textHeightBehavior,}
)
```

Properties:

- **children:** The widgets below this widget in the tree.
- **hashCode:** The hash code for this object.
- **key:** Controls how one widget replaces another widget in the tree.
- **runtimeType:** A representation of the runtime type of the object.
- **text:** The text to display in this widget.
- **textAlign:** How the text should be aligned horizontally.
- **local:** This property takes in *Locale* class as the object. It controls the font used for the text depending on the language used.
- **maxLines:** The *maxLines* property takes in an int value as the object. It controls the maximum number of the line that can be there for the text to expand and wrap.
- **overflow:** *TextOverflow* enum is the object given to its class it controls the text in case of overflow.
- **softWrap:** This property takes in a *boolean* value as the object. If it is set to false the gulphs in the text become wider.
- **textDirection:** This property takes in *TextDirection* class as the object to decide the direction of the text. It can be either from left-to-right or right-to-left.
- **textHightBehaviour:** *TextHeightBehavior* class is the object given to this property. It controls how the text will be highlighted.
- **textScaleFactor:** This property taken in a *double* value as the object to determine the relative size of the font.
- **textWidthBasis:** *TextWidthBasis* enum is the object of this property. It controls the width of a single line of text is being measured.

11. Installation Steps / Performance Steps and Results –

Q1	Use SafeArea Widget in flutter App development

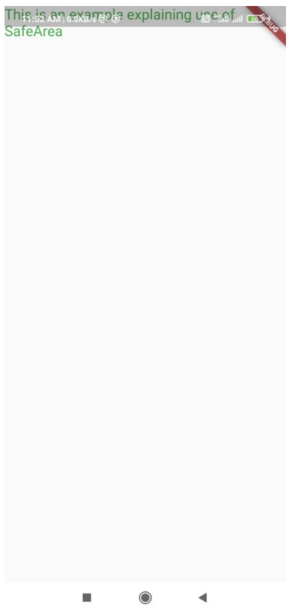
Source code:

Without SafeArea :

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

//This example explains view of scattered App without using SafeArea
void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        body: Text(
          'This is an example explaining use of SafeArea',
          style: TextStyle(color: Colors.green, fontSize: 20),
        ),
      ),
    );
  }
}
```

In the above code, we have not used SafeArea and as a result Text is printed on the status bar. Notice that, we have not used App Bar here, because App Bar automatically calculates the values and adds the required padding.

Output:



This is an example explaining use of SafeArea

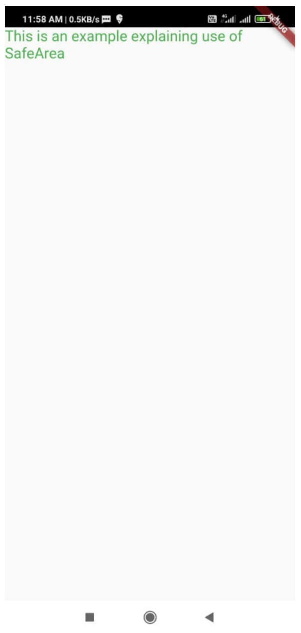
With SafeArea :

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

//This example explains effect of SafeArea widget on view of App
void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: SafeArea(
        top: true,
        child: Scaffold(
          body: Text(
            'This is an example explaining use of SafeArea',
            style: TextStyle(color: Colors.green, fontSize: 20),
          ),
        ),
      ),
    );
  }
}
```

In the above code, we have used SafeArea and we can see that automatically padding is added around the text and is not covering the status bar.

Note: It is not necessary to use SafeArea above Scaffold. You can use it above any widget, which you think can scatter or create obstruction on screen elements.

Output:



This is an example explaining use of
SafeArea

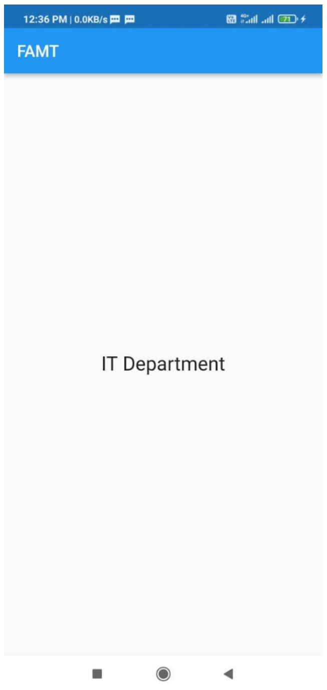
Q2	Use AppBar Widget in flutter App development

Source code:

Example 1:

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

//Use of AppBar Example 1
void main() {
  runApp(MaterialApp(
    home: Scaffold(
      appBar: AppBar(
        title: Text('FAMT'),
      ), //AppBar
      body: const Center(
        child: Text(
          'IT Department',
          style: TextStyle(fontSize: 24),
        ), //Text
      ), // center
    ), //Scaffold
    debugShowCheckedModeBanner: false, //Removing Debug Banner
  )); //MaterialApp
}
```

Output:



Explanation:

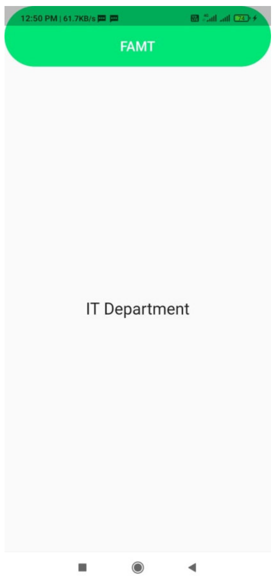
First, we have imported the material.dart file as the AppBar widget utilizes it. Then we have our main function calling runApp. At top, we have MaterialApp widget followed by Scaffold. The MaterialApp widget provided Style to AppBar and the Scaffold widget by default places the AppBar widget at the top of the screen. This is just the bare basic out of the box AppBar provided by flutter. This AppBar is utilizing only the title property of the AppBar class, which takes in the main widget to be displayed in the AppBar. In this case, it is a Text widget.

In the body, we have a child text widget inside the center widget displaying the text 'IT Department', with a font size of 24. In the end, the debug banner has been disabled

Example 2:

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

//Use of AppBar Example 2
void main() {
  runApp(MaterialApp(
    home: Scaffold(
      appBar: AppBar(
        title: Text("FAMT"),
        titleSpacing: 00.0,
        centerTitle: true,
        toolbarHeight: 60.2,
        shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(360)),
        elevation: 0.00,
        backgroundColor: Colors.greenAccent[400],
      ), //AppBar
      body: const Center(
        child: Text(
          'IT Department',
          style: TextStyle(fontSize: 24),
        ), //Text
      ), //Center
    ), //Scaffold
    debugShowCheckedModeBanner: false, //Removing Debug Banner
  )); //MaterialApp
}
```

Output:



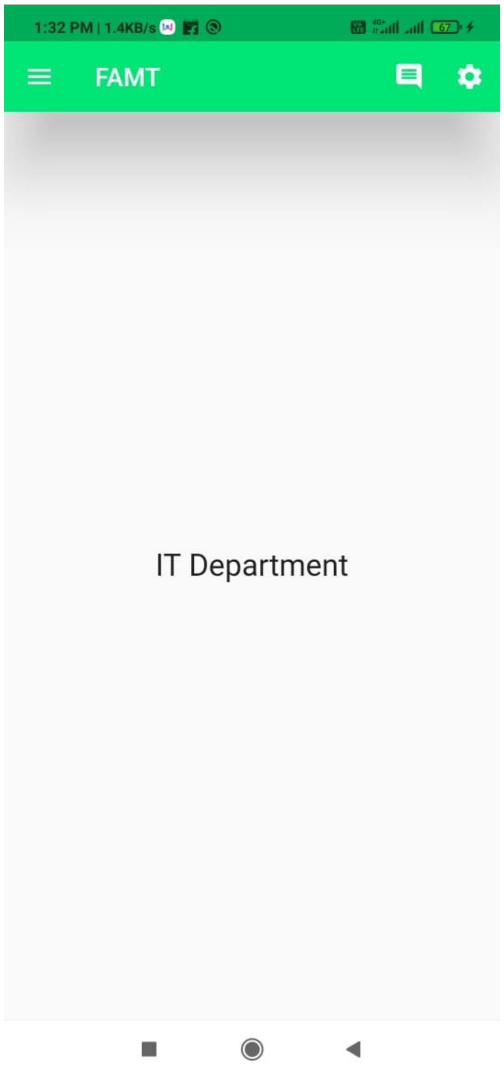
IT Department

Explanation:
Here the AppBar widget is utilizing seven properties in total. It starts with the title 'FAMT. The second is the titlespacing which takes in double as a parameter and in this case, it is set to 00.0 to keep text close together. The third property is centerTitle which takes in boolean as a parameter and is set to true here. The fourth property is toolbarHeight which also takes in double as a parameter. This property provides a shadow underneath the AppBar which in turn makes it look elevated. The fifth property is shape it is utilized to give a different shape to the AppBar by modifying the border of the AppBar. The sixth property is elevation, it defines the z-coordinates at which the AppBar is to be placed with respect to its parent. It also takes in double as a parameter. And the last is the backgroundColor which controls the background color of the AppBar, in this case, we have the signature IT Department greenAccect.

Example 3:

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

void main() {
  runApp(MaterialApp(
    home: Scaffold(
      appBar: AppBar(
        title: Text("FAMT"),
        actions: <Widget>[
          IconButton(
            icon: Icon(Icons.comment),
            tooltip: 'Comment Icon',
            onPressed: () {},
          ), //IconButton
          IconButton(
            icon: Icon(Icons.settings),
            tooltip: 'Setting Icon',
            onPressed: () {},
          ), //IconButton
        ], //<Widget>[]
        backgroundColor: Colors.greenAccent[400],
        elevation: 50.0,
        leading: IconButton(
          icon: Icon(Icons.menu),
          tooltip: 'Menu Icon',
          onPressed: () {},
        ), //IconButton
        brightness: Brightness.dark,
      ), //AppBar
      body: const Center(
        child: Text(
          "IT Department",
          style: TextStyle(fontSize: 24),
        ), //Text
      ), //Center
    ), //Scaffold
    debugShowCheckedModeBanner: false, //Removing Debug Banner
  )); //MaterialApp
}
```

Output:



IT Department

Explanation:

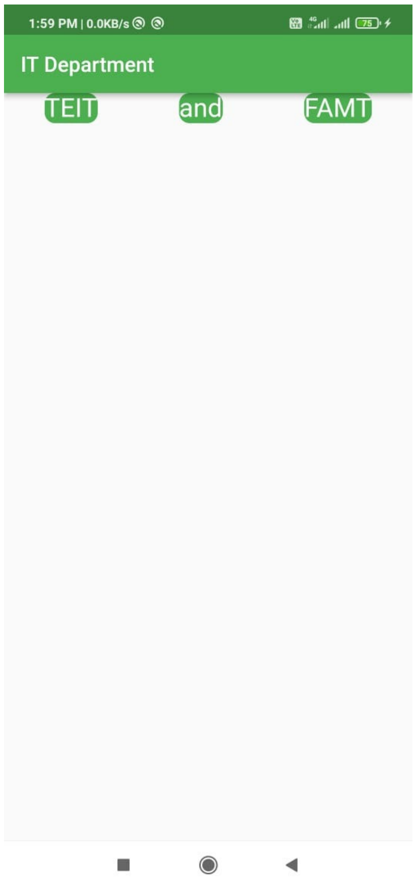
Here we can see that in addition to the title on the app Bar we have three more icons on the AppBar, one on the left and two on the right of the title. This AppBar widget starts with the usual title property which is taking Text widget as a parameter. The title is followed by the action property which takes in a list of widgets as a parameter to be displayed after the title if the AppBar is a row. In this case, we can see the two icons namely the comment and setting. These two icons are IconsButton widgets, utilizing three properties namely icon, tooltip, and onPressed function. The onPressed function is not specified in any of the IconButton so it is null. The icon property takes in string as a parameter which is the name of the specific icon. The tooltip property also takes in a string as the parameter which is displays in a floating label, while hovering with the mouse or on the long-press. In the first IconButton we have Icons.comment & Comment Icon and in the second IconButton we have Icons.setting & Setting Icon as the parameter for the icon and tooltip respectively. Now, all this is followed but the backgroundColor and elevation are set to Colors.greenAccent[400] and 50.0 respectively. After that, we have the leading property which takes in a widget as a parameter, to be displayed before the title in the AppBar. In this case, the leading is also a IconButton, which displays a menu icon. The onPressed property is not mentioned and the tooltip property is given a parameter of string ‘Menu Icon’. And the body is similar to the first and second examples.

Q3	Use Row Widget in flutter App development

Source code:

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

//Example for Row Widget
void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'FAMT',
      theme: ThemeData(
        primarySwatch: Colors.green,
      ),
      home: MyHomePage(),
      debugShowCheckedModeBanner: false,
    );
  }
}

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

class _MyHomePageState extends State<MyHomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("IT Department"),
      ),
      body: Row(
          mainAxisAlignment: MainAxisAlignment.spaceAround,
          children: <Widget>[
            Container(
              decoration: BoxDecoration(
                  borderRadius: BorderRadius.circular(10), color: Colors.green),
              child: Text(
                "TEIT",
                style: TextStyle(color: Colors.white, fontSize: 25),
              ),
            ),
            Container(
              decoration: BoxDecoration(
                  borderRadius: BorderRadius.circular(10), color: Colors.green),
              child: Text(
                "and",
                style: TextStyle(color: Colors.white, fontSize: 25),
              ),
            ),
            Container(
              decoration: BoxDecoration(
                  borderRadius: BorderRadius.circular(10), color: Colors.green),
              child: Text(
                "FAMT",
                style: TextStyle(color: Colors.white, fontSize: 25),
              ),
            )
          ]),
    );
  }
}
```

Output:

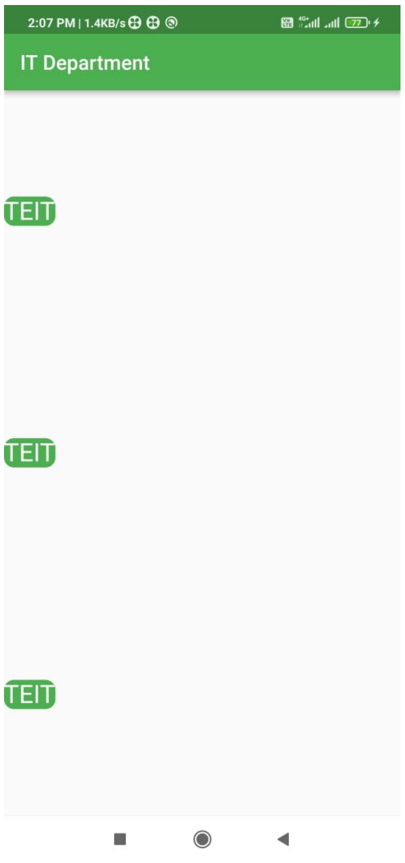


Q4	Use Column Widget in flutter App development

Source code:

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

//Example for Column Widget
void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'FAMT',
      theme: ThemeData(
        primarySwatch: Colors.green,
      ),
      home: MyHomePage(),
      debugShowCheckedModeBanner: false,
    );
  }
}

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

class _MyHomePageState extends State<MyHomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("IT Department"),
      ),
      body: Column(
          mainAxisAlignment: MainAxisAlignment.spaceAround,
          children: <Widget>[
            Container(
              decoration: BoxDecoration(
                  borderRadius: BorderRadius.circular(10), color: Colors.green),
              child: Text(
                "TEIT",
                style: TextStyle(color: Colors.white, fontSize: 25),
              ),
            ),
            Container(
              decoration: BoxDecoration(
                  borderRadius: BorderRadius.circular(10), color: Colors.green),
              child: Text(
                "TEIT",
                style: TextStyle(color: Colors.white, fontSize: 25),
              ),
            ),
            Container(
              decoration: BoxDecoration(
                  borderRadius: BorderRadius.circular(10), color: Colors.green),
              child: Text(
                "TEIT",
                style: TextStyle(color: Colors.white, fontSize: 25),
              ),
            )
          ]),
    );
  }
}
```

Output:

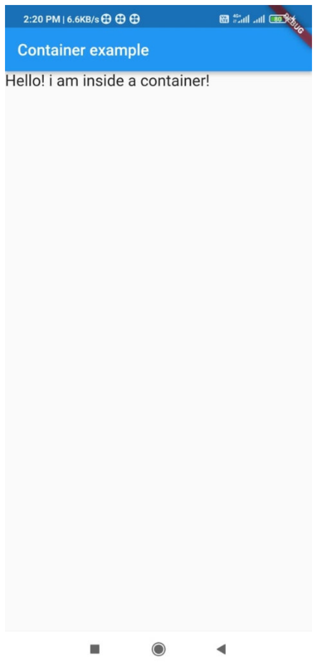


Q.5	Use Container Widget with child property in flutter App development

Source code:

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

//This Example is for demonstrating child property of Container
void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text("Container example"),
        ),
        body: Container(
          child: Text("Hello! i am inside a container!",
              style: TextStyle(fontSize: 20)),
        ),
      ),
    );
  }
}
```

Output:



Container example

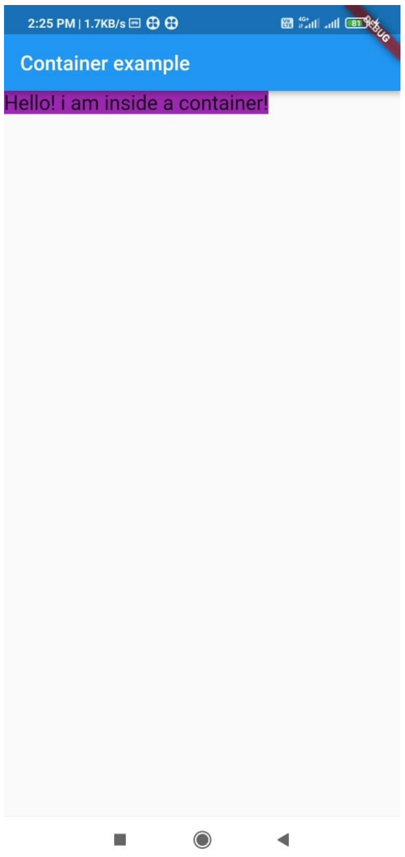
Hello! i am inside a container!

Q6	Use Container Widget with color property in flutter App development

Source code:

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

//This Example is for demonstrating color property of Container
void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text("Container example"),
        ),
        body: Container(
          color: Colors.purple,
          child: Text("Hello! i am inside a container!",
              style: TextStyle(fontSize: 20)),
        ),
      ),
    );
  }
}
```

Output:



Container example

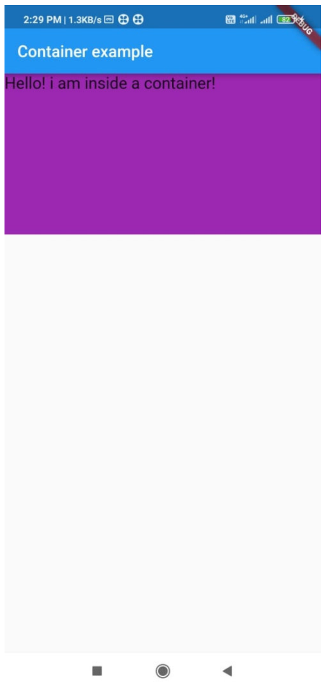
Hello! i am inside a container!

Q.7	Use Container Widget with height and width property in flutter App development

Source code:

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

//This Example is for demonstrating Height and Width property of Container
void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text("Container example"),
        ),
        body: Container(
          height: 200,
          width: double.infinity,
          color: Colors.purple,
          child: Text("Hello! i am inside a container!",
              style: TextStyle(fontSize: 20)),
        ),
      ),
    );
  }
}
```

Output:



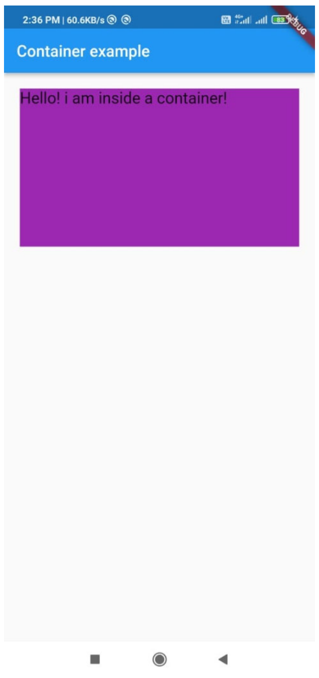
Hello! i am inside a container!

Q.8	Use Container Widget with margin property in flutter App development

Source code:

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

//This Example is for demonstrating margin property of Container
void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text("Container example"),
        ),
        body: Container(
          height: 200,
          width: double.infinity,
          color: Colors.purple,
          margin: EdgeInsets.all(20),
          child: Text("Hello! i am inside a container!",
              style: TextStyle(fontSize: 20)),
        ),
      ),
    );
  }
}
```

Output:



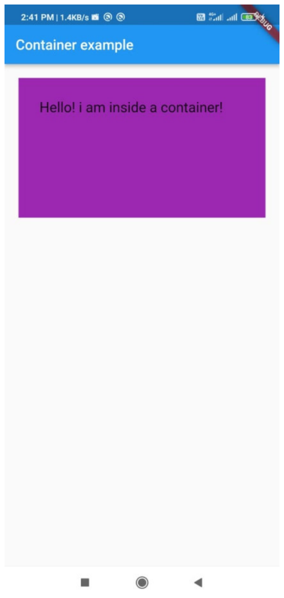
Hello! i am inside a container!

Q.9	Use Container Widget with padding property in flutter App development

Source code:

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

//This Example is for demonstrating padding property of Container
void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text("Container example"),
        ),
        body: Container(
          height: 200,
          width: double.infinity,
          color: Colors.purple,
          margin: EdgeInsets.all(20),
          padding: EdgeInsets.all(30),
          child: Text("Hello! i am inside a container!",
              style: TextStyle(fontSize: 20)),
        ),
      ),
    );
  }
}
```

Output:

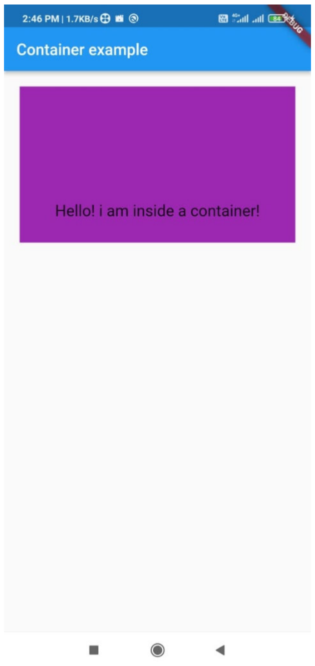


Q.10	Use Container Widget with alignment property in flutter App development

Source code:

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

//This Example is for demonstrating alignment property of Container
void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text("Container example"),
        ),
        body: Container(
          height: 200,
          width: double.infinity,
          color: Colors.purple,
          alignment: Alignment.bottomCenter,
          margin: EdgeInsets.all(20),
          padding: EdgeInsets.all(30),
          child: Text("Hello! i am inside a container!",
              style: TextStyle(fontSize: 20)),
        ),
      ),
    );
  }
}
```

Output:

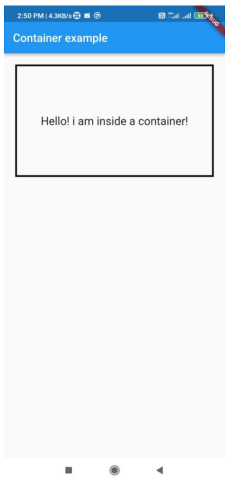


Q.11	Use Container Widget with decoration property in flutter App development

Source code:

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

//This Example is for demonstrating decoration property of Container
void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text("Container example"),
        ),
        body: Container(
          height: 200,
          width: double.infinity,
          //color: Colors.purple,
          alignment: Alignment.center,
          margin: EdgeInsets.all(20),
          padding: EdgeInsets.all(30),
          decoration: BoxDecoration(
            border: Border.all(color: Colors.black, width: 3),
          ),
          child: Text("Hello! i am inside a container!",
              style: TextStyle(fontSize: 20)),
        ),
      ),
    );
  }
}
```

Output:



Container example

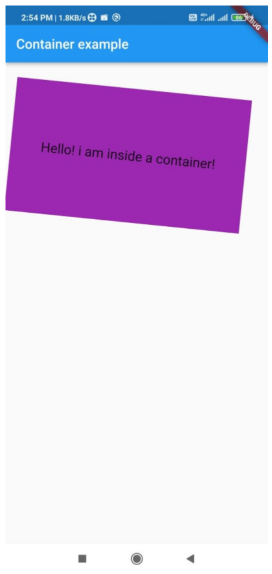
Hello! i am inside a container!

Q.12	Use Container Widget with transform property in flutter App development

Source code:

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

//This Example is for demonstrating transform property of Container
void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text("Container example"),
        ),
        body: Container(
          height: 200,
          width: double.infinity,
          color: Colors.purple,
          alignment: Alignment.center,
          margin: EdgeInsets.all(20),
          padding: EdgeInsets.all(30),
          transform: Matrix4.rotationZ(0.1),
          child: Text("Hello! i am inside a container!",
              style: TextStyle(fontSize: 20)),
        ),
      ),
    );
  }
}
```

Output:



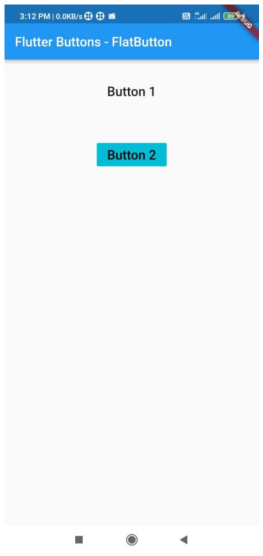
Hello! i am inside a container!

Q.13	Use FlatButton Widget in flutter App development

Source code:

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

//This example explains FlatButten Widget
void main() {
  runApp(MyApp());
}

class MyApp extends StatefulWidget {
  @override
  _MyAppState createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
          appBar: AppBar(
            title: Text('Flutter Buttons - FlatButton'),
          ),
          body: Center(
              child: Column(children: <Widget>[
            Container(
              margin: EdgeInsets.all(25),
              child: FlatButton(
                child: Text(
                  'Button 1',
                  style: TextStyle(fontSize: 20.0),
                ),
                onPressed: () {},
              ),
            ),
            Container(
              margin: EdgeInsets.all(25),
              child: FlatButton(
                child: Text(
                  'Button 2',
                  style: TextStyle(fontSize: 20.0),
                ),
                color: Colors.cyan,
                textColor: Colors.black,
                onPressed: () {},
              ),
            ),
          ]))),
    );
  }
}
```

Output:

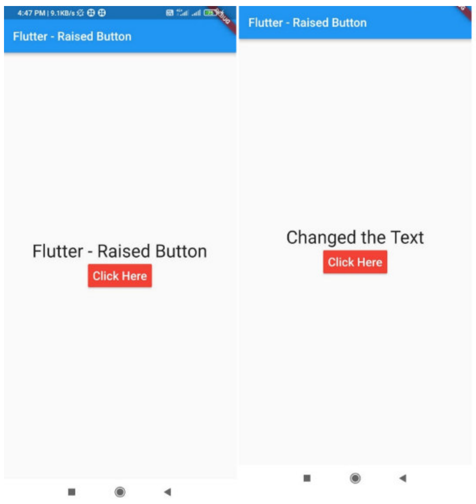


Q.14	Use RaisedButton Widget in flutter App development

Source code:

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

//This example explains Raised Button Widget
void main() {
  runApp(MyApp());
}

class MyApp extends StatefulWidget {
  @override
  _MyAppState createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
  String msg = 'Flutter - Raised Button';
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Flutter - Raised Button'),
        ),
        body: Container(
          child: Center(
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Text(
                  msg,
                  style: TextStyle(fontSize: 30, fontStyle: FontStyle.normal),
                ),
                RaisedButton(
                  child: Text(
                    "Click Here",
                    style: TextStyle(fontSize: 20),
                  ),
                  onPressed: _changeText,
                  color: Colors.red,
                  textColor: Colors.white,
                  padding: EdgeInsets.all(8.0),
                  splashColor: Colors.grey,
                )
              ],
            ),
          ),
        ),
      ),
    );
  }

  _changeText() {
    setState(() {
      if (msg == 'Flutter - Raised Button') {
        msg = 'Changed the Text';
      } else {
        msg = 'Flutter - Raised Button';
      }
    });
  }
}
```

Output:

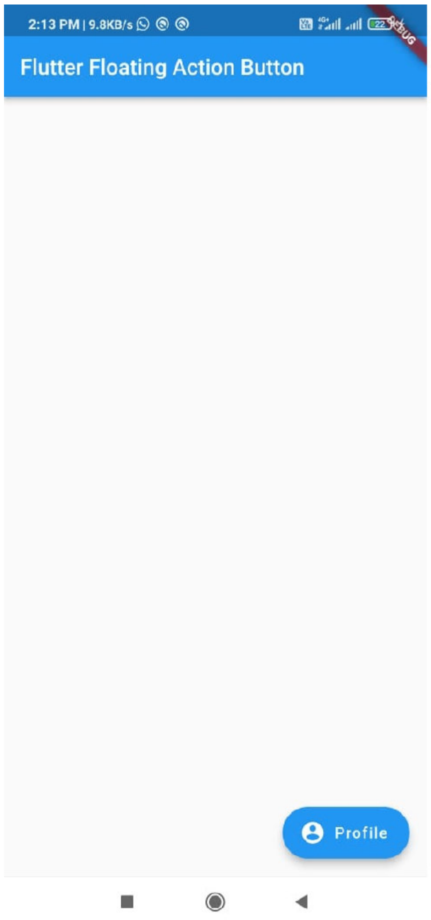


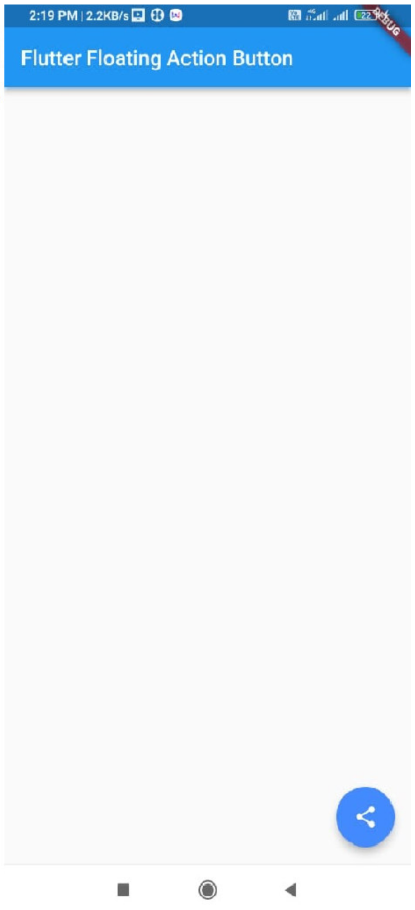
Q.15	Use Floating Action Button Widget in flutter App development

Source code:

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

//This example explains Floating Action Button Widget
void main() {
  runApp(MyApp());
}

class MyApp extends StatefulWidget {
  @override
  _MyAppState createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text("Flutter Floating Action Button"),
          backgroundColor: Colors.blue,
        ),
        // floatingActionButton: FloatingActionButton(
        //   child: Icon(Icons.share),
        //   backgroundColor: Colors.blueAccent,
        //   foregroundColor: Colors.white,
        //   onPressed: () => {},
        // ),
        floatingActionButton: FloatingActionButton.extended(
          onPressed: () {},
          icon: Icon(Icons.account_circle),
          label: Text("Profile"),
        ),
      ),
    );
  }
}
```

Output:



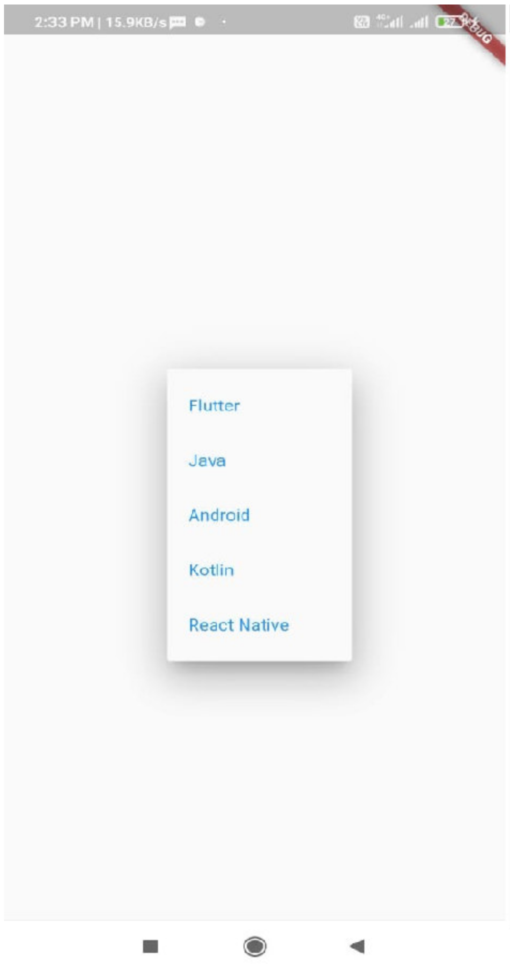


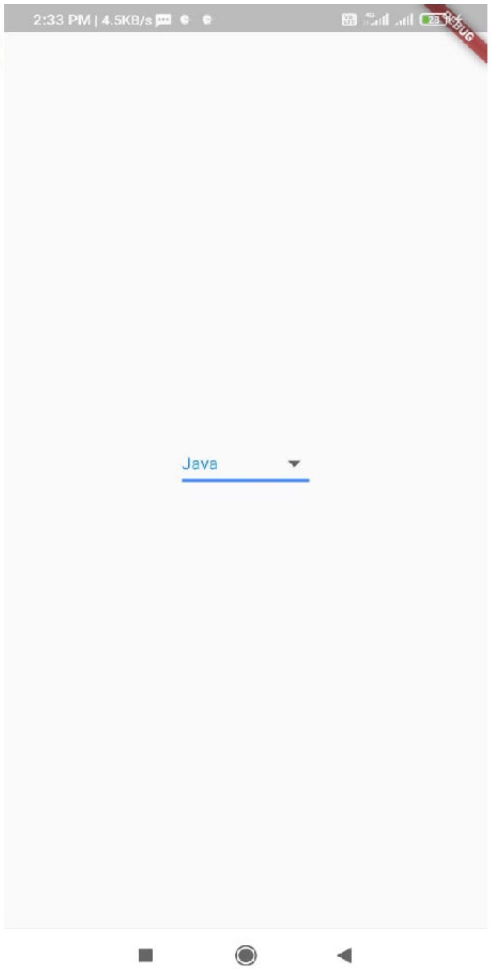
Q.16	Use Dropdown Button Widget in flutter App development

Source code:

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

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

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ),
      home: Scaffold(
        body: Center(
          child: MyHomePage(),
        ),
      ),
    );
  }
}

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

class _MyHomePageState extends State<MyHomePage> {
  String? dropdownvalue = 'Flutter';

  @override
  Widget build(BuildContext context) {
    return DropdownButton<String>(
      value: dropdownvalue,
      icon: Icon(Icons.arrow_drop_down),
      iconSize: 25,
      elevation: 16,
      style: TextStyle(color: Colors.blue),
      underline: Container(
        height: 3,
        color: Colors.blueAccent,
      ),
      onChanged: (String? newValue) {
        setState(() {
          dropdownvalue = newValue;
        });
      },
      items: <String>["Flutter", "Java", "Android", "Kotlin", "React Native"]
          .map<DropdownMenuItem<String>>((String val) {
        // ignore: missing_required_param
        return DropdownMenuItem<String>(
          value: val,
          child: Text(val),
        );
      }).toList(),
    );
  }
}
```

Output:



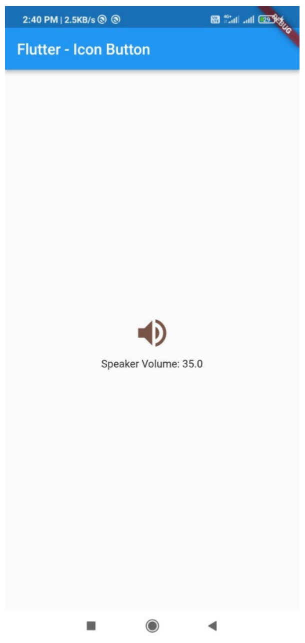


Q.17	Use Icon Button Widget in flutter App development

Source code:

```
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("Flutter - Icon Button"),
        ),
        body: Center(
          child: MyStatefulWidget(),
        ),
      ),
    );
  }
}

double _speakervolume = 0.0;

class MyStatefulWidget extends StatefulWidget {
  @override
  _MyStatefulWidgetState createState() => _MyStatefulWidgetState();
}

class _MyStatefulWidgetState extends State<MyStatefulWidget> {
  Widget build(BuildContext context) {
    return Column(
      mainAxisSize: MainAxisSize.min,
      children: <Widget>[
        IconButton(
          icon: Icon(Icons.volume_up),
          iconSize: 50,
          color: Colors.brown,
          tooltip: 'Increase volume by 5',
          onPressed: () {
            setState(() {
              _speakervolume += 5;
            });
          },
        ),
        Text('Speaker Volume: $_speakervolume')
      ],
    );
  }
}
```

Output:



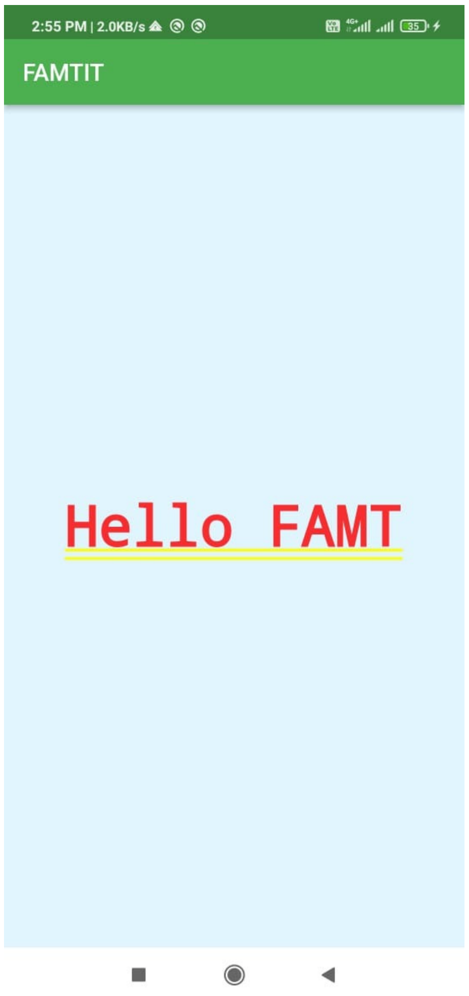
Speaker Volume: 35.0

Q.18	Use RichText Widget in flutter App development

Source code:

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

//This example exlains RichText Widget
void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
// This widget is
//the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'ClipOval',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: MyHomePAGE(),
      debugShowCheckedModeBanner: false,
    );
  }
}

class MyHomePAGE extends StatefulWidget {
  @override
  _MyHomePAGEState createState() => _MyHomePAGEState();
}

class _MyHomePAGEState extends State<MyHomePAGE> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('FAMTIT'),
        backgroundColor: Colors.green,
      ),
      body: Center(
          child: RichText(
        text: TextSpan(
          text: 'Hello ',
          style: DefaultTextStyle.of(context).style,
          children: <TextSpan>[
            TextSpan(
                text: 'FAMT', style: TextStyle(fontWeight: FontWeight.bold)),
          ],
          ),
        )),
        backgroundColor: Colors.lightBlue[50],
      );
    }
  }

class MyClip extends CustomClipper<Rect> {
  Rect getClip(Size size) {
    return Rect.fromLTWH(0, 0, 100, 100);
  }

  bool shouldReclip(oldClipper) {
    return false;
  }
}
```

Output:


2:55 PM | 2.0KB/s
FAMTIT
Hello FAMT