PRN: **121A3014**

Experiment No: 3

AIM: To design a layout of Flutter App using layout widgets and images.

THEORY:

Flutter Layout

- **Types of Layout Widgets:**

1. Single Child Widget : Center, Padding, Container, Align, etc.

Multiple Child Widget : Row, Column, ListView, GridView, Expanded, Stack.

- **Single Child Widget :** The single child layout widget is a type of widget, which can have only one child widget inside the parent layout widget.

1. **Center**: This widget allows you to center the child widget within itself.
2. **Align**: It is a widget, which aligns its child widget within itself and sizes it based on the child's size. It provides more control to place the child widget in the exact position where you need it.
3. **Container**: It is the most popular layout widget that provides customizable options for painting, positioning, and sizing of widgets.
   Example :
```
Center(
  child:
    Container(
    height:
    110.0,
    width:     110.0,
    color:
    Colors.blue,
    child: Align(
    alignment:
      Alignment.topLeft,  child:
      FlutterLogo(
        size: 50,
      ),
    ),
  ),
)
```
4. **Padding**: It is a widget that is used to arrange its child widget by the given padding. It contains EdgeInsets and EdgeInsets.fromLTRB for the desired side where you want to provide padding. Example :
```
const
  Greetings(
```

PRN: **121A3014**

```
  child:
  Padding(
    padding:
    EdgeInsets.all(14.0), child:
    Text('Hello World!'),
  ),
)
```

5. **SizedBox**: This widget allows you to give the specified size to the child widget through all screens.

   <u>Example :</u>

```
SizedBox(
  width:
  300.0,
  height: 450.0,
  child: const Card(child: Text('Hello JavaTpoint!')),  )
```

6. **AspectRatio**: This widget allows you to keep the size of the child widget to a specified aspect ratio.

```
AspectRatio(
  aspectRatio:
  5/3,          child:
  Container(
  color:
  Colors.bluel,
  ),
),
```

7. **Baseline**: This widget shifts the child widget according to the child's baseline.

```
child: Baseline(
         baseline: 30.0,
         baselineType:
         TextBaseline.alphabetic,         child:
         Container(
            height: 60,
            width: 50,
            color: Colors.blue,
         ),
)
```

8. **ConstrainedBox:** It is a widget that allows you to force the additional constraints on its child widget. It means you can force the child widget to have a specific constraint without changing the properties of the child widget.

```
ConstrainedBox(
  constraints:    new    BoxConstraints(
    minHeight: 150.0,
    minWidth: 150.0,
    maxHeight: 300.0,
    maxWidth: 300.0,
  ),
  child: new DecoratedBox(
    decoration: new BoxDecoration(color: Colors.red),
  ),
),
```

PRN: **121A3014**

FittedBox: It scales and positions the child widget according to the specified fit.

- **Multiple Child widgets :** The multiple child widgets are a type of widget, which contains more than one child widget, and the layout of these widgets are unique.
  1. GridView : Flutter GridView is a widget that is similar to a 2-D Array in any programming language. As the name suggests, a GridView Widget is used when we have to display something on a Grid. We can display images, text, icons, etc on GridView. We can implement GridView in various ways in Flutter :

     GridView.count()

     GridView.builder()

     GridView.custom()

     GridView.extent()

- **Properties of GridView:**

anchor: This property takes in a double value as the object to control the zero scroll effect.

childrenDelegate: sliverChildDelegate is the object of this property. It provides a delegate that serves the children for the GridView.

GridView.count() is one which is used frequently and it is used when we already know the size of Grids. Whenever we have to implement GridView dynamically, we use GridView.builder(). Both are just like a normal array and dynamic array. In Flutter, the two GridView is mostly used.

GridView.count() is used with some named parameters. The properties that we can use with GridView.count() are:

crossAxisCount: It defines the number of columns in GridView.

crossAxisSpacing: It defines the number of pixels between each child listed along the cross axis.

mainAxisSpacing: It defines the number of pixels between each child listed along the main axis.

padding(EdgeInsetsGeometry): It defines the amount of space to surround the whole list of widgets.

primary: If true, it’s ‘Scroll Controller’ is obtained implicitly by the framework.

scrollDirection: It defines the direction in which the items on GridView will move, by default it is vertical.

PRN: **121A3014**

reverse: If it is set to true, it simply reverses the list of widgets in opposite direction along the main axis.

shrinkWrap: By default, it values is false then the scrollable list takes as much as space for scrolling in scroll direction which is not good because it takes memory that is wastage of memory and performance of app reduces and might give some error, so to avoid leakage of memory while scrolling, we wrap our children widgets using shrinkWrap by setting shrinkWrap to true and then scrollable list will be as big as it's children widgets will allow.

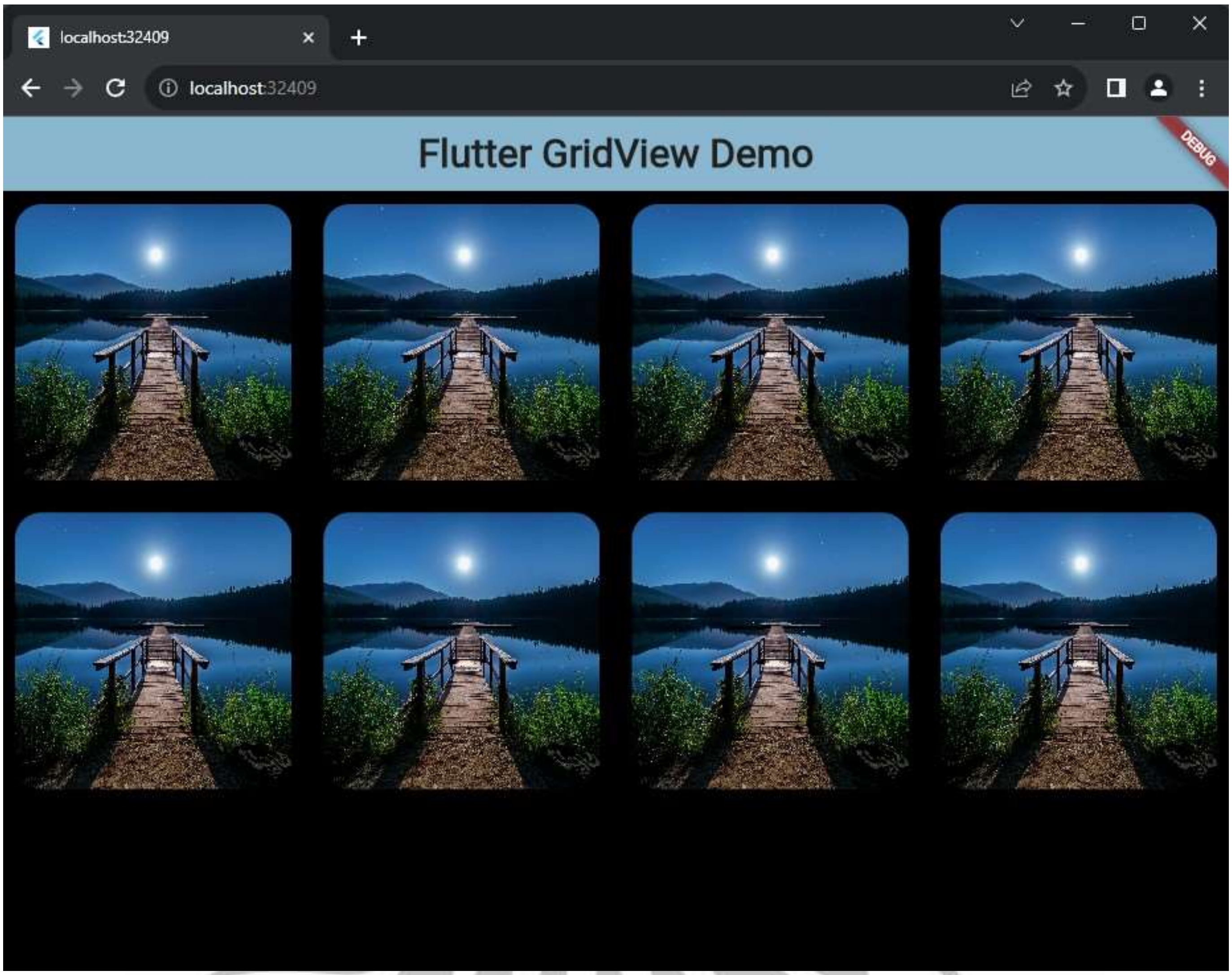
Demonstration of Grid View Properties:

```
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(
        home: Scaffold(
      backgroundColor: Colors.black,
      appBar: AppBar(
        backgroundColor: const Color.fromARGB(255, 136, 184, 208),
        title: Center(
          child: Text(
            'Flutter GridView Demo',
            style: TextStyle(
              color: Color.fromARGB(255, 32, 32, 30),
              fontWeight: FontWeight.bold,
              fontSize: 30.0,
            ),
          ),
        ),
      ),
      body: GridView.count(
        crossAxisCount: 4,
        crossAxisSpacing: 4,
        mainAxisSpacing: 4,
        shrinkWrap: true,
        // scrollDirection: Axis.horizontal,
        children: List.generate(8, (index) {
          return Padding(
            padding: const EdgeInsets.all(10),
            child: Container(
              decoration: BoxDecoration(
```

PRN: **121A3014**

```
            image: DecorationImage(
              image: NetworkImage(
                  'https://images.pexels.com/photos/414612/pexels-photo-414612.jpeg'),
              fit: BoxFit.cover,
            ),
            borderRadius: BorderRadius.all(
              Radius.circular(20),
            ),
          ),
        ),
      );
    }),
  ),

  // body: ListView(
  //   padding: EdgeInsets.all(10),
  //   shrinkWrap: true,
  //   reverse: true,
  //   // itemExtent: 100,
  //   // scrollDirection: Axis.horizontal,
  //   children: [
  //     const Card(child: ListTile(title: Text("Item 1"))),
  //     const Card(child: ListTile(title: Text("Item 2"))),
  //     const Card(child: ListTile(title: Text("Item 3"))),
  //   ],
  // ),
 ));
 }
}
```

PRN: **121A3014**



PRN: **121A3014**

2. **ListView** is a very important widget in flutter. It is used to create the list of children But when we want to create a list recursively without writing code again and again then ListView.builder is used instead of ListView. ListView.builder creates a scrollable, linear array of widgets. ListView.builder by default does not support child reordering.
   This default ListView is suitable if we want to display a small number of children. Use it to display a static list of children whose count don't change. For example, we can use this listview for displaying a menu in our app.

   To create a ListView call the constructor of the ListView class provided by flutter and provide required properties. There are no required properties for a listview widget. But we have to provide data to the children property, in order to display the listview.

```
ListView(
{Key? key,
Axis        scrollDirection        =
Axis.vertical,   bool   reverse   =
false,             ScrollController?
controller,
bool?                primary,
ScrollPhysics?  physics,
bool shrinkWrap = false,
EdgeInsetsGeometry?
padding,                 double?
itemExtent,
bool addAutomaticKeepAlives =
true, bool addRepaintBoundaries
```

PRN: **121A3014**

```
     = true, bool addSemanticIndexes
     = true, double? cacheExtent,
    List<Widget> children = const <Widget>[],
    int? semanticChildCount,
    DragStartBehavior dragStartBehavior = DragStartBehavior.start,

    ScrollViewKeyboardDismissBehavior keyboardDismissBehavior =  ScrollViewKeyboardDismissBehavior.manual,
     String? restorationId,
    Clip clipBehavior = Clip.hardEdge}
  )
```

Basic implementation of ListView.

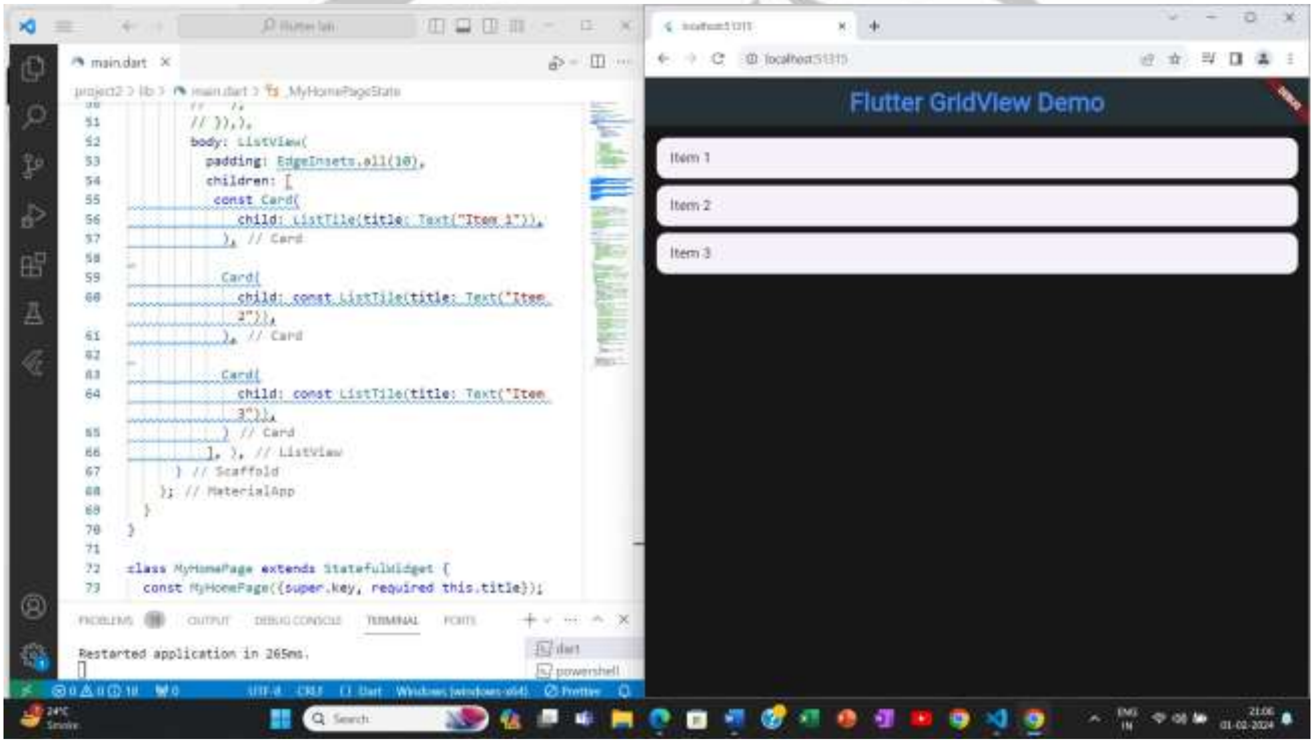
```
ListView(
      children: [
        child
        widget1,
        child
        widget2,
        .....
      ],
      scrollDirection: Axis.horizontal,
    )
```

Demonstration of ListView Properties:

Flutter ListView Properties

The important properties of a ListView are :

- Padding : It takes EdgeInsetsGeometry as value. Use this property to apply padding to the listview.
  shrinkWrap : same as gridview shrinkwrap



```
          // }),),
          body: ListView(
            padding: EdgeInsets.all(10),
            children: [
             const Card(
                child: ListTile(title: Text("Item 1")),
              ), // Card

              Card(
                child: const ListTile(title: Text("Item
                2")),
              ), // Card

              Card(
                child: const ListTile(title: Text("Item
                3")),
              ) // Card
            ], ), // ListView
        ) // Scaffold
      ); // MaterialApp
    }
  }

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});
```

```
Restarted application in 265ms.
```

Flutter GridView Demo

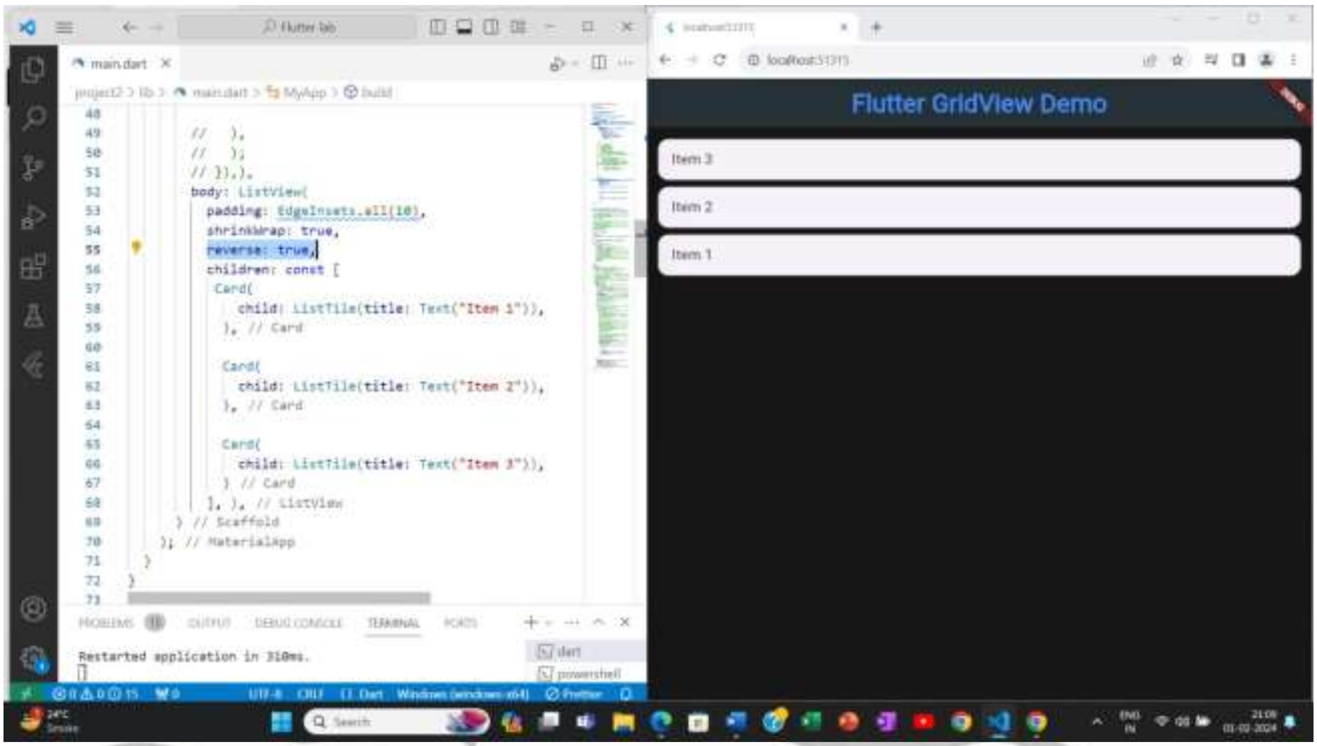
Item 1

Item 2

Item 3

PRN: **121A3014**

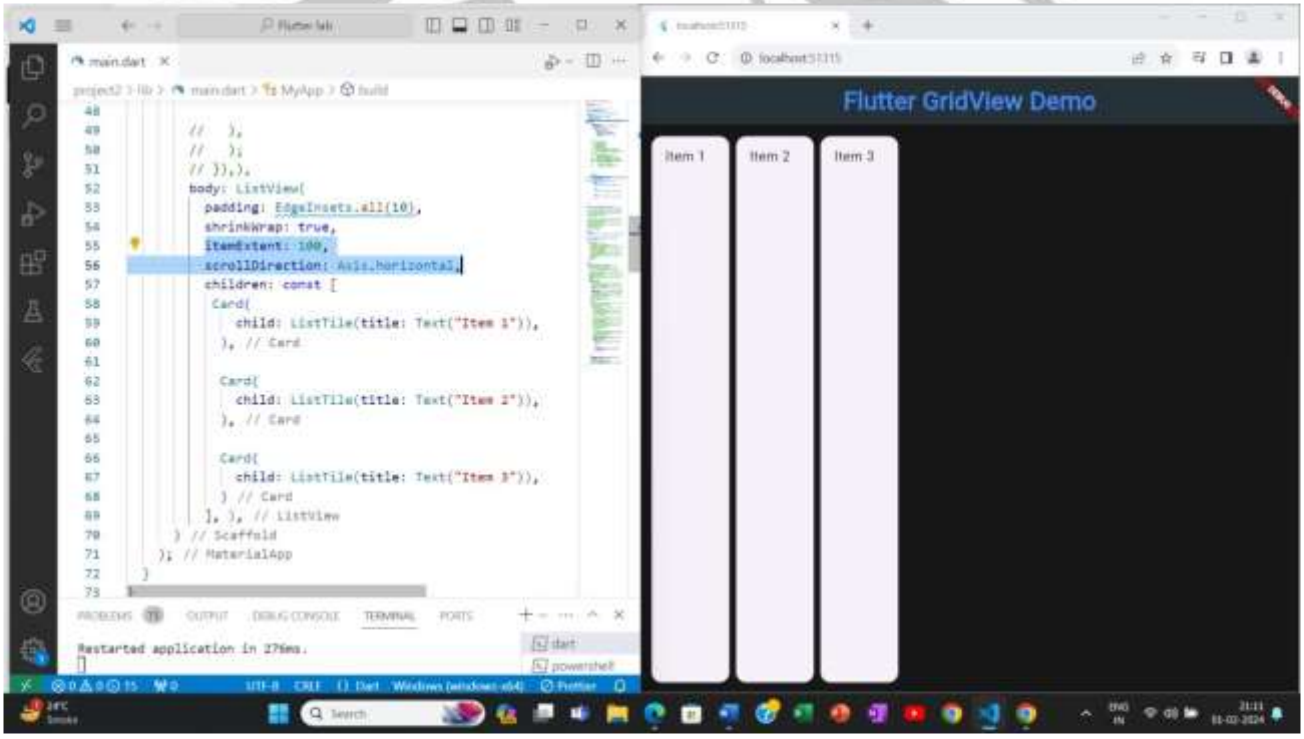
reverse : reverses the order of items in a list



```
            //   ),
            //   );
            // }),),
            body: ListView(
              padding: EdgeInsets.all(10),
              shrinkWrap: true,
              reverse: true,
              children: const [
               Card(
                  child: ListTile(title: Text("Item 1")),
                ), // Card

                Card(
                  child: ListTile(title: Text("Item 2")),
                ), // Card

                Card(
                  child: ListTile(title: Text("Item 3")),
                ) // Card
              ], ), // ListView
          ) // Scaffold
        ); // MaterialApp
      }
    }
```

```
Restarted application in 310ms.
```

- scrollDirection : It takes Axis as value. Use this property to change the scroll direction of the ListView. By default the value is Axis.vertical.

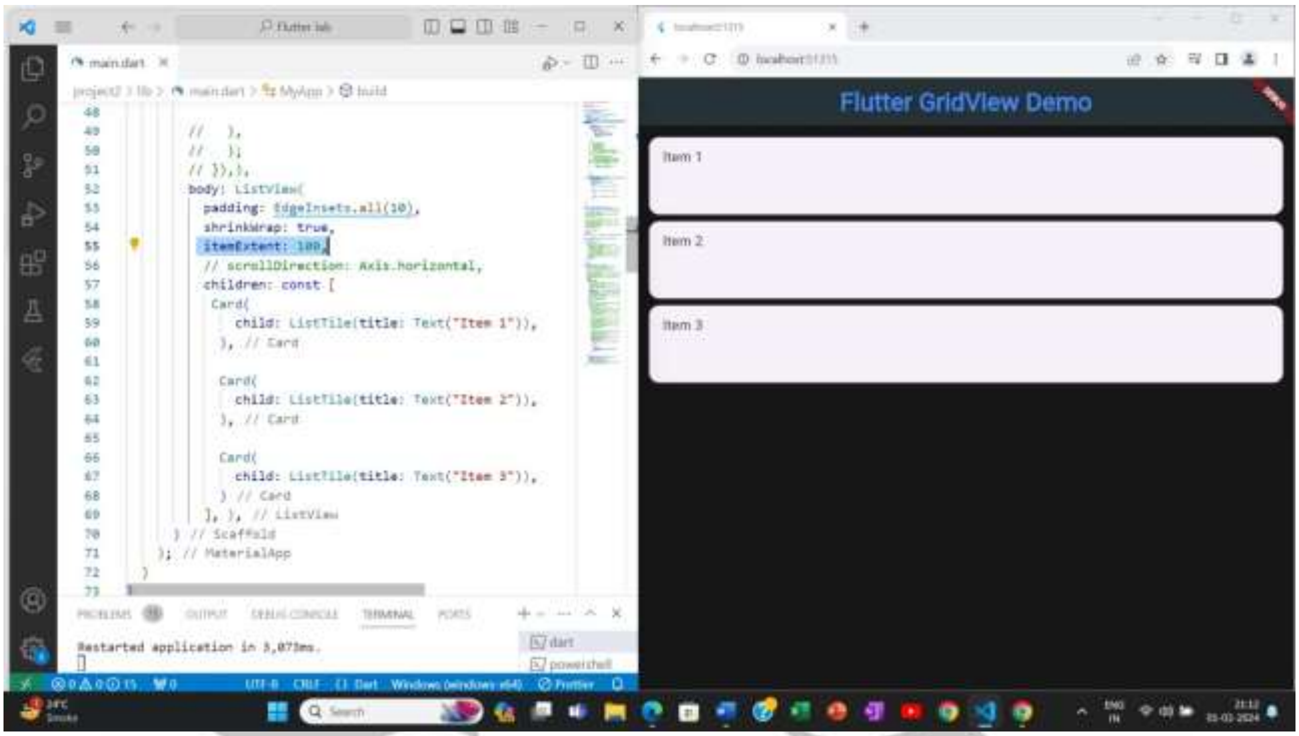


```
            //   ),
            //   );
            // }),),
            body: ListView(
              padding: EdgeInsets.all(10),
              shrinkWrap: true,
              itemExtent: 100,
              scrollDirection: Axis.horizontal,
              children: const [
               Card(
                  child: ListTile(title: Text("Item 1")),
                ), // Card

                Card(
                  child: ListTile(title: Text("Item 2")),
                ), // Card

                Card(
                  child: ListTile(title: Text("Item 3")),
                ) // Card
              ], ), // ListView
          ) // Scaffold
        ); // MaterialApp
      }
```

```
Restarted application in 276ms.
```

PRN: **121A3014**

- itemExtent : It takes double as value. Use this property to extend (increase) the item in scroll direction. when the scroll direction is vertical it increases height and when vertical it increases the width of the item.



```
              //   ),
              //   );
              // }),),
              body: ListView(
                padding: EdgeInsets.all(10),
                shrinkWrap: true,
                itemExtent: 100,
                // scrollDirection: Axis.horizontal,
                children: const [
                 Card(
                    child: ListTile(title: Text("Item 1")),
                  ), // Card

                  Card(
                    child: ListTile(title: Text("Item 2")),
                  ), // Card

                  Card(
                    child: ListTile(title: Text("Item 3")),
                  ) // Card
                ], ), // ListView
            ) // Scaffold
          ); // MaterialApp
        }
```

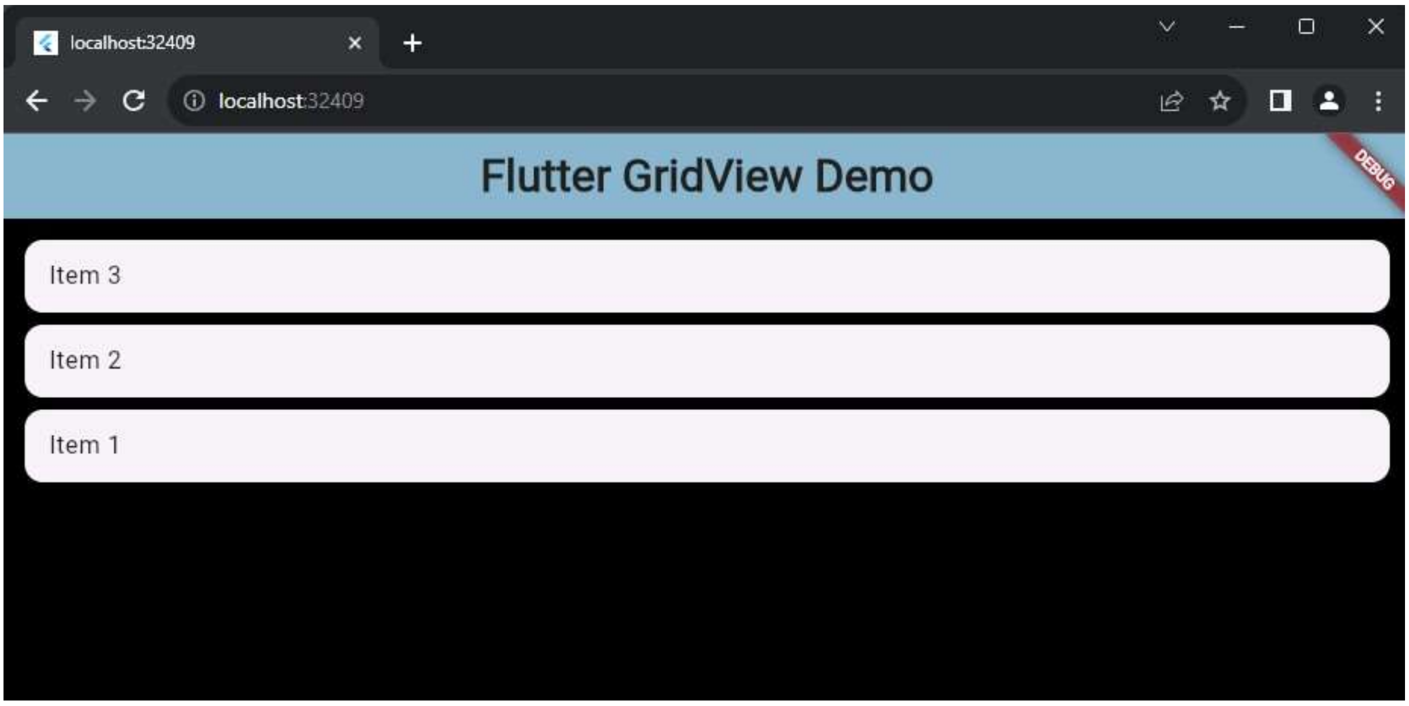
```
Restarted application in 3,073ms.
```

```
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(
        home: Scaffold(
      backgroundColor: Colors.black,
      appBar: AppBar(
        backgroundColor: const Color.fromARGB(255, 136, 184, 208),
        title: Center(
          child: Text(
            'Flutter GridView Demo',
            style: TextStyle(
              color: Color.fromARGB(255, 32, 32, 30),
              fontWeight: FontWeight.bold,
```

PRN: **121A3014**

```
          fontSize: 30.0,
        ),
      ),
    ),
  ),
  // body: GridView.count(
  //   crossAxisCount: 4,
  //   crossAxisSpacing: 4,
  //   mainAxisSpacing: 4,
  //   shrinkWrap: true,
  //   // scrollDirection: Axis.horizontal,
  //   children: List.generate(8, (index) {
  //     return Padding(
  //       padding: const EdgeInsets.all(10),
  //       child: Container(
  //         decoration: BoxDecoration(
  //           image: DecorationImage(
  //             image: NetworkImage(
  //                 'https://images.pexels.com/photos/414612/pexels-photo-414612.jpeg'),
  //             fit: BoxFit.cover,
  //           ),
  //           borderRadius: BorderRadius.all(
  //             Radius.circular(20),
  //           ),
  //         ),
  //       ),
  //     );
  //   }),
  // ),

  body: ListView(
    padding: EdgeInsets.all(10),
    shrinkWrap: true,
    reverse: true,
    // itemExtent: 100,
    // scrollDirection: Axis.horizontal,
    children: [
      const Card(child: ListTile(title: Text("Item 1"))),
      const Card(child: ListTile(title: Text("Item 2"))),
      const Card(child: ListTile(title: Text("Item 3"))),
    ],
  ),
 ));
}
}
```

PRN: **121A3014**



Flutter GridView Demo

Item 3

Item 2

Item 1

CONCLUSION:

Hence we have successfully designed a layout of Flutter App using layout widgets and images.