CHAPITER IV: WIDGETS ET ANIMATIONS IN FLUTTER

I. ListView Widget

In Flutter, ListView is a scrollable list of widgets arranged linearly. It displays its children one after another in the scroll direction i.e, vertical or horizontal.

The ListView widget provides several constructors that allow you to customize its behaviour according to your needs. Some of the most common constructors include

1. ListView

It is a simple widget where we provide the widget list inside the children parameter.

```
ListView(
  children: const <Widget>[],
),
```

2. ListView.builder

This constructor creates a list of items based on a builder function. This is useful when you have a large number of items to display and you don't want to create them all upfront. It allows you to lazily generate and display a list of widgets based on a builder callback.

```
ListView.builder(
  itemCount: itemCount, // Required: The total number of items in the list
  itemBuilder: (BuildContext context, int index) {
    // Required: Builder callback function that creates a widget for each item
    // 'context' is the BuildContext for the widget tree
    // 'index' is the index of the current item being built

    // Return the widget for the current item
    return YourItemWidget(data: dataList[index]);
  },
)
```

3. ListView.separated

This constructor creates a list of items separated by a divider. This is useful when you want to visually distinguish between items in the list.

```
ListView.separated(
  itemCount: itemCount, // Required: The total number of items in the list
  separatorBuilder: (BuildContext context, int index) {
    // Required: Separator builder callback function that creates a widget for each separator
    // 'context' is the BuildContext for the widget tree
    // 'index' is the index of the current separator being built

    // Return the widget for the current separator
    return YourSeparatorWidget();
  },
  itemBuilder: (BuildContext context, int index) {
    // Required: Builder callback function that creates a widget for each item
    // 'context' is the BuildContext for the widget tree
    // 'index' is the index of the current item being built

    // Return the widget for the current item
    return YourItemWidget(data: dataList[index]);
  },
)
```

4. ListView.custom

The ListView.custom constructor in Flutter provides a way to create a highly customizable list view by specifying a custom SliverChildDelegate that defines how the list items should be built and laid out.

```
ListView.custom(
  childrenDelegate: SliverChildDelegate(
    findChildIndexCallback: (Key key) {
      // Required: Returns the index of the widget with the given key
      // Return null if the key is not associated with any widget
      // You can use this callback for optimizations like item removal or reordering
      return findIndexByKeyFunction(key);
    },
    childCount: itemCount, // Required: The total number of items in the list
    itemBuilder: (BuildContext context, int index) {
      // Required: Builder callback function that creates a widget for each item
      // 'context' is the BuildContext for the widget tree
      // 'index' is the index of the current item being built

      // Return the widget for the current item
      return YourItemWidget(data: dataList[index]);
    },
  ),
)
```

II. DataTable Widget

A DataTable is a material design used to display data on a table or in rows and columns. A Data table is used to show the data which have columns and rows as child, a Column is used to set the name of the column, and a Row is used to set the values of the columns.

```
DataTable(
    // datatable widget
    columns: [
        // column to set the name
        DataColumn(label: Text('Col1'),),
        DataColumn(label: Text('Col2'),),
    ],
    rows: [
        // row to set the values
        DataRow(cells: [
            DataCell(Text('ValCol1')),
            DataCell(Text('ValCol2')),
        ]),
    ],
)
```

III. Stack Widget

Stack widget is a built-in widget in flutter SDK which allows us to make a layer of widgets by putting them on top of each other. The stack allows developers **to overlap multiple widgets into a single screen** and renders them from bottom to top. Hence, the **first widget** is the **bottommost** item, and the **last widget** is the **topmost** item. Many of the times a simple row and column layout is not enough, we need a way to overlay one widget on top of the other, for example, we might want to show some text over an image,

Properties of Stack Widget:

- **alignment:** This property takes a parameter of Alignment Geometry, and controls how a child widget which is non-positioned or partially-positioned will be aligned in the Stack.
- **clipBehaviour:** This property decided whether the content will be clipped or not.
- **fit:** This property decided how the non-positioned children in the Stack will fill the space available to it.
- **overflow:** This property controls whether the overflow part of the content will be visible or not,
- **textDirection:** With this property, we can choose the text direction from right to left. or left to right.

The child widget in a stack can be either **positioned** or **non-positioned**. **positioned** is not the stack parameter but can be used in the stack to locate the children widgets.

```
Stack(
  children: <Widget>[
    // Max Size
    Container(
      color: Colors.green,
    ),
    Container(
      color: Colors.blue,
    ),
   Positioned(
       top: 30,
       right: 20,
       child: Container(
                color: Colors.yellow,
        ),
    ),
  ],
),
```

IV. SnackBar Widget

Snackbar is a widget provided by flutter to display a dismissible pop-up message on your application. It is used to show users if certain actions take place in our applications. For example, if the user login process fails due to some reason, so to inform the user to try again we can use snackbar. It pops up on the screen, and it can also perform operations like undoing the action which has taken place. Snackbar displays the informative message for a very short period of time and when the time is completed it disappears on its own. The recommended onscreen time for a snackbar is 5-10s.

```
ElevatedButton(
      style: ButtonStyle(backgroundColor: MaterialStateProperty.all(Colors.green)),
      onPressed: () {
        const snackdemo = SnackBar(
          content: Text('Hii this is GFG\'s SnackBar'),
          backgroundColor: Colors.green,
          elevation: 10,
          behavior: SnackBarBehavior.floating,
          margin: EdgeInsets.all(5),
        );
        ScaffoldMessenger.of(context).showSnackBar(snackdemo);

        // 'showSnackBar' is deprecated and shouldn't be used.
        //Use ScaffoldMessenger.showSnackBar.
        // Scaffold.of(context).showSnackBar(snackdemo);
      },
      child: const Text('Click Here'),
    ),
```

V. AnimatedWidget

Animated Widget is a widget that rebuilds when the given Listenable changes value. **AnimatedWidget** is most commonly used with Animation objects, which are Listenable, but it can be used with any Listenable, including **ChangeNotifier** and **ValueNotifier**. See file aninamtion.dart