Experiment No : 06

Aim : To include bar charts in Flutter application

Theory :

A chart is a graphical representation of data where data is represented by a symbol such as a line, bar, pie, etc. In Flutter, the chart behaves the same as a normal chart. We use a chart in Flutter to represent the data in a graphical way that allows the user to understand them in a simple manner. We can also plot a graph to represents the rise and fall of our values. The chart can easily read the data and helps us to know the performance on a monthly or yearly basis whenever we need it.

Supported Chart Types in Flutter

Flutter supports mainly three types of charts, and each chart comes with several configuration options. The following are the chart used in Flutter application:

Line Chart

Bar Chart

Pie and Donut Chart

Line Chart

A line chart is a graph that uses lines for connecting individual data points. It displays the information in a series of data points. It is mainly used to track changes over a short and long period of time.

```
LineChart(
  LineChartData(
    // write your logic
  ),
);
```

Bar Chart

It is a graph that represents the categorical data with rectangular bars. It can be horizontal or vertical.

We can use it as below:

```
BarChart(
  BarChartData(
    // write your logic
  ),
);
```

Pie or Donut Chart

It is a graph that displays the information in a circular graph. In this graph, the circle is divided into sectors, and each shows the percentage or proportional data.

We can use it as below:

```
PieChart(
  PieChartData(
    // write your logic
  ),
);
```

Code :

Main.dart

```
import 'dart:core';

import 'package:flutter/material.dart';

import 'package:charts_flutter/flutter.dart' as charts;
```

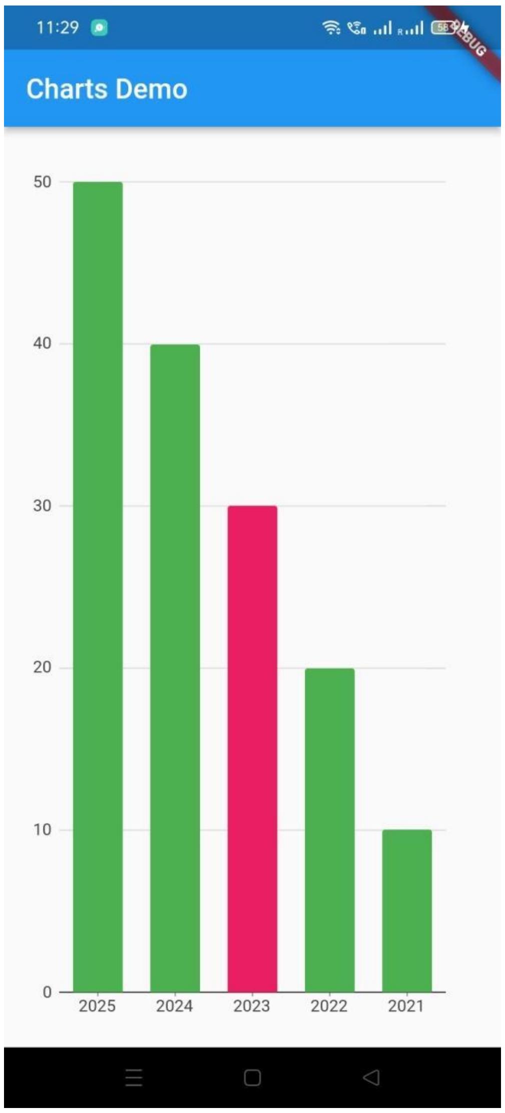
```
void main() {
  runApp(const MyApp());
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      //theme: ThemeData(

      // primarySwatch: Colors.blue,
      // ),
      home:  MyHomePage(),
    );
  }
}

class MyHomePage extends StatefulWidget {
  MyHomePage(): super();
  final String title="Charts Demo";

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  late List<charts.Series<dynamic, String>> seriesList;
  static List<charts.Series<Sales, String>> _createRandomData()
  {
    final desktopSalesData = [
      Sales( '2025',50),
      Sales( '2024',40),
      Sales( '2023',30),
      Sales( '2022',20),
      Sales( '2021',10),
    ];return[
      charts.Series<Sales, String>(
        id: 'Sales',
        domainFn: (Sales sales,_) =>sales.year,
        measureFn: (Sales sales,_) =>sales.sales,
        data: desktopSalesData,
        fillColorFn: (Sales sales, _)
        {return (sales.year == '2023')
            ?charts.MaterialPalette.pink.shadeDefault
            :charts.MaterialPalette.green.shadeDefault;},)];}
  barChart() {return charts.BarChart(
      seriesList = _createRandomData()
  );
  }
  @override
  Widget build(BuildContext context) => Scaffold(
    appBar: AppBar(
      title: Text(widget.title),
    ),
    body: Container(
      padding: EdgeInsets.all(20.0),
      child:barChart(),
    ),
  );
}
class Sales{
  final String year;
  final int sales;
  Sales(this.year,this.sales);
}
```

Pubspec.yaml

```
name: exp6
description: A new Flutter project.

# The following line prevents the package from being accidentally published to
# pub.dev using `flutter pub publish`. This is preferred for private packages.
publish_to: 'none' # Remove this line if you wish to publish to pub.dev

# The following defines the version and build number for your application.
# A version number is three numbers separated by dots, like 1.2.43
# followed by an optional build number separated by a +.
# Both the version and the builder number may be overridden in flutter
# build by specifying --build-name and --build-number, respectively.
# In Android, build-name is used as versionName while build-number used as
versionCode.
# Read more about Android versioning at
https://developer.android.com/studio/publish/versioning
# In iOS, build-name is used as CFBundleShortVersionString while build-number
is used as CFBundleVersion.
# Read more about iOS versioning at
#
https://developer.apple.com/library/archive/documentation/General/Reference/In
foPlistKeyReference/Articles/CoreFoundationKeys.html
# In Windows, build-name is used as the major, minor, and patch parts
# of the product and file versions while build-number is used as the build
suffix.
version: 1.0.0+1

environment:
  sdk: '>=2.18.6 <3.0.0'

# Dependencies specify other packages that your package needs in order to
work.
# To automatically upgrade your package dependencies to the latest versions
# consider running `flutter pub upgrade --major-versions`. Alternatively,
# dependencies can be manually updated by changing the version numbers below
to
# the latest version available on pub.dev. To see which dependencies have
newer
# versions available, run `flutter pub outdated`.
dependencies:
  flutter:
    sdk: flutter

  charts_flutter: ^0.12.0

  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.2

dev_dependencies:
  flutter_test:
    sdk: flutter
```

```yaml
  # The "flutter_lints" package below contains a set of recommended lints to
  # encourage good coding practices. The lint set provided by the package is
  # activated in the `analysis_options.yaml` file located at the root of your
  # package. See that file for information about deactivating specific lint
  # rules and activating additional ones.
  flutter_lints: ^2.0.0

# For information on the generic Dart part of this file, see the
# following page: https://dart.dev/tools/pub/pubspec

# The following section is specific to Flutter packages.
flutter:

  # The following line ensures that the Material Icons font is
  # included with your application, so that you can use the icons in
  # the material Icons class.
  uses-material-design: true

  # To add assets to your application, add an assets section, like this:
  # assets:
  #   - images/a_dot_burr.jpeg
  #   - images/a_dot_ham.jpeg

  # An image asset can refer to one or more resolution-specific "variants",
see
  # https://flutter.dev/assets-and-images/#resolution-aware

  # For details regarding adding assets from package dependencies, see
  # https://flutter.dev/assets-and-images/#from-packages

  # To add custom fonts to your application, add a fonts section here,
  # in this "flutter" section. Each entry in this list should have a
  # "family" key with the font family name, and a "fonts" key with a
  # list giving the asset and other descriptors for the font. For
  # example:
  # fonts:
  #   - family: Schyler
  #     fonts:
  #       - asset: fonts/Schyler-Regular.ttf
  #       - asset: fonts/Schyler-Italic.ttf
  #         style: italic
  #   - family: Trajan Pro
  #     fonts:
  #       - asset: fonts/TrajanPro.ttf
  #       - asset: fonts/TrajanPro_Bold.ttf
  #         weight: 700
  #
  # For details regarding fonts from package dependencies,
  # see https://flutter.dev/custom-fonts/#from-packages
```

Output :



Conclusion : Therefore we have successfully implemented bar charts in Flutter application