**Task 7 : Report**

**Goals of Responsive Design**

1. **Wider Audience** – The more flexible and responsive your app is, the more audience it will attract.
2. **Improved SEO** – Flexible and Responsive web design helps in enhancing search engine optimization.
3. **Improved browsing experience** – Responsive and flexible web app designs lead to an overall improved browsing experience and even lead to increased buyers.
4. **Brand visibility** –  A breakthrough, fast, and beautiful app design improves the overall brand visibility.

Why should your Flutter app be Responsive?

A Flutter application can run on any mobile device, tablet, or TV screen. Today, Mobile Devices have so many different screen sizes and resolutions, and the apps must cater to all such screen sizes. And this doesn’t end here; users can rotate their phones, and the applications should adjust when viewing or testing on landscape or portrait modes. Catering to so many user requirements is not an easy task, so the apps should be Responsive enough to meet all such needs and ensure a seamless user experience.

**Flutter Responsive App Examples**

Some of the responsive web apps made using Flutter that are widely used in the market are:

1. **Google Ads** – It is a mobile application that allows managing Google ad campaigns directly from smartphones.
2. **Reflectly** – It is an AI-powered personal journaling application that helps users in coping up with daily stress and negative thoughts by combining cognitive behavioral therapy and positive psychology.
3. **Lunching** – It is a smartphone app that makes it easier to order delivery food. It is one of the most successful food delivery apps running today.
4. **Watermaniac** – It is a lightweight water monitoring application made using the Flutter framework. It assists consumers in keeping an eye on their daily water consumption.
5. **Cryptograph** – This Flutter app helps in monitoring and tracking the latest updates on over 1600+ global cryptocurrencies such as Bitcoin, Ripple, Ethereum, etc.

### How To Create A Responsive Flutter App?

Let us now learn how to make a Flutter app responsive.

**1. Media Query**

Media Query can be used to get the real-time size (width/height) and orientation (portrait/landscape) of the window screen. It suggests the orientation and size of the app.

**2. Layout Builder**

Layout Builder is just a simplified version of Media Query. The main difference between Media Query and Layout Builder is that Media Query is based on the full context of the screen rather than just the size of a particular widget; on the other hand, Layout Builder determines the maximum width and height of a specific widget only.

Layout Builder class provides the Box Constraints object that can be used for determining the maxWidth and maxHeight of the widget.

**3. Orientation Builder**

Orientation Builder class can be used to determine a widget’s current orientation.

The Orientation Builder widget is similar to the Layout Builder class. One can get the Orientation object using the builder property of OrientationBuilder class. For example, you can use the OrientationBuilder class to change the number of columns of GridView

**4. Expanded and Flexible Widgets**

Expanded and Flexible Widgets are two widgets that can be used inside Row, Column, or Flex to give their children the flexibility in order to expand to fill the available space. The only difference is that the Expanded widget requires the child to fill all the available space, whereas Flexible does not. Also, Expanded and Flexible widgets can be used to get a responsive Flutter UI that works with percentages instead of hardcoded values.

**6. Custom MutiChild Layout Class**

CustomMultiChildLayout is a widget that utilizes a delegate to specify the size and position of multiple children within it. This delegate is responsible for defining the layout constraints of each child and determining their positioning within the parent widget. While the delegate can define the size of the parent, it cannot be dependent on the size of its children.

CustomMultiChildLayout is a suitable choice when there are intricate connections between the positioning and size of various widgets.