Name:Kshitij Hundre Experiment-3 D15C/18

AIM: - To Include icons , images & fonts in the flutter app/

Theory:

Flutter provides robust support for customizing the UI using **icons**, **images**, and **custom fonts**. These elements enhance the visual appeal and user experience of the application. Below is a detailed explanation of each component:

1. Icons in Flutter:

Icons are graphical representations of actions, files, devices, or app features. Flutter uses Material Design Icons by default, but also allows adding custom icons.

Types of Icons in Flutter:

- Built-in Icons: Provided by Flutter in the Icons class (Material Design).
- Custom Icons: You can use .ttf icon fonts (like Font Awesome) by adding them in the pubspec.yaml and referring via IconData.

How to Use Icons:

- Add icons using the Icon widget.
- Change size, color, and action by setting properties.
- Combine with buttons like IconButton for interactivity.

2. Images in Flutter:

Flutter supports both **network** and **local images** to display pictures, logos, banners, and illustrations.

Types of Images:

- **Asset Images**: Stored in the app's assets folder.
- **Network Images**: Fetched from a URL at runtime.

How to Add Asset Images:

1. Create an assets/ folder in the root of the project.
2. Place image files inside this folder (e.g., logo.png).
3. Declare the image in pubspec.yaml:
4. Use the image in code:

Customization:

- Set width, height, fit (BoxFit.cover, BoxFit.contain, etc.
- Apply rounded corners or shadows using ClipRRect or Container.

3. Fonts in Flutter:

Flutter allows the use of **custom fonts** for branding and better typography.

How to Add Custom Fonts:

1. Create a fonts/ folder and place .ttf or .otf files.
2. Declare fonts in pubspec.yaml:
3. Use the custom font in your app:

Font Properties You Can Customize:

- fontSize
- fontWeight (e.g., FontWeight.bold)
- fontStyle (e.g., FontStyle.italic)
- letterSpacing and wordSpacing

Best Practices:

- Keep asset names in lowercase and avoid spaces.
- Use appropriate image resolutions for better performance.
- Use responsive sizing for images and text for multi-device compatibility.
- Group similar assets (e.g., images/icons/fonts) in separate folders.

Code:

pubspec.yaml

```yaml
assets:
  - assets/logo.png

fonts:
  - family: Poppins
    fonts:
      - asset: fonts/Poppins-Regular.ttf
```

qr_generator_screen.dart

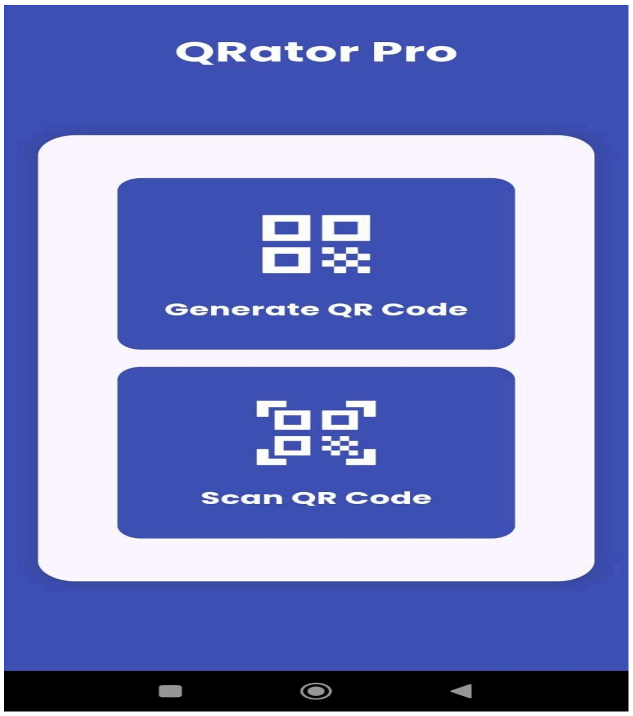
```dart
import 'package:flutter/material.dart';

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Common Widgets Demo',
      theme: ThemeData(
        fontFamily: 'Poppins', // using custom font
        primarySwatch: Colors.blue,
      ),
      home: Scaffold(
        appBar: AppBar(
          title: const Text("Common Widgets"),
          actions: [
            IconButton(
              icon: Icon(Icons.notifications),
              onPressed: () {},
        )
       ],
      ),
      body: Center(
       child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
         Text(
          'Welcome to Flutter!',
          style: TextStyle(
           fontSize: 24,
           fontWeight: FontWeight.bold,
           color: Colors.black87,
          ),
         ),
         SizedBox(height: 20),
         Icon(
          Icons.star,
          color: Colors.amber,
          size: 50,
         ),
         SizedBox(height: 20),
         Image.asset(
          'assets/logo.png', // Make sure this path is correct
          width: 100,
          height: 100,
         ),
         SizedBox(height: 20),
         Text(
          'Custom Font Text Example',
          style: TextStyle(
           fontSize: 18,
           fontFamily: 'Poppins',
           color: Colors.deepPurple,
          ),
         ),
        ],
       ),
      ),
     ),
    );
   }
```

Output:



Conclusion:

In Flutter, incorporating **icons, images, and fonts** is essential for crafting a visually appealing and functional UI. Icons add intuitive controls, images make interfaces vibrant, and custom fonts bring personality and branding to the application. Mastering how to properly use and manage these assets ensures a consistent, professional, and user-friendly design across different platforms.