Experiment No: 3

Experiment No 3 **Aim: To include icons, images, fonts in Flutter app**	
ROLL NO	**30**
NAME	**Kaushik Kotian**
CLASS	**D15-B**
SUBJECT	**MAD & PWA Lab**
LO-MAPPED	

Aim: To include icons, images, fonts in Flutter app

Theory:
Including icons, images, and fonts in a Flutter app enhances its visual appeal and user experience. Flutter provides straightforward ways to add these resources, making your app more engaging and personalized. Here's a concise theoretical overview of how to include icons, images, and fonts in a Flutter app:

Icons: Flutter has a rich set of built-in material design icons available through the Icons class. To use these icons, you simply reference them in your widget tree. For custom icons, you can use font packages or image assets.

- Built-in Icons: Utilize icons directly from the Icons class. For example, Icon(Icons.home) displays a home icon.

- Custom Icons: Add custom icons by including them as font files or images. For font icons, include the font file in your pubspec.yaml file and reference the icons using the Icon widget with the appropriate Unicode value.

Images: Flutter supports including images in your app, which can be stored in assets or fetched from the network.

- Asset Images: Include images in your project's asset folder and reference them in the pubspec.yaml file. Use the Image.asset('path/to/asset.png') widget to display the image.

- Network Images: Display images from the internet using the Image.network('url') widget. Ensure your app has the necessary permissions to access the internet if required.

Fonts: Custom fonts can significantly impact your app's aesthetics and branding. Flutter allows you to include custom fonts by specifying them in the pubspec.yaml file.

- Including Fonts: Place your custom font files in the assets folder. In the pubspec.yaml file, under the fonts section, specify the font family and the asset path for each font style and weight.

- Using Fonts: Once included, specify the font family in your widget's TextStyle property to apply the font, e.g., TextStyle(fontFamily: 'YourCustomFont').

Steps to Include Icons, Images, and Fonts:

- Prepare Your Assets: Gather your icons, images, and font files. For images and custom icons, ensure they are in the correct resolutions and formats. For fonts, ensure you have the necessary licenses to use them in your app.

- Update pubspec.yaml: Include your assets in the pubspec.yaml file. This file is used by Flutter to define assets and dependencies. For each type of asset (icons, images, fonts), specify the paths to the files or directories containing those assets.

- Use Assets in Your App: Once declared in pubspec.yaml, you can use these assets throughout your app. Utilize the appropriate widgets (Icon, Image.asset, Image.network, TextStyle) to display icons, images, and apply fonts.

Code:

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Experiment 3'),
        ),
```

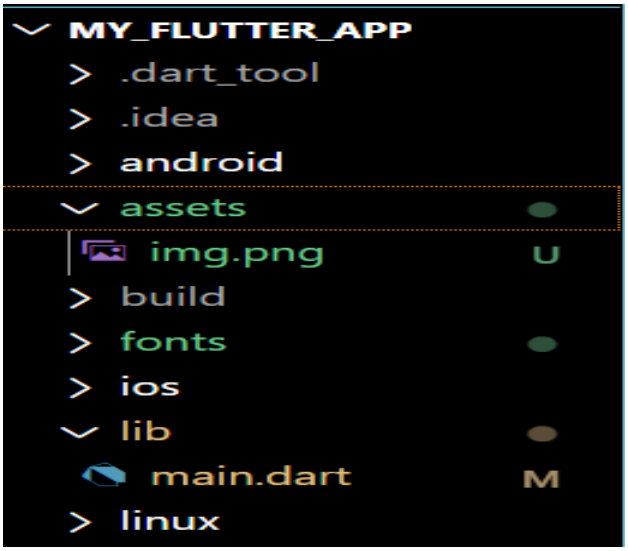
```
body: Center(
  child: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: <Widget>[
      Image.asset('assets/img.png', width: 500),
      const Text(
        'Hello, Kaushik!',
        style: TextStyle(
            fontSize: 50,
            fontWeight: FontWeight.bold,
            color: Colors.blue,
            fontFamily: 'CustomFont'),
      ),
      Container(
        margin: const EdgeInsets.all(20),
        padding: const EdgeInsets.all(10),
        color: Colors.amber,
        child: const Text('Please enter your Message',
            style: TextStyle(
              fontFamily: 'CustomFont',
              color: Colors.red,
              fontSize: 20,
            )),
      ),
      const Padding(
        padding: EdgeInsets.all(8.0),
        child: TextField(
          decoration: InputDecoration(
            border: OutlineInputBorder(),
            labelText: 'Input Field',
          ),
        ),
      ),
      Row(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          const Icon(Icons.send),
```

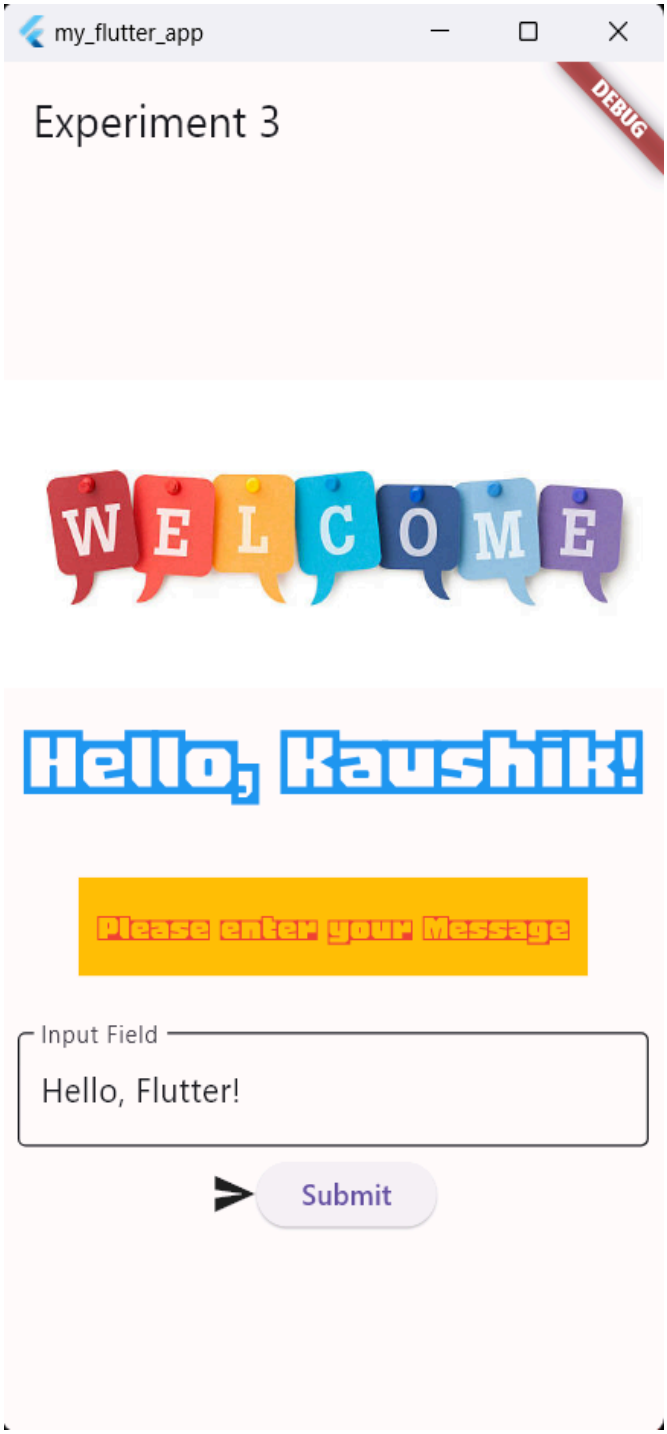
```
                ElevatedButton(
                  onPressed: () {},
                  child: const Text('Submit'),
                ),
                const SizedBox(width: 10),
              ],
            ),
          ],
        ),
      ),
    ),
  );
 }
}
```

Pubspec.yaml:

```
flutter:
  uses-material-design: true
  assets:
    - assets/img.png
  fonts:
    - family: CustomFont
      fonts:
        - asset: fonts/CustomFont.ttf
          weight: 700
```

File structure:





Conclusion:

Including icons, images, and fonts in your Flutter app is a powerful way to enhance its appearance and user experience. By carefully selecting and incorporating these resources, you can create a visually appealing and unique app that stands out. Always remember to manage your assets efficiently and follow best practices for optimal app performance.