Experiment No:03

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

Theory:

Including icons, images, and custom fonts in a Flutter app significantly enhances both its visual appeal and functionality. Here's a detailed explanation of how to integrate these assets into your Flutter project:

Icons: In Flutter, icons are an essential part of UI design, aiding in visual communication and navigation within the app. The Icons class in Flutter provides a comprehensive set of Material Design icons, covering a wide range of commonly used symbols and actions. These icons are vector-based, ensuring crisp rendering across various screen sizes and resolutions. Developers can easily incorporate icons into their app by utilizing the Icon widget, which takes an IconData object representing the desired icon from the Icons class. Additionally, developers can customize the appearance of icons by adjusting properties such as size, color, and opacity, allowing for seamless integration with the app's design language.
Images: images are integral to creating visually engaging and informative user interfaces in Flutter apps. To include images, developers typically place image files in the assets directory of the Flutter project. This directory serves as a centralized location for storing static assets like images, ensuring easy access and management. The Image widget in Flutter provides various constructors for loading images, with Image.asset() being the most commonly used for loading images from the asset bundle. By specifying the asset path within the Image.asset() constructor, developers can display images within their app's UI. Additionally, Flutter supports advanced image features such as caching, resizing, and network image loading, enabling developers to optimize image performance and user experience.
Fonts: Custom fonts play a crucial role in shaping the visual identity and brand recognition of a Flutter app. Flutter allows developers to include custom font files (e.g., TrueType or OpenType) within the project's fonts directory. These font files contain typeface designs that can be applied to text elements within the app. To use custom fonts, developers need to declare them in the pubspec.yaml file under the flutter section using the fonts property. This declaration specifies the font family name and the path to the font file within the project directory. Once declared, developers can apply the custom font to text elements by setting the fontFamily property in the TextStyle widget. By leveraging custom fonts, developers can achieve a unique typographic style that aligns with the app's design aesthetic and brand identity, enhancing readability and user engagement.
In summary, integrating icons, images, and custom fonts into a Flutter app involves a straightforward process that significantly enhances its visual appeal and functionality. By leveraging built-in support for icons, asset management for images, and custom font declarations, developers can create visually stunning and brand-consistent user interfaces that elevate the overall user experience of their Flutter applications.

Code:

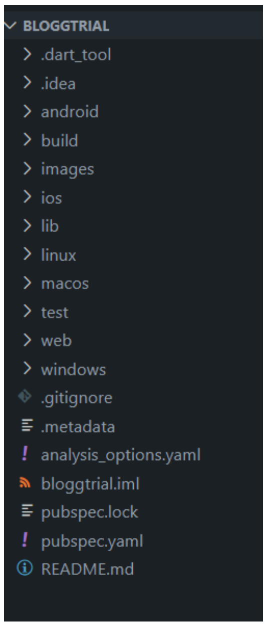
```
import 'dart:async';

import 'package:bloggerapp/sceens/option_screen.dart';
import 'package:flutter/material.dart';

class SplashScreen extends StatefulWidget {
  const SplashScreen({super.key});

  @override
  State<SplashScreen> createState() => _SplashScreenState();
}

class _SplashScreenState extends State<SplashScreen> {
  @override
  void initState() {
    super.initState();
    Timer(Duration(seconds: 5), () {
      Navigator.push(context, MaterialPageRoute(builder: (context) =>
OptionScreen()));
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Column(
       mainAxisAlignment: MainAxisAlignment.center,
       crossAxisAlignment: CrossAxisAlignment.center,
        children: [
          Container(
            color: Colors.blueGrey.shade300 ,
            child: Image(
              height: MediaQuery.of(context).size.height *.3,
              width: MediaQuery.of(context).size.width *.6,
              image: AssetImage('images/blog.png')
              ),
          ),
          const Padding(
            padding: EdgeInsets.symmetric(vertical: 30),
            child: Align(
              alignment: Alignment.center,
              child: Text(
                'Blog',
                style: TextStyle(
                    fontStyle: FontStyle.italic,
                    fontSize: 30,
                    fontWeight: FontWeight.w300),
              ),
            ),
           )
        ],
      ),
    );
  }
}
```

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

import 'package:bloggerapp/components/round_button.dart';



class RegisterScreen extends StatefulWidget {

  const RegisterScreen({super.key});
  @override
  State<RegisterScreen> createState() => _RegisterScreenState();
}


class _RegisterScreenState extends State<RegisterScreen> {
  @override
  Widget build(BuildContext context) {
    return  Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.blueGrey,
        centerTitle: true,
        title: const Text("Create Account", style: TextStyle(color:
Colors.white),
        ),
      ),
      body:  Padding(
        padding: const EdgeInsets.symmetric(horizontal: 20),


        child: Column(
            crossAxisAlignment: CrossAxisAlignment.center,

            mainAxisAlignment: MainAxisAlignment.center,

            children: [

              const Text(

                'Register',

                style: TextStyle(fontWeight: FontWeight.bold, fontSize:
35),

                ),

                Padding(

                padding:const EdgeInsets.symmetric(vertical: 30),

                child: Column(

                  children: [

                    TextFormField(

                      keyboardType: TextInputType.emailAddress ,

                      decoration: const InputDecoration(

                        hintText: 'Email',

                        labelText: 'Email',

                        prefixIcon: Icon(Icons.email),

                        border: OutlineInputBorder(),

                      )
                ),

                Padding(

                padding:const EdgeInsets.symmetric(vertical: 15),

                child: TextFormField(


                      keyboardType: TextInputType.visiblePassword ,

                      decoration: const InputDecoration(

                        hintText: 'Password',

                        labelText: 'Password',

                        prefixIcon: Icon(Icons.lock),

                        border: OutlineInputBorder(),

                      ),

                ),

                ),

                RoundButton(

                  title: 'Register',

                  onPress:() {}

                  )

                  ],
                ),

                ),

            ],

        ),

      ),

    );

  }

}
```

Pubsec.yaml:

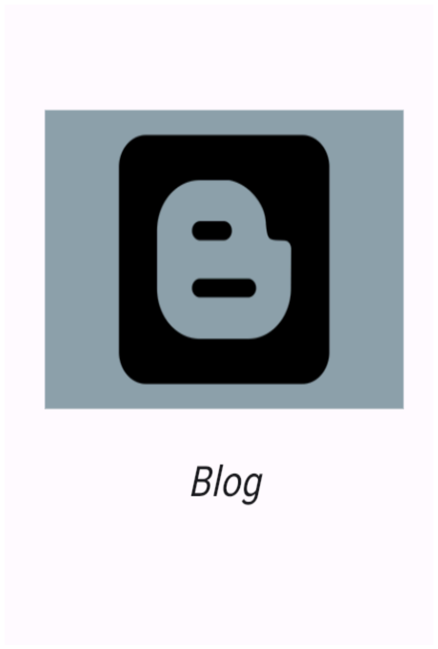
```
assets:
 - images/

fonts:
 - family: Roboto
   fonts:
     - asset: fonts/Roboto-BoldCondensedItalic.ttf
```

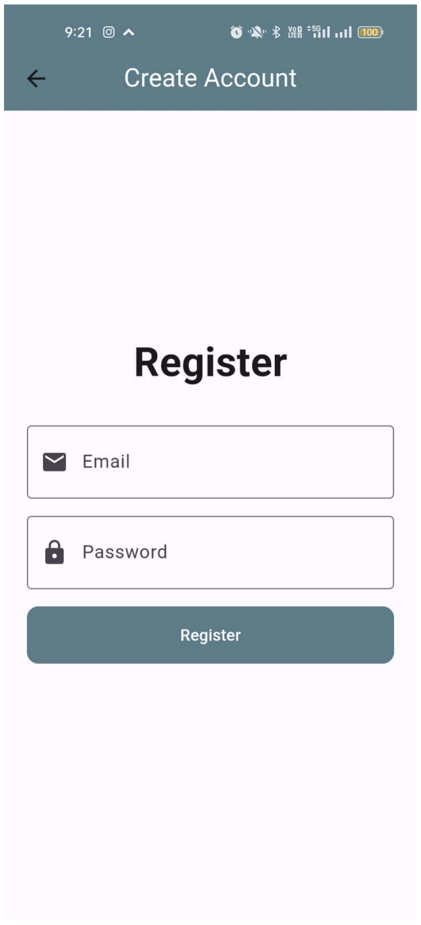
File Structure :



```
BLOGGTRIAL
  > .dart_tool
  > .idea
  > android
  > build
  > images
  > ios
  > lib
  > linux
  > macos
  > test
  > web
  > windows
  .gitignore
  .metadata
  analysis_options.yaml
  bloggtrial.iml
  pubspec.lock
  pubspec.yaml
  README.md
```

Output:





Conclusion:

I have successfully understood and implemented the images , fonts and Icons in a Flutter

Application.