Experiment 3

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

Theory:

Icons:

Icons in Flutter are primarily used to represent actions or features in the user interface. Flutter provides a wide range of built-in icons through the Icons class. These icons cover common actions, such as navigation, file operations, communication, and more.

To use icons in your Flutter app, follow these steps:

- Import the material.dart package if you haven't already, as it contains the Icons class.
- Use the Icon widget to display an icon. You can set the icon using the Icons class directly or by specifying a custom icon.Optionally, wrap the Icon widget with other widgets like IconButton to add interactivity.

Example:

```
import 'package:flutter/material.dart';
        // Using a built-in icon
Icon(Icons.home)

// Using a custom icon
Icon(
 IconData(0xe801, fontFamily: 'MyCustomIcons'), // Replace 0xe801 with your icon's code point
 size: 24.0,
 color: Colors.red,
)
```

- Images:
  Images play a crucial role in enhancing the visual appeal of your Flutter app. Flutter supports various image formats, including JPEG, PNG, GIF, WebP, and animated WebP.
  Here's how to use images in your Flutter app:
  - Store your image files in the project directory. Typically, images are stored under the assets folder.
  - Declare the image assets in the pubspec.yaml file to make them accessible within your app.
  - Use the Image widget to load and display the images. You can use either Image.asset for images stored in the assets folder or Image.network for images fetched from the internet.

Example:

Flutter:
assets:
- assets/images/my_image.png

Image.asset('assets/images/my_image.png')

- Fonts:
  Custom fonts allow you to give your Flutter app a unique typographic identity. You can use custom fonts for text styling throughout your app.
  To include custom fonts in your Flutter app:
  - Add the font files (usually .ttf or .otf files) to your project directory.
  - Declare the font files in the pubspec.yaml file, specifying the font family name and the file paths.
  - Apply the custom font using the fontFamily property of the TextStyle widget when styling text.
- Example:

```
flutter:
 fonts:
 - family: MyCustomFont
 fonts:
 - asset: assets/fonts/my_font.ttf
```

dart
Copy code

```
Text(
 'Custom Font Text',
 style: TextStyle(
 fontFamily: 'MyCustomFont',
 fontSize: 20.0,
 ),
)
```

Code:

Make a folder screens in the lib folder. Under screens folder make a home.dart,feed.dart,post_screen.dart,favorite_screen.dart,post_screen.dart,sear ch.dart,profile_screen.dart

Main.dart

```
import 'package:flutter/material.dart';
 import 'screens/login.dart'; 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(), home:const
LoginScreen(), );

} }
```

Favorite_Screen.dart

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

import '../model/suggested_follower.dart';

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

  @override
  State<FavoriteScreen> createState() => _FavoriteScreenState();
}

class _FavoriteScreenState extends State<FavoriteScreen> {
  int _currentIndexTab = 0;

  @override
  Widget build(BuildContext context) {
    return DefaultTabController(
      length: 3,
      child: Scaffold(
        appBar: AppBar(
          elevation: 0,
          backgroundColor: Colors.white,
          title: const Text(
            'Activity',
```

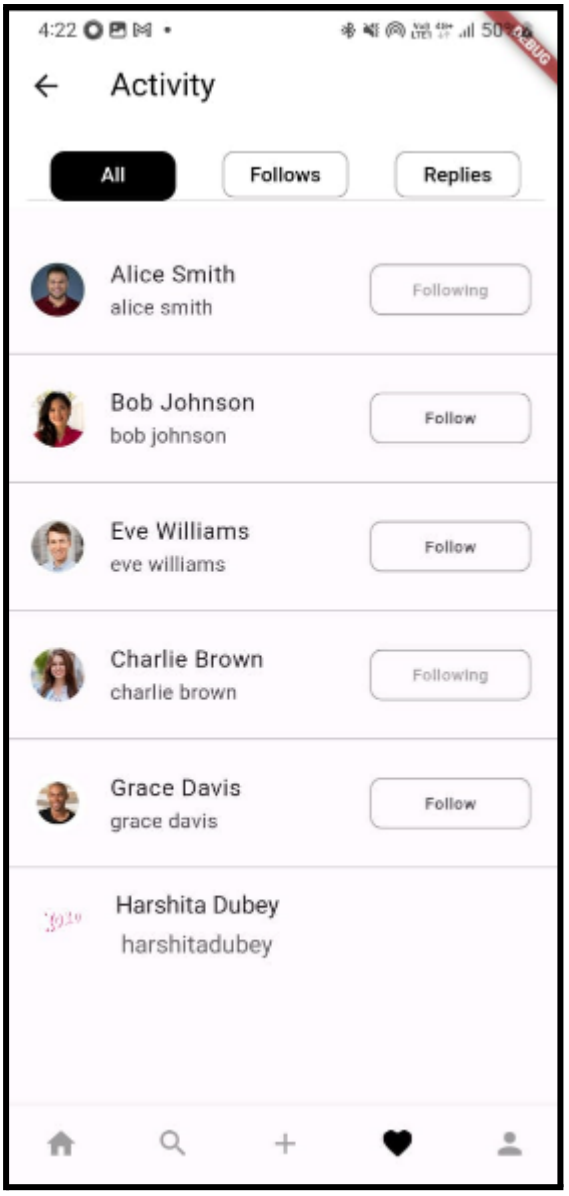
```
  style: TextStyle(color: Colors.black),
),
bottom: PreferredSize(
  preferredSize: Size.fromHeight(AppBar().preferredSize.height),
  child: Container(
    height: 44,
    padding: const EdgeInsets.symmetric(horizontal: 15, vertical: 5),
    child: TabBar(
      labelColor: Colors.white,
      unselectedLabelColor: Colors.black,
      indicator: BoxDecoration(
          borderRadius: BorderRadius.circular(10),
          color: Colors.black),
      onTap: (index) => setState(() => _currentIndexTab = index),
      tabs: [
        Container(
          width: double.infinity,
          decoration: BoxDecoration(
            border: Border.all(
                color: _currentIndexTab == 0
                    ? Colors.transparent
                    : Colors.grey),
            borderRadius: BorderRadius.circular(8),
          ),
          child: const Tab(text: 'All'),
        ),
        Container(
          width: double.infinity,
          decoration: BoxDecoration(
            border: Border.all(
                color: _currentIndexTab == 1
                    ? Colors.transparent
                    : Colors.grey),
            borderRadius: BorderRadius.circular(8),
          ),
          child: const Tab(text: 'Follows'),
        ),
        Container(
          width: double.infinity,
          decoration: BoxDecoration(
            border: Border.all(
                color: _currentIndexTab == 2
                    ? Colors.transparent
                    : Colors.grey),
```

```
                  borderRadius: BorderRadius.circular(8),
                ),
                child: const Tab(text: 'Replies'),
              ),
            ],
          ),
        ),
      ),
      body: Padding(
        padding: const EdgeInsets.only(top: 20.0),
        child: TabBarView(
          children: [
            Column(
              children: [
                ...suggestedFollowers.map((follower) {
                  return SuggestedFollowerWidget(follower: follower);
                }).toList()
              ],
            ),
            const Center(child: Text('Nothing to see here yet')),
            const Center(child: Text('Nothing to see here yet')),
          ],
        ),
      ),
    ),
  );
 }
}

class SuggestedFollowerWidget extends StatelessWidget {
 const SuggestedFollowerWidget({super.key, required this.follower});

 final SuggestedFollower follower;
 @override
 Widget build(BuildContext context) {
  return Column(
   children: [
    ListTile(
     leading: CircleAvatar(
      backgroundImage: AssetImage(follower.profileImageUrl),
      backgroundColor: Colors.white,
     ),
     title: Text(follower.username),
```

```
          subtitle: Text(follower.username.toLowerCase()),
          trailing: Row(
            mainAxisSize: MainAxisSize.min,
            children: [
              Container(
                alignment: Alignment.center,
                width: 110,
                height: 35,
                decoration: BoxDecoration(
                  border: Border.all(color: Colors.grey),
                  borderRadius: BorderRadius.circular(8),
                ),
                child: follower.isFollowing
                    ? const Text(
                        'Following',
                        style: TextStyle(color: Colors.grey),
                      )
                    : const Text('Follow'),
              )
            ],
          ),
        ),
        const Divider(),
      ],
    );
  }
}
```



CONCLUSION

In conclusion, integrating icons, images, and custom fonts into my Flutter app enhances its visual appeal, usability, and brand identity.