Experiment No. 3

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

Theory:
To include icons, images, and fonts in a Flutter app, you need to understand the following core concepts related to asset management in Flutter. Here's the theory behind including these resources:

1. **Assets in Flutter:**
   Assets are files or resources such as images, fonts, icons, or sounds that you include in your app and bundle within the app package. In Flutter, you can include these assets in your project and then use them in your app.

2. **Adding Assets to pubspec.yaml:**
   In Flutter, you declare assets in the pubspec.yaml file. This is where you specify which assets should be bundled with your app during the build process.
   Example for adding assets:

   ```
   flutter:
     assets:
       - assets/images/
       - assets/icons/
   ```

   In this example, the images are stored in the assets/images directory, and icons in the assets/icons directory. You can also specify specific files instead of directories.

3. **Including Images:**
   Flutter provides several ways to include images in your app, including network images, asset images, and file images. To use asset images, you reference them by their file path relative to the assets directory.

   Example of including an asset image:

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

   For this to work, the image (my_image.png) must be listed in the pubspec.yaml file under the flutter section, like this:

```
flutter:
  assets:
    - assets/images/my_image.png
```

4. Including Icons:
Flutter allows you to use custom icons in your app. You can add icon files (e.g., .png or .svg) to your assets folder and use them in the app. Alternatively, Flutter provides built-in icons via the Icons class.

Example of using an asset icon:
```
Image.asset('assets/icons/my_icon.png')
```

5. **Including Fonts:**
To include custom fonts, you place your font files (e.g., .ttf or .otf files) in a folder inside your assets directory. Then, you declare these fonts in the pubspec.yaml file and use them in your app.

Example of adding custom fonts in pubspec.yaml:
```
flutter:
  fonts:
    - family: CustomFont
      fonts:
        - asset: assets/fonts/CustomFont-Regular.ttf
        - asset: assets/fonts/CustomFont-Bold.ttf
```

6. Font Weight and Style:
When specifying fonts, you can also define specific font weights and styles (like bold, italic) to support different text styles in your app.

7. Working with Icon Libraries:
While you can use custom icon files, Flutter also supports popular icon libraries like FontAwesome, MaterialIcons, etc. For example, Flutter's built-in Icons class provides access to the Material Design icons.

Example of using a Material icon:
```
Icon(Icons.home)
```

8. Caching and Optimization:
● **Images:** Flutter caches images, but you might want to use libraries like cached_network_image for better image loading and caching.

● **Fonts:** Custom fonts are loaded from assets when the app is first started, and they remain available for the lifecycle of the app.

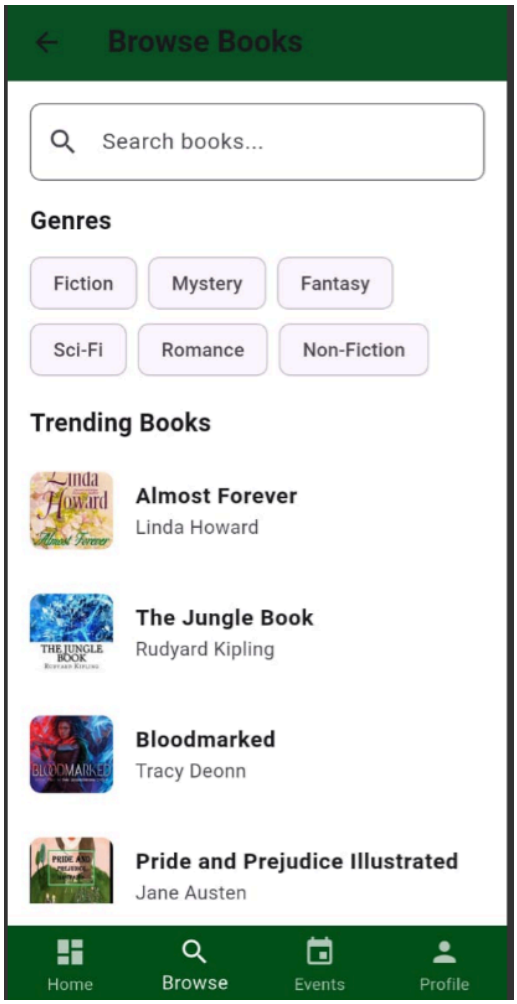
9. SVG Images:

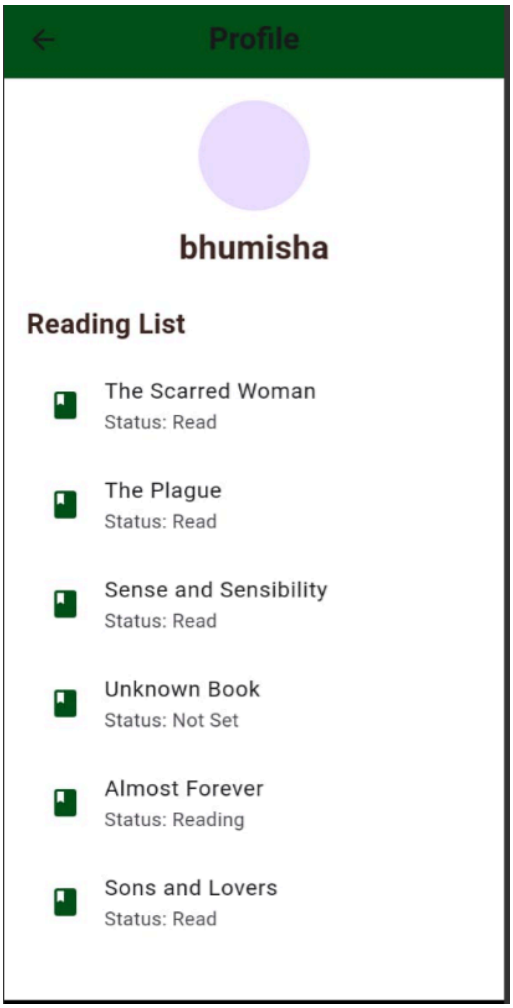
If you want to use vector-based images (like SVG), you can include them using packages like flutter_svg, which allows you to display scalable vector graphics (SVG files) in Flutter.

Example of including an SVG:

```
import 'package:flutter_svg/flutter_svg.dart';
SvgPicture.asset('assets/icons/my_icon.svg')
```

Screenshots:





Code Snippets:

Fonts:

```
Text(
  userData!["name"] ?? "Unknown User",
  style: const TextStyle(
    fontSize: 24,
    fontWeight: FontWeight.bold,
    color: Color(0xFF3E2723),
  ),
),
```

```
title: const Text(
  "Profile",
  style: TextStyle(
    fontSize: 22,
    fontWeight: FontWeight.bold,
  ),
),
```

Icons:

```
leading: const Icon(
  Icons.book,
  color: Color(0xFF04511A),
),
```

Image:

```
CircleAvatar(
  radius: 40,
  backgroundImage: userData!["profilePic"] != null
      ? NetworkImage(userData!["profilePic"])
      : const AssetImage("assets/media/default_avatar.png") as ImageProvider,
),
```

Conclusion:

In conclusion, incorporating icons, images, and fonts into a Flutter app significantly enhances the user interface and user experience. By correctly managing assets through the pubspec.yaml file and understanding how to use built-in and custom resources, developers can build visually appealing and functional applications. Whether using asset images, network images, Material icons, SVGs, or custom fonts, Flutter provides flexible and efficient ways to integrate these elements. Mastery of these features empowers developers to design clean, modern, and engaging mobile applications that align with branding and usability standards.