Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

Experiment 3

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

Theory:
In modern app development, visual design plays a crucial role in engaging users and delivering an intuitive experience. In Flutter, a UI toolkit developed by Google, the integration of icons, images, and fonts is streamlined to allow developers to build beautiful, branded, and responsive applications across multiple platforms using a single codebase.

1. Icons in Flutter
Icons are compact, universally understood symbols that visually represent actions, types of content, or app functionality. They enhance navigation and improve usability without requiring text labels for every element.

Flutter provides two main built-in icon sets:

- Material Icons – Follows Google's Material Design guidelines and is typically used for Android-style apps. These icons are accessed via the Icons class.
- Cupertino Icons – Designed for iOS-style interfaces, following Apple's Human Interface Guidelines. These are accessed using the CupertinoIcons class.

Key Features:

- Scalable and resolution-independent
- Customizable in terms of color, size, and style
- Hardware-accelerated for smooth rendering

2. Image Integration
Images are essential for visual storytelling, branding, and improving user interaction. Flutter supports image rendering from multiple sources, each suitable for different use cases:

- Asset Images: Stored locally within the application bundle. Ideal for logos, static illustrations, and decorative UI components. Must be declared in the pubspec.yaml file.
- Network Images: Loaded dynamically from the internet via URLs. Useful for content that changes frequently or is fetched from APIs.
- Memory Images: Rendered from raw byte data. These are used in advanced use cases like dynamically generated images or images received from databases/APIs in base64 format.

Considerations for Using Images:

- Optimize image size and resolution for performance
- Use caching strategies for network images
- Handle loading errors and placeholders gracefully

Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

3. Font Management

Fonts define the style and tone of textual content and contribute significantly to the overall aesthetic and branding of an app. Flutter allows the integration of custom fonts, enabling developers to move beyond the default system fonts.

Steps to Add Custom Fonts:

1. Place .ttf or .otf files in the project directory (commonly under assets/fonts/).
2. Declare the font in the pubspec.yaml file.
3. Apply the font in widgets using the TextStyle class.

Benefits of Custom Fonts:

- Support for different font weights and styles (e.g., bold, italic, light)
- Enhances readability and visual hierarchy
- Aligns with branding guidelines for a cohesive look across platforms

Code:

1. **Images:**

```
child: CircleAvatar(
    radius: 60,
    backgroundImage: AssetImage('assets/images/Profile.jpg'),
    backgroundColor: Colors.grey[300],
  ),
```

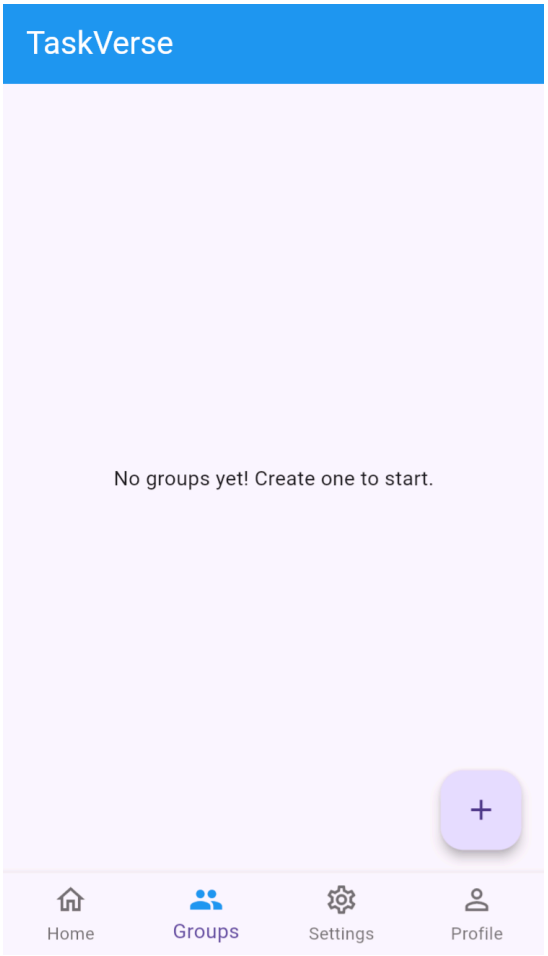
Asset image:

Pubsec.yaml:

```
uses-material-design: true
assets:
  - assets/images/Profile.jpg
  - assets/images/TaskVerse_Splash.png
```

2. **Fonts and Text**

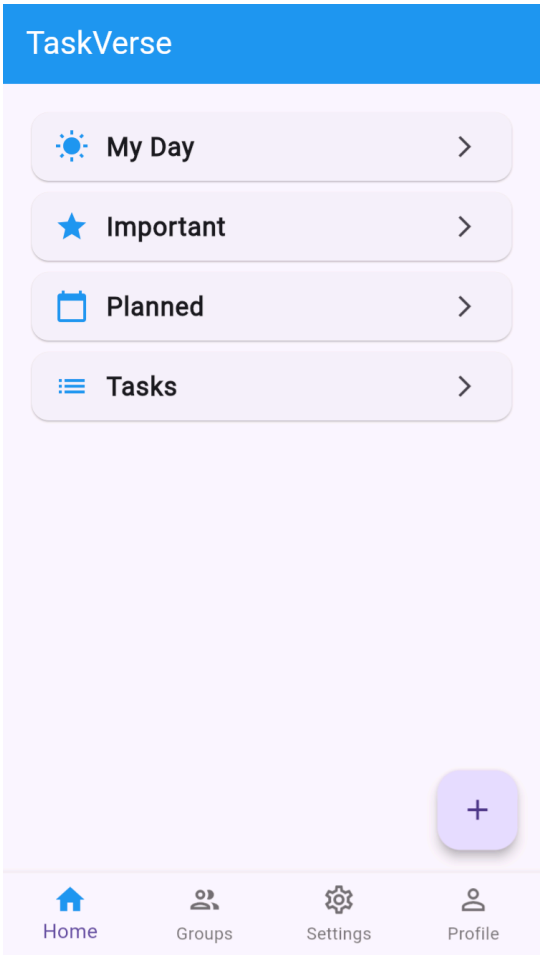
```
title: Text(
            widget.task.title,
            style: TextStyle(
              fontSize: 16,
              decoration: widget.task.isCompleted ? TextDecoration.lineThrough : null,
            ),
          ),
Text(
                    "${DateFormat('EEE, MMM d').format(widget.task.dueDateTime)} at
${DateFormat('hh:mm a').format(widget.task.dueDateTime)}",
              style: const TextStyle(fontSize: 14, color: Colors.grey),
            ),
```



3. **Icons**

```
BottomNavigationBarItem(
            icon: const Icon(Icons.home_outlined),
            activeIcon: const Icon(Icons.home, color: Colors.blue),
```

```
            label: AppLocalizations.of(context).home,
          ),
BottomNavigationBarItem(
            icon: const Icon(Icons.group_outlined),
            activeIcon: const Icon(Icons.group, color: Colors.blue),
            label: AppLocalizations.of(context).groups,
          ),
```



GitHub Link: https://github.com/BKCODE2003/ToDo

Conclusion:

This experiment demonstrated how to effectively integrate icons, images, and fonts in a Flutter app to enhance UI/UX.By successfully including icons, images, and custom fonts in a Flutter application, you have learned how to enhance the visual design and branding of your app. Understanding how to manage assets and configure them in the pubspec.yaml file is essential for creating a polished and engaging user interface. This knowledge enables you to build more dynamic and visually appealing applications tailored to user preferences and design standards.