Name:Komal Singh
Div:D15B
Roll No: 56

Practical:3

Aim:

To include icons, images, and custom fonts in a Flutter application to enhance its visual appearance and user experience.

Theory:

Flutter allows developers to integrate various media assets such as icons, images, and fonts to create visually appealing applications. These assets can be added from local files or external sources.

1. **Icons in Flutter:**
    - Flutter provides built-in icons through the `Icons` class (`Icons.home`, `Icons.settings`, etc.).
    - Custom icons can be added using packages like `flutter_launcher_icons` or `FontAwesomeIcons`.
2. **Images in Flutter:**

Asset Images: Stored in the `assets` folder and defined in `pubspec.yaml`. Example:
dart
CopyEdit
```
Image.asset('assets/images/logo.png')
```

    -

Network Images: Loaded from a URL. Example:
dart
CopyEdit
```
Image.network('https://example.com/image.png')
```

    -
    - **Memory & File Images:** Used for dynamic images stored in device memory.
3. **Fonts in Flutter:**
    - Custom fonts can be added by including `.ttf` or `.otf` files in the `assets/fonts` directory.

Defined in `pubspec.yaml`:
yaml

CopyEdit

```
fonts:

  - family: CustomFont

    fonts:

      - asset: assets/fonts/CustomFont-Regular.ttf

      - asset: assets/fonts/CustomFont-Bold.ttf

        weight: 700
```

-

Applied using `TextStyle`:
dart
CopyEdit

```
Text('Hello', style: TextStyle(fontFamily: 'CustomFont'))
```

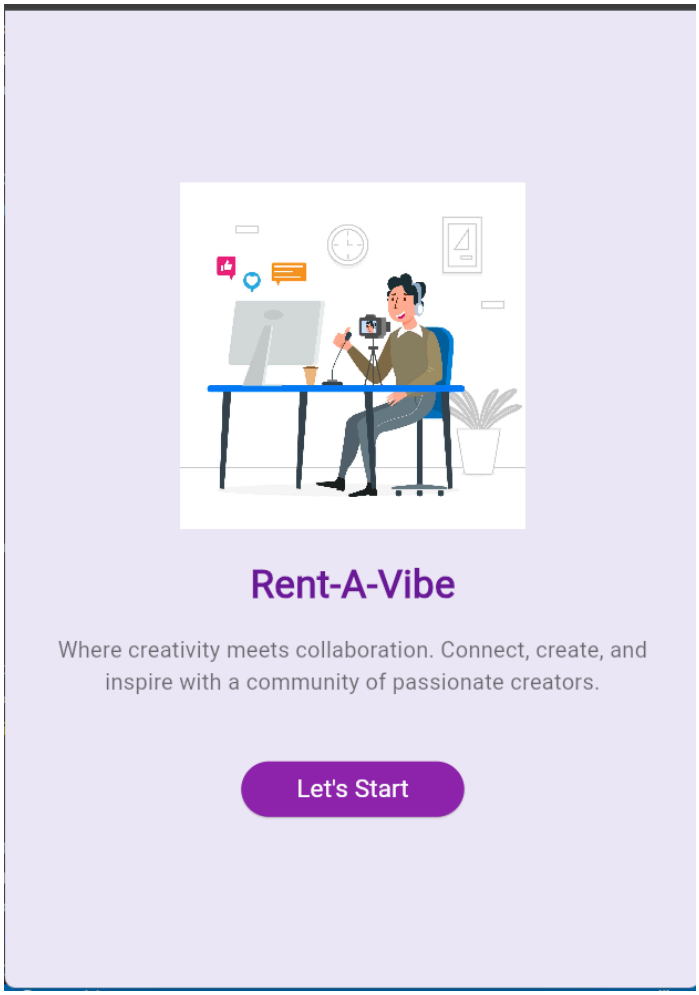
-

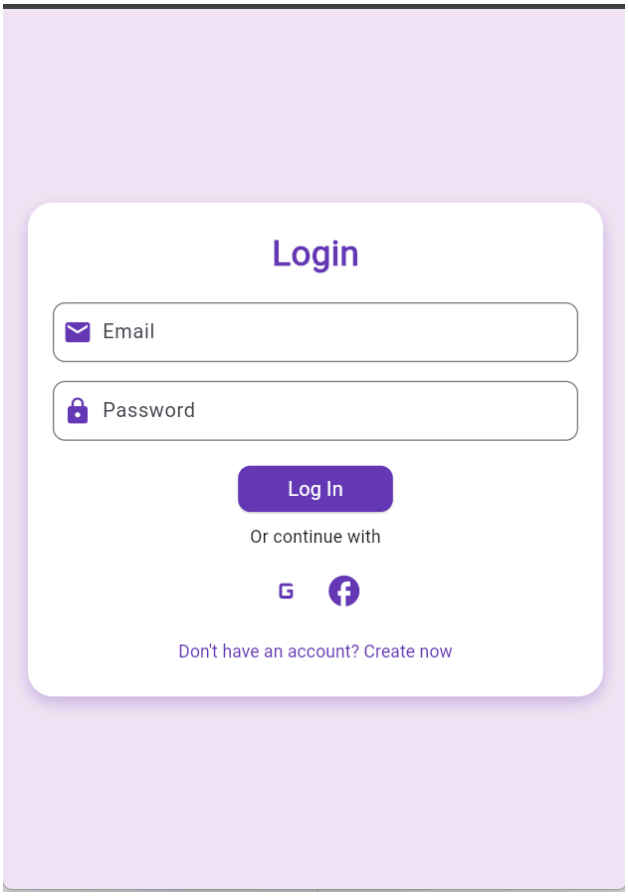
By configuring these assets correctly, developers can enhance the UI/UX of a Flutter application.

Steps to Implement the Start and Login Screens in Flutter

1. **Project Setup**
    - Created a Flutter project and organized it with separate Dart files (start.dart, login_screen.dart, etc.).
2. **Start Screen Implementation**
    - Used StatefulWidget with Scaffold and SafeArea.
    - Added an image using Image.asset('assets/start.jpg'), ensuring the image is placed inside the assets folder and declared in pubspec.yaml.
    - Styled the title text using TextStyle with fontSize, fontWeight, and color.
    - Wrapped the description text inside a Padding widget for spacing.
    - Created a button using ElevatedButton with styleFrom for custom color, padding, and shape.
    - Used Navigator.push to navigate to LoginScreen when the button is pressed.
3. **Login Screen Implementation**
    - Created a StatefulWidget and used TextEditingController for email and password input.

- Wrapped UI inside a Container with BoxDecoration, giving it a white background, rounded corners, and a shadow effect.
- Added input fields using TextField, each having InputDecoration with labelText, prefixIcon, and OutlineInputBorder for styling.
- Used an ElevatedButton with custom styling for login functionality, calling the login() function when pressed.
- Implemented authentication using AuthService.signIn(), navigating to HomeScreen on success or showing a SnackBar on failure.
- Included social login icons using IconButton with Icons.g_mobiledata and Icons.facebook, styled with size and color.
- Provided a TextButton for navigation to the registration screen using Navigator.push.





Conclusion:

Including icons, images, and fonts in a Flutter app improves its visual appeal and usability. Proper asset management ensures efficient loading and better performance, enabling developers to create visually rich and engaging applications.