EXPERIMENT NO. 3
MAD and PWA Lab

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

Theory:

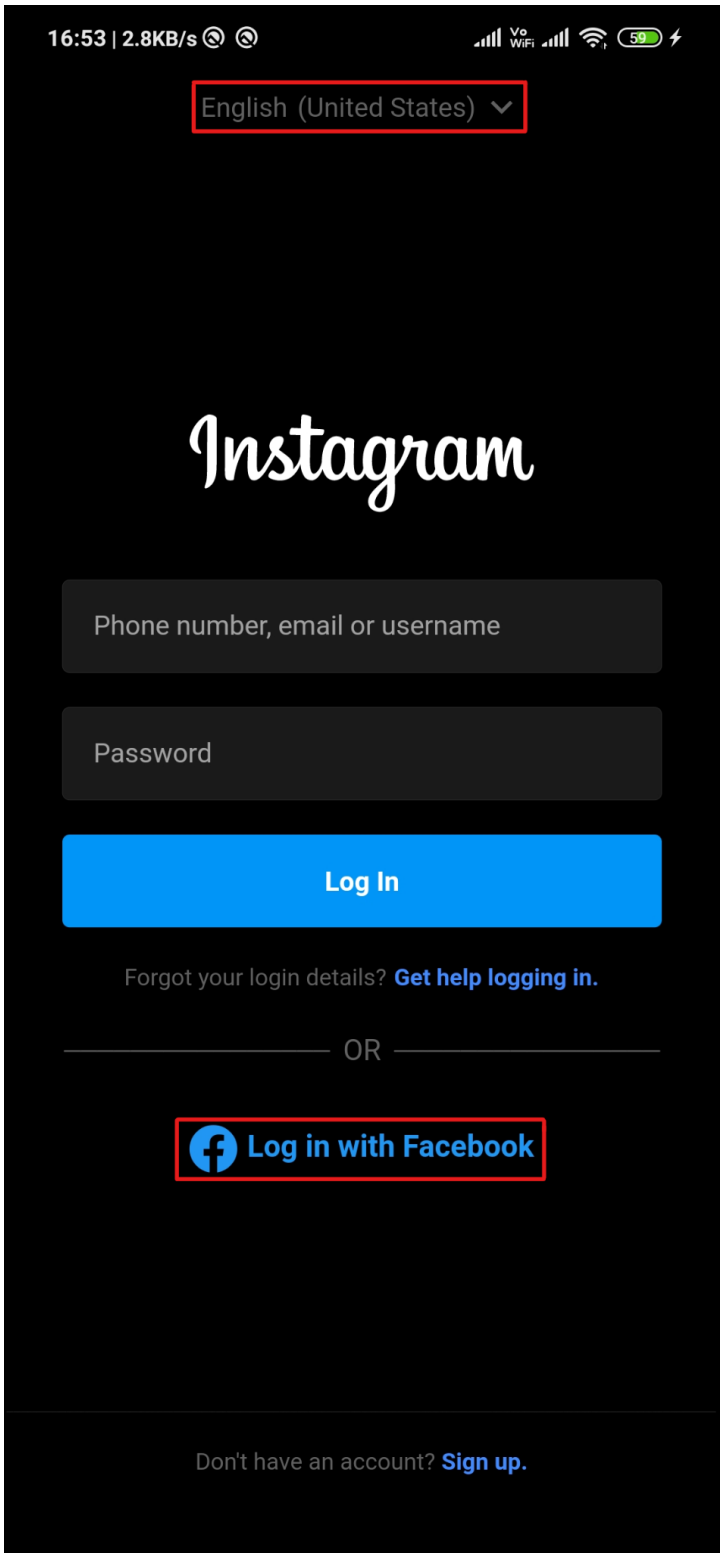
An **Icon** is a graphic image representing an application or any specific entity containing meaning for the user. It can be selectable and unselectable. Flutter provides an **Icon widget** to create icons in our applications. We can create icons in Flutter, either using inbuilt icons or with the custom icons. Flutter provides the list of all icons in the Icons class.

To use this class, make sure you set **uses-material-design: true** in your project's **pubspec.yaml** file in the flutter section. This ensures that the Material Icons font is included in your application. This font is used to display the icons.

Property	Description
icon	It is used to specify the icon name to display in the application. Generally, Flutter uses material design icons that are symbols for common actions and items.
color	It is used to specify the color of the icon.
size	It is used to specify the size of the icon in pixels. Usually, icons have equal height and width.
textDirection	It is used to specify to which direction the icon will be rendered.

In the following **Login Screen** of the application (Instagram), I have used two Icons.

```
Icon(
      Icons.expand_more,                // Icon on the top of the screen
      color: Color(0xff616161),
),
Icon(
      FontAwesomeIcons.facebook,        // Icon below the OR text
      color: Colors.blue,
      size: 27,
),
```



The first **icon** selected is `expand_more` which is a downward arrow icon. The expand_more icon name is selected from the list of the icons that Flutter provides.

The **color** selected is a hex code of `0xff616161` which is a light grey color.

The second **icon** selected is `facebook` which is a facebook logo. The facebook icon is selected from the **FontAwesomeIcons** list which is imported from the package `font_awesome_flutter` which is installed separately.

The **color** selected is blue which is from the colors.dart file provided by the Flutter.

The **size** provided to the icon is 27 pixels.

Displaying **Images** is the fundamental concept of most of the mobile apps. Flutter has an **Image widget** that allows displaying different types of images in the mobile application.

Flutter apps can include both code and assets (sometimes called resources). An asset is a file that is bundled and deployed with your app, and is accessible at runtime. Common types of assets include static data (for example, JSON files), configuration files, icons, and images (JPEG, WebP, GIF, animated WebP/GIF, PNG, BMP, and WBMP).

How to display the image in Flutter:
To display an image in Flutter, do the following steps :

1. First, we need to create a new folder inside the root of the Flutter project and name it assets. We can also give it any other name if you want.
2. Next, inside this folder, add an image manually.
3. Update the pubspec.yaml file. Suppose the image name is image1.jpg, then pubspec.yaml file will be:

```
assets:
    - assets/image1.jpg
```

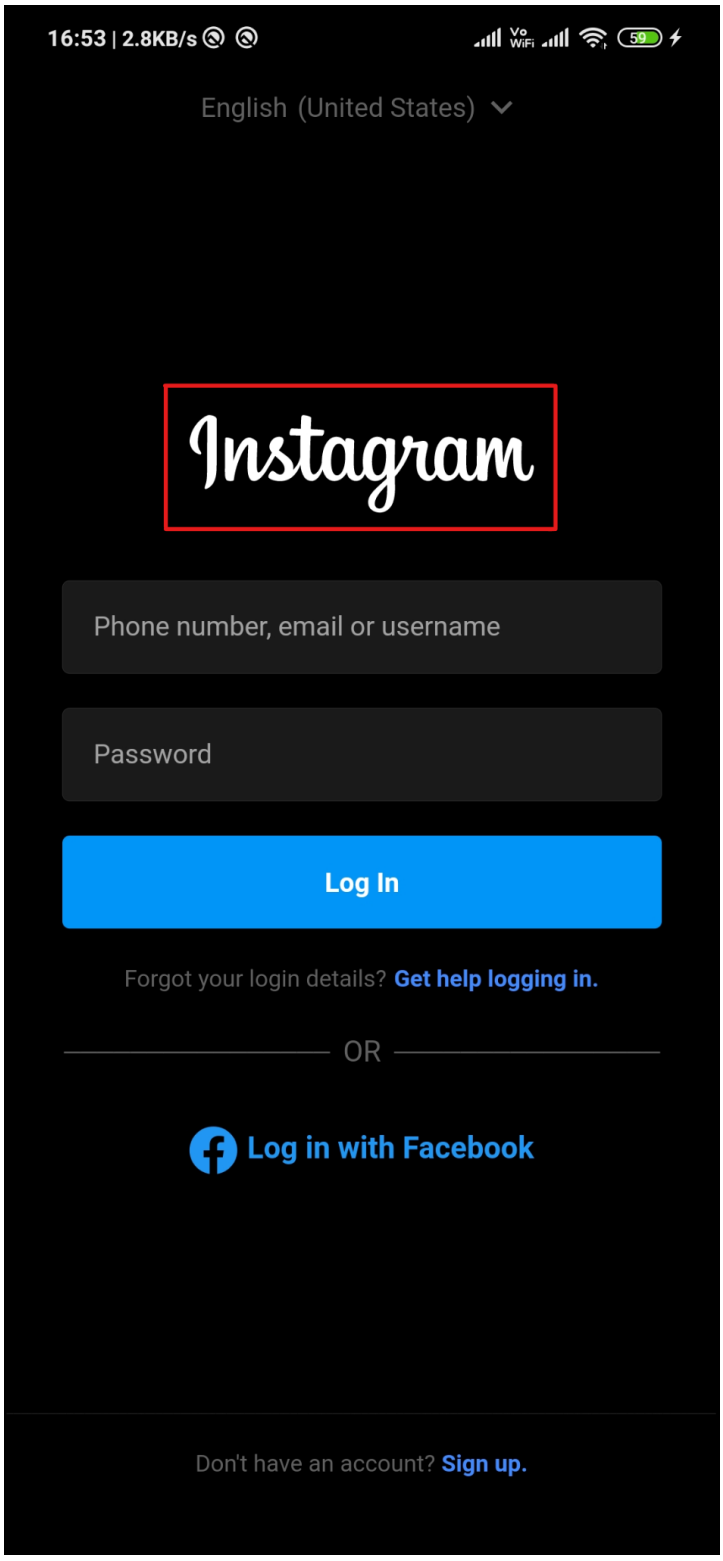
Displaying images from the internet or network is very simple. Flutter provides a built-in method **Image.network** to work with images from a URL. The Image.network method also allows you to use some optional properties, such as height, width, color, fit, and many more.

When we display an image, it simply pops onto the screen as they are loaded. It does not assume usefulness between the users. We can display an image Fade-In Images to overcome this issue.

The Image uses a **FadeInImage widget** that shows a placeholder image while the target image is loading, then fades in the new image when it loads. The FadeInImage can work with various types of images, such as local assets, in-memory, or images from the internet.

In the following **Login Screen** of the application (Instagram), I have used a SVG image to display. It is similar to the **Image** class.

```
SvgPicture.asset(
      'assets/ic_instagram.svg',
      color: primaryColor,
      height: 54,
),
```



First I have used a **SVG image** to display in the center of the screen. Therefore I have made a folder called **assets** and then added this image in that folder.

Next, I updated the **pubspec.yaml** file where I uncommented the line of assets and added the following line :

```
# To add assets to your application, add an assets section, like this:
  assets:
    - assets/ic_instagram.svg
```

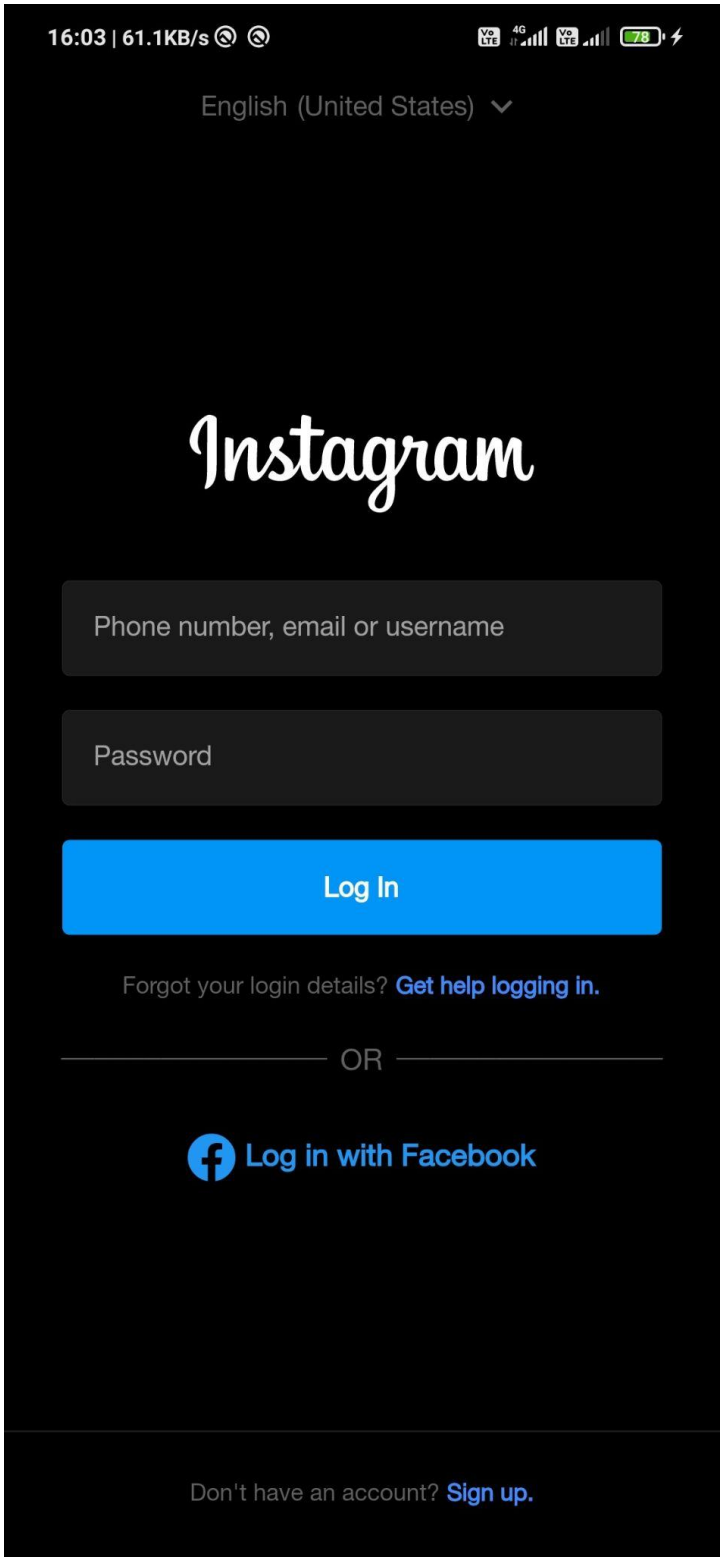
After this, I used the class **SvgPicture.asset()**. This widget will parse SVG data into a Picture using a PictureProvider. To use this widget, we need to install and import the package `flutter_svg` in the login screen dart file.

The **color** selected is **primaryColor** which is called from the colors.dart file which is provided by the Flutter. So primaryColor is white color.

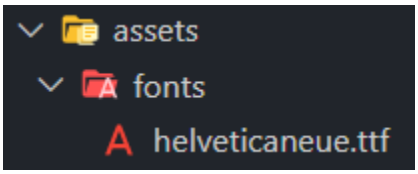
The **height** selected is 54 pixels. The **width** is kept by default as the image has.

Although Android and iOS offer high quality system **fonts**, one of the most common requests from designers is for custom fonts. Flutter works with **custom fonts** and you can apply a custom font across an entire app or to individual widgets.

In the following **Login Screen** of the application (Instagram), I have used Helvetica Neue font everywhere.



To work with a font, **import the font files** into the project. It's common practice to put font files in a fonts or assets folder at the root of a Flutter project.



Flutter supports the following font formats :

- .ttf
- .otf

Once you've identified a font, tell Flutter where to find it. You can do this by including a **font definition** in the **pubspec.yaml** file.

```
fonts:
    - family: Helvetica
      fonts:
        - asset:
assets/fonts/helveticaneue.ttf
```

You have two options for how to apply fonts to text :

1. Default font for everything
2. Font only within specific widgets

To use a **font as the default**, set the fontFamily property as part of the app's theme. The value provided to fontFamily must match the family name declared in the pubspec.yaml.

```
theme: ThemeData(
      fontFamily: 'Helvetica',
      brightness: Brightness.dark,
      scaffoldBackgroundColor: mobileBackgroundColor,
),
```

If you want to apply the **font to a specific widget**, such as a Text widget, provide a TextStyle to the widget.

```
child: Text(
      'This font is Helvetica Neue',
      style: TextStyle(
            fontFamily: 'Helvetica',
      ),
),
```

I have used the font as default for my application.

Conclusion: Hence, we understood how to include icons, images, fonts that have been used in making the Login Screen of our application (Instagram).