

### MPL - 3

**AIM:-** To add advanced Flutter UI by including widgets like Image, Fonts, Icons.

#### **THEORY:-**

Flutter provides robust mechanisms for working with images, fonts, and icons in your app's user interface. Here's a summary of their functionalities and considerations:

##### **Images:**

- **Loading and Displaying:** Use the Image widget to load and display images from various sources like assets, network URLs, or files. Adjust properties like fit, alignment, and opacity for customization.
- **Asset Management:** Store images within your app's assets directory (usually under assets/images/). Flutter automatically handles different screen resolutions and densities.
- **Network Images:** Use the Image.network constructor to directly load images from URLs. Ensure proper internet connectivity and consider caching mechanisms for efficiency.
- **Caching and Performance:** Flutter automatically caches downloaded images. For complex scenarios, explore advanced caching libraries like cached\_network\_image.

##### **Fonts:**

- **Using System Fonts:** Access system fonts available on the device using the Text widget's fontFamily property.
- **Custom Fonts:** Include custom fonts in your app's pubspec.yaml file and integrate them using the GoogleFonts package or by loading font files manually.
- **Font Styling:** Control font properties like size, weight, color, and more using the TextStyle class within the Text widget.
- **Text Layouts and Effects:** Flutter offers rich text editing and layout features. Explore properties like textAlign, overflow, and textSpan for advanced text formatting and effects.

## Icons:

- **Material Icons:** Flutter provides built-in access to a vast collection of Material Design icons through the Icons class. Use them with the Icon widget for simple icon display.
- **Custom Icons:** You can create custom vector icons or use icon fonts. Popular packages like flutter\_icons and font\_awesome\_flutter provide diverse icon sets.
- **Icon Styling:** Modify icons' colors, sizes, and other properties directly through the Icon widget's parameters
- **Animations and Interactions:** Integrate icon animations and interactions using gestures, animations, and state management techniques.

Code:

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

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'FreeLancer',
      theme: ThemeData.light(), // Changed theme to light
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Upworkr'),
        ),
        body: Center(
```

```
child: Column(  
  mainAxisAlignment: MainAxisAlignment.center,  
  children: [  
    const Text(  
      'Welcome to Upwork',  
      style: TextStyle(  
        fontSize: 24,  
        fontWeight: FontWeight.bold,  
      ),  
    ),  
    SizedBox(height: 20),  
    Icon(  
      Icons.work,  
      size: 50,  
      color: Colors.blue,  
    ),  
    SizedBox(height: 20),  
    ElevatedButton.icon(  
      onPressed: () {  
        // Add your functionality here  
      },  
      icon: const Icon(Icons.login),  
      label: const Text('Login'),  
    ),  
    SizedBox(height: 10),  
    ElevatedButton.icon(  
      onPressed: () {  
        // Add your functionality here  
      },  
      icon: const Icon(Icons.person_add),  
      label: const Text('Sign Up'),  
    ),  
    SizedBox(height: 20),
```

```
const Text(  
  'Powered by Flutter',  
  style: TextStyle(  
    fontSize: 16,  
    fontStyle: FontStyle.italic,  
  ),  
)  
],  
)  
)  
)  
)  
)  
)  
);  
}  
}
```

OutPut:

Upworkr

**Welcome to Upwork**



→ Login

+ Sign Up

*Powered by Flutter*

**Conclusion:** We have successfully added advanced Flutter UI by including widgets like Image, Fonts, Icons.

