EXPERIMENT NO.3

Experiment No 3 **3: To design Flutter UI by including common widgets.**	
ROLL NO	**35**
NAME	**Gaurang Milind Mapuskar**
CLASS	**D15-B**
SUBJECT	**MAD & PWA Lab**
LO-MAPPED	

Theory:

Flutter, Google's UI toolkit, offers a rich set of features and widgets for building cross-platform apps with high performance and native-like experiences. Its hot reload feature allows rapid development and debugging. Widgets like MaterialApp provide a consistent look and feel across platforms. Core layout widgets such as Column and Row enable flexible UI designs. Material Design and Cupertino widgets ensure native-like experiences on Android and iOS. Customization options like themes, animations, and gestures enhance user interactions. With access to device APIs, plugins, and third-party libraries, Flutter empowers developers to create beautiful, responsive, and feature-rich apps for mobile, web, and desktop platforms.

Widgets:

Image:

The Image widget in Flutter is used to display images in the app's user interface. It supports various image sources such as assets, the internet, or memory. Developers can use the AssetImage for asset images, NetworkImage for images from the internet, or MemoryImage for images from memory. Images can be displayed with different formats like JPEG, PNG, GIF, and WebP.

Fonts:

Font Family:

- In Flutter, the fontFamily property of the TextStyle widget specifies the typeface used for rendering text.
- Developers can choose from system fonts like 'Roboto' or 'Arial', or include custom fonts in the app and reference them.
- Custom fonts are declared in the pubspec.yaml file and associated with a font family name.

Font Size:

- The fontSize property of the TextStyle widget determines the size of the text in logical pixels.
- Developers can specify the font size as a double value, allowing for precise control over text sizing.
- Adjusting the font size ensures readability and aesthetics, aligning with the app's design guidelines and user preferences.

Icon:
The Icon widget in Flutter is used to display graphical symbols or icons in the app's user interface. It supports built-in Material Design icons through the Icons class or custom icons imported as image assets. Developers can customize the icon's size, color, and alignment to match the app's design and theme. Icons are commonly used in IconButton widgets for actions or navigation items.

Code:

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

void main() {
  runApp(AlphabetApp());
}

class AlphabetApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Alphabet Learning App',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: AlphabetScreen(),
```

```
    );
  }
}

class AlphabetScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Alphabet Learning App'),
        actions: <Widget>[
          IconButton(
            icon: Icon(Icons.settings), // Add the Icon widget here
            onPressed: () {
              // Handle settings button press
            },
          ),
        ],
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: <Widget>[
            Column(
              mainAxisAlignment: MainAxisAlignment.spaceEvenly,
              children: <Widget>[
                AlphabetCard(
                  letter: 'A',
                  imagePath: 'assets/a.png',
                ),
                AlphabetCard(
                  letter: 'B',
                  imagePath: 'assets/b.png',
```

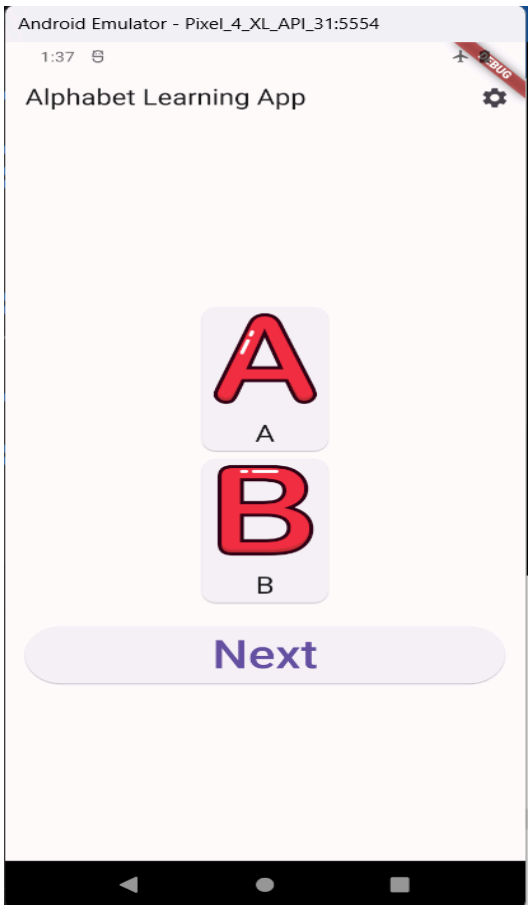
```
            ),
            // Add more AlphabetCards for other letters
          ],
        ),
        SizedBox(height: 20.0),
        ElevatedButton(
          onPressed: () {
            print("Button is clicked");
          },
          child: const Text(
            'Next',
            style: TextStyle(
              // fontFamily: 'Schyler',
              fontSize: 40,
            ),
          ),
        ),
      ],
    ),
  ),
 );
 }
}

class AlphabetCard extends StatelessWidget {
  final String letter;
  final String imagePath;

  AlphabetCard({required this.letter, required this.imagePath});

  @override
  Widget build(BuildContext context) {
   return Card(
     child: Column(
       children: <Widget>[
```

```
          Image.asset(
            imagePath,
            width: 100,
            height: 100,
          ),
          SizedBox(height: 10),
          Text(
            letter,
            style: TextStyle(fontSize: 24),
          ),
        ],
      ),
    );
  }
}
```



Conclusion:
Experiment successfully demonstrated the effective utilization of Flutter's common widgets to design a dynamic and user-friendly UI. By integrating these widgets creatively, we enhanced both functionality and aesthetics.