Name: Mitesh A. Dalvi **Roll no.: 11**
Class: D15B

MAD Experiment 3

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

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.

Flutter Widgets:

Icon:
An icon is a graphic image representing an application or any specific entity containing meaning for the user. It can be selectable and non-selectable. 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.

Icon Widget Properties:
Flutter icons widget has different properties for customizing the icons. These properties are explained below:

Property	Descriptions
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.

Images:
When you create an app in Flutter, it includes both code and assets (resources). An asset is a file, which is bundled and deployed with the app and is accessible at runtime. The asset can include static data, configuration files, icons, and images. The Flutter supports many image formats, such as JPEG, WebP, PNG, GIF, animated WebP/GIF, BMP, and WBMP.
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.

How to display the image in Flutter:
To display an image in Flutter, do the following steps:
Step 1: First, we need to create a new folder inside the root of the Flutter project and named it assets. We can also give it any other name if you want.
Step 2: Next, inside this folder, add one image manually.
Step 3: Update the pubspec.yaml file.

```
assets:
  - assets/tablet.png
  - assets/background.png
```

Display images from the internet:
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. We can use the following syntax to display an image from the internet.

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.

Program:

main.dart() :

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

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

class RecipeApp extends StatefulWidget {
  @override
  _RecipeAppState createState() => _RecipeAppState();
}

class _RecipeAppState extends State<RecipeApp> {
  int _selectedIndex = 0;

  static List<Widget> _widgetOptions = <Widget>[
    HomePage(),
    // Add more screens here as needed
    Placeholder(), // Example additional screen
    Placeholder(), // Example additional screen
  ];

  void _onItemTapped(int index) {
    setState(() {
      _selectedIndex = index;
    });
  }

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Recipe App',
```

```
      theme: ThemeData(
        primarySwatch: Colors.blue,
        // Add your custom font here
        fontFamily: 'Roboto',
      ),
      home: Scaffold(
        appBar: AppBar(
          title: Text('Recipes'),
        ),
        body: Center(
          child: _widgetOptions.elementAt(_selectedIndex),
        ),
        bottomNavigationBar: BottomNavigationBar(
          items: const <BottomNavigationBarItem>[
            BottomNavigationBarItem(
              icon: Icon(Icons.home),
              label: 'Home',
            ),
            // Add more bottom navigation bar items as needed
            BottomNavigationBarItem(
              icon: Icon(Icons.search),
              label: 'Search',
            ),
            BottomNavigationBarItem(
              icon: Icon(Icons.favorite),
              label: 'Favorites',
            ),
          ],
          currentIndex: _selectedIndex,
          selectedItemColor: Colors.blue,
          onTap: _onItemTapped,
        ),
      ),
    );
  }
}

class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return ListView(
      children: <Widget>[
        RecipeCard(
```

```
        title: 'Paneer Butter Masala',
        description:
            'Indian cottage cheese cubes are smothered in a creamy, lightly spiced
tomato sauce that is downright delicious.',
        imagePath: 'assets/paneer.jpg',
      ),
      RecipeCard(
        title: 'Chicken Stir Fry',
        description:
            'Quick and easy stir fry with chicken, vegetables, and soy sauce.',
        imagePath: 'assets/chickenstirfry.jpg',
      ),
      RecipeCard(
        title: 'Chocolate Chip Cookies',
        description: 'Classic homemade chocolate chip cookies.',
        imagePath: 'assets/chocolatechip.jpg',
      ),
    ],
  );
 }
}

class RecipeCard extends StatelessWidget {
 final String title;
 final String description;
 final String imagePath;

 const RecipeCard({
  Key? key,
  required this.title,
  required this.description,
  required this.imagePath,
 }) : super(key: key);

 @override
 Widget build(BuildContext context) {
  return Card(
   margin: EdgeInsets.all(8),
   child: Padding(
    padding: EdgeInsets.all(16),
    child: Column(
     crossAxisAlignment: CrossAxisAlignment.start,
     children: <Widget>[
```

```
            Image.asset(
              imagePath,
              height: 150,
              width: double.infinity,
              fit: BoxFit.cover,
            ),
            SizedBox(height: 8),
            Text(
              title,
              style: TextStyle(
                fontSize: 20,
                fontWeight: FontWeight.bold,
              ),
            ),
            SizedBox(height: 8),
            Text(
              description,
              style: TextStyle(fontSize: 16),
            ),
          ],
        ),
      ),
    );
  }
}
```

pubspec.yaml :

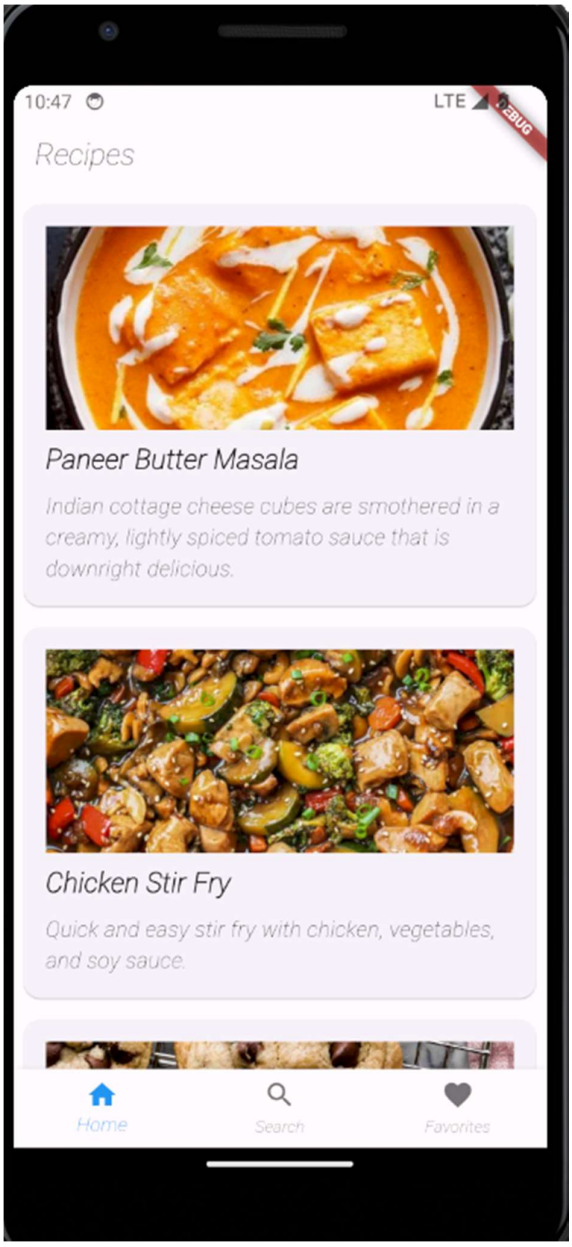
```
  fonts:

    - family: Roboto
      fonts:

        - asset: fonts/Roboto-ThinItalic.ttf

    - family: Lora
      fonts:
        - asset: fonts/Lora-SemiBold.ttf
  assets:
    - assets/paneer.jpg
    - assets/chickenstirfry.jpg
    - assets/chocolatechip.jpg
```

Output:



Conclusion: In this experiment, we have added images and icons on our UI designed. Also, we have downloaded fonts and inserted in the assets folder and updated the pubspec.yaml to access the assets folder in main.dart(). The output is updated successfully.