Experiment No: 3

Experiment No 3 3: Exploring Flutter Widgets.	
ROLL NO	**03**
NAME	**Ansari Mohammed Sharjeel**
CLASS	**D15B**
SUBJECT	**MAD & PWA Lab**
LO-MAPPED	

Experiment 3

Aim: To include icons, images, fonts in flutter app

Theory:

Flutter is a powerful framework for building cross-platform applications with beautiful an advanced user interfaces. It offers a wide range of built-in widgets that allow developers to incorporate various elements such as images, fonts, and icons seamlessly into their applications.

Images: Flutter provides the Image widget for displaying images in different formats, including network images, asset images, and memory images. Developers can use the Image.asset() constructor to display images stored locally within the app, while the Image.network() constructor allows fetching images from the internet. Images can be resized, cropped, and styled according to the application's design requirements.

Fonts: Flutter supports custom fonts, enabling developers to enhance the visual appeal of their applications by using unique typography. Fonts can be loaded into Flutter applications by including font files (e.g., .ttf, .otf) within the project directory. The TextStyle widget allows developers to specify various font properties such as font family, size, weight, style, and color. By utilizing custom fonts, developers can create visually striking UI designs that reflect the app's branding and style.

Icons: Icons are essential elements in UI design, providing users with intuitive visual cues for navigation, actions, and information. Flutter offers a vast collection of built-in icons through the Icons class, covering a wide range of categories such as action, alert, communication, content, and more. Developers can easily integrate icons into their Flutter applications using the Icon widget. Additionally, Flutter supports custom icon sets,
allowing developers to import and use custom icon libraries to achieve a unique visual identity for their applications.

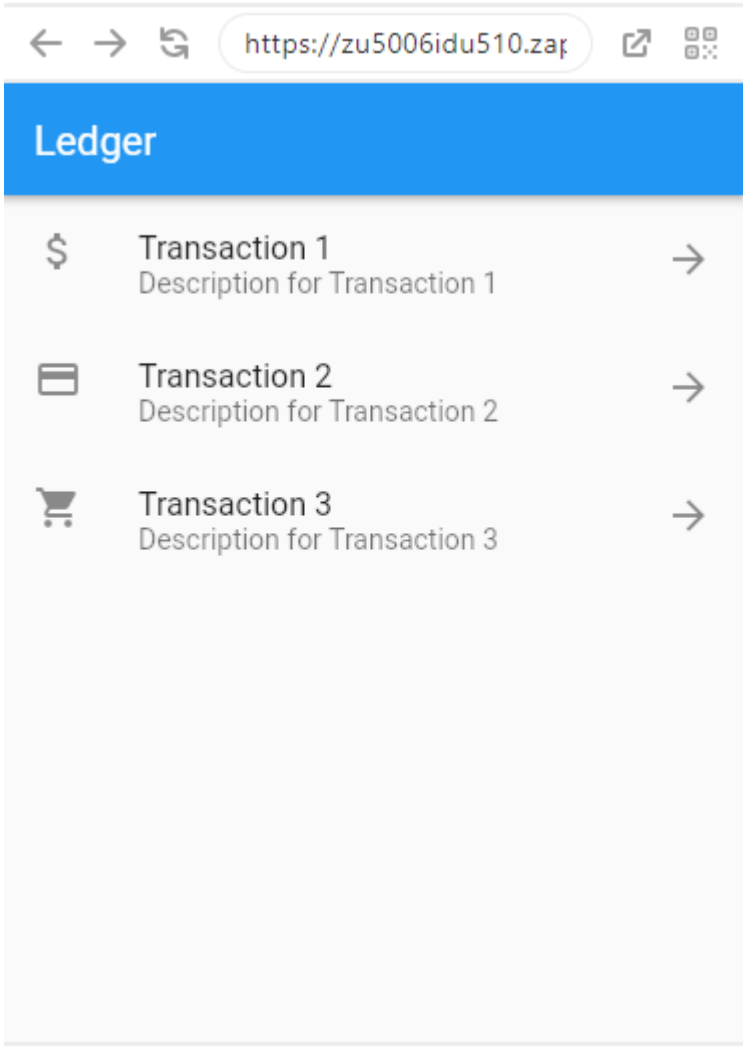
Code:

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

class Transaction {
  final String title;
  final String description;
```

```
  final IconData iconData;

  Transaction({required this.title, required this.description, required
this.iconData});
}

class LedgerScreen extends StatelessWidget {
  final List<Transaction> transactions = [
    Transaction(
      title: 'Transaction 1',
      description: 'Description for Transaction 1',
      iconData: Icons.attach_money,
    ),
    Transaction(
      title: 'Transaction 2',
      description: 'Description for Transaction 2',
      iconData: Icons.payment,
    ),
    Transaction(
      title: 'Transaction 3',
      description: 'Description for Transaction 3',
      iconData: Icons.shopping_cart,
    ),
  ];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Ledger'),
      ),
      body: ListView.builder(
        itemCount: transactions.length,
        itemBuilder: (context, index) {
          return ListTile(
            leading: Icon(transactions[index].iconData),
            title: Text(transactions[index].title),
            subtitle: Text(transactions[index].description),
            trailing: Icon(Icons.arrow_forward),
          );
        },
      ),
    );
  }
}
```

```
void main() {
  runApp(MaterialApp(
    debugShowCheckedModeBanner: false,
    title: 'Ledger App',
    theme: ThemeData(
      primarySwatch: Colors.blue,
      fontFamily: 'Roboto', // Custom font family
    ),
    home: LedgerScreen(),
  ));
}
```

Output:



Conclusion:
Hence, 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.