Experiment No.2
MAD & PWA LAB

- **Aim:** To design Flutter UI by including common widgets

- **Theory:**

→ **Introduction to Flutter –**
Flutter is an open-source UI software development toolkit created by Google. It allows developers to build natively compiled applications for mobile, web, and desktop from a single codebase. Flutter uses the Dart programming language and provides a rich set of pre-designed widgets for creating beautiful and responsive user interfaces.

→ **Common Widgets in Flutter:**
Widgets are the building blocks of a Flutter application. They are elements of the user interface, ranging from simple buttons to complex layouts. Some common widgets include:

1) Container:
- ✓ A box model that can contain other widgets.
- ✓ Properties like padding, margin, and decoration can be applied.

2) Row and Column:
- ✓ Used for horizontal (Row) and vertical (Column) arrangements of widgets.
- ✓ Children are arranged in a linear manner.

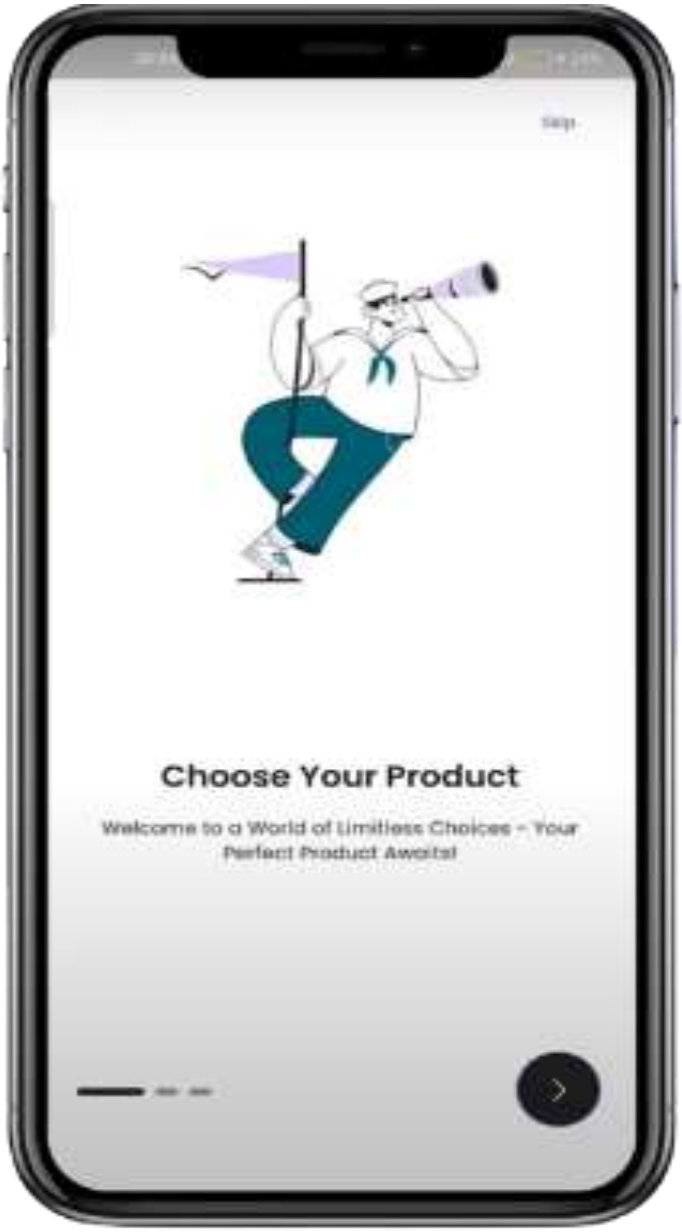
3) ListView:
- ✓ A scrollable list of widgets.
- ✓ Useful for displaying a large number of items.

4) AppBar:
- ✓ Represents the top app bar.
- ✓ Typically contains the app's title, icons, and actions.

5) FlatButton, RaisedButton, and IconButton:
- ✓ Different types of buttons with varying visual styles.

6) TextField:
- ✓ Allows users to input text.
- ✓ Options for customization include keyboard type and input validation.

7) Image:
- ✓ Displays images.
- ✓ Supports various image formats and sources.




```
import 'package:flutter/material.dart';
import
'package:t_store/utils/constants/image_s
trings.dart';
import
'package:t_store/utils/helpers/helper_fu
nctions.dart';
class OnBoardingScreen extends
StatelessWidget {
  const OnBoardingScreen({Key? key})
: super(key: key);
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Stack(
        children: [
          PageView(
            children: [
              Column(
                mainAxisAlignment:
MainAxisAlignment.center,
                crossAxisAlignment:
CrossAxisAlignment.center,
                children: [
                  Image(
                    width:
THelperFunctions.screenWidth() * 0.8,
                    height:
THelperFunctions.screenHeight() * 0.6,
                    image:
AssetImage(TImages.onBoardingImage
1),
                  ),
                ],
              ),
            ],
          ),
        ],
      ),
    );
  }
}
```

→ **Designing a Flutter UI Practically –**

1) **Create a new Flutter Project:**
✓ Use the flutter create command to initialize a new Flutter project.

2) **Understanding the Project Structure:**
✓ Familiarize ourself with the project structure, especially the lib directory where the main Dart code resides.

3) **Modify the main.dart File:**
✓ Open the main.dart file and start building our UI.
✓ Import necessary packages and widgets.

4) **Use Common Widgets:**
✓ Incorporate common widgets like Container, Row, Column, ListView, etc., based on our UI requirements.

5) **Apply Styling:**
✓ Use the properties of widgets to apply styles such as colors, fonts, and sizes.

6) **Handle User Interactions:**
✓ Implement interactivity using gesture detectors or callbacks.

7) **Testing:**
✓ Regularly test our UI on different devices and screen sizes.

8) **Debugging:**
✓ Use Flutter's debugging tools to identify and fix issues.

9) **Optimization:**
✓ Optimize our code for performance and maintainability.

10) **Documentation:**
✓ Document our code for future reference and collaboration.

→ **Types Of Widgets:**

1) **Stateless Widgets –**
Stateless widgets are immutable, meaning their properties (data) cannot change once they are initialized.
They are used for UI elements that don't need to change dynamically.
Examples include Container, Text, Icon, Image, and Row/Column for layout.

```
class MyTextWidget extends StatelessWidget {
  final String text;
  MyTextWidget(this.text);
  @override
  Widget build(BuildContext context) {
    return Text(text);
  }
}
```

2) **Stateful Widgets –**
Stateful widgets can be modified during the lifetime of the widget.
They are used for UI elements that need to update dynamically in response to user interactions or data changes.
Examples include TextField, Checkbox, Radio, and ListView.

```
class MyTextFieldWidget extends StatefulWidget {
  @override
  _MyTextFieldWidgetState createState() => _MyTextFieldWidgetState();
}

class _MyTextFieldWidgetState extends State<MyTextFieldWidget> {
  TextEditingController _controller = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return TextField(
      controller: _controller,
      decoration: InputDecoration(labelText: 'Enter text'),
    );
  }
}
```

- **Conclusion:**
In conclusion, we've gained insights into crafting the Flutter UI for our application's Boarding Screen by incorporating the essential widgets utilized in TradeSphere's [E-Commerce] design.