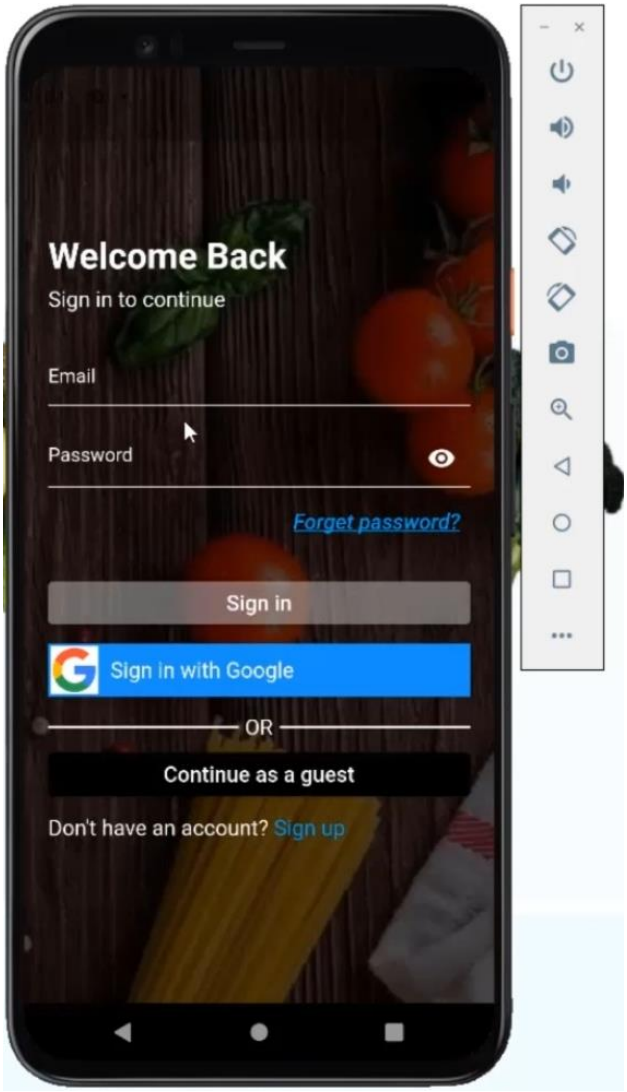
Experiment No.2
MAD & PWA LAB

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

- **Theory:**

  Flutter widgets are constructed using a contemporary framework influenced by React. These widgets define the appearance of their view based on their current configuration and state. Whenever a widget's state changes, it reconstructs its description. Subsequently, the framework compares this new description with the previous one to identify the minimal changes required in the underlying render tree for transitioning between states.



1. MaterialApp
2. Scaffold
3. Container
4. SingleChildScrollView
5. Text
6. TextField
7. LinearGradient
8. BoxDecoration
9. Padding
10. SizedBox
11. Row
12. Column
13. MaterialButton
14. FadeInUp

1. **MaterialApp**:
- Description: MaterialApp is a widget provided by the Flutter framework. It represents the root of a Flutter application and provides various configuration options for the entire application, such as the title, theme, and initial route.
- Details:
  - `title`: Sets the title of the application, which appears in the device's app switcher and task manager.
  - `theme`: Configures the overall theme of the application, including colors, typography, and shape.
  - `home`: Specifies the widget to be displayed as the home screen of the application.

2. **Scaffold:**
- Description: Scaffold is a layout structure widget in Flutter that implements the basic material design visual layout structure. It provides a framework for building app screens with components like an app bar, floating action button, drawer, and bottom navigation.
- Details:
  - `appBar`: Configures the app bar at the top of the screen.
  - `body`: Specifies the main content of the screen.
  - `floatingActionButton`: Configures a button that floats above the main content.
  - `bottomNavigationBar`: Configures a bottom navigation bar for navigating between screens.

3. Container:

- Description: Container is a versatile widget in Flutter used to contain other widgets and apply various styling options like padding, margin, borders, and background color.
- Details:
  - `width`, `height`: Sets the width and height of the container.
  - `decoration`: Applies decorations like colors, gradients, borders, and shadows to the container.
  - `child`: Specifies the widget(s) contained within the container.

4.SingleChildScrollView:

- Description: SingleChildScrollView is a widget used to enable scrolling when the content exceeds the size of the viewport. It allows users to scroll vertically to view content that would otherwise be cut off.
- Details:
  - `child`: Specifies the child widget that will be scrolled if its content overflows.

5. Text:

- Description: Text widget is used to display textual information on the screen. It allows customization of the text style, including color, font size, font weight, alignment, and more.
- Details:
  - `data`: Specifies the text content to be displayed.
  - `style`: Defines the text style properties like color, font size, font weight, and more.

6. **TextField**:

- Description: TextField widget is used to capture user input as text. It provides a text input field where users can enter text, such as usernames, passwords, or search queries.
- Details:
  - `decoration`: Configures the visual appearance of the input field, including placeholder text and border styling.
  - `onChanged`: Callback function invoked when the text input changes.

7. **LinearGradient:**

- Description: LinearGradient widget is used to create a linear gradient, which is a smooth transition between two or more colors that appear in a straight line.
- Details:
  - `begin`, `end`: Specifies the starting and ending points of the gradient.

- `colors`: Defines the list of colors that make up the gradient.

8. BoxDecoration:

- Description: BoxDecoration is used to apply decorations like colors, gradients, borders, and shadows to a container. It allows for advanced customization of the visual appearance of the container.
- Details:
  - `color`: Sets the background color of the container.
  - `border`: Configures the border properties, such as color, width, and style.
  - `borderRadius`: Defines the rounded corners of the container.

10. SizedBox:

- Description: SizedBox widget is used to give a specific size to its child widget or to create space between widgets. It constrains its child to a fixed width and height.
- Details:
  - `width`, `height`: Sets the width and height of the SizedBox.

11. Row:

- Description: Row widget is used to arrange its children widgets horizontally in a row, allowing for side-by-side alignment of multiple widgets.
- Details:
  - `children`: Specifies the list of child widgets to be arranged horizontally.

12. Column:

- Description: Column widget is used to arrange its children widgets vertically in a column, allowing for stacked alignment of multiple widgets.
- Details:
  - `children`: Specifies the list of child widgets to be arranged vertically.

13. MaterialButton:

- Description: MaterialButton is a button widget in Flutter that follows the Material Design guidelines. It can be customized with various properties like onPressed, color, shape, etc., to create interactive buttons.
- Details:

- `onPressed`: Callback function invoked when the button is pressed.
- `color`: Sets the background color of the button.
- `shape`: Defines the shape of the button, such as rounded corners or a circular shape.

14. FadeInUp:

- Description: FadeInUp is a widget provided by the 'animate_do' package. It applies a fade-in animation with an upward movement to its child widget, providing a visually appealing entrance animation.
- Details:
  - `duration`: Specifies the duration of the animation.
  - `child`: Specifies the child widget to which the animation will be applied.

Code:

```
import 'package:card_swiper/card_swiper.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/gestures.dart';
import 'package:flutter/material.dart';
import 'package:grocery_app/screens/auth/forget_pass.dart';
import 'package:grocery_app/screens/auth/register.dart';
import 'package:grocery_app/screens/btm_bar.dart';
import 'package:grocery_app/screens/loading_manager.dart';
import 'package:grocery_app/services/global_methods.dart';

import '../../consts/contss.dart';
import '../../consts/firebase_consts.dart';
import '../../fetch_screen.dart';
import '../../widgets/auth_button.dart';
import '../../widgets/google_button.dart';
import '../../widgets/text_widget.dart';

class LoginScreen extends StatefulWidget {
  static const routeName = '/LoginScreen';
  const LoginScreen({Key? key}) : super(key: key);

  @override
  State<LoginScreen> createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
```

```
final _emailTextController = TextEditingController();
final _passTextController = TextEditingController();
final _passFocusNode = FocusNode();
final _formKey = GlobalKey<FormState>();
var _obscureText = true;
@override
void dispose() {
  _emailTextController.dispose();
  _passTextController.dispose();
  _passFocusNode.dispose();
  super.dispose();
}

bool _isLoading = false;
void _submitFormOnLogin() async {
  final isValid = _formKey.currentState!.validate();
  FocusScope.of(context).unfocus();

  if (isValid) {
    _formKey.currentState!.save();
    setState(() {
      _isLoading = true;
    });
    try {
      await authInstance.signInWithEmailAndPassword(
          email: _emailTextController.text.toLowerCase().trim(),
          password: _passTextController.text.trim());
      Navigator.of(context).pushReplacement(
        MaterialPageRoute(
          builder: (context) => const FetchScreen(),
        ),
      );
      print('Succefully logged in');
    } on FirebaseException catch (error) {
      GlobalMethods.errorDialog(
          subtitle: '${error.message}', context: context);
      setState(() {
        _isLoading = false;
      });
    } catch (error) {
      GlobalMethods.errorDialog(subtitle: '$error', context: context);
```

```
      setState(() {
        _isLoading = false;
      });
    } finally {
      setState(() {
        _isLoading = false;
      });
    }
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: LoadingManager(
      isLoading: _isLoading,
      child: Stack(children: [
        Swiper(
          duration: 800,
          autoplayDelay: 8000,
          itemBuilder: (BuildContext context, int index) {
            return Image.asset(
              Constss.authImagesPaths[index],
              fit: BoxFit.cover,
            );
          },
          autoplay: true,
          itemCount: Constss.authImagesPaths.length,
        ),
        Container(
          color: Colors.black.withOpacity(0.7),
        ),
        SingleChildScrollView(
          child: Padding(
            padding: const EdgeInsets.all(20.0),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              crossAxisAlignment: CrossAxisAlignment.start,
              mainAxisSize: MainAxisSize.max,
              children: [
                const SizedBox(
```

```
  height: 120.0,
),
TextWidget(
  text: 'Welcome Back',
  color: Colors.white,
  textSize: 30,
  isTitle: true,
),
const SizedBox(
  height: 8,
),
TextWidget(
  text: "Sign in to continue",
  color: Colors.white,
  textSize: 18,
  isTitle: false,
),
const SizedBox(
  height: 30.0,
),
Form(
    key: _formKey,
    child: Column(
      children: [
        TextFormField(
          textInputAction: TextInputAction.next,
          onEditingComplete: () => FocusScope.of(context)
              .requestFocus(_passFocusNode),
          controller: _emailTextController,
          keyboardType: TextInputType.emailAddress,
          validator: (value) {
            if (value!.isEmpty || !value.contains('@')) {
              return 'Please enter a valid email address';
            } else {
              return null;
            }
          },
          style: const TextStyle(color: Colors.white),
          decoration: const InputDecoration(
            hintText: 'Email',
            hintStyle: TextStyle(color: Colors.white),
```

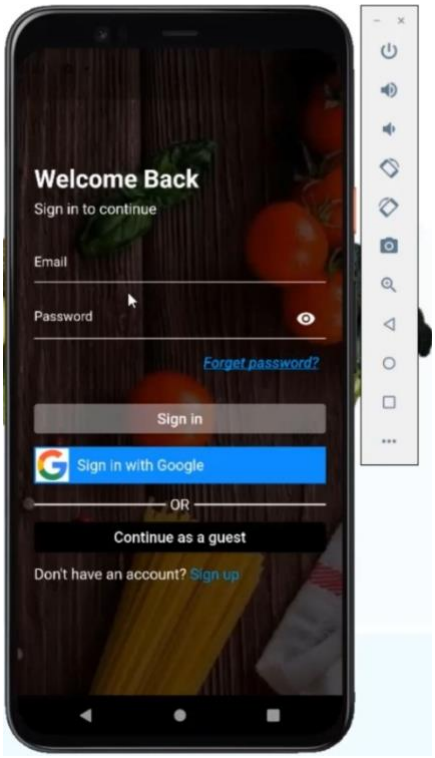
```
    enabledBorder: UnderlineInputBorder(
      borderSide: BorderSide(color: Colors.white),
    ),
    focusedBorder: UnderlineInputBorder(
      borderSide: BorderSide(color: Colors.white),
    ),
  ),
),
SizedBox(
  height: 12,
),
//Password

TextFormField(
  textInputAction: TextInputAction.done,
  onEditingComplete: () {
    _submitFormOnLogin();
  },
  controller: _passTextController,
  focusNode: _passFocusNode,
  obscureText: _obscureText,
  keyboardType: TextInputType.visiblePassword,
  validator: (value) {
    if (value!.isEmpty || value.length < 7) {
      return 'Please enter a valid password';
    } else {
      return null;
    }
  },
  style: const TextStyle(color: Colors.white),
  decoration: InputDecoration(
    suffixIcon: GestureDetector(
        onTap: () {
          setState(() {
            _obscureText = !_obscureText;
          });
        },
        child: Icon(
          _obscureText
              ? Icons.visibility
              : Icons.visibility_off,
```

```
                color: Colors.white,
              )),
            hintText: 'Password',
            hintStyle: const TextStyle(color: Colors.white),
            enabledBorder: const UnderlineInputBorder(
              borderSide: BorderSide(color: Colors.white),
            ),
            focusedBorder: const UnderlineInputBorder(
              borderSide: BorderSide(color: Colors.white),
            ),
          ),
        ),
      ],
    )),
const SizedBox(
  height: 10,
),
Align(
  alignment: Alignment.topRight,
  child: TextButton(
    onPressed: () {
      GlobalMethods.navigateTo(
          ctx: context,
          routeName: ForgetPasswordScreen.routeName);
    },
    child: const Text(
      'Forget password?',
      maxLines: 1,
      style: TextStyle(
          color: Colors.lightBlue,
          fontSize: 18,
          decoration: TextDecoration.underline,
          fontStyle: FontStyle.italic),
    ),
  ),
),
const SizedBox(
  height: 10,
),
AuthButton(
  fct: _submitFormOnLogin,
```

```
  buttonText: 'Login',
),
const SizedBox(
  height: 10,
),
GoogleButton(),
const SizedBox(
  height: 10,
),
Row(
  children: [
    const Expanded(
      child: Divider(
        color: Colors.white,
        thickness: 2,
      ),
    ),
    const SizedBox(
      width: 5,
    ),
    TextWidget(
      text: 'OR',
      color: Colors.white,
      textSize: 18,
    ),
    const SizedBox(
      width: 5,
    ),
    const Expanded(
      child: Divider(
        color: Colors.white,
        thickness: 2,
      ),
    ),
  ],
),
const SizedBox(
  height: 10,
),
AuthButton(
  fct: () {
```

```
            Navigator.of(context).push(
              MaterialPageRoute(
                builder: (context) => const FetchScreen(),
              ),
            );
          },
          buttonText: 'Continue as a guest',
          primary: Colors.black,
        ),
        const SizedBox(
          height: 10,
        ),
        RichText(
            text: TextSpan(
              text: 'Don\'t have an account?',
              style: const TextStyle(
                  color: Colors.white, fontSize: 18),
              children: [
               TextSpan(
                  text: '  Sign up',
                  style: const TextStyle(
                      color: Colors.lightBlue,
                      fontSize: 18,
                      fontWeight: FontWeight.w600),
                  recognizer: TapGestureRecognizer()
                    ..onTap = () {
                      GlobalMethods.navigateTo(
                          ctx: context,
                          routeName: RegisterScreen.routeName);
                    }),
            ]))
      ],
    ),
  )
```



- **Conclusion:**

  Hence, we understood how to design Flutter UI by including the widgets that have been used in making the **Login Screen** of our application (Grocery App).