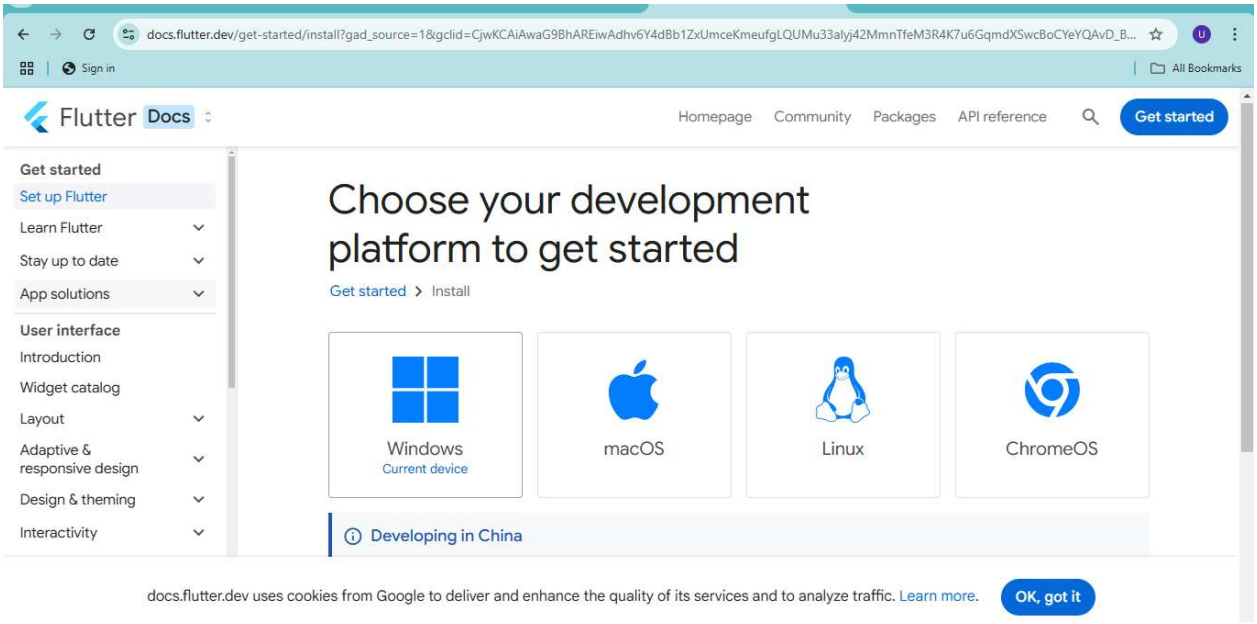
Install the Flutter SDK

Step 1: Download the installation bundle of the Flutter Software Development Kit for windows. To download Flutter SDK, Go to its official website https://docs.flutter.dev/get-started/install , you will get the following screen.



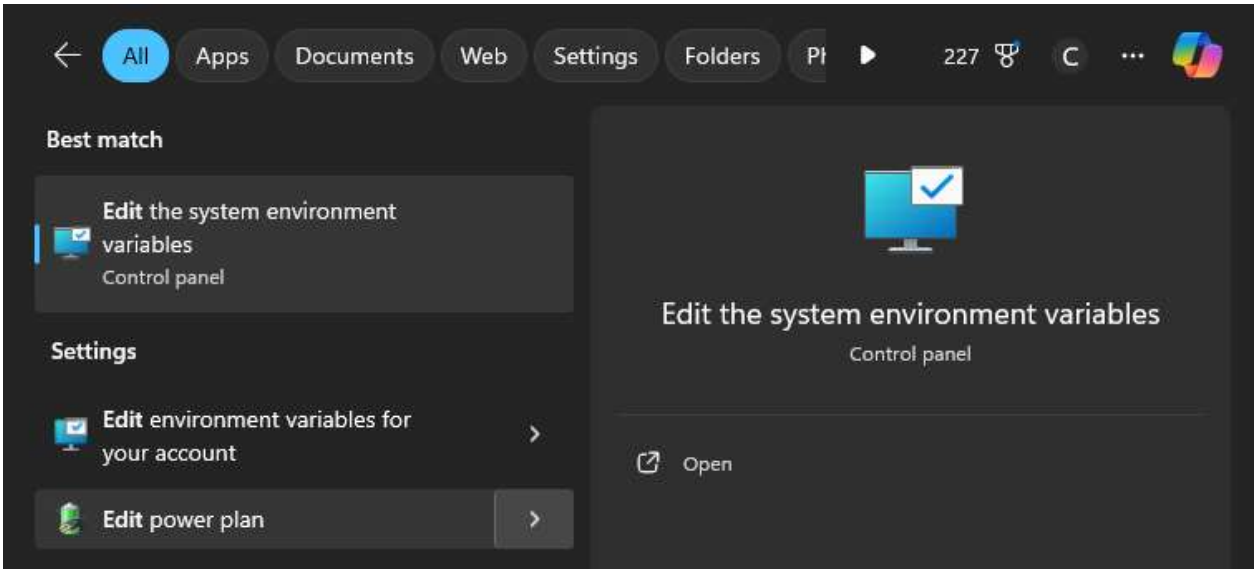
Step 2: Next, to download the latest Flutter SDK, click on the Windows icon. Here, you will find the download link for SDK.

Step 3: Extract the Files

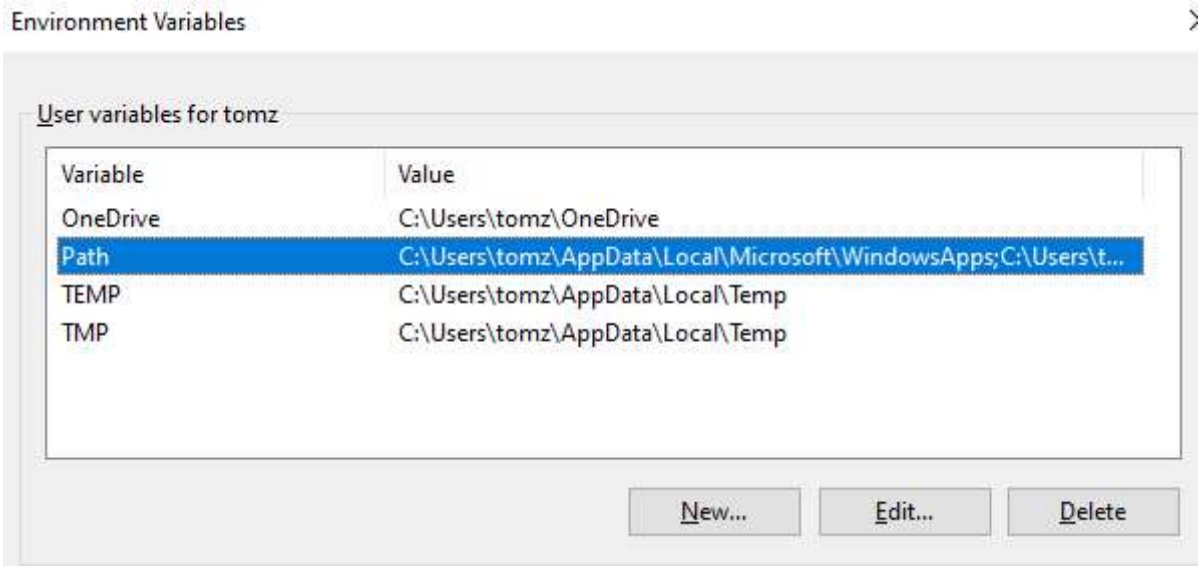
Extract the downloaded zip file and move it to the desired location you want to install Flutter SDK. Do not install it in a folder or directory that requires elevated privileges, (such as *C:\Program Files*\) to ensure the program runs properly. For this tutorial, it will be stored in *C:\development\flutter*.

Step 3: Update Path Variable for Windows PowerShell

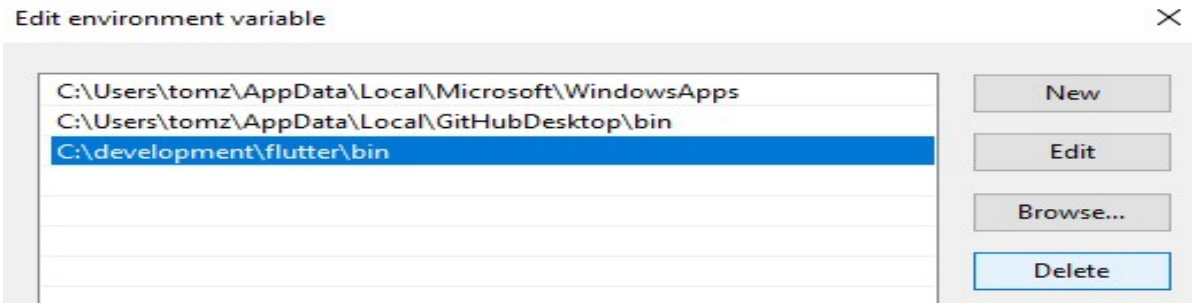
Next, you need to update your Path environment variable to run Flutter commands in Windows consoles PowerShell and Command Prompt (CMD). First, click the Start button and type to search for and then click on **Edit environment variables for your account**.



Under User variables, click on and highlight **Path**. Click **Edit**.



On the next screen, click **New** and add the full path to your *flutter\bin* directory. For this guide, it is shown below. Click OK on both windows to enable running Flutter commands in Windows consoles.



Step 4: Confirm Installed Tools for Running Flutter

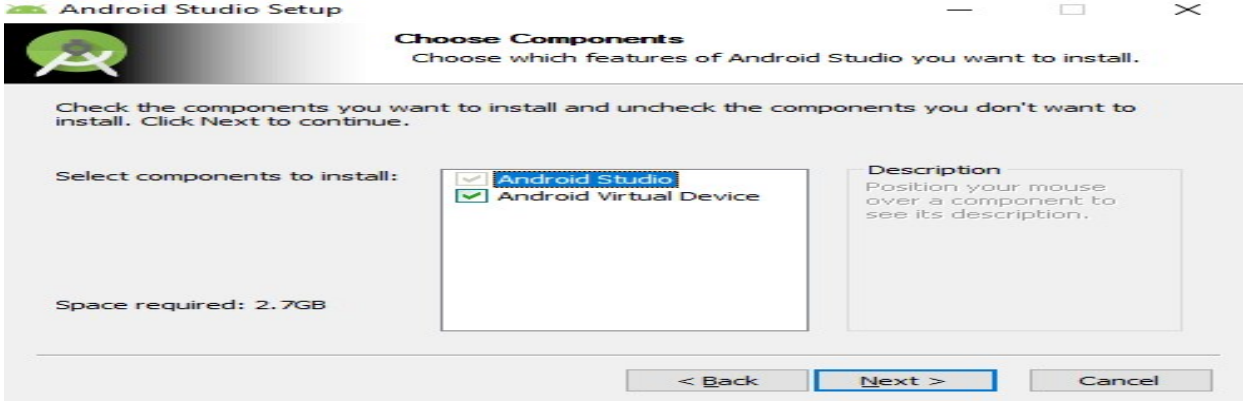
In CMD, run the *flutter doctor* command to confirm the installed tools along with brief descriptions.

```
C:\Users\tomz>flutter doctor
Running "flutter pub get" in flutter_tools... 8,9s
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 2.10.4, on Microsoft Windows [Version 10.0.19041.746], locale
en-US)
[X] Android toolchain - develop for Android devices
    X Unable to locate Android SDK.
    Install Android Studio from: https://developer.android.com/studio/index.html
    On first launch it will assist you in installing the Android SDK components.
    (or visit https://flutter.dev/docs/get-started/install/windows#android-setup for
detailed instructions).
    If the Android SDK has been installed to a custom location, please use
    'flutter config --android-sdk' to update to that location.

[√] Chrome - develop for the web
[X]] Visual Studio - develop for Windows
     X Visual Studio not installed; this is necessary for Windows development.
       Download at https://visualstudio.microsoft.com/downloads/.
       Please install the "Desktop development with C++" workload, including all of its
default components
[!] Android Studio (not installed)
[√] Connected device (2 available)
[√] HTTP Host Availability

! Doctor found issues in 3 categories.
```

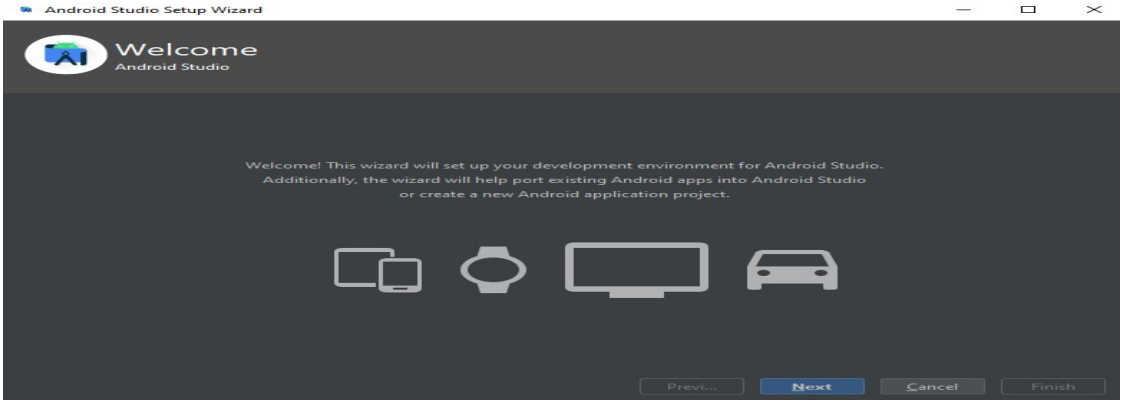
As visible, several components still need to be installed to complete the installation.

Step 5: Download and Install Android Studio

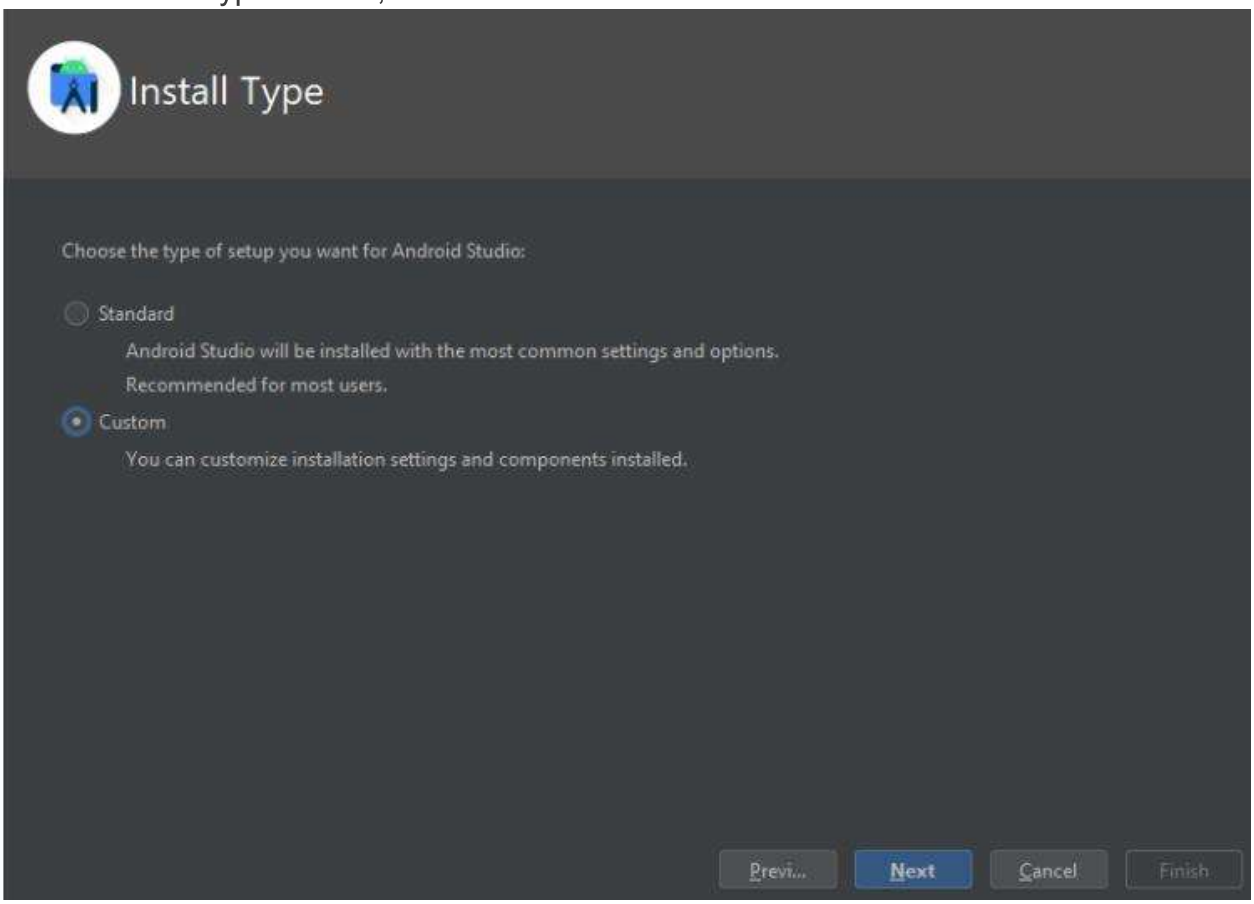
Continue by downloading Android Studio. In the setup, unless you have specific requirements, you can click Next on all screens leaving the default settings. Ensure that the Android Virtual Device option is selected on the Choose Components screen so that you can have an Android emulator running for Android app development.



Afterward, Android Studio Setup Wizard will start and you can proceed by clicking **Next.**



On the Install Type screen, select Custom and click **Next.**



Verify the selections and click **Next.**

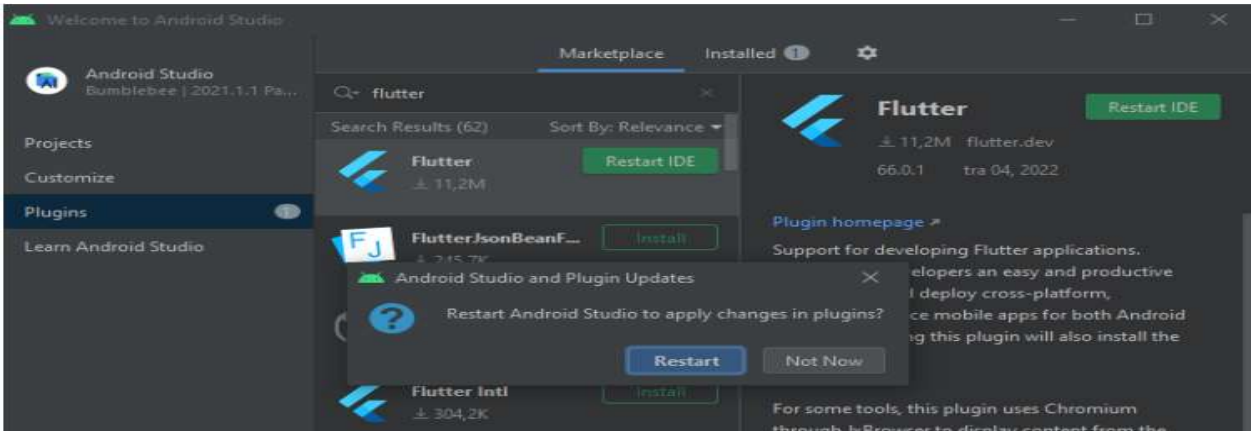
On the next screen, accept the License Agreement and click **Finish.**

The download of the components will start and Android Studio install. Once completed, click **Finish.**

After the installation, start Android Studio. On the left side, click **Plugins**. Search for Flutter and click **Install** to install the Flutter plugin.

It will also prompt you to install Dart, a programming language used to create Flutter apps. Click **Install** at the prompt.

Finally, click **Restart IDE** so that the plugin changes are applied. Click **Restart** at the prompt to confirm this action.



Afterward, run the *flutter doctor* command in CMD to confirm the Android Studio installation.

```
C:\Users\tomz>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 2.10.4, on Microsoft Windows [version 10.0.19041.746), locale
en-US)
[!] Android toolchain - develop for Android devices (Android SDK version 32.1.0-rc1)
    ! Some Android licenses not accepted. To resolve this, run: flutter doctor --android-
licenses
[√] Chrome - develop for the web
[X] Visual Studio - develop for Windows
    X Visual Studio not installed; this is necessary for Windows development.
      Download at https://visualstudio.microsoft.com/downloads/.
      Please install the "Desktop development with C++- workload, including all of its
default components
[√] Android Studio (version 2021.1)
[√] Connected device (2 available)
[√] HTTP Host Availability

! Doctor found issues in 2 categories.
```

Aim:
The aim of this experiment is to design a Flutter UI by using common Flutter widgets such as Text, Containers, Buttons, ListViews, and more to create a functional and visually appealing interface. The experiment helps in understanding how to structure and build UIs effectively using Flutter's pre-built components.

Theory:
Flutter is an open-source framework developed by Google for building natively compiled applications for mobile, web, and desktop from a single codebase. It provides a rich set of pre-designed widgets that help in building user interfaces with ease. These widgets are categorized into material design widgets and Cupertino widgets, which are designed to follow Google's Material Design principles and Apple's Human Interface Guidelines respectively. Common widgets like Text, Row, Column, Container, Button, ListView, and others play a critical role in creating dynamic layouts and interactive elements in Flutter applications.

Code:
loginpage.dart

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

import 'package:nutrilab/authservice.dart';

import 'package:nutrilab/forgot.dart';

import 'package:nutrilab/registerpage.dart';

import './deco.dart';

class GoToLoginPage extends StatefulWidget {

  const GoToLoginPage({super.key});

  @override

  State<GoToLoginPage> createState() => _GoToLoginPageState();

}


class _GoToLoginPageState extends State<GoToLoginPage> {

  final _auth = AuthService();
```

```
final _email = TextEditingController();
final _password = TextEditingController();

@override
void dispose() {
 _email.dispose();
 _password.dispose();
 super.dispose();
}

@override
Widget build(BuildContext context) {
 final screenWidth = MediaQuery.of(context).size.width;

 // Base font sizes
 final double baseFontSize = screenWidth  * 0.05;
 final double smallFontSize = baseFontSize * 0.65;
 final double mediumFontSize = baseFontSize * 0.8;
 final double largeFontSize = baseFontSize*1.4;
 final double extraLargeFontSize = baseFontSize * 2.5;

 return Scaffold(
  backgroundColor: Color.fromARGB(255, 225, 226, 209),
  body: Center(
```

```
child: Container(
  width: screenWidth - 40,
  child: Column(
    children: [
      Spacer(
        flex: 16,
      ),
      Stack(
        children: <Widget>[
          Text(
            'NUTRIEATS',
            style: TextStyle(
              fontSize: extraLargeFontSize,
              fontFamily: 'Genos',
              fontWeight: FontWeight.w500,
              foreground: Paint()
                ..style = PaintingStyle.stroke
                ..strokeWidth = 3
                ..color = Color.fromARGB(255, 27, 78, 23),
            ),
          ),
          Text(
            'NUTRIEATS',
            style: TextStyle(
```

```
      fontSize: extraLargeFontSize,
      fontWeight: FontWeight.w500,
      fontFamily: 'Genos',
      color: Color.fromARGB(255, 122, 185, 120),
    ),
   ),
 ],
),
Spacer(flex: 2),
Text(
 "Welcome back, you have been missed!",
 style: TextStyle(
  fontFamily: 'Gayathri',
  fontWeight: FontWeight.w700,
  fontSize: mediumFontSize,
 ),
),
Spacer(flex: 4),
TextFormField(
 controller: _email,
 cursorColor: Color.fromARGB(255, 24, 79, 87),
 style: TextStyle(
  color: const Color.fromARGB(255, 0, 0, 0),
  fontFamily: 'Gayathri',
```

```
    fontWeight: FontWeight.w700,
    fontSize: mediumFontSize,
   ),
   decoration: myDecorationField.copyWith(
    hintText: 'Email',
   ),
   keyboardType: TextInputType.emailAddress,
  ),
  Spacer(),
  TextFormField(
   controller: _password,
   obscureText: true,
   cursorColor: Color.fromARGB(255, 24, 79, 87),
   style: TextStyle(
    color: const Color.fromARGB(255, 0, 0, 0),
    fontFamily: 'Gayathri',
    fontWeight: FontWeight.w700,
    fontSize: mediumFontSize,
   ),
   decoration: myDecorationField.copyWith(
    hintText: 'Password',
   ),
   keyboardType: TextInputType.visiblePassword,
  ),
```

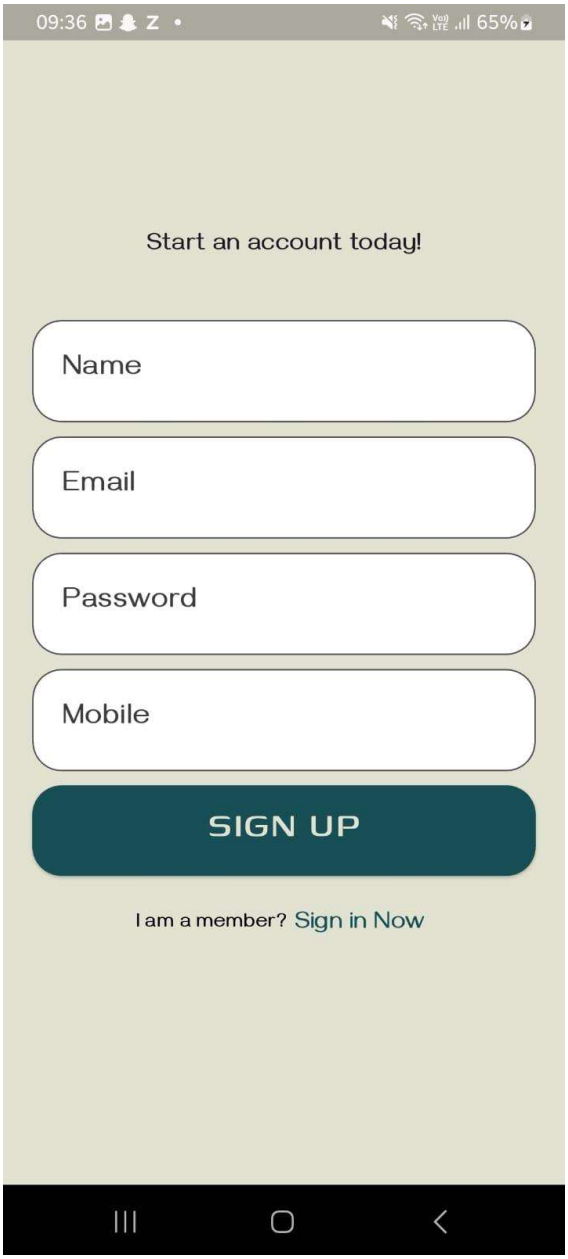
```
Align(
 alignment: Alignment.topRight,
 child: TextButton(
  onPressed: () {
   Navigator.push(
    context,
    MaterialPageRoute(
     builder: (context) => ForgotPage(),
    ),
   );
  },
  child: Text(
   'Forgot Password?',
   style: TextStyle(
    fontWeight: FontWeight.w700,
    fontFamily: 'Gayathri',
    fontSize: smallFontSize,
    color: Color.fromARGB(255, 24, 79, 87),
   ),
  ),
 ),
),
Spacer(),
SizedBox(
```

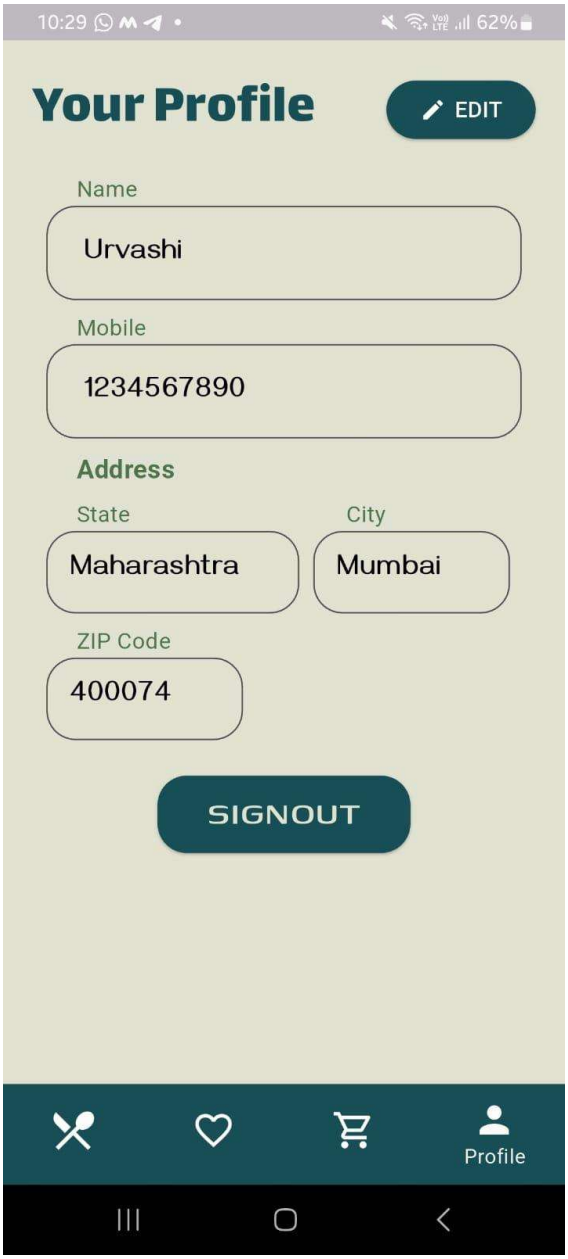
```
 width: screenWidth - 40,
 child: ElevatedButton(
  onPressed: _login,
  style: ElevatedButton.styleFrom(
   padding: EdgeInsets.fromLTRB(100, 5, 100, 15),
   backgroundColor: Color.fromARGB(255, 24, 79, 87),
   shape: RoundedRectangleBorder(
    borderRadius: BorderRadius.circular(20),
   ),
  ),
  child: Text(
   "SIGN IN",
   style: TextStyle(
    fontSize: largeFontSize,
    fontWeight: FontWeight.w500,
    fontFamily: 'Genos',
    color: Color.fromARGB(255, 225, 226, 209),
   ),
  ),
 ),
),
Spacer(),
Row(
 mainAxisAlignment: MainAxisAlignment.center,
```

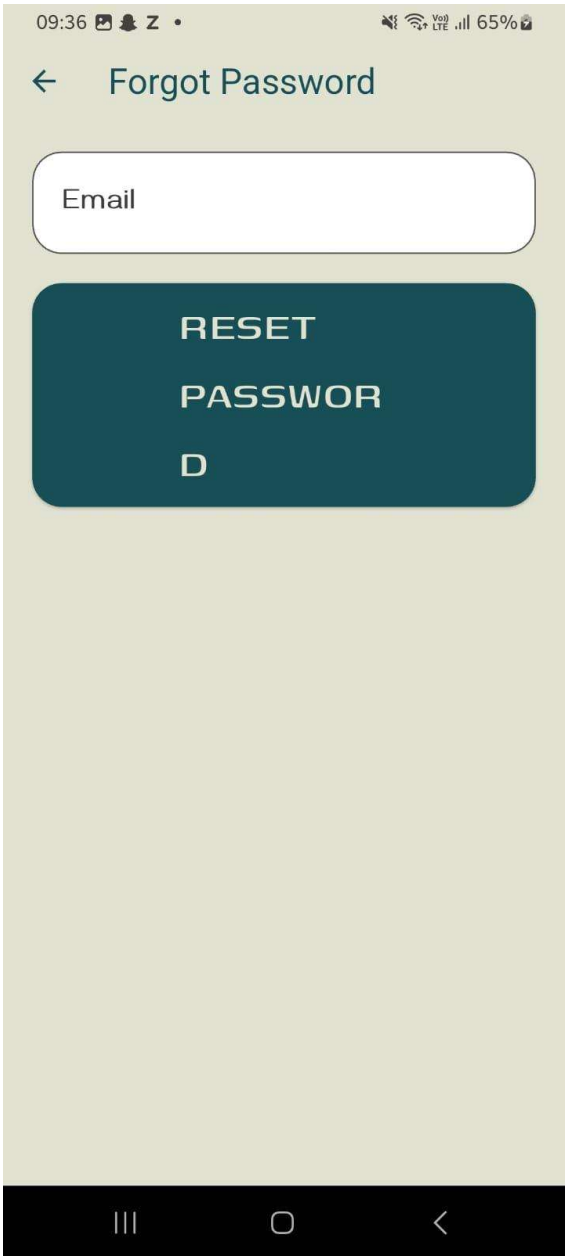
```
children: [
 Text(
  "Not a member?",
  style: TextStyle(
   color: Color.fromARGB(255, 0, 0, 0),
   fontFamily: 'Gayathri',
   fontWeight: FontWeight.w700,
   fontSize: smallFontSize,
  ),
 ),
 TextButton(
  style: TextButton.styleFrom(
   splashFactory: NoSplash.splashFactory,
   padding: EdgeInsets.all(5),
  ),
  onPressed: () {
   Navigator.push(
    context,
    MaterialPageRoute(
     builder: (context) => GoToRegisterPage(),
    ),
   );
  },
  child: Text(
```

```
                "Register Now",
                style: TextStyle(
                  color: Color.fromARGB(255, 24, 79, 87),
                  fontFamily: 'Gayathri',
                  fontWeight: FontWeight.w700,
                  fontSize: mediumFontSize,
                ),
              ),
            ),
          ],
        ),
        Spacer(flex: 16),
      ],
    ),
  ),
 ),
);
}

_login() async {
 await _auth.loginUserWithEmailAndPassword(
  context,
  _email.text,
  _password.text,
```

```
    );
  }
}
```


09:36
65%
Start an account today!
Name
Email
Password
Mobile
SIGN UP
I am a member? Sign in Now


10:29
62%
Your Profile
EDIT
Name
Urvashi
Mobile
1234567890
Address
State
Maharashtra
City
Mumbai
ZIP Code
400074
SIGNOUT
Profile



Conclusion:

In this experiment, we successfully designed a Flutter UI by integrating various common widgets. By leveraging these widgets, we were able to build an intuitive and responsive user interface. This experiment helped us understand the flexibility and power of Flutter in UI design, enabling the creation of highly customizable and interactive mobile applications. The knowledge gained is foundational for further exploring advanced UI and app development in Flutter.

Aim:
To learn how to include icons, images, and custom fonts in a Flutter application to enhance its visual design.

Theory:
Flutter allows the integration of icons, images, and custom fonts to improve UI. Icons represent actions, images display rich content, and custom fonts personalize typography. The `Icon` widget, `Image` widget, and the `pubspec.yaml` file help in adding these elements to the app.

Code:

menupage.dart

```
import 'package:flutter/material.dart';
import 'package:nutrilab/menuitem.dart';
class GoToMenuPage extends StatefulWidget {
 final Map<String, int> cart; // Accept cart as a parameter
 final Function(Map<String, int>) onCartUpdated; // Callback to update parent cart state
 const GoToMenuPage({
  Key? key,
  required this.cart,
  required this.onCartUpdated, // Add this parameter
 }) : super(key: key);
 @override
 State<GoToMenuPage> createState() => _GoToMenuPageState();
}
class _GoToMenuPageState extends State<GoToMenuPage> {
 String _searchTerm = '';
 final TextEditingController _searchController = TextEditingController();
 String _selectedCategory = 'All';
 // Local cart state
 Map<String, int> _cart = {};
 // Local data
```

```
 final List<Map<String, dynamic>> menuItems = [
  {
    'Name': 'Pizza',
    'Description': 'Delicious cheese pizza WheatBased',
    'Image': 'https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcTyMkRu4MFF5OT6HzojWJOq9YTAsvjNYqAYmw&
s',
    'Ingr': 'WheatDough,Cheese,Capsicum,Tomato',
    'Type': 'WheatBased',
    'Calories': 300,
    'Price': 250,
    'CarbonFootprint': '300 gms',
    'showInAll': true, // This item will appear in the "All" category
  },
  {
    'Name': 'Juice',
    'Description': 'Pure Orange Juice',
    'Image': 'https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcRu5r5P2zouN2MCLj2foGmRwE_xG25s9Lwxkw&
s',
    'Ingr': 'Orange',
    'Type': 'WeightLoss',
    'Calories': 100,
    'Price': 100,
    'CarbonFootprint': '200 gms',
    'showInAll': true, // This item will appear in the "All" category
  },
  {
    'Name': 'Fruit Salad',
    'Description': 'Healthy Fruit Salad',
```

'Image': 'https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRXKIQ3slSvcMrIYRs-iIJXxIfCVuTwa9QgrQ&s',

'Ingr': 'Strawbeery,Blueberry,Raspberry,Kiwi,pomogrenate',

'Type': 'WeightLoss',

'Calories': 350,

'Price': 300,

'CarbonFootprint': '200gms',

'showInAll': true, // This item will appear in the "All" category

},

{

'Name': 'Avacado Toast',

'Description': 'Avacado Toast with Cottage Cheese Nutrious,Filling and Delicious',

'Image': 'https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQ-Q-mQTnFMqYdTeYyyHqyo6dZJOL95ISmFPY_MJENSs5RPhDoMYZCZgGgKWk5Wa9ZL240&usqp=CAU',

'Ingr': 'Avacado,Dough,Cottage Cheese',

'Type': 'WeightGain',

'Calories': 350,

'Price': 300,

'CarbonFootprint': '500gms',

'showInAll': true, // This item will appear in both "All" and "WeightGain"

},

{

'Name': 'Popcorn',

'Description': 'Yummy Salted Popcorn',

'Image': 'https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQWYksOSYUtzxmteDlS5AAbkM5fJpT7qGRIZg&s',

'Ingr': 'Corn,Salt',

'Type': 'LightSnack',

'Calories': 93,

```
    'Price': 150,
    'CarbonFootprint': '200 gms',
    'showInAll': true, // This item will appear in the "All" category
  },
  {
    'Name': 'SugarFree Brownie',
    'Description': 'Low Calorie Brownie',
    'Image': 'https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcT63vcW-
Cw75HeR0YZcRoHj0yF40uEb21szqyL2VJ2UJajbd1ziebQq0d3jU5AxEp-dBDA&usqp=CAU',
    'Ingr': 'Cocoa,Dough,Milk,DatesBakingPowder',
    'Type': 'SugarFree',
    'Calories': 70,
    'Price': 150,
    'CarbonFootprint': '200 gms',
    'showInAll': true, // This item will appear in the "All" category
  },
 ];

 @override
 void initState() {
  super.initState();
  // Initialize the local cart state with the passed cart
  _cart = Map.from(widget.cart);
 }
 // Callback to handle quantity changes
 void _handleQuantityChanged(String itemId, int quantity) {
  setState(() {
    _cart[itemId] = quantity; // Update the local cart state
  });
```

```
    print('Cart updated - Item ID: $itemId, New Quantity: $quantity');
    print('Updated Cart: $_cart');
    // Notify the parent widget of the updated cart state
    widget.onCartUpdated(_cart);
  }
  @override
  Widget build(BuildContext context) {
    final double screenWidth = MediaQuery.of(context).size.width;
final double mainAxisSpacing = screenWidth * 0.02;
    final double crossAxisSpacing = screenWidth * 0.02;
    // Filter menu items based on the selected category and search term
    List<Map<String, dynamic>> filteredMenuItems = menuItems.where((item) {
      bool matchesCategory = _selectedCategory == 'All'
          ? item['showInAll'] // Only show items with showInAll: true in the "All" category
          : item['Type'] == _selectedCategory; // Show items of the selected category
      bool matchesSearch = _searchTerm.isEmpty ||
          item['Name'].toLowerCase().contains(_searchTerm.toLowerCase()) ||
          item['Description'].toLowerCase().contains(_searchTerm.toLowerCase());
      return matchesCategory && matchesSearch;
    }).toList();
    return Padding(
      padding: const EdgeInsets.fromLTRB(0, 10, 0, 10),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Center(
            child: Text(
              'NUTRIEATS',
```

```
    style: TextStyle(
     color: Color.fromARGB(255, 24, 79, 87),
     fontSize: 45,
     fontWeight: FontWeight.w500,
     fontFamily: 'Genos',
    ),
   ),
  ),
 ),
 Padding(
  padding: const EdgeInsets.symmetric(horizontal: 25, vertical: 10),
  child: TextField(
   controller: _searchController,
   cursorColor: Color.fromARGB(255, 24, 79, 87),
   decoration: InputDecoration(
    contentPadding: EdgeInsets.symmetric(vertical: 15.0, horizontal: 10.0),
    border: OutlineInputBorder(
     borderRadius: BorderRadius.circular(30),
     borderSide: BorderSide(
      width: 0.8,
      color: Color.fromARGB(255, 24, 79, 87),
     ),
    ),
    enabledBorder: OutlineInputBorder(
     borderRadius: BorderRadius.circular(30),
     borderSide: BorderSide(
      width: 0.8,
      color: Color.fromARGB(255, 49, 49, 49),
     ),
```

```
  ),
  focusedBorder: OutlineInputBorder(
   borderRadius: BorderRadius.circular(30),
   borderSide: BorderSide(
    width: 1.5,
    color: Color.fromARGB(255, 71, 71, 71),
   ),
  ),
  hintText: 'Search',
  hintStyle: TextStyle(fontSize: 18),
  prefixIcon: Icon(
   Icons.search,
   size: 30,
  ),
  suffixIcon: IconButton(
   onPressed: () {
    _searchController.clear();
    if (mounted) {
     setState(() {
      _searchTerm = '';
     });
    }
   },
   icon: Icon(Icons.clear),
  ),
 ),
 onChanged: (value) {
  if (mounted) {
```

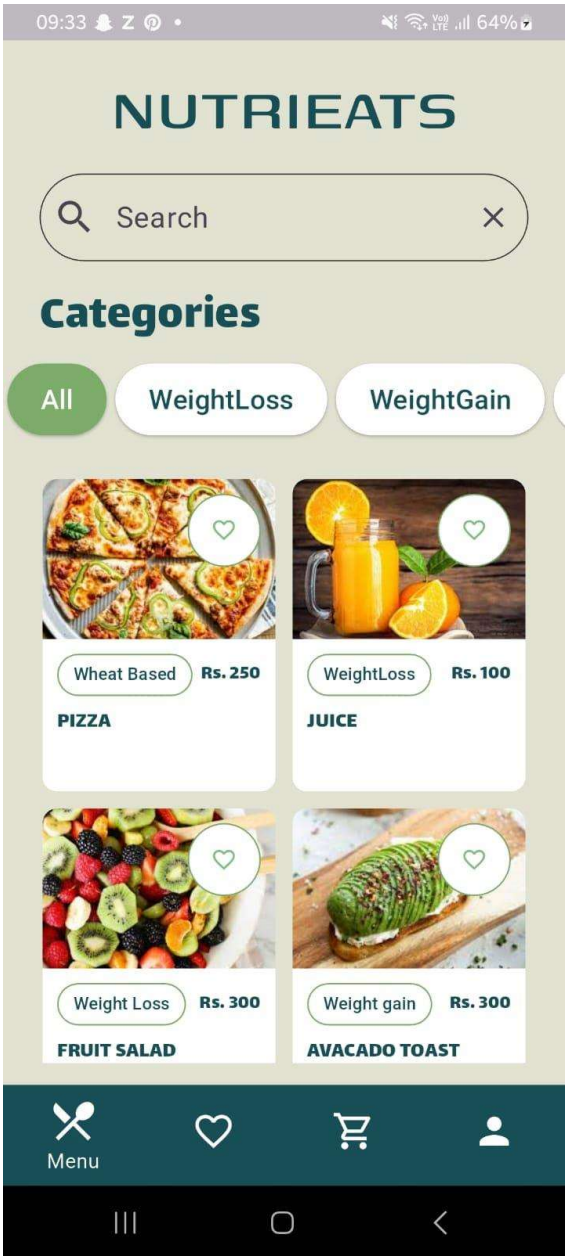
```
        setState(() {
          _searchTerm = value.trim();
        });
      }
    },
  ),
),
Padding(
  padding: const EdgeInsets.only(top: 5, left: 25),
  child: Text(
    'Categories',
    style: TextStyle(
      fontFamily: 'Lalezar',
      color: Color.fromARGB(255, 24, 79, 87),
      fontSize: 30,
    ),
  ),
),
Padding(
  padding: const EdgeInsets.fromLTRB(0, 5, 0, 20),
  child: SingleChildScrollView(
    scrollDirection: Axis.horizontal,
    child: Container(
      height: 55,
      child: Row(
        children: [
          _buildCategoryButton('All'),
          _buildCategoryButton('WeightLoss'),
```

```
        _buildCategoryButton('WeightGain'),
        _buildCategoryButton('WheatBased'),
        _buildCategoryButton('SugarFree'),
        _buildCategoryButton('LightSnack'),
      ],
    ),
   ),
  ),
),
Expanded(
 child: Padding(
  padding: const EdgeInsets.symmetric(horizontal: 25, vertical: 5),
  child: GridView.builder(
   gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
    crossAxisCount: 2,
    childAspectRatio: 0.75,
    mainAxisSpacing: mainAxisSpacing,
    crossAxisSpacing: crossAxisSpacing,
   ),
   itemCount: filteredMenuItems.length,
   itemBuilder: (context, index) {
    final data = filteredMenuItems[index];

    return MenuItemWidget(
     name: data['Name'],
     des: data['Description'],
     img: data['Image'],
     ingr: data['Ingr'],
```

```
            type: data['Type'],
            cal: data['Calories'],
            price: data['Price'],
            carbonfootprint: data['CarbonFootprint'],
            itemId: index.toString(),
            quantity: _cart[index.toString()] ?? 0, // Get quantity from cart or default to 0
            onQuantityChanged: _handleQuantityChanged, // Pass the callback
          );
        },
      ),
    ),
  ),
 ],
 ),
 );
}

Widget _buildCategoryButton(String category) {
 return Padding(
  padding: const EdgeInsets.all(3.0),
  child: SizedBox(
   height: 50,
   child: ElevatedButton(
    onPressed: () {
     if (mounted) {
      setState(() {
       _selectedCategory = category;
      });
```

```
        }
      },
      style: ElevatedButton.styleFrom(
       backgroundColor: _selectedCategory == category
          ? Color.fromARGB(255, 125, 172, 106)
          : Colors.white,
       foregroundColor: _selectedCategory == category
          ? Colors.white
          : Color.fromARGB(255, 24, 79, 87),
      ),
      child: Text(
       category,
       style: TextStyle(
        fontSize: 17,
       ),
      ),
     ),
    ),
   );
 }
}
```

Output:



Conclusion:

This experiment demonstrated how to add icons, images, and custom fonts in a Flutter app, improving its appearance and user experience. It highlighted the flexibility of Flutter in creating a customized and engaging interface.

Aim:

To learn how to create an interactive form using the `Form` widget in Flutter to collect and validate user input.

Theory:

Flutter's `Form` widget is used to manage form elements, including text fields, checkboxes, and other input widgets. It allows for form validation, which ensures that user inputs are correct before submission. The `TextFormField` widget is commonly used for text input in forms. The `FormState` object helps in managing the state of the form, including validation and resetting the form.

Code:

```
import 'package:cloud_firestore/cloud_firestore.dart';
import "package:flutter/material.dart";
import "package:nutrilab/authservice.dart";
import 'dart:developer';
import "package:nutrilab/navbarwidget.dart";
import "./deco.dart";
class GoToRegisterPage extends StatefulWidget {
  const GoToRegisterPage({super.key});
  @override
  State<GoToRegisterPage> createState() => _GoToRegisterPageState();
}
class _GoToRegisterPageState extends State<GoToRegisterPage> {
  final _auth = AuthService();
  final _name = TextEditingController();
  final _email = TextEditingController();
  final _password = TextEditingController();
  final _mobile = TextEditingController();
  @override
  void dispose() {
    super.dispose();
    _email.dispose();
    _password.dispose();
    _name.dispose();
    _mobile.dispose();
  }
  Future addUser(String name, email, int mobile) async {
    await FirebaseFirestore.instance.collection('users').doc(email).set({
      'name': name,
      'email': email,
      'mobile': mobile,
      'liked': [],
```

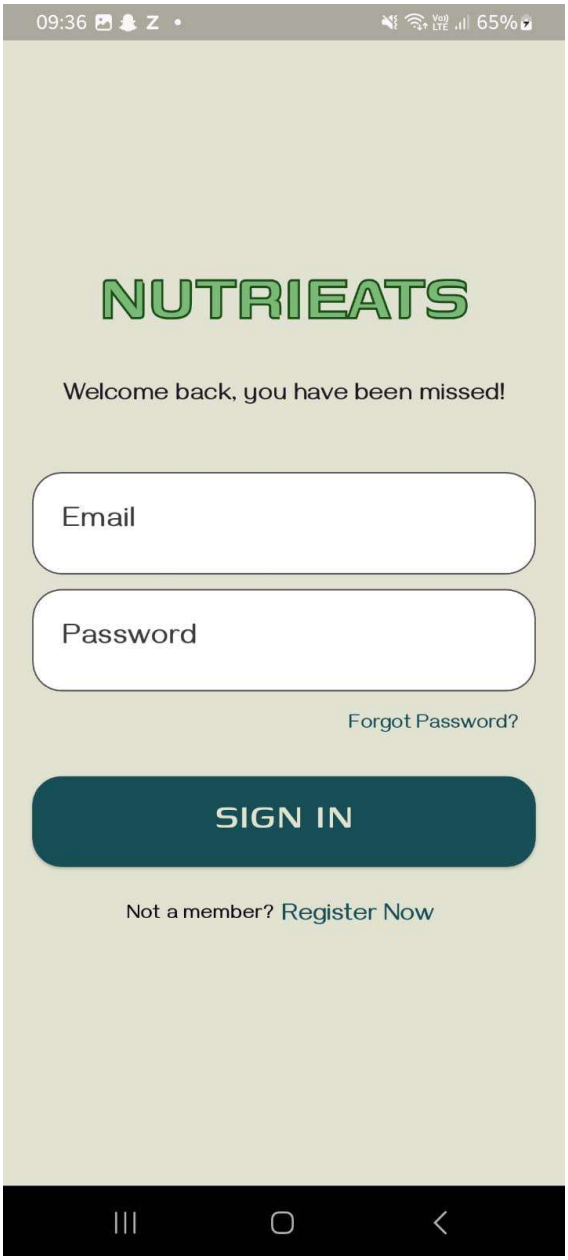
```
    'cart': {},
    'address': {},
  });
}
@override
Widget build(BuildContext context) {
  final screenWidth = MediaQuery.of(context).size.width;
  // Base font sizes
  final double baseFontSize = screenWidth * 0.05;
  final double smallFontSize = baseFontSize * 0.65;
  final double mediumFontSize = baseFontSize * 0.8;
  final double largeFontSize = baseFontSize * 1.4;
  return Scaffold(
    backgroundColor: Color.fromARGB(255, 225, 226, 209),
    body: Center(
      child: Container(
        width: MediaQuery.of(context).size.width - 40,
        child: Column(
          children: [
            Spacer(flex: 16),
            Text(
              "Start an account today!",
              style: TextStyle(
                fontFamily: 'Gayathri',
                fontWeight: FontWeight.w700,
                fontSize: mediumFontSize,
              ),
            ),
            Spacer(flex: 4),
            TextFormField(
              controller: _name,
              cursorColor: Color.fromARGB(255, 24, 79, 87),
              style: TextStyle(
                color: const Color.fromARGB(255, 0, 0, 0),
                fontFamily: 'Gayathri',
                fontWeight: FontWeight.w700,
                fontSize: mediumFontSize,
              ),
              decoration: myDecorationField.copyWith(hintText: 'Name'),
              keyboardType: TextInputType.emailAddress,
            ),
            Spacer(),
            TextFormField(
              controller: _email,
              cursorColor: Color.fromARGB(255, 24, 79, 87),
              style: TextStyle(
                color: const Color.fromARGB(255, 0, 0, 0),
```

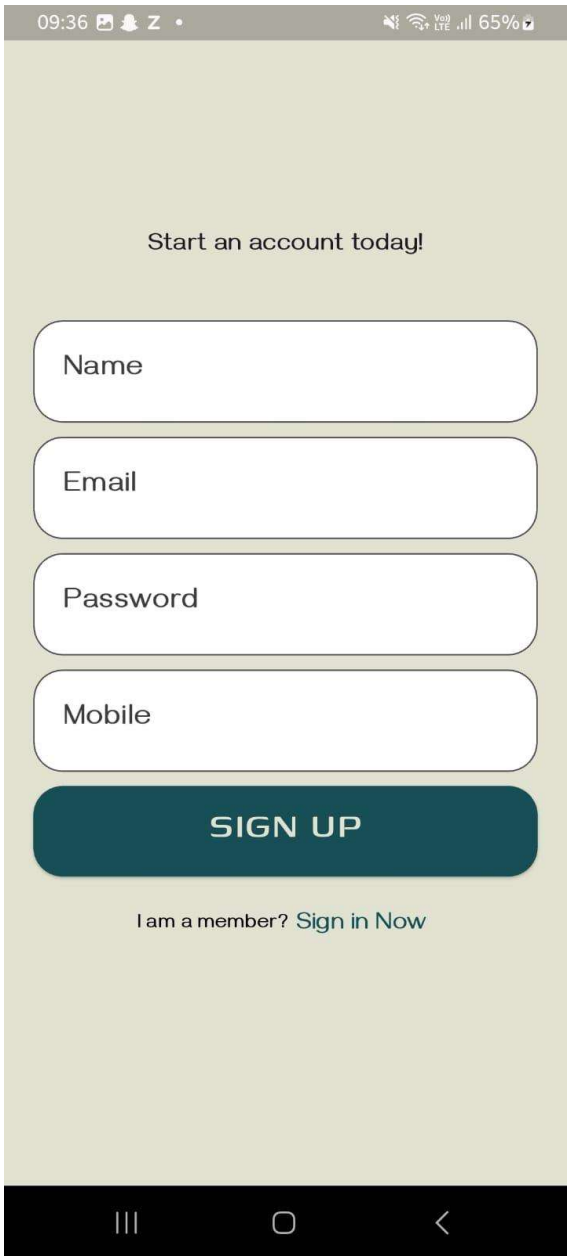
```
    fontFamily: 'Gayathri',
    fontWeight: FontWeight.w700,
    fontSize: mediumFontSize,
   ),
   decoration: myDecorationField.copyWith(hintText: 'Email'),
   keyboardType: TextInputType.emailAddress,
  ),
  Spacer(),
  TextFormField(
   obscureText: true,
   controller: _password,
   cursorColor: Color.fromARGB(255, 24, 79, 87),
   style: TextStyle(
    color: const Color.fromARGB(255, 0, 0, 0),
    fontFamily: 'Gayathri',
    fontWeight: FontWeight.w700,
    fontSize: mediumFontSize,
   ),
   decoration: myDecorationField.copyWith(hintText: 'Password'),
  ),
  Spacer(),
  TextFormField(
   controller: _mobile,
   cursorColor: Color.fromARGB(255, 24, 79, 87),
   style: TextStyle(
    color: const Color.fromARGB(255, 0, 0, 0),
    fontFamily: 'Gayathri',
    fontWeight: FontWeight.w700,
    fontSize: mediumFontSize,
   ),
   decoration: myDecorationField.copyWith(hintText: 'Mobile'),
   keyboardType: TextInputType.number,
  ),
  Spacer(),
  SizedBox(
   width: MediaQuery.of(context).size.width - 40,
   child: ElevatedButton(
    onPressed: _signup,
    style: ElevatedButton.styleFrom(
     padding: EdgeInsets.fromLTRB(100, 5, 100, 15),
     backgroundColor: Color.fromARGB(255, 24, 79, 87),
     shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(20),
     ),
    ),
    child: Text(
     "SIGN UP",
```

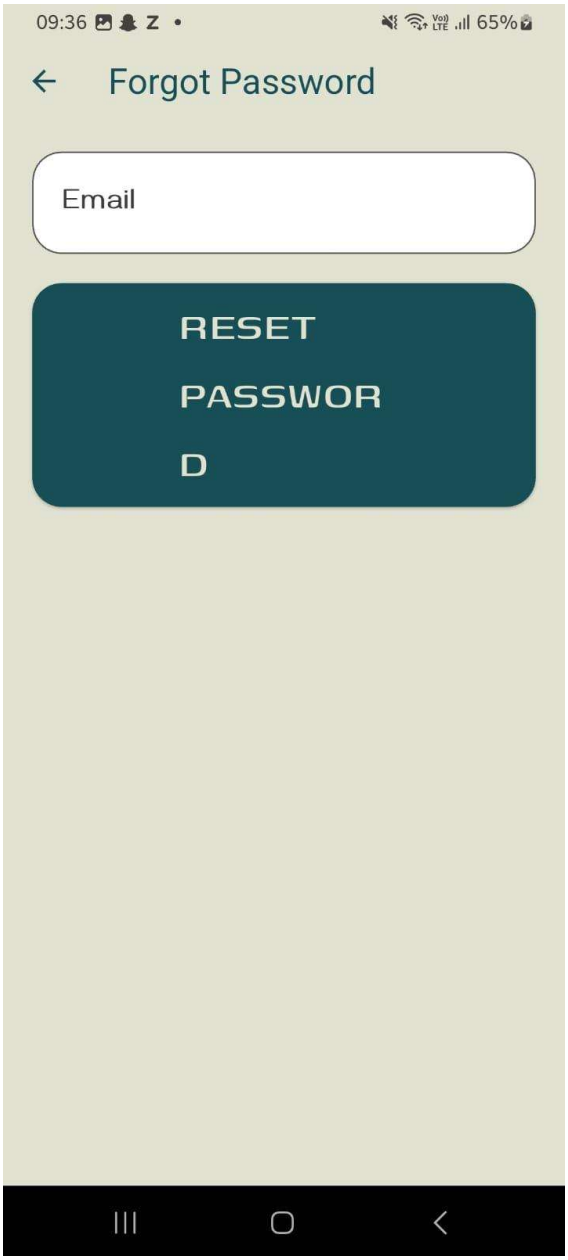
```
            style: TextStyle(
              fontSize: largeFontSize,
              fontWeight: FontWeight.w500,
              fontFamily: 'Genos',
              color: Color.fromARGB(255, 225, 226, 209),
            ),
          ),
        ),
      ),
      Spacer(),
      Row(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Text(
            "I am a member?",
            style: TextStyle(
              color: Color.fromARGB(255, 0, 0, 0),
              fontFamily: 'Gayathri',
              fontWeight: FontWeight.w700,
              fontSize: smallFontSize,
            ),
          ),
          TextButton(
            style: TextButton.styleFrom(
              splashFactory: NoSplash.splashFactory,
              padding: EdgeInsets.all(5),
            ),
            onPressed: () {
              Navigator.pop(context);
            },
            child: Text(
              "Sign in Now",
              style: TextStyle(
                color: Color.fromARGB(255, 24, 79, 87),
                fontFamily: 'Gayathri',
                fontWeight: FontWeight.w700,
                fontSize: mediumFontSize,
              ),
            ),
          ),
        ],
      ),
      Spacer(flex: 16),
    ],
  ),
 ),
),
```

```
  );
 }
 goToHome(BuildContext context) => Navigator.push(
  context,
  MaterialPageRoute(
   builder: (context) => BottomNav(cart: {}), // Pass an empty cart map
  ),
 );
 _signup() async {
  final user = await _auth.createUserWithEmailAndPassword(
      context, _email.text, _password.text);
  if (user != null) {
   log("User Created Successfully");
   addUser(_name.text, _email.text, int.parse(_mobile.text));
   goToHome(context);
  }
 }
```

Output:


09:36
65%
NUTRIEATS
Welcome back, you have been missed!
Email
Password
Forgot Password?
SIGN IN
Not a member? Register Now


09:36
65%
Start an account today!
Name
Email
Password
Mobile
SIGN UP
I am a member? Sign in Now



Conclusion: This experiment demonstrated how to create an interactive form in Flutter using the `Form` widget, allowing for efficient input collection and validation. It highlighted the importance of form management and validation in building user-friendly and error-free applications.

Aim:
To learn how to implement navigation, routing, and gestures in a Flutter app for better user interaction and app flow.

Theory:
In Flutter, navigation allows users to move between different screens (pages), and routing defines how the app handles these transitions. Flutter uses `Navigator` for navigation, while routing is managed through `MaterialPageRoute` or named routes. Gestures enable interactive actions like tapping, swiping, or dragging. Flutter provides gesture detection through widgets like `GestureDetector` to handle user inputs.

Code :

navbarwidget.dart

```
import 'package:flutter/material.dart';
import 'package:nutrilab/cartpage.dart';
import 'package:nutrilab/menupage.dart';
import 'package:nutrilab/profile.dart';
import 'package:nutrilab/savedpage.dart';

class BottomNav extends StatefulWidget {
final Map<String, int> cart;

const BottomNav({Key? key, required this.cart}) : super(key: key);

@override
State<BottomNav> createState() => _BottomNavState();
}

class _BottomNavState extends State<BottomNav> {
int myIndex = 0;

// Local cart state
Map<String, int> cart = {};

@override
void initState() {
super.initState();
// Initialize the local cart state with the passed cart
cart = Map.from(widget.cart);
```

```
}

// Callback to handle cart updates
void _handleCartUpdated(Map<String, int> updatedCart) {
setState(() {
cart = updatedCart; // Update the local cart state
});
print('Parent Cart Updated: $cart');
}

@override
Widget build(BuildContext context) {
List<Widget> pages = [
GoToMenuPage(
cart: cart, // Pass the cart map
onCartUpdated: _handleCartUpdated, // Pass the callback
),
GoToSavedPage(cart: cart), // Pass the cart map
GoToCartPage(cart: cart), // Pass the cart map
GoToProfile(),
];

return SafeArea(
child: Scaffold(
backgroundColor: Color.fromARGB(255, 225, 226, 209),
body: pages[myIndex],
bottomNavigationBar: BottomNavigationBar(
type: BottomNavigationBarType.fixed,
showUnselectedLabels: false,
selectedItemColor: Color.fromARGB(255, 253, 253, 253),
unselectedItemColor: Colors.white,
iconSize: 35,
backgroundColor: Color.fromARGB(255, 24, 79, 87),
onTap: (index) {
setState(() {
myIndex = index;
});
},
currentIndex: myIndex,
items: [
BottomNavigationBarItem(
icon: Icon(Icons.local_dining),
```

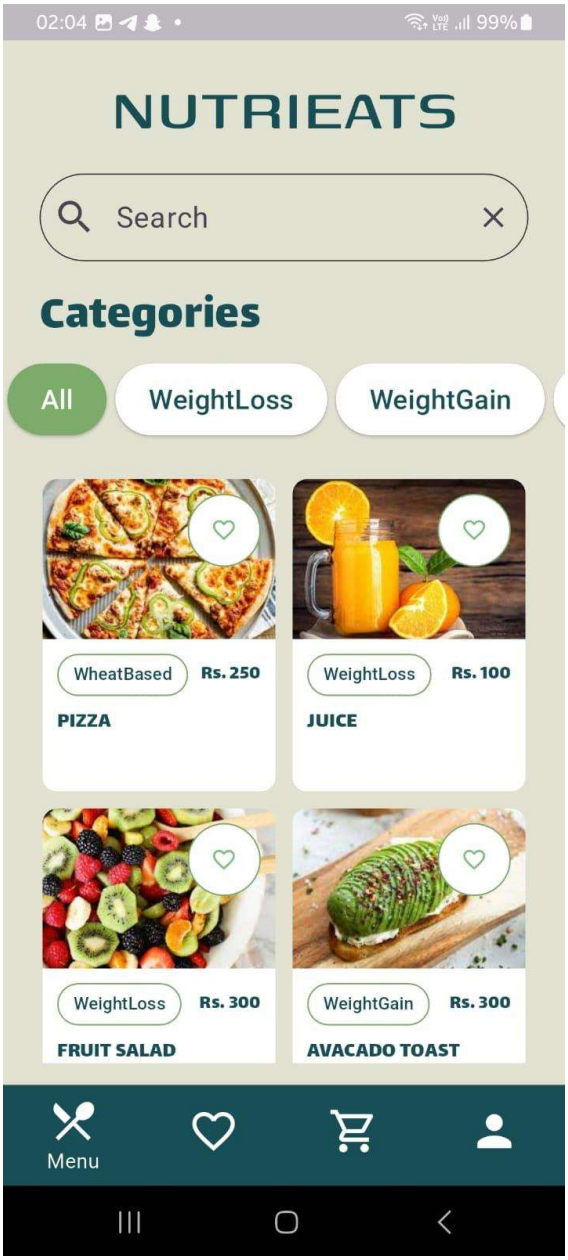
```
label: 'Menu',
),
BottomNavigationBarItem(
icon: Icon(Icons.favorite_border_outlined),
label: 'Saved',
),
BottomNavigationBarItem(
icon: Icon(Icons.shopping_cart_outlined),
label: 'Cart',
),
BottomNavigationBarItem(
icon: Icon(Icons.person_sharp),
label: 'Profile',
    ),

   ],

  ),

 ),

);

}

}
```

Cartpage.dart

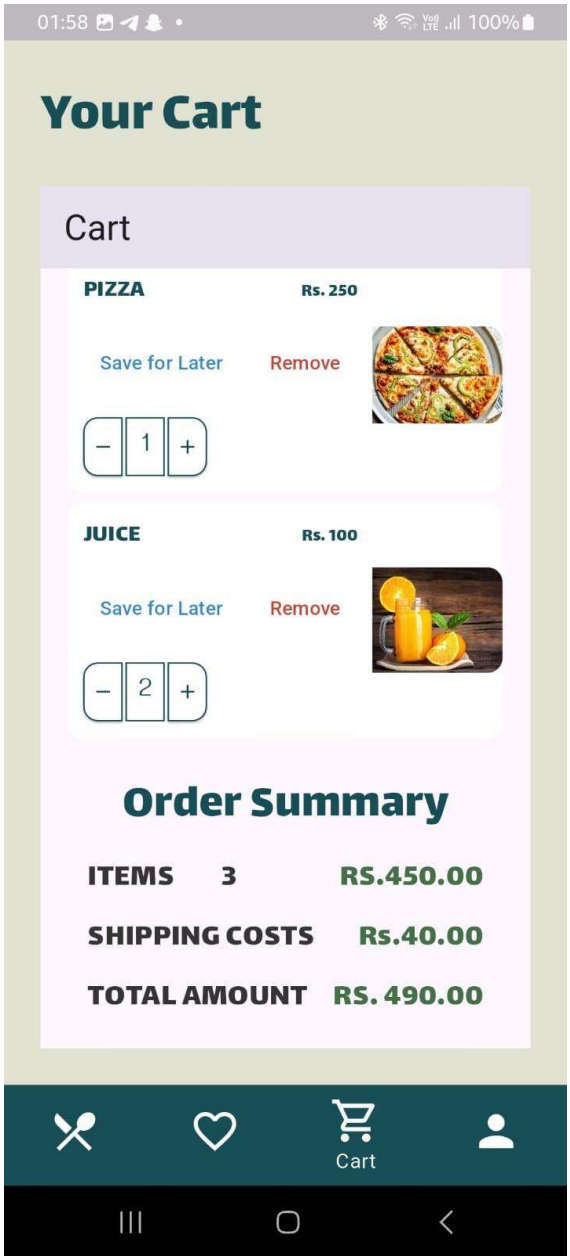
```
import 'package:flutter/material.dart';
import 'package:nutrilab/buildcart.dart';
class GoToCartPage extends StatefulWidget {
final Map<String, int> cart;
const GoToCartPage({Key? key, required this.cart}) : super(key: key);

@override
State<GoToCartPage> createState() => _GoToCartPageState();
}

class _GoToCartPageState extends State<GoToCartPage> {
@override
Widget build(BuildContext context) {
return Padding(padding: const EdgeInsets.all(25),
child: Column(
children: [
Align(
```

```
alignment: Alignment.centerLeft,
child: Padding(
padding: const EdgeInsets.only(bottom: 20.0),
child: Text(
'Your Cart',
style: TextStyle(
color: Color.fromARGB(255, 24, 79, 87),
fontSize: 35,
fontWeight: FontWeight.w500,
fontFamily: 'Lalezar'),
),
),
),
Expanded(
child: BuildCart(cart:widget.cart),
),


],
)
);
}
}
```

Output:


02:04
99%
NUTRIEATS
Search
Categories
All
WeightLoss
WeightGain
WheatBased
Rs. 250
PIZZA
WeightLoss
Rs. 100
JUICE
WeightLoss
Rs. 300
FRUIT SALAD
WeightGain
Rs. 300
AVACADO TOAST
Menu


01:05
96%
PIZZA
Wheat Based
Rs. 250
Delicious cheese pizza WheatBased
Calories: 300cal
Ingredients:
WheatDough,Cheese,Capsicum,Tomato
CarbonFootPrint: 300 gms
Quantity:
1
1 Item added to cart!



Conclusion:

This experiment showed how to use navigation, routing, and gestures in a Flutter app to create a smooth user experience. By applying these concepts, we can easily manage app flow and add interactivity, improving overall usability and app responsiveness.

Aim:
To learn how to implement navigation, routing, and gestures in a Flutter app for better user interaction and app flow.

Theory:
In Flutter, navigation allows users to move between different screens (pages), and routing defines how the app handles these transitions. Flutter uses `Navigator` for navigation, while routing is managed through `MaterialPageRoute` or named routes. Gestures enable interactive actions like tapping, swiping, or dragging. Flutter provides gesture detection through widgets like `GestureDetector` to handle user inputs.

Code :

navbarwidget.dart

```
import 'package:flutter/material.dart';
import 'package:nutrilab/cartpage.dart';
import 'package:nutrilab/menupage.dart';
import 'package:nutrilab/profile.dart';
import 'package:nutrilab/savedpage.dart';

class BottomNav extends StatefulWidget {
final Map<String, int> cart;

const BottomNav({Key? key, required this.cart}) : super(key: key);

@override
State<BottomNav> createState() => _BottomNavState();
}

class _BottomNavState extends State<BottomNav> {
int myIndex = 0;

// Local cart state
Map<String, int> cart = {};

@override
void initState() {
super.initState();
// Initialize the local cart state with the passed cart
cart = Map.from(widget.cart);
```

```
}

// Callback to handle cart updates
void _handleCartUpdated(Map<String, int> updatedCart) {
setState(() {
cart = updatedCart; // Update the local cart state
});
print('Parent Cart Updated: $cart');
}

@override
Widget build(BuildContext context) {
List<Widget> pages = [
GoToMenuPage(
cart: cart, // Pass the cart map
onCartUpdated: _handleCartUpdated, // Pass the callback
),
GoToSavedPage(cart: cart), // Pass the cart map
GoToCartPage(cart: cart), // Pass the cart map
GoToProfile(),
];

return SafeArea(
child: Scaffold(
backgroundColor: Color.fromARGB(255, 225, 226, 209),
body: pages[myIndex],
bottomNavigationBar: BottomNavigationBar(
type: BottomNavigationBarType.fixed,
showUnselectedLabels: false,
selectedItemColor: Color.fromARGB(255, 253, 253, 253),
unselectedItemColor: Colors.white,
iconSize: 35,
backgroundColor: Color.fromARGB(255, 24, 79, 87),
onTap: (index) {
setState(() {
myIndex = index;
});
},
currentIndex: myIndex,
items: [
BottomNavigationBarItem(
icon: Icon(Icons.local_dining),
```

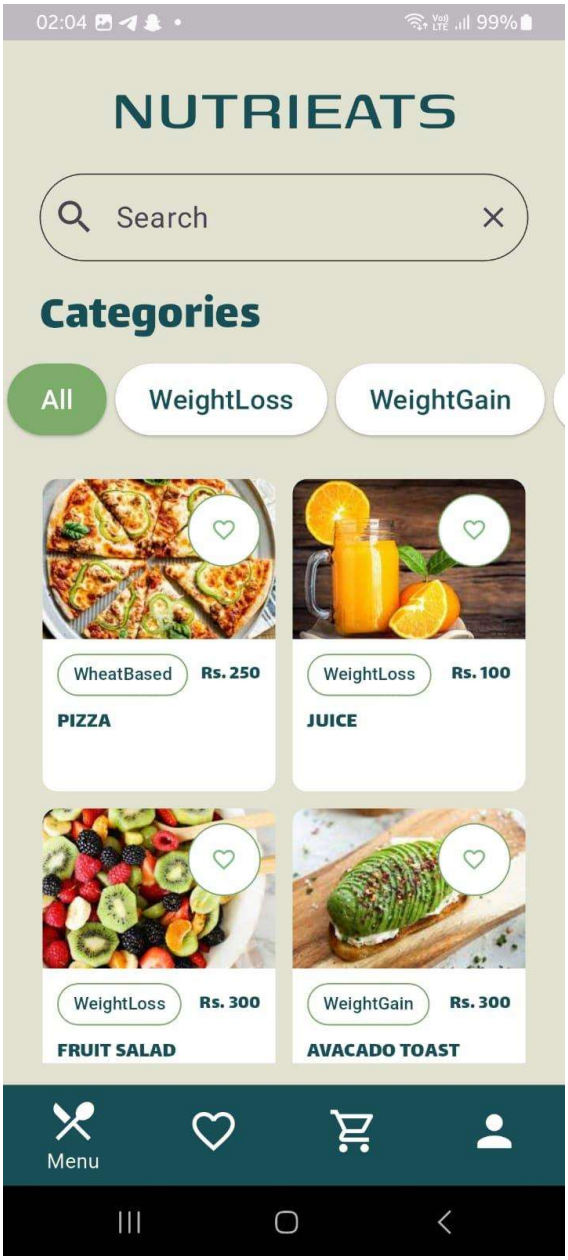
```
label: 'Menu',
),
BottomNavigationBarItem(
icon: Icon(Icons.favorite_border_outlined),
label: 'Saved',
),
BottomNavigationBarItem(
icon: Icon(Icons.shopping_cart_outlined),
label: 'Cart',
),
BottomNavigationBarItem(
icon: Icon(Icons.person_sharp),
label: 'Profile',
    ),

   ],

  ),

 ),

);

}

}
```

Cartpage.dart

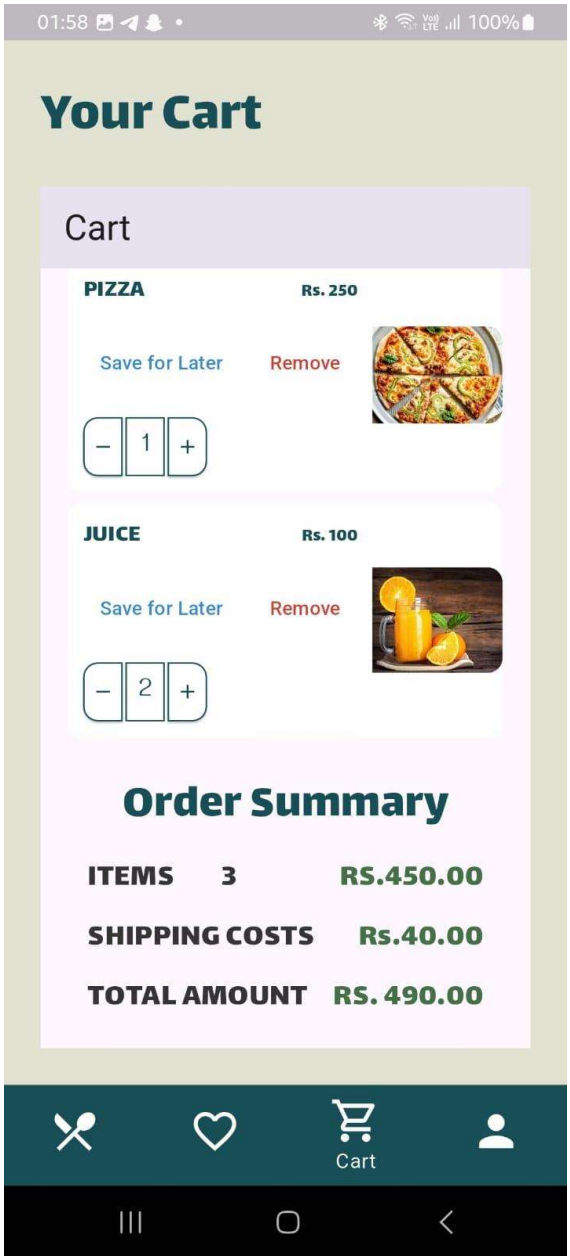
```
import 'package:flutter/material.dart';
import 'package:nutrilab/buildcart.dart';
class GoToCartPage extends StatefulWidget {
final Map<String, int> cart;
const GoToCartPage({Key? key, required this.cart}) : super(key: key);

@override
State<GoToCartPage> createState() => _GoToCartPageState();
}

class _GoToCartPageState extends State<GoToCartPage> {
@override
Widget build(BuildContext context) {
return Padding(padding: const EdgeInsets.all(25),
child: Column(
children: [
Align(
```

```
alignment: Alignment.centerLeft,
child: Padding(
padding: const EdgeInsets.only(bottom: 20.0),
child: Text(
'Your Cart',
style: TextStyle(
color: Color.fromARGB(255, 24, 79, 87),
fontSize: 35,
fontWeight: FontWeight.w500,
fontFamily: 'Lalezar'),
),
),
),
Expanded(
child: BuildCart(cart:widget.cart),
),


],
)
);
}
}
```

Output:


02:04
99%
NUTRIEATS
Search
Categories
All
WeightLoss
WeightGain
WheatBased
Rs. 250
PIZZA
WeightLoss
Rs. 100
JUICE
WeightLoss
Rs. 300
FRUIT SALAD
WeightGain
Rs. 300
AVACADO TOAST
Menu


01:05
96%
PIZZA
Wheat Based
Rs. 250
Delicious cheese pizza WheatBased
Calories: 300cal
Ingredients:
WheatDough,Cheese,Capsicum,Tomato
CarbonFootPrint: 300 gms
Quantity:
1
1 Item added to cart!



Conclusion:

This experiment showed how to use navigation, routing, and gestures in a Flutter app to create a smooth user experience. By applying these concepts, we can easily manage app flow and add interactivity, improving overall usability and app responsiveness.

Experiment 6: Firebase Integration

Aim: Connect Flutter with Firebase

Steps:

Add Firebase to Project:

Create a Firebase project at console.firebase.google.com.

Follow setup steps for Android/iOS/web.

Add Dependencies:

```
dependencies:
  firebase_core: ^2.15.0
  cloud_firestore: ^4.8.0
```

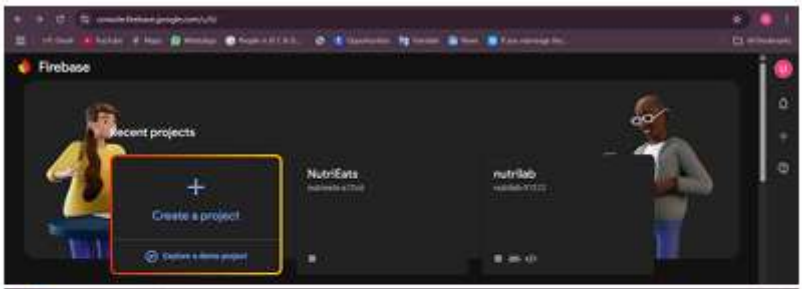
Initialize Firebase:

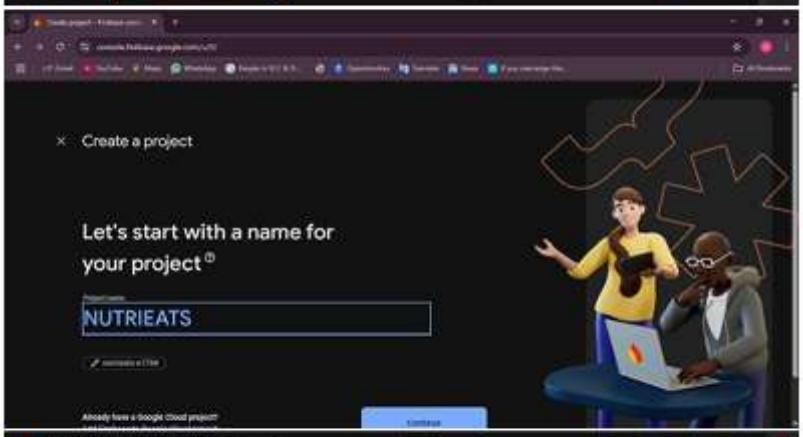
```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(MyApp());
}
```

Fetch Data:

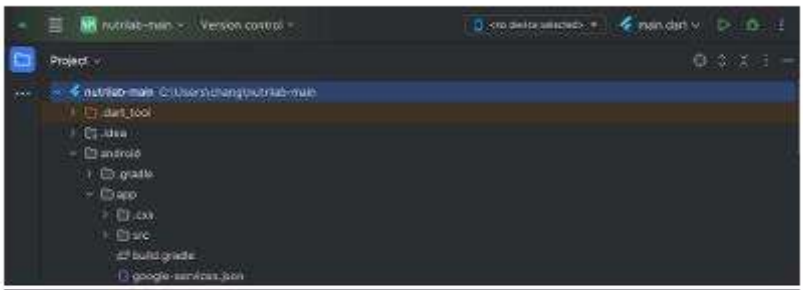
```
StreamBuilder(
  stream:
FirebaseFirestore.instance.collection('products').snapshots(),
  builder: (context, snapshot) {
    // Display data
  },
)
```

Conclusion: Firebase enables real-time data synchronization.


Firebase
Recent projects
Create a project
NutriEats
nutrilab


Create a project
Let's start with a name for your project
NUTRIEATS


Firebase
Project Overview
Authentication


nutrilab-main
Version control
Project
android
app
build.gradle
google-services.json


Firebase
Project Overview
Authentication
Users
Sign-in method
Templates
Usage
Settings
Extensions
Add user

Experiment No 7

Aim: To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.

Theory: **Steps**:

Create `manifest.json` in `web` folder:

json

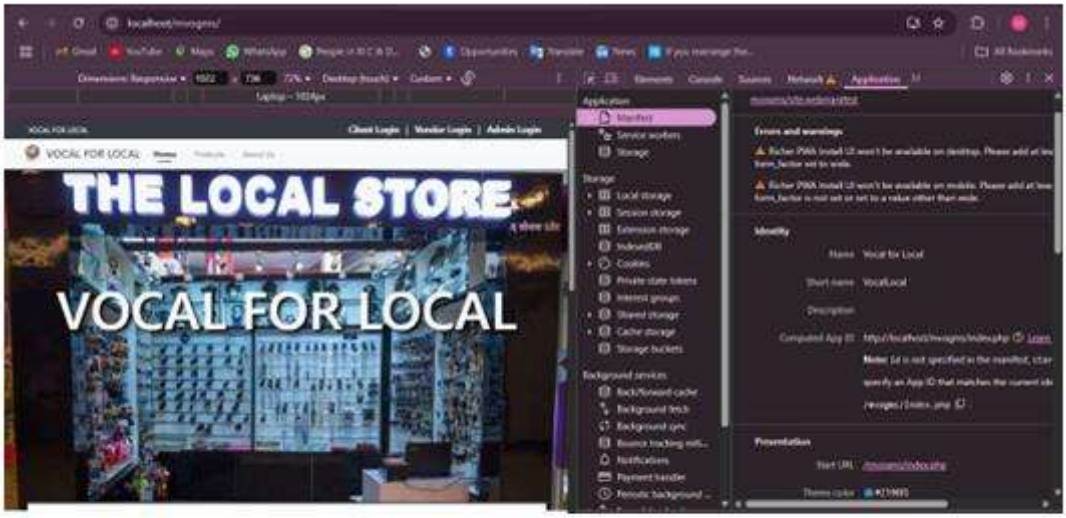
Copy

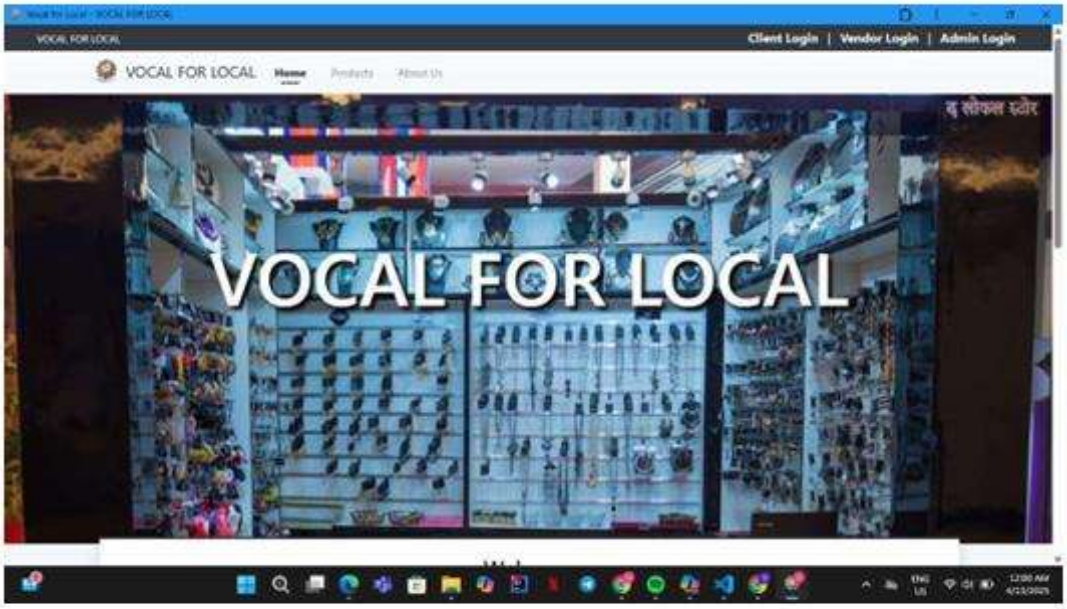
```
{
  "name": "E-Commerce PWA",
  "short_name": "Shop",
  "start_url": "/",
  "display": "standalone",
  "icons": [
    {
      "src": "icons/icon-192.png",
      "sizes": "192x192",
      "type": "image/png"
    }
  ]
}
```

Link in `web/index.html`:

```
<link rel="manifest" href="manifest.json">
```

Conclusion: The manifest enables "Add to Home Screen" functionality.




VOCAL FOR LOCAL
Client Login | Vendor Login | Admin Login
VOCAL FOR LOCAL
Home
Products
About Us
VOCAL FOR LOCAL

Experiment No 8

Aim: To implement Service worker events like fetch, sync and push for E-commerce PWA

Theory :

Steps:

Build the project:

bash

Copy

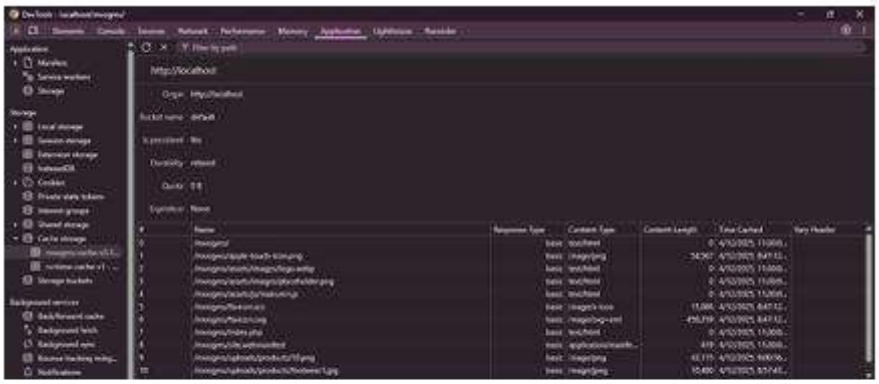
```
flutter build web
```

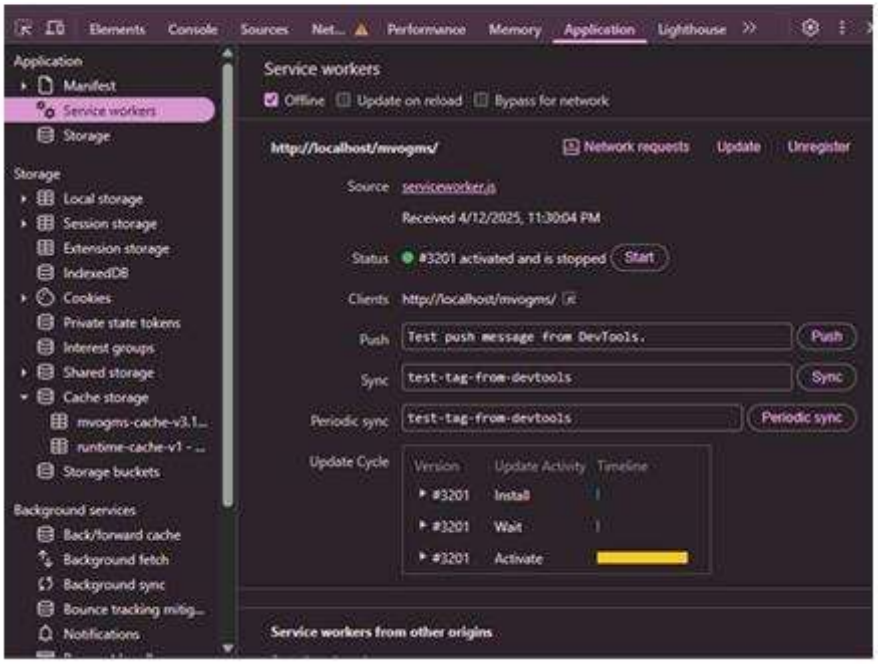
Commit and push the `build/web` folder to a GitHub repository.

Enable GitHub Pages:

Go to repo Settings → Pages → Select `main` branch and `/ (root)` folder.

Conclusion: GitHub Pages hosts PWAs for free.





Experiment No 9

Aim: To implement Service worker events like fetch, sync and push for E-commerce PWA

Theory:

Steps:

Create `sw.js` in the `web` folder.

Register Service Worker in `index.html`:

html

Copy

```
<script>
  if ('serviceWorker' in navigator) {
    navigator.serviceWorker.register('sw.js');
  }
</script>
```

Run HTML

Cache Assets in `sw.js`:

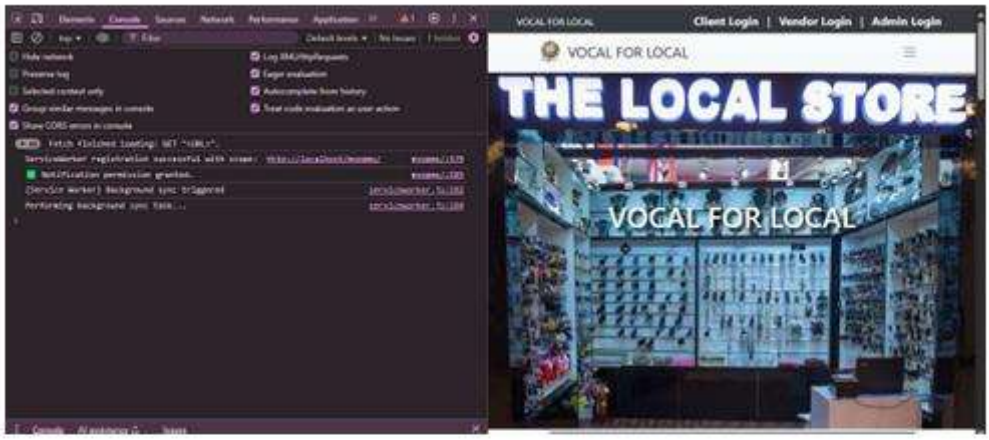
javascript

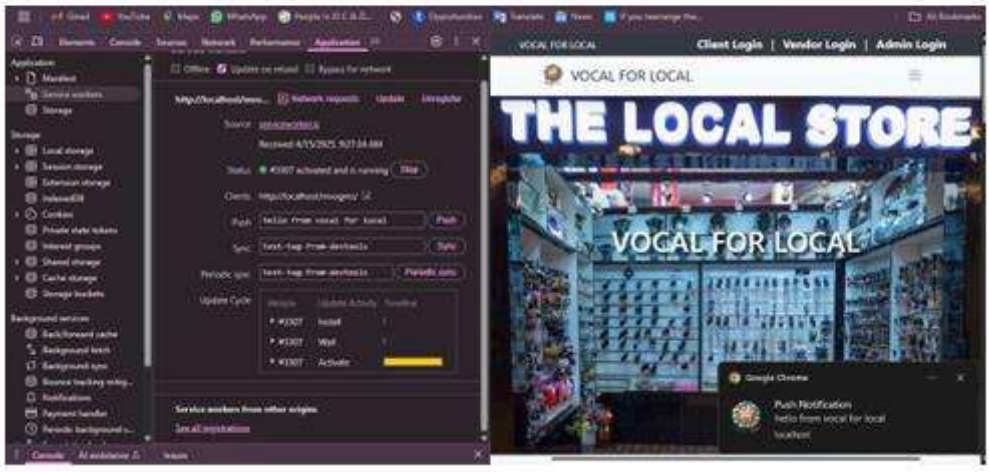
Copy

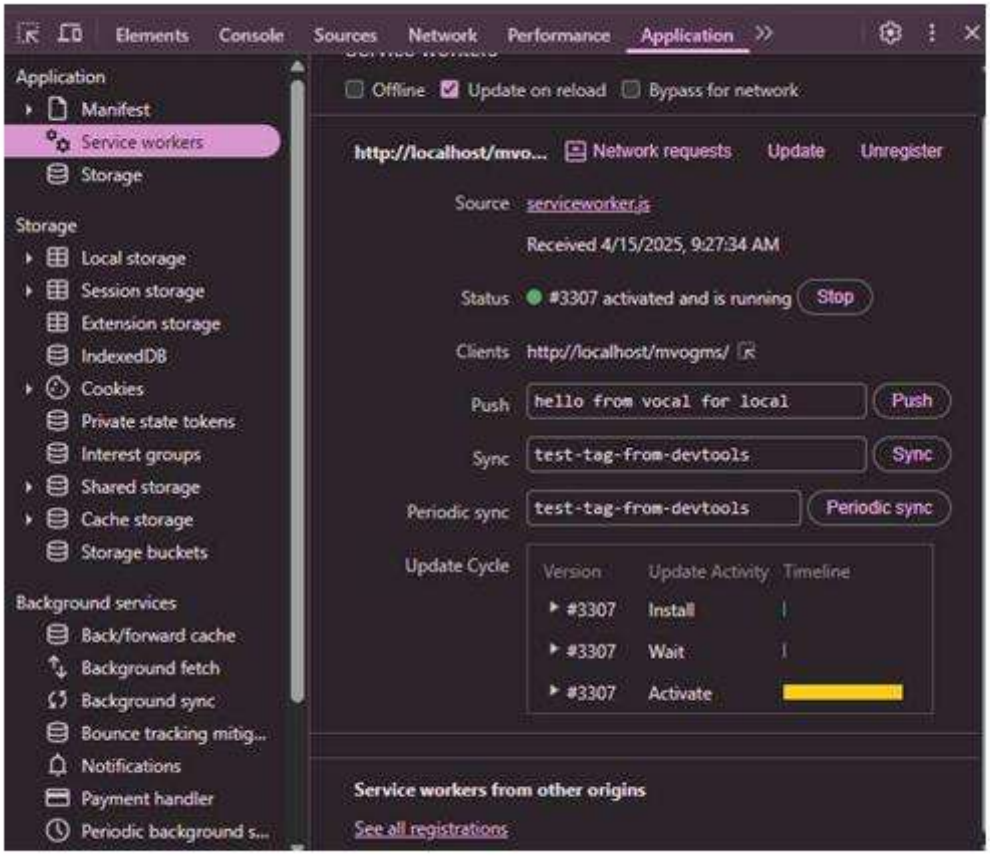
```
const CACHE_NAME = 'my-pwa-cache-v1';
const urlsToCache = ['/', '/main.dart.js', '/assets/...'];

self.addEventListener('install', (event) => {
  event.waitUntil(
    caches.open(CACHE_NAME).then((cache) =>
```

```
cache.addAll(urlsToCache))
  );
});

self.addEventListener('fetch', (event) => {
  event.respondWith(
    caches.match(event.request).then((response) => response ||
fetch(event.request))
  );
});
```

Conclusion: Service workers enable offline functionality and caching.






Elements
Console
Sources
Network
Performance
Application
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Storage buckets
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking mitig...
Notifications
Payment handler
Periodic background s...
Offline
Update on reload
Bypass for network
http://localhost/mvo...
Network requests
Update
Unregister
Source
serviceworker.js
Received 4/15/2025, 9:27:34 AM
Status
#3307 activated and is running
Stop
Clients
http://localhost/mvogms/
Push
hello from vocal for local
Push
Sync
test-tag-from-devtools
Sync
Periodic sync
test-tag-from-devtools
Periodic sync
Update Cycle
Version
Update Activity
Timeline
#3307
Install
#3307
Wait
#3307
Activate
Service workers from other origins
See all registrations

Experiment No 10

Aim: To study and implement deployment of Ecommerce PWA to GitHub Pages.

Theory:

Steps:
Build the project:

```
flutter build web
```

Commit and push the `build/web` folder to a GitHub repository.
Enable GitHub Pages:
Go to repo Settings → Pages → Select `main` branch and `/ (root)` folder.
Conclusion: GitHub Pages hosts PWAs for free.

Experiment No 11

Aim: To use google Lighthouse PWA Analysis Tool to test the PWA functioning.

Theory :

Steps:

Open the deployed PWA in Chrome.

Open DevTools → **Lighthouse**.

Run audit for **PWA**, **Performance**, and **Accessibility**.

Fix issues reported (e.g., missing icons, service worker errors).

Conclusion: Lighthouse ensures PWA compliance and performance.