Vivekanand Education Society's Institute of Technology

An Autonomous Institute Affiliated to University of Mumbai
Hashu Advani Memorial Complex, Collector Colony, Chembur East, Mumbai - 400074.

Since 1962

Department of Information Technology

CERTIFICATE

This is to certify that **Bhavesh Wadhwani** of **D15A** semester **VI,** have successfully completed necessary experiments in the **MAD & PWA Lab** under my supervision in **VES Institute of Technology** during the academic year **2023-2024.**

Lab Assistant

Subject Teacher

Mrs. Kajal Joseph

Principal

Head of Department

Dr. Mrs. Shalu Chopra

Name of the Course : MAD & PWA Lab **Course Code :** ITL604

Year/Sem/Class : D15A/D15B **A.Y.:** 23-24

Faculty Incharge : Mrs. Kajal Joseph.

Lab Teachers : Mrs. Kajal Jewani.

Email : kajal.jewani@ves.ac.in

Programme Outcomes: The graduate will be able to:

PO1) Basic Engineering knowledge: An ability to apply the fundamental knowledge in mathematics, science and engineering to solve problems in Computer engineering.

PO2) Problem Analysis: Identify, formulate, research literature and analyze computer engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and computer engineering and sciences.

PO3) Design/ Development of Solutions: Design solutions for complex computer engineering problems and design system components or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal and environmental considerations.

PO4) Conduct investigations of complex engineering problems using research-based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of information to provide valid conclusions.

PO5) Modern Tool Usage: Create, select and apply appropriate techniques, resources and modern computer engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6) The Engineer and Society: Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to computer engineering practice.

PO7) Environment and Sustainability: Understand the impact of professional computer engineering solutions in societal and environmental contexts and demonstrate knowledge of and need for sustainable development.

PO8) Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of computer engineering practice.

PO9) Individual and Team Work: Function effectively as an individual, and as a member or leader in diverse teams and in multidisciplinary settings.

PO10) Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations and give and receive clear instructions.

PO11) Project Management and Finance: Demonstrate knowledge and understanding of computer engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12) Life-long Learning: Recognize the need for and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

Program specific Outcomes

PSO1) An ability to manage and analyze data / information effectively for making better decisions.

PSO2) Demonstrate the ability to use state of the art technologies and tools including Free and Open Source Software (FOSS) tools in developing software.

Lab Objectives:

Sr. No.	Lab Objectives
The Lab experiments aims:	
1	Learn the basics of the Flutter framework.
2	Develop the App UI by incorporating widgets, layouts, gestures and animation
3	Create a production ready Flutter App by including files and firebase backend service.
4	Learn the Essential technologies, and Concepts of PWAs to get started as quickly and efficiently as possible
5	Develop responsive web applications by combining AJAX development techniques with the jQuery JavaScript library.
6	Understand how service workers operate and also learn to Test and Deploy PWA.

Lab Outcomes:

Sr. No.	Lab Outcomes	Cognitive levels of attainment as per Bloom's Taxonomy
On Completion of the course the learner/student should be able to:		
1	Understand cross platform mobile application development using Flutter framework	L1, L2
2	Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation	L3
3	Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS	L3, L4
4	Understand various PWA frameworks and their requirements	L1, L2
5	Design and Develop a responsive User Interface by applying PWA Design techniques	L3
6	Develop and Analyse PWA Features and deploy it over app hosting solutions	L3, L4

Index

Sr. No	Experiment Title	LO	DOP	DOS	Grade
1.	To install and configure the Flutter Environment	LO1	16/1	23/1	10
2.	To design Flutter UI by including common widgets.	LO2	23/1	30/1	11
3.	To include icons, images, fonts in Flutter app	LO2	30/1	6/2	11
4.	To create an interactive Form using form widget	LO2	6/2	13/2	10
5.	To apply navigation, routing and gestures in Flutter App	LO2	13/2	20/2	11
6.	To Connect Flutter UI with fireBase database	LO3	20/2	5/3	11
7.	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.	LO4	5/3	12/3	14
8.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA	LO5	12/3	19/3	14
9.	To implement Service worker events like fetch, sync and push for E-commerce PWA	LO5	19/3	26/3	14
10.	To study and implement deployment of Ecommerce PWA to GitHub Pages.	LO5	26/3	2/4	14
11.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.	LO6	5/3	12/3	14
12.	Assignment-1	LO1,LO2 ,LO3	2/2	5/2	4
13.	Assignment-2	LO4,LO5 ,LO6	19/3	21/3	4

MAD & PWA Lab Journal

Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	69
Name	Bhavesh Wadhwani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	10

Experiment 01

Aim: Create a Flutter Project using flutter to get Hello World output.

Code:

```
import 'package:flutter/material.dart';
void main() {
 runApp(const MyApp());
}
class MyApp extends StatelessWidget {
 const MyApp({Key? key}) : super(key: key);
 @override
 Widget build(BuildContext context)
   {return MaterialApp(
     title: 'Welcome to Flutter',
     home: Scaffold(
       appBar: AppBar(
         title: const Text('Welcome to Flutter'),
       ),
       body: const Center(
         child: Text('Hello bhavesh Wadhwani'),
       ),
     ),
   );
 }
}
```

1:35
DEBUG
Welcome to Flutter
Hello Bhavesh Wadhwani Roll no:-69

```
Administrator: Command Pro

! Doctor found issues in 1 category.

C:\Users\INFT505-22>flutter
Manage your Flutter app development.

Common commands:

  flutter create <output directory>
    Create a new Flutter project in the specified directory.

  flutter run [options]
    Run your Flutter application on an attached device or in an emulator.

Usage: flutter <command> [arguments]

Global options:
-h, --help                  Print this usage information.
-v, --verbose               Noisy logging, including all shell commands executed.
                            If used with "--help", shows hidden options. If used with "flutter doctor", shows additional
                            diagnostic information. (Use "-vv" to force verbose logging in those cases.)
-d, --device-id             Target device id or name (prefixes allowed).
    --version               Reports the version of this tool.
    --suppress-analytics    Suppress analytics reporting when this command runs.

Available commands:

Flutter SDK
  bash-completion   Output command line shell completion setup scripts.
  channel           List or switch Flutter channels.
```

```
C:\Users\INFT505-22>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.3.10, on Microsoft Windows [Version 10.0.22621.2861], locale en-US)
[✓] Android toolchain - develop for Android devices (Android SDK version 33.0.1)
[✓] Chrome - develop for the web
[✓] Visual Studio - develop for Windows (Visual Studio Community 2022 17.4.3)
[!] Android Studio (version 2023.1)
    X Unable to find bundled Java version.
[✓] VS Code (version 1.76.2)
[✓] Connected device (3 available)
[✓] HTTP Host Availability

! Doctor found issues in 1 category.

C:\Users\INFT505-22>
```

Output:

Conclusion: Thus we installed flutter and android studio and created our first flutter app withtext field.

MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	69
Name	Bhavesh Wadhwani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	11

Experiment 2

Aim: To design simple flutter UI by including common widgets.

Theory:
Common Widgets used are:

Scaffold: Represents the basic material design visual structure. It provides functionality like the AppBar, Drawer, BottomNavigationBar, and body of the app.

AppBar: Represents the top app bar that usually contains the app's title and actions.

Padding: Adds padding around its child widget.

Container: A box model that can contain other widgets and provides styling properties.

Positioned: Positions a child within a stack relative to the top-left corner.

Text: Displays a paragraph of text with various styling options.

TextField: Allows the user to enter text.

Column: A widget that displays its children in a vertical array.

Row: A widget that displays its children in a horizontal array.

Image.asset: Loads an image from an asset bundle.

GestureDetector: A widget that detects gestures.

BottomNavigationBar: A material design widget that displays a horizontal row of three to five icons and labels at the bottom of the screen.import 'package:flutter/material.dart';

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

class User {
 String name;
 String email;
 String address;
 User({required this.name, required this.email, required this.address});
}

class UserInfoPage extends StatefulWidget
 { @override
 _UserInfoPageState createState() => _UserInfoPageState();
}

class _UserInfoPageState extends State<UserInfoPage>
 { late User _user;

 @override
```

```
void initState()
  { super.initState();
  _user = User(
      name: "Bhavesh Wadhwani",
      email: "BhaveshWadhwani@gmail.com",
      address: "VESIT, Kurla");
}

@override
Widget build(BuildContext context)
  { return Scaffold(
    appBar:
      AppBar( backgroundColor:
      Colors.red, title: Text("User
      Information"),
    ),
    body:
      SingleChildScrollView( paddi
      ng: EdgeInsets.all(16.0), child:
      Column(
        crossAxisAlignment: CrossAxisAlignment.stretch,
        children: [
          SizedBox(height: 20.0),
          CircleAvatar(
            radius: 60,
            backgroundImage: AssetImage('assets/comp.png'),
          ),
          SizedBox(height: 20.0),
          ListTile(
            title: Text("Name"),
            subtitle: Text(_user.name),
            trailing: Icon(Icons.edit),
            onTap: () {
              // Add functionality to edit name
            },
          ),
          ListTile(
            title: Text("Email"),
            subtitle: Text(_user.email),
            trailing: Icon(Icons.edit),
            onTap: () {
              // Add functionality to edit email
            },
          ),
          ListTile(
            title: Text("Address"),
            subtitle: Text(_user.address),
            trailing: Icon(Icons.edit),
            onTap: () {
              // Add functionality to edit address
            },
          ),
        ],
      ),
    ),
  );
 }
}
```

```
void main() { runApp(MaterialApp( home:
 UserInfoPage(), theme: ThemeData(
    primaryColor: Colors.red, hintColor:
    Colors.yellow,
  ),
 ));
}
```

Output:

Conclusion:

To design simple flutter UI by including common widgets.

MAD & PWA Lab Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	69
Name	Bhavesh Wadhwani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	11

MAD PWA LAB 3

Aim: Exploring Flutter Widgets

Theory:

Images:
● Flutter supports the integration of various image formats, such as PNG, JPEG, GIF, and WebP.
● Images can be displayed using the Image widget, and different aspects like width, height, and alignment can be customized.
● Asset images are commonly used and are declared in the pubspec.yaml file under the flutter section. The Image.asset constructor is then used to load these assets
Fonts:
● Fonts in Flutter can be customized by specifying the TextStyle for text widgets.
● Custom fonts, including TrueType and OpenType fonts, can be used by declaring them in the pubspec.yaml file and specifying them in the TextStyle with the fontFamily property.
Icons:
● Flutter provides a wide range of icons through the Icons class, including Material Icons and Cupertino Icons.
● The Icon widget is used to display icons, and you can customize their appearance using properties like color, size, and opacity.
Button Types:
● Flutter provides various button types, including ElevatedButton, TextButton, OutlinedButton, and more.
● These buttons offer different visual styles, allowing developers to choose the one that best fits the application's design.
Button Properties:
● Buttons can be customized using properties such as onPressed (to define the action when the button is pressed), style (to modify the button's appearance), and child (to set the content inside the button)

Code:

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

import 'search.dart';

class Shop extends StatefulWidget
 {@override
 _ShopState createState() => _ShopState();
}

class _ShopState extends State<Shop> {
 final List<Map<dynamic, dynamic>> products = [
   {'name': 'Wallet', 'rating': 3.0, 'image': 'https://m.media-
amazon.com/images/I/71ZKFRCkdkL._SL1500_.jpg'},
   {'name': 'Mixer', 'rating': 3.0, 'image': 'https://m.media-
amazon.com/images/I/61isgOoxj8L._SL1500_.jpg'},
```

```
  {'name': 'IPhone 11', 'rating': 4.0, 'image':
'https://rukminim2.flixcart.com/image/416/416/k2jbyq80pkrrdj/mobile-refurbished/x/j/s/iphone-11-
128-d-mwm02hn-a-apple-0-original-imafkg242ugz8hwc.jpeg?q=70&crop=false'},
  {'name': 'IPhone 11', 'rating': 4.0, 'image': 'https://images.unsplash.com/photo-1519125323398-
675f0ddb6308?ixlib=rb-
0.3.5&ixid=eyJhcHBfaWQiOjEyMDd9&s=94a1e718d89ca60a6337a6008341ca50&auto=format&fit=c
rop&w=1950&q=80'},
  {'name': 'IPhone 11', 'rating': 4.0, 'image': 'https://images.unsplash.com/photo-1519125323398-
675f0ddb6308?ixlib=rb-
0.3.5&ixid=eyJhcHBfaWQiOjEyMDd9&s=94a1e718d89ca60a6337a6008341ca50&auto=format&fit=c
rop&w=1950&q=80'},
  {'name': 'IPhone 11', 'rating': 4.0, 'image': 'https://images.unsplash.com/photo-1519125323398-
675f0ddb6308?ixlib=rb-
0.3.5&ixid=eyJhcHBfaWQiOjEyMDd9&s=94a1e718d89ca60a6337a6008341ca50&auto=format&fit=c
rop&w=1950&q=80'},
  {'name': 'IPhone 11', 'rating': 4.0, 'image': 'https://images.unsplash.com/photo-1519125323398-
675f0ddb6308?ixlib=rb-
0.3.5&ixid=eyJhcHBfaWQiOjEyMDd9&s=94a1e718d89ca60a6337a6008341ca50&auto=format&fit=c
rop&w=1950&q=80'},
  {'name': 'IPhone 11', 'rating': 4.0, 'image': 'https://images.unsplash.com/photo-1519125323398-
675f0ddb6308?ixlib=rb-
0.3.5&ixid=eyJhcHBfaWQiOjEyMDd9&s=94a1e718d89ca60a6337a6008341ca50&auto=format&fit=c
rop&w=1950&q=80'},
  {'name': 'IPhone 11', 'rating': 4.0, 'image': 'https://images.unsplash.com/photo-1519125323398-
675f0ddb6308?ixlib=rb-
0.3.5&ixid=eyJhcHBfaWQiOjEyMDd9&s=94a1e718d89ca60a6337a6008341ca50&auto=format&fit=c
rop&w=1950&q=80'},
  {'name': 'IPhone 12', 'rating': 5.0, 'image': 'https://images.unsplash.com/photo-1523205771623-
e0faa4d2813d?ixlib=rb-
0.3.5&ixid=eyJhcHBfaWQiOjEyMDd9&s=89719a0d55dd05e2deae4120227e6efc&auto=format&fit=c
rop&w=1953&q=80'},
 ];
 final GlobalKey<ScaffoldState> scaffoldKey = GlobalKey<ScaffoldState>();
 @override
 Widget build(BuildContext context)
  {return
  DefaultTabController( length: 2,
    child:
      Scaffold( key:
      scaffoldKey,
      appBar: AppBar(
        actions:
          <Widget>[IconBu
          tton(
            icon: Icon(Icons.search, color: Colors.white),
            onPressed: () {
              scaffoldKey.currentState!
                  .showBottomSheet((context) => ShopSearch());
            },
          )
        ],
        title: Text('Shop'),
      ),
      body: Builder(
        builder: (BuildContext context)
          {return DefaultTabController(
              length: 2, child:
              Column(
                children:
                  <Widget>[Contain
                  er(
                    constraints: BoxConstraints(maxHeight: 150.0),
                    child: Material(
                      color: Theme.of(context).colorScheme.secondary,
                      child: TabBar(
                        indicatorColor: Colors.red,
```

```
      tabs: [
        Tab(icon: Icon(Icons.view_list)),
        Tab(icon: Icon(Icons.grid_on)),
      ],
    ),
  ),
),
Expanded(
  child:
    TabBarView(child
    ren: [ Container(
        child: ListView(
          children: products.map((product)
            {return Builder(
              builder: (BuildContext context)
                {return InkWell(
                  onTap: ()
                    { print('Card
                    tapped.');
                  },
                  child: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: <Widget>[
                      Divider( h
                        eight: 0,
                      ),
                      Padding(
                        padding: const EdgeInsets.only(top: 10.0, bottom: 10.0),
                        child: ListTile(
                          trailing: Icon(Icons.navigate_next),
                          leading: ClipRRect(
                            borderRadius: BorderRadius.circular(5.0),child:
                            Container(
                              decoration:
                                  BoxDecoration(color:
                                  Colors.red
                              ),
                              child:
                                CachedNetworkImage(fit:
                                BoxFit.cover,
                                imageUrl: product['image'],
                                placeholder: (context, url) => Center(
                                    child: CircularProgressIndicator()
                                ),
                                errorWidget: (context, url, error) => new Icon(Icons.error),
                              ),
                            ),
                          ),
                          title:
                            Text( product['n
                            ame'],style:
                            TextStyle(
                                fontSize: 14
                            ),
                          ),
                          subtitle: Column(
                            crossAxisAlignment: CrossAxisAlignment.start,
                            children: <Widget>[
                              Row(
                                children:
                                  <Widget>[Padding
                                  (
                                    padding: const EdgeInsets.only(top: 2.0, bottom: 1),
                                    child: Text('\$200', style: TextStyle(
                                      color: Theme.of(context).colorScheme.secondary,
                                      fontWeight: FontWeight.w700,
                                    )),
                                  ),
```

```
                    Padding(
                      padding: const EdgeInsets.only(left: 6.0),
                      child: Text('(\$400)', style: TextStyle(
                          fontWeight: FontWeight.w700,
                          fontStyle: FontStyle.italic,
                          color: Colors.grey,
                          decoration: TextDecoration.lineThrough
                      )),
                    )
                  ],
                ),
                Row(
                  children:
                    <Widget>[RatingS
                    tars(
                      value: product['rating'],
                      starSize: 16,
                      valueLabelColor: Colors.amber,
                      starColor: Colors.amber,
                    )
                  ],
                )
              ],
            ),
          ),
        ),
      ],
    ),
  );
},
);
}).toList(),
),
),
Container(
  child:
    GridView.count(shrin
    kWrap: true,
    crossAxisCount: 2,
    childAspectRatio: 0.7,
    padding: EdgeInsets.only(top: 8, left: 6, right: 6, bottom: 12),
    children: List.generate(products.length, (index) {
      return
        Container(child:
        Card(
          clipBehavior: Clip.antiAlias,
          child: InkWell(
            onTap: ()
              { print('Card
              tapped.');
            },
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: <Widget>[
                SizedBox(
                  height: (MediaQuery.of(context).size.width / 2 - 5),
                  width: double.infinity,
                  child:
                    CachedNetworkImage(fit:
                    BoxFit.cover,
                    imageUrl: products[index]['image'],
                    placeholder: (context, url) => Center(
                        child: CircularProgressIndicator()
                    ),
                    errorWidget: (context, url, error) => new Icon(Icons.error),
                  ),
                ),
                Padding(
```

```
                    padding: const EdgeInsets.only(top: 5.0),
                    child: ListTile(
                     title: Text(
                      'Two Gold Rings',
                      style: TextStyle(
                         fontWeight: FontWeight.bold,
                         fontSize: 16
                      ),
                     ),
                     subtitle: Column(
                      crossAxisAlignment: CrossAxisAlignment.start,
                      children: <Widget>[
                       Row(
                        children:
                         <Widget>[Padding
                         (
                          padding: const EdgeInsets.only(top: 2.0, bottom: 1),
                          child: Text('\$200', style: TextStyle(
                           color: Theme.of(context).colorScheme.secondary,
                           fontWeight: FontWeight.w700,
                          )),
                         ),
                         Padding(
                          padding: const EdgeInsets.only(left: 6.0),
                          child: Text('(\$400)', style: TextStyle(
                             fontWeight: FontWeight.w700,
                             fontStyle: FontStyle.italic,
                             color: Colors.grey,
                             decoration: TextDecoration.lineThrough
                          )),
                         )
                        ],
                       ),
                       Row(
                        children:
                         <Widget>[RatingS
                         tars(
                          value: products[index]['rating'],
                          starSize: 16,
                          valueLabelColor: Colors.amber,
                          starColor: Colors.amber,
                         )
                        ],
                       )
                      ],
                     ),
                    ),
                   ),
                  ),
                 ),
                );
               }),
              ),
             ),
            ],
           ),
          ),
         ],
        ));
       },
      ),
     ),
    );
```

```
 }
}
```

Output:

Shop
Back
storite
MorphyRichards
Two Gold Rings
$200 ($400)
3.0/5.0
Two Gold Rings
$200 ($400)
3.0/5.0

-

Conclusion:

Thus we studied how to design UI using various icons ,images,buttons.

MAD & PWA Lab Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	69
Name	Bhavesh Wadhwani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	10

MAD PWA LAB 4

Aim: To design Flutter UI by including common widgets.

Widgets:

1. Column
2. Container
3. Row
4. Stack
5. Padding
6. Textfield
7. Button

Code:

```
import 'dart:js_interop';

// import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:flutter_ecommerce_ui_kit/blocks/auth_block.dart';
import 'package:flutter_ecommerce_ui_kit/models/user1.dart';
import 'package:provider/provider.dart';

class SignUp extends StatefulWidget
 {@override
 _SignUpState createState() => _SignUpState();
}

class _SignUpState extends State<SignUp>
 {final _formKey = GlobalKey<FormState>();
 final User1 user = User1(password: '', username: '', email: '');
 late String confirmPassword;

 @override
 Widget build(BuildContext context)
  {return Center(
    child: Form( key:
      _formKey,
      child: SingleChildScrollView(
```

```
child: Padding(
  padding: const EdgeInsets.only(left: 15.0, right: 15.0),
  child: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: <Widget>[
      Padding(
        padding: const EdgeInsets.only(bottom: 12.0),
        child: TextFormField(
          validator: (value) {
            if (value!.isEmpty) {
              return 'Please Enter Username';
            }
            return null;
          },
          onSaved: (value)
            { setState(()
            { user.username = value!;
            });
          },
          decoration:
            InputDecoration(    hintText:
            'Enter Username', labelText:
            'Username',
          ),
        ),
      ),
      Padding(
        padding: const EdgeInsets.only(bottom: 12.0),
        child: TextFormField(
          validator: (value) {
            if (value!.isEmpty) {
              return 'Please Enter Email Address';
            }
            return null;
          },
          onSaved: (value)
            { setState(()
            { user.email = value!;
            });
          },
          decoration: InputDecoration(
```

```
      hintText: 'Enter Email',
      labelText: 'Email',
    ),
  ),
),
Padding(
  padding: const EdgeInsets.only(bottom: 12.0),
  child: TextFormField(
      validator: (value) {
        if (value!.isEmpty) {
          return 'Please Enter Password';
        }
        return null;
      },
      onSaved: (value)
        { setState(()
        { user.password = value!;
        });
      },
      onChanged: (text)
        { user.password =
        text;
      },
      decoration:
        InputDecoration(hintText:
        'Enter Password', labelText:
        'Password',
      ),
      obscureText: true),
),
TextFormField( vali
  dator: (value) {
    if (value!.isEmpty) {
      return 'Please Enter Confirm Password';
    } else if (user.password != confirmPassword)
      {return 'Password fields dont match';
    }
    return null;
  },
  onChanged: (text)
    { confirmPassword =
    text;
  },
```

```
            decoration:
              InputDecoration( hintText:
              'Enter Same Password',labelText:
              'Confirm Password',
            ),
            obscureText: true,
          ),
          Padding(
            padding: const EdgeInsets.only(top: 25.0),
            child: SizedBox(
              width: double.infinity,
              height: 50,
              child: Consumer<AuthBlock>(builder:
                  (BuildContext context, AuthBlock auth, Widget? child)
                {return ElevatedButton(
                  style: ElevatedButton.styleFrom(
                    primary: Theme.of(context).primaryColor,
                  ),
                  child: auth.loading && auth.loadingType == 'register'
                      ?
                        CircularProgressIndicator
                        (valueColor:
                            AlwaysStoppedAnimation<Color>(Colors.white),
                      )
                    : Text(
                        'Sign Up',
                        style: TextStyle(color: Colors.white),
                      ),
                  onPressed: () {
                    if (_formKey.currentState!.validate() &&
                        !auth.loading) {
                      _formKey.currentState!.save();
                      // If the form is valid, display a snackbar. In the real world,
                      // you'd often call a server or save the information in a
database.
                      auth.register(user);
                    } else{
                      String email = user.email;
                      String password = user.password;

FirebaseAuth.instance.createUserWithEmailAndPassword(email: email,password:
password).then((credential) {
```

```
                    // Handle successful user creation
                    print('User ${credential.user!.uid} created successfully.');
                  }).catchError((error) {
                    // Handle error during user creation
                    print('Error creating user: $error');
                  });
                  ScaffoldMessenger.of(context).showSnackBar(SnackB
                    ar(
                      content: Text('Sign up successful'),
                      backgroundColor: Colors.green, // Set color here
                    ),
                  );
                }
              },
            );
          },
        ),
      )
    ],
  ),
),
),
),
),
);
}
}
```

Output:

localhost:50318
localhost:50318/#/auth
Create Account
Username
Email
Password
Confirm Password
Sign Up
Sign In
Create Account

MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	69
Name	Bhavesh Wadhwani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	11

MAD and PWA Lab 5

AIM: To apply navigation, routing and gestures in Flutter App

THEORY:
In a Flutter app, implementing navigation, routing, and gestures eff ectively is crucial for creating a seamless user experience. Let's delve into a theory on how to apply these concepts in your Flutter app:

A) Navigation and Routing:

a. Navigation:
Navigation in Flutter refers to moving between diff erent screens or "routes" within your app. It's essential to organize your app's navigation in a clear and intuitive manner.

b. Routing:
Routing involves defining routes and managing transitions between them. Flutter provides a built-in routing system that allows you to define routes using a Route object. Common types of routes include MaterialPageRoute, CupertinoPageRoute, and Custom routes.

Application:

1. Route Management:
Define routes for each screen in your app using MaterialApp or CupertinoApp's "routes" parameter. This maps route names to builder functions.

```
routes: {
'/home': (context) => HomeScreen(),
'/profile': (context) => ProfileScreen(),
// Add more routes as needed
}
```

2. Navigation:
Use Navigator.push() to navigate to a new route when a user interacts with UI elements such as buttons or gestures.

```
onPressed: ()
{ Navigator.pushNamed(context, '/profile');
}
```

3.Navigation with Data: Pass data between routes using arguments.

```
Navigator.pushNamed(context, '/profile', arguments: userData);
```

4. Back Navigation:
Allow users to navigate back using Navigator.pop().

```
onPressed: () {
Navigator.pop(context);
}
```

5. Named Routes vs. Route Builders:
Decide whether to use named routes or route builders based on your app's complexity and navigation requirements. Named routes are simpler, while route

builders off er more customization.
C) Gestures:
a. Gestures:
Gestures enable users to interact with your app through touch-based actions like tapping, swiping, dragging, etc. Flutter provides GestureDetector and various gesture recognizers to handle these interactions.
b. Theory/Application:
1. GestureDetector:
Wrap your UI elements with GestureDetector to detect gestures.GestureDetector(onTap: () {
// Handle tap gesture
},
onDoubleTap: () {
// Handle double tap gesture
},
// Add more gesture callbacks as needed
child: YourWidget(),
)
CODE:
Main.dart:

```
import 'package:flutter/material.dart';
import 'package:flutter_localizations/flutter_localizations.dart';
import 'package:flutter_ecommerce_ui_kit/auth/auth.dart';
import 'package:flutter_ecommerce_ui_kit/blocks/auth_block.dart';
import 'package:flutter_ecommerce_ui_kit/cart.dart';
import 'package:flutter_ecommerce_ui_kit/categorise.dart';
import 'package:flutter_ecommerce_ui_kit/home/home.dart';
import 'package:flutter_ecommerce_ui_kit/localizations.dart';
import 'package:flutter_ecommerce_ui_kit/product_detail.dart';
import  'package:flutter_ecommerce_ui_kit/settings.dart';
import 'package:flutter_ecommerce_ui_kit/shop/shop.dart';
import  'package:flutter_ecommerce_ui_kit/wishlist.dart';
import 'package:provider/provider.dart';
import 'package:firebase_core/firebase_core.dart';



void main()
 async{ WidgetsFlutterBinding.ensureInitializ
 ed(); await Firebase.initializeApp(
   // Replace with actual values
   options: FirebaseOptions(
     apiKey: "AIzaSyClVw9QJlN2fi3_7thVm7HdmA_SWTYMtw",
     appId: "1:560299755679:android:7701dcc4e602a6bc493aaa",
     messagingSenderId: "560299755679",
     projectId: "bhaveshs-826b8",
   ),
 );

 final Locale locale = Locale('en');
 runApp(MultiProvider(
   providers: [ChangeNotifierProvider<AuthBlock>.value(value: AuthBlock())],
   child: MaterialApp(
     localizationsDelegates: [
```

```
      AppLocalizations.delegate,
      GlobalMaterialLocalizations.delegate,
      GlobalWidgetsLocalizations.delegate,
      GlobalCupertinoLocalizations.delegate,
    ],
    supportedLocales: [Locale('en'), Locale('ar')],
    locale: locale,
    debugShowCheckedModeBanner: false,
    theme: ThemeData(
      primaryColor: Colors.red,
      colorScheme: ColorScheme.fromSwatch()
          .copyWith(secondary: Colors.black),
      fontFamily: locale.languageCode == 'ar' ? 'Dubai' : 'Lato',
    ),
    initialRoute: '/',
    routes: <String, WidgetBuilder>{
      '/': (BuildContext context) => Home(),
      '/auth': (BuildContext context) => Auth(),
      '/shop': (BuildContext context) => Shop(),
      '/categorise': (BuildContext context) => Categorise(),
      '/wishlist': (BuildContext context) => WishList(),
      '/cart': (BuildContext context) => CartList(),
      '/settings': (BuildContext context) => Settings(),
      '/products': (BuildContext context) => Products()
    },
  ),
));
}
```

Shop.dart:

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

import 'search.dart';

class Shop extends StatefulWidget
 {@override
 _ShopState createState() => _ShopState();
}

class _ShopState extends State<Shop> {
 final List<Map<dynamic, dynamic>> products = [
  {'name': 'Wallet', 'rating': 3.0, 'image': 'https://m.media-
amazon.com/images/I/71ZKFRCkdkL._SL1500_.jpg'},
  {'name': 'Mixer', 'rating': 3.0, 'image': 'https://m.media-
amazon.com/images/I/61isgOoxj8L._SL1500_.jpg'},
  {'name': 'IPhone 11', 'rating': 4.0, 'image':
'https://rukminim2.flixcart.com/image/416/416/k2jbyq80pkrrdj/mobile-
refurbished/x/j/s/iphone-11-128-d-mwm02hn-a-apple-0-original-
imafkg242ugz8hwc.jpeg?q=70&crop=false'},
  {'name': 'IPhone 11', 'rating': 4.0, 'image': 'https://images.unsplash.com/photo-
1519125323398-675f0ddb6308?ixlib=rb-
0.3.5&ixid=eyJhcHBfaWQiOjEyMDd9&s=94a1e718d89ca60a6337a6008341ca50&auto=format&fi
t=crop&w=1950&q=80'},
```

```
  {'name': 'IPhone 11', 'rating': 4.0, 'image': 'https://images.unsplash.com/photo-
1519125323398-675f0ddb6308?ixlib=rb-
0.3.5&ixid=eyJhcHBfaWQiOjEyMDd9&s=94a1e718d89ca60a6337a6008341ca50&auto=format&fi
t=crop&w=1950&q=80'},
  {'name': 'IPhone 11', 'rating': 4.0, 'image': 'https://images.unsplash.com/photo-
1519125323398-675f0ddb6308?ixlib=rb-
0.3.5&ixid=eyJhcHBfaWQiOjEyMDd9&s=94a1e718d89ca60a6337a6008341ca50&auto=format&fi
t=crop&w=1950&q=80'},
  {'name': 'IPhone 11', 'rating': 4.0, 'image': 'https://images.unsplash.com/photo-
1519125323398-675f0ddb6308?ixlib=rb-
0.3.5&ixid=eyJhcHBfaWQiOjEyMDd9&s=94a1e718d89ca60a6337a6008341ca50&auto=format&fi
t=crop&w=1950&q=80'},
  {'name': 'IPhone 11', 'rating': 4.0, 'image': 'https://images.unsplash.com/photo-
1519125323398-675f0ddb6308?ixlib=rb-
0.3.5&ixid=eyJhcHBfaWQiOjEyMDd9&s=94a1e718d89ca60a6337a6008341ca50&auto=format&fi
t=crop&w=1950&q=80'},
  {'name': 'IPhone 11', 'rating': 4.0, 'image': 'https://images.unsplash.com/photo-
1519125323398-675f0ddb6308?ixlib=rb-
0.3.5&ixid=eyJhcHBfaWQiOjEyMDd9&s=94a1e718d89ca60a6337a6008341ca50&auto=format&fi
t=crop&w=1950&q=80'},
  {'name': 'IPhone 12', 'rating': 5.0, 'image': 'https://images.unsplash.com/photo-
1523205771623-e0faa4d2813d?ixlib=rb-
0.3.5&ixid=eyJhcHBfaWQiOjEyMDd9&s=89719a0d55dd05e2deae4120227e6efc&auto=format&fi
t=crop&w=1953&q=80'},
 ];
 final GlobalKey<ScaffoldState> scaffoldKey = GlobalKey<ScaffoldState>();
 @override
 Widget build(BuildContext context)
  {return
  DefaultTabController( length: 2,
    child:
      Scaffold( key:
      scaffoldKey,
      appBar: AppBar(
        actions:
          <Widget>[IconBu
          tton(
            icon: Icon(Icons.search, color: Colors.white),
            onPressed: () {
              scaffoldKey.currentState!
                  .showBottomSheet((context) => ShopSearch());
            },
          )
        ],
        title: Text('Shop'),
      ),
      body: Builder(
        builder: (BuildContext context)
          {return DefaultTabController(
              length: 2,
              child: Column(
                children:
                  <Widget>[Contain
                  er(
                    constraints: BoxConstraints(maxHeight: 150.0),
                    child: Material(
                      color: Theme.of(context).colorScheme.secondary,
```

```
    child:
      TabBar( indicatorColor:
      Colors.red,tabs: [
        Tab(icon: Icon(Icons.view_list)),
        Tab(icon: Icon(Icons.grid_on)),
      ],
    ),
  ),
),
Expanded(
  child:
    TabBarView(child
    ren: [ Container(
        child: ListView(
          children: products.map((product)
            {return Builder(
              builder: (BuildContext context)
                {return InkWell(
                  onTap: () {
                    print('Card tapped.');
                  },
                  child: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: <Widget>[
                      Divider( h
                        eight: 0,
                      ),
                      Padding(
                        padding: const EdgeInsets.only(top: 10.0, bottom: 10.0),
                        child: ListTile(
                          trailing: Icon(Icons.navigate_next),
                          leading: ClipRRect(
                            borderRadius: BorderRadius.circular(5.0),
                            child: Container(
                              decoration:
                                  BoxDecoration(color:
                                  Colors.red
                              ),
                              child:
                                CachedNetworkImage(fit:
                                BoxFit.cover,
                                imageUrl: product['image'],
                                placeholder: (context, url) => Center(
                                    child: CircularProgressIndicator()
                                ),
                                errorWidget: (context, url, error) => new Icon(Icons.error),
                              ),
                            ),
                          ),
                          title:
                            Text( product['n
                            ame'],style:
                            TextStyle(
                                fontSize: 14
                            ),
                          ),
                          subtitle: Column(
```

```
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: <Widget>[
                      Row(
                        children:
                          <Widget>[Padding
                          (
                            padding: const EdgeInsets.only(top: 2.0, bottom: 1),
                            child: Text('\$200', style: TextStyle(
                              color: Theme.of(context).colorScheme.secondary,
                              fontWeight: FontWeight.w700,
                            )),
                          ),
                          Padding(
                            padding: const EdgeInsets.only(left: 6.0),
                            child: Text('(\$400)', style: TextStyle(
                                fontWeight: FontWeight.w700,
                                fontStyle: FontStyle.italic, color:
                                Colors.grey,
                                decoration: TextDecoration.lineThrough
                            )),
                          )
                        ],
                      ),
                      Row(
                        children:
                          <Widget>[RatingSt
                          ars(
                            value: product['rating'],
                            starSize: 16,
                            valueLabelColor: Colors.amber,
                            starColor: Colors.amber,
                          )
                        ],
                      )
                    ],
                  ),
                ),
              ),
            ],
          ),
        );
      },
    );
  }).toList(),
  ),
),
Container(
  child:
    GridView.count(shrin
    kWrap: true,
    crossAxisCount: 2,
    childAspectRatio: 0.7,
    padding: EdgeInsets.only(top: 8, left: 6, right: 6, bottom: 12),
    children: List.generate(products.length, (index) {
      return
        Container(child:
        Card(
          clipBehavior: Clip.antiAlias,
```

```
child:
  InkWell(onTa
  p: () {
    print('Card tapped.');
  },
  child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: <Widget>[
      SizedBox(
        height: (MediaQuery.of(context).size.width / 2 - 5),
        width: double.infinity,
        child:
          CachedNetworkImage(fit:
          BoxFit.cover,
          imageUrl: products[index]['image'],
          placeholder: (context, url) => Center(
              child: CircularProgressIndicator()
          ),
          errorWidget: (context, url, error) => new Icon(Icons.error),
        ),
      ),
      Padding(
        padding: const EdgeInsets.only(top: 5.0),
        child: ListTile(
          title: Text(
            'Two Gold Rings',
            style: TextStyle(
                fontWeight: FontWeight.bold,
                fontSize: 16
            ),
          ),
          subtitle: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: <Widget>[
              Row(
                children:
                  <Widget>[Padding
                  (
                    padding: const EdgeInsets.only(top: 2.0, bottom: 1),
                    child: Text('\$200', style: TextStyle(
                      color: Theme.of(context).colorScheme.secondary,
                      fontWeight: FontWeight.w700,
                    )),
                  ),
                  Padding(
                    padding: const EdgeInsets.only(left: 6.0),
                    child: Text('(\$400)', style: TextStyle(
                        fontWeight: FontWeight.w700,
                        fontStyle: FontStyle.italic, color:
                        Colors.grey,
                        decoration: TextDecoration.lineThrough
                    )),
                  )
                ],
              ),
              Row(
                children: <Widget>[
```

```
                                    RatingStars(
                                      value: products[index]['rating'],
                                      starSize: 16,
                                      valueLabelColor: Colors.amber,
                                      starColor: Colors.amber,
                                    )
                                  ],
                                )
                              ],
                            ),
                          ),
                        )
                      ],
                    ),
                  ),
                ),
              );
            }),
          ),
        ),
      ],
    ),
  ),
  ),
 );
}
}
```

Signin.dart:

```
import 'dart:collection';
import 'dart:html';

import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:flutter_ecommerce_ui_kit/blocks/auth_block.dart';
import 'package:flutter_ecommerce_ui_kit/models/user1.dart';
import 'package:provider/provider.dart';


class SignIn extends StatefulWidget
 {@override
 _SignInState createState() => _SignInState();
 static Future<List<dynamic>> _getEventsFromFirestore() async
  {var collection = FirebaseFirestore.instance.collection('users');
  var querySnapshot = await collection.get();
  var email;
  var password;
  for (var queryDocumentSnapshot in querySnapshot.docs)
    { Map<String, dynamic> data =
    queryDocumentSnapshot.data();email = data['email'];
     password = data['password'];
```

```
  }
  return [email, password];
 }
}

class _SignInState extends State<SignIn>
 { final _formKey =
 GlobalKey<FormState>();final
 UserCredential userCredential =
    UserCredential(usernameOrEmail: '', password: '');

 @override
 Widget build(BuildContext context)
  {return Center(
    child: Form( key:
      _formKey,
      child:
        SingleChildScrollView(child:
        Padding(
          padding: const EdgeInsets.only(left: 15.0, right: 15.0),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              Padding(
                padding: const EdgeInsets.only(bottom: 12.0),
                child: TextFormField(
                  keyboardType: TextInputType.emailAddress,
                  validator: (value) {
                    if (value!.isEmpty) {
                      return 'Please Enter Email or Username';
                    }
                    return null;
                  },
                  onSaved: (value)
                    {setState(() {
                      userCredential.usernameOrEmail = value!;
                    });
                  },
                  decoration:
                    InputDecoration( hintText: 'Enter
                    Username Or Email',labelText:
                    'Email',
                  ),
                ),
              ),
              TextFormField( validat
                or: (value) {
                  if (value!.isEmpty) {
                    return 'Please Enter Password';
                  }
                  return null;
                },
                onSaved: (value)
                  {setState(() {
                    userCredential.password = value!;
                  });
                },
```

```
            decoration:
              InputDecoration(hintText:
              'Enter Password', labelText:
              'Password',
            ),
            obscureText: true,
          ),
          Padding(
            padding: const EdgeInsets.only(top: 25.0),
            child: SizedBox(
              width: double.infinity,
              height: 50,
              child: Consumer<AuthBlock>(
                builder: (BuildContext context, AuthBlock auth,
                    Widget? child) {
                  return ElevatedButton(
                    style: ElevatedButton.styleFrom(
                      primary: Theme.of(context).primaryColor,
                    ),
                    child: auth.loading && auth.loadingType == 'login'
                        ? CircularProgressIndicator(
                            valueColor:
                                AlwaysStoppedAnimation<Color>(Colors.w
                                hite),
                          )
                        : Text(
                            'Sign In',
                            style: TextStyle(color: Colors.white),
                          ),
                    onPressed: () {
                      // Validate form
                      if (_formKey.currentState!.validate() &&
                          !auth.loading) {
                        // Update values
                        _formKey.currentState!.save();
                        // Hit Api
                        auth.login(userCredential);
                      }
                    },
                  );
                },
              ),
            ),
          )
        ],
      ),
    ),
  ),
),
);
}
}
```

OUTPUT:
Location Screen:
After clicking on ‘Shop’ button

It will navigate to this pageNavigation to Shop items screen:

CONCLUSION:

We have implemented navigation ,routing and gestures in the pages.

MAD & PWA Lab
Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	69
Name	Bhavesh Wadhwani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	11

MAD and PWA Lab 6

Name: Bhavesh Wadhwani
Class: D15A
Roll no:66

AIM: To connect flutter UI with firebase database.

THEORY:
Firebase is a great backend solution for anyone that wants to use authentication, databases, cloud functions, ads, and countless other features within an app.:

OUTPUT:
1. Create a new firebase project:

Create a project (Step 3 of 3)

Configure Google Analytics

Choose or create a Google Analytics account

BhaveshWadhwani

Automatically create a new property in this account

Upon project creation, a new Google Analytics property will be created in your chosen Google Analytics account and linked to your Firebase project. This link will enable data flow between the products. Data exported from your Google Analytics property into Firebase is subject to the Firebase terms of service, while Firebase data imported into Google Analytics is subject to the Google Analytics terms of service. Learn more.

Previous

Create project

Snapdeal

Your Firebase project is ready

Continue

2. DataBase Connection Part:

main.dart build.gradle

```
sourceSets {
    main.java.srcDirs += 'src/main/kotlin'
}

lintOptions {
    disable 'InvalidPackage'
}

defaultConfig {
    applicationId "com.bhaveshwadhwani.snapdeal"
    minSdkVersion 16
    targetSdkVersion 28
    versionCode flutterVersionCode.toInteger()
    versionName flutterVersionName
    multiDexEnabled true
}
```

Add Firebase to your Android app
Register app
Android package name: com.bhaveshwadhwani.snapdeal
2 Download and then add config file
Instructions for Android Studio below | Unity C++
Download google-services.json
Switch to the Project view in Android Studio to see your project root directory.
Move your downloaded google-services.json file into your module (app-level) root directory.
google-services.json
Project
MyApplication [My Application]
.gradle
.idea
app
libs
src
.gitignore
build.gradle.kts
google-services.json
proguard-rules.pro
gradle
Next
3 Add Firebase SDK

Download and then add config file

3 Add Firebase SDK

Instructions for Gradle | Unity C++

Are you still using the `buildscript` syntax to manage plugins? Learn how to add Firebase plugins using that syntax.

1. To make the `google-services.json` config values accessible to Firebase SDKs, you need the Google services Gradle plugin.

   Kotlin DSL (build.gradle.kts) Groovy (build.gradle)

   Add the plugin as a dependency to your **project-level** `build.gradle` file:

   Root-level (project-level) Gradle file (`<project>/build.gradle`):

```
plugins {
  // ...

  // Add the dependency for the Google services Gradle plugin
  id 'com.google.gms.google-services' version '4.4.0' apply false
}
```

2. Then, in your **module (app-level)** `build.gradle` file, add both the `google-services` plugin and any Firebase SDKs that you want to use in your app:

   Module (app-level) Gradle file (`<project>/<app-module>/build.gradle`):

Firebase
Snapdeal
What's new
Extensions NEW
Product categories
Build
Release & Monitor
Analytics
Engage
All products
Customize your nav!
You can now focus your console experience by customizing your navigation
Learn more
Got it
Spark
No-cost $0/month
Upgrade
Create database
Set name and location
2 Secure rules
After you define your data structure, you will need to write rules to secure your data.
Learn more
Start in production mode
Your data is private by default. Client read/write access will only be granted as specified by your security rules.
Start in test mode
Your data is open by default to enable quick setup. However, you must update your security rules within 30 days to enable long-term client read/write access.
rules_version = '2';
service cloud.firestore {
match /databases/{database}/documents {
match /{document=**} {
allow read, write: if false;
}
}
}
All third party reads and writes will be denied
Cancel
Create
Watch later
Share
How much will Cloud

Storage
Store and retrieve user-generated files like
Set up Cloud Storage
Secure rules for Cloud Storage
2 Set Cloud Storage location
Your location setting is where your default Cloud Storage bucket and its data will be stored.
Cloud Storage location
nam5 (us-central)
Blaze Plan customers can choose other locations for additional buckets click here to upgrade
Cancel
Done
Introducing Cloud Storage for Firebase

:

Google-services.json


```
{
 "project_info": {
   "project_number": "560299755679",
   "project_id": "bhaveshs-826b8",
   "storage_bucket": "bhaveshs-826b8.appspot.com"
 },
 "client": [
   {
     "client_info":
       { "mobilesdk_app_id":
"1:560299755679:android:7701dcc4e602a6bc493aaa",
       "android_client_info": {
         "package_name": "com.furqan.flutter_ecommerce_ui_kit"
       }
     },
     "oauth_client": [],
     "api_key": [
       {
         "current_key": "AIzaSyClVw9QJIN2fi3_7thVm7HdmA___SWTYMtw"
       }
     ],
     "services":
       { "appinvite_service":
       {
         "other_platform_oauth_client":  []
       }
     }
   }
 ],
 "configuration_version": "1"
}
```


Add dependencies related to firebase in pubsec.yaml:

```
name: olx_clone
description: "A new Flutter project."

version: 1.0.0+1

environment:
 sdk: ">=2.12.0 <3.0.0"

dependencies:
 flutter:
   sdk: flutter
 flutter_localizations:
   sdk: flutter
 firebase_core: ^2.25.4
 firebase_auth: ^4.17.4
 cloud_firestore: ^4.2.0

 intl: 0.18.1
 carousel_slider: ^4.0.0
 cached_network_image: ^3.0.0
 flutter_rating_stars: ^1.0.3+4
 provider: ^6.0.3
 flutter_secure_storage: ^5.0.2
 fluttertoast: ^8.0.7

 # The following adds the Cupertino Icons font to your application.
 # Use with the CupertinoIcons class for iOS style icons.
 cupertino_icons: ^1.0.2

dev_dependencies:
 flutter_test:
   sdk: flutter
```

Build.gradle:

```
  dependencies {
      classpath 'com.google.gms:google-services:4.4.1'
      classpath 'com.android.tools.build:gradle:7.0.0'
      classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
      classpath 'com.google.gms:google-services:4.3.15'
  }
}
```

CONCLUSION:

We have implemented navigation ,routing and gestures in the pages.

MAD & PWA Lab Journal

Experiment No.	07
Experiment Title.	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.
Roll No.	69
Name	Bhavesh Wadhwani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	14

Experiment - 7

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

Theory:

Regular Web App

A regular web app is a website that is designed to be accessible on all mobile devices such that the content gets fit as per the device screen. It is designed using a web technology stack (HTML, CSS, JavaScript, Ruby, etc.) and operates via a browser. They offer various native-device features and functionalities. However, it entirely depends on the browser the user is using. In other words, it might be possible that you can access a native-device feature on Chrome but not on Safari or Mozilla Firefox because the browsers are incompatible with that feature.

Progressive Web App

Progressive Web App (PWA) is a regular web app, but some extras enable it to deliver an excellent user experience. It is a perfect blend of desktop and mobile application experience to give both platforms to the end-users.

Difference between PWAs vs. Regular Web Apps:

A Progressive Web is different and better than a Regular Web app with features like:

1. Native Experience

Though a PWA runs on web technologies (HTML, CSS, JavaScript) like a Regular web app, it gives user experience like a native mobile application. It can use most native device features, including push notifications, without relying on the browser or any other entity. It offers a seamless and integrated user experience that it is quite tough for one to differentiate between a PWA and a Native application by considering its look and feel.

2. Ease of Access

Unlike other mobile apps, PWAs do not demand longer download time and make memory space available for installing the applications. The PWAs can be shared and installed by a link, which cuts down the number of steps to install and use. These applications can easily keep an app icon on the user's home screen, making the app easily accessible to the users and helps the brands remain in the users' minds, and improving the chances of interaction.

3. Faster Services

PWAs can cache the data and serve the user with text stylesheets, images, and other web content even before the page loads completely. This lowers the waiting time for the end-users and helps the

brands improve the user engagement and retention rate, which eventually adds value to their business.

4. Engaging Approach
As already shared, the PWAs can employ push notifications and other native device features more efficiently. Their interaction does not depend on the browser user uses. This eventually improves the chances of notifying the user regarding your services, offers, and other options related to your brand and keeping them hooked to your brand. In simpler words, PWAs let you maintain the user engagement and retention rate.

5. Updated Real-Time Data Access
Another plus point of PWAs is that these apps get updated on their own. They do not demand the end-users to go to the App Store or other such platforms to download the update and wait until installed.In this app type, the web app developers can push the live update from the server, which reaches the apps residing on the user's devices automatically. Therefore, it is easier for the mobile app developer to provide the best of the updated functionalities and services to the end-users without forcing them to update their app.

The main features are:
Progressive — They work for every user, regardless of the browser chosen because they are built at the base with progressive improvement principles.

Responsive — They adapt to the various screen sizes: desktop, mobile, tablet, or dimensions that can later become available.

App-like — They behave with the user as if they were native apps, in terms of interaction and navigation.

Updated — Information is always up-to-date thanks to the data update process offered by service workers.

Secure — Exposed over HTTPS protocol to prevent the connection from displaying information or altering the contents.

Searchable — They are identified as “applications” and are indexed by search engines.

Reactivable — Make it easy to reactivate the application thanks to capabilities such as web notifications.

Code:
```
import { useEffect, useState } from 'react';
import './App.css';
import { v4 as uuidv4 } from 'uuid';
import Header from './components/Header';
import { ProjectList } from './components/ProjectList';
```

```
import { LogsTable } from './components/LogsTable';

function App() {
const [ projects, setProjects] =
useState(JSON.parse(localStorage.getItem('projects')) || []);
const [ isSidebarVisible, setIsSidebarVisible] = useState(false);
projects.map(project=> console.log(project.state))

useEffect(() => {
localStorage.setItem('projects', JSON.stringify(projects));
},[projects]);

const handleSidebarClose = () => setIsSidebarVisible(false);
const handleSidebarShow = () => setIsSidebarVisible(true);

const handleAddProject = (project) => {
  const projectToBeAdded = {...project, id: uuidv4(), elapsed:0,
runningSince:null,logs:[{Starttime:0,Endtime:0,Description:""}]};
setProjects([...projects, projectToBeAdded]);
setIsSidebarVisible(false);
}

const handleUpdateProject = (projectId, updatedProject) =>
  {const updatedProjects = projects.map(project =>
  { if(project.id === projectId) {
      return{
        ...updatedProject,
        elapsed: project.elapsed,
        runningSince: project.runningSince,
        id: projectId
      }
    }
    return project;
  });
  setProjects(updatedProjects);
}

const handleDeleteProject = (projectId) => {
  const updatedProjects = projects.filter(project => project.id !== projectId);
  setProjects(updatedProjects);
```

```
}

const handleStartTimer = (projectId) => {
  const updatedProjects = projects.map(project => project.id === projectId ?
{...project, runningSince: Date.now()}: project);
  setProjects(updatedProjects);
}
const handleStopTimer = (projectId) => {
  const updatedProjects = projects.map(project =>
    {if(project.id !== projectId) return project;

    return {
      ...project,
      elapsed:project.elapsed + (Date.now() - project.runningSince),
      runningSince:null
    }
  });
  setProjects(updatedProjects);
}
  return (
    <>
      <Header
        onAddProject = { handleAddProject }
        onSidebarShow = { handleSidebarShow}
        onSidebarClose = { handleSidebarClose}
        isSidebarVisible = { isSidebarVisible }
      />
      <ProjectList
        onDeleteProject = { handleDeleteProject }
        projects={ projects }
        onSidebarShow= { handleSidebarShow }
        onStartTimer ={ handleStartTimer }
        onStopTimer = { handleStopTimer }
        onUpdateProject={ handleUpdateProject }
       />
       <LogsTable
        projects={projects}
       />
    </>
  );
```

```
}

export default App;
```

Open folder in VS code and click go live at bottom right corner

Open your hosted site on Microsoft Edge

Click on 3 dots
click on Apps
select install app option
Click on Install

App icon will appear at the bottom

Output:

Desktop App Created Successfully.

Open Desktop App:

\

Conclusion:

In this experiment, we have successfully created a basic progressive web app of our webpage and installed it in our desktop successfully.

\

MAD & PWA Lab Journal

Experiment No.	08
Experiment Title.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA
Roll No.	69
Name	Bhavesh Wadhwani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	14

\

Experiment No:08

Aim: To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA.

Theory:

Service Worker

Service Worker is a script that works on browser background without user interaction independently. Also, It resembles a proxy that works on the user side. With this script, you can track network traffic of the page, manage push notifications and develop “offline first” web applications with Cache API.

Things to note about Service Worker:

- A service worker is a programmable network proxy that lets you control how network requests from your page are handled.
- Service workers only run over HTTPS. Because service workers can intercept network requests and modify responses, "man-in-the-middle" attacks could be very bad.
- The service worker becomes idle when not in use and restarts when it's next needed. You cannot rely on a global state persisting between events. If there is information that you need to persist and reuse across restarts, you can use IndexedDB databases.

What can we do with Service Workers?

- You can dominate **Network Traffic**
  You can manage all network traffic of the page and do any manipulations. For example, when the page requests a CSS file, you can send plain text as a response or when the page requests an HTML file, you can send a png file as a response. You can also send a true response too.

- You can **Cache**
  You can cache any request/response pair with Service Worker and Cache API and you can access these offline content anytime.

- You can manage **Push Notifications**
  You can manage push notifications with Service Worker and show any information message to the user.
- You can **Continue**

\

Although Internet connection is broken, you can start any process with Background Sync of Service Worker.

What can't we do with Service Workers?

- You can't access the **Window**
  You can't access the window, therefore, You can't manipulate DOM elements. But, you can communicate to the window through post Message and manage processes that you want.

- You can't work it on **80 Port**
  Service Worker just can work on HTTPS protocol. But you can work on localhost during development.

A service worker goes through three steps in its life cycle:

- Registration
- Installation
- Activation

Registration
To install a service worker, you need to register it in your main JavaScript code. Registration tells the browser where your service worker is located, and to start installing it in the background. Let's look at an example:

main.js

```
if ('serviceWorker' in navigator)
  { navigator.serviceWorker.register('/s
  ervice-worker.js')
  .then(function(registration) {
    console.log('Registration successful, scope is:', registration.scope);
  })
  .catch(function(error) {
    console.log('Service worker registration failed, error:', error);
  });
}
```

\

This code starts by checking for browser support by examining **navigator.serviceWorker**. The service worker is then registered with navigator.serviceWorker.register, which returns a promise that resolves when the service worker has been successfully registered. The scope of the service worker is then logged with registration.scope. If the service worker is already installed, navigator.serviceWorker.register returns the registration object of the currently active service worker.

The scope of the service worker determines which files the service worker controls, in other words, from which path the service worker will intercept requests. The default scope is the location of the service worker file, and extends to all directories below. So if service-worker.js is located in the root directory, the service worker will control requests from all files at this domain.

You can also set an arbitrary scope by passing in an additional parameter when registering. For example:

main.js

```
navigator.serviceWorker.register('/service-worker.js', {scope:
  '/app/'
});
```

In this case we are setting the scope of the service worker to /app/, which means the service worker will control requests from pages like /app/, /app/lower/ and /app/lower/lower, but not from pages like /app or /, which are higher.

If you want the service worker to control higher pages e.g. /app (without the trailing slash) you can indeed change the scope option, but you'll also need to set the Service-Worker-Allowed HTTP Header in your server config for the request serving the service worker script.

\

Code

Index.html

manifest.json M | app.js U | service-worker.js U | index.html M

index.html > html > body.auth-body.dark > script

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="fuzzy" />
    <meta name="keywords" content="fuzzy" />
    <meta name="author" content="fuzzy" />
    <link rel="manifest" href="manifest.json" />
    <link rel="icon" href="assets/images/logo/favicon.png" type="image/x-icon" />
    <title>fuzzy</title>
    <link rel="apple-touch-icon" href="assets/images/logo/favicon.png" />
    <meta name="theme-color" content="#122636" />
    <meta name="apple-mobile-web-app-capable" content="yes" />
    <meta name="apple-mobile-web-app-status-bar-style" content="black" />
    <meta name="apple-mobile-web-app-title" content="fuzzy" />
    <meta name="msapplication-TileImage" content="assets/images/logo/favicon.png" />
    <meta name="msapplication-TileColor" content="#FFFFFF" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
```

```html
</head>

<body class="auth-body dark">
  <!-- onboarding section start -->
  <section class="section-b-space">
    <div class="swiper intro slider-1">
      <div class="swiper-wrapper">
        <div class="swiper-slide">
          <div class="theme-logo pb-3">
            <img class="img-fluid logo-img" src="assets/images/logo/logo.png" alt="logo" />
          </div>
          <div class="onboarding-design">
            <img class="img-fluid design-img" src="assets/images/onboarding/design1.png" alt="b

            <img class="img-fluid slider-img1" src="assets/images/onboarding/1.png" alt="slider

            <img class="img-fluid vector1" src="assets/images/onboarding/vector1.png" alt="v1"
            <img class="img-fluid vector2" src="assets/images/onboarding/vector2.png" alt="v2"
            <img class="img-fluid vector3" src="assets/images/onboarding/vector3.png" alt="v3"
          </div>
          <div class="product-details">
            <h1>Office Furniture</h1>
            <span></span>
```

\

```html
<!-- pwa install app popup start -->
<div class="offcanvas offcanvas-bottom addtohome-popup theme-offcanvas" tabindex="-1" id="offc
  <button type="button" class="btn-close text-reset" data-bs-dismiss="offcanvas" aria-label="(
  <div class="offcanvas-body small">
    <div class="app-info">
      <img src="assets/images/logo/48.png" class="img-fluid" alt="" />
      <div class="content">
        <h4>fuzzy app</h4>
        <a href="#">www.fuzzy-app.com</a>
      </div>
    </div>
    <a href="#!" class="btn theme-btn install-app btn-inline home-screen-btn m-0" id="installa
  </div>
</div>
<!-- pwa install app popup start -->

<!-- swiper js -->
<script src="assets/js/swiper-bundle.min.js"></script>
<script src="assets/js/custom-swiper.js"></script>

<!-- iconsax js -->
<script src="assets/js/iconsax.js"></script>
```

Style.css

```css
@-webkit-keyframes fireworkLine {
  0% {
    right: 20%;
    -webkit-transform: scale(0, 0);
            transform: scale(0, 0);
  }
  25% {
    right: 20%;
    width: 6px;
    -webkit-transform: scale(1, 1);
            transform: scale(1, 1);
  }
  35% {
    right: 0;
    width: 35%;
  }
  70% {
    right: 0;
    width: 4px;
    -webkit-transform: scale(1, 1);
            transform: scale(1, 1);
  }
  100% {
    right: 0;
    -webkit-transform: scale(0, 0);
            transform: scale(0, 0);
```

```css
@keyframes fireworkLine {
  0% {
    right: 20%;
    -webkit-transform: scale(0, 0);
            transform: scale(0, 0);
  }
  25% {
    right: 20%;
    width: 6px;
    -webkit-transform: scale(1, 1);
            transform: scale(1, 1);
  }
  35% {
    right: 0;
    width: 35%;
  }
  70% {
    right: 0;
    width: 4px;
    -webkit-transform: scale(1, 1);
            transform: scale(1, 1);
  }
  100% {
    right: 0;
    -webkit-transform: scale(0, 0);
```

\

App.js

manifest.json M | app.js U | service-worker.js U | index.html M | style.css 5

app.js > ...

```js
if ('serviceWorker' in navigator) {
    window.addEventListener('load', () => {
    navigator.serviceWorker.register('/service-worker.js')
    .then(registration => {
    console.log('Service Worker registered with scope:', registration.scope);
    })
    .catch(error => {
    console.error('Service Worker registration failed:', error);
    });
    });
    }
```

Service-Worker.js

manifest.json M | app.js U | service-worker.js U | index.html M

service-worker.js > assetsToCache

```js
const cacheName = 'fuzzy_app';
const assetsToCache = [
'/',
'/index.html',
'/assets/css/style.css',
'/app.js'
]
self.addEventListener('install', event => {
event.waitUntil(
caches.open(cacheName)
.then(cache => {
return cache.addAll(assetsToCache);
})
);
});
self.addEventListener('activate', event => {
event.waitUntil(
caches.keys().then(cacheNames => {
return Promise.all(
cacheNames.filter(name => {
return name !== cacheName;
}).map(name => {
return caches.delete(name);
})
);
})
```

\

OUTPUT:

Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
IndexedDB
Web SQL
Cookies
Private state tokens
Service workers
Offline
Update on reload
Bypass for network
http://127.0.0.1:5500/
Network requests
Update
Unregister
Source
service-worker.js
Received 2/4/2024, 11:22:03 am
Status
#1879 activated and is running
stop
Clients
http://127.0.0.1:5500/index.html
focus

\

Cache Storage

Conclusion: In this experiment, we have registered a service worker, and completed the install and activationprocess for a new service worker for the E-commerce PW

\

MAD & PWA Lab Journal

Experiment No.	09
Experiment Title.	To implement Service worker events like fetch, sync and push for E-commerce PWA
Roll No.	69
Name	Bhavesh Wadhwani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	14

\

Experiment No:09

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

Theory:

Service Worker

Service Worker is a script that works on browser background without user interaction independently. Also, It resembles a proxy that works on the user side. With this script, you can track network traffic of the page, manage push notifications and develop “offline first” web applications with Cache API.

Things to note about Service Worker:

- A service worker is a programmable network proxy that lets you control how network requests from your page are handled.
- Service workers only run over HTTPS. Because service workers can intercept network requests and modify responses, "man-in-the-middle" attacks could be very bad.
- The service worker becomes idle when not in use and restarts when it's next needed. You cannot rely on a global state persisting between events. If there is information that you need to persist and reuse across restarts, you can use IndexedDB databases.
- Service workers make extensive use of promises, so if you're new to promises, then you should stop reading this and check out Promises, an introduction.

Fetch Event

You can track and manage page network traffic with this event. You can check existing cache, manage “cache first” and “network first” requests and return a response that you want.

Of course, you can use many different methods but you can find in the following example a “cache first” and “network first” approach. In this example, if the request’s and current location’s origin are thesame (Static content is requested.), this is called “cacheFirst” but if you request a targeted external URL, this is called “networkFirst”.

- **CacheFirst** - In this function, if the received request has cached before, the cached response is returned to the page. But if not, a new response requested from the network.
- **NetworkFirst** - In this function, firstly we can try getting an updated response from the network, if this process completed successfully, the new response will be cached and returned. But if this process fails, we check whether the request has been cached before or not. If a cache exists, it is returned to the page, but if not, this is up to you. You can return dummy content or information messages to the page.

\

Code:

Service-Worker.js

```
self.addEventListener("install",  function  (event)
   { event.waitUntil(preLoad());
   });
   self.addEventListener("fetch", function (event)
   { event.respondWith( checkResponse(event.requ
   est).catch(function () { console.log("Fetch from
   cache successful!"); return
   returnFromCache(event.request);
   })
   );
   console.log("Fetch successful!");
   event.waitUntil(addToCache(event.requesst));
   });
   self.addEventListener("sync", (event) =>
   { if (event.tag === "syncMessage")
   { console.log("Sync successful!");
   }
   });
   self.addEventListener("push",  function  (event)
   { if (event && event.data)
   {try {
   var data = event.data.json();
   if (data && data.method === "pushMessage")
   { console.log("Push notification sent");
   self.registration.showNotification("Ecommerce  website",
   { body: data.message,
   });
   }
   } catch (error) {
   console.error("Error parsing push data:", error);
   }
   }
   });
   var preLoad = function () {
   return caches.open("offline").then(function (cache) {
```

\

```
// caching index and important routes
return cache.addAll([
"/",
"/index.html",
"/landing.html",
"/profile.html",
"/shop.html",
"/cart.html",
"/css/main.css",
]);
});
};
var checkResponse = function (request)
{ return new Promise(function (fulfill, reject)
{ fetch(request)
.then(function (response)
{if (response.status !==
404)
{ fulfill(response);
} else {
reject(new Error("Response not found"));
}
})
.catch(function (error)
{ reject(error);
});
});
};
var returnFromCache = function (request) {
return caches.open("offline").then(function (cache)
{ return cache.match(request).then(function (matching)
{ if (!matching || matching.status == 404)
{return cache.match("offline.html");
} else {
return matching;
}
});
});
```

\

```
};
var addToCache = function (request) {
return  caches.open("offline").then(function  (cache)
{ return fetch(request).then(function (response) {
return cache.put(request, response.clone()).then(function ()
{ return response;
});
});
});
};
```

Output:

Conclusion : In this experiment, we have successfully implemented service worker events like fetch, sync and push for E-commerce PWA and found out output for above implementation.

\

MAD & PWA Lab Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	69
Name	Bhavesh Wadhwani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	14

\

Experiment No. 10

Aim:

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

Theory:

GitHub Pages

Public web pages are freely hosted and easily published. Public webpages hosted directlyfrom your GitHub repository. Just edit, push, and your changes are live.

GitHub Pages provides the following key features:

1. Blogging with Jekyll
2. Custom URL
3. Automatic Page Generator

Reasons for favoring this over Firebase:

1. Free to use
2. Right out of github
3. Quick to set up

GitHub Pages is used by Lyft, CircleCI, and HubSpot.
GitHub Pages is listed in 775 company stacks and 4401 developer stacks.

Pros

1. Very familiar interface if you are already using GitHub for your projects.
2. Easy to set up. Just push your static website to the gh-pages branch and your website is ready.
3. Supports Jekyll out of the box.
4. Supports custom domains. Just add a file called CNAME to the root of your site, add an A record in the site's DNS configuration, and you are done.

Cons

1. The code of your website will be public, unless you pay for a private repository.

\

2. Currently, there is no support for HTTPS for custom domains. It's probably coming soon though.
3. Although Jekyll is supported, plug-in support is rather spotty.

Firebase

The Realtime App Platform. Firebase is a cloud service designed to power real-time, collaborative applications. Simply add the Firebase library to your application to gain access to a shared data structure; any changes you make to that data are automatically synchronized with the Firebase cloud and with other clients within milliseconds.

Some of the features offered by Firebase are:

1. Add the Firebase library to your app and get access to a shared data structure. Any changes made to that data are automatically synchronized with the Firebase cloud and with other clients within milliseconds.
2. Firebase apps can be written entirely with client-side code, update in real-time out-of-the-box, interoperate well with existing services, scale automatically, and provide strong data security.
3. Data Accessibility- Data is stored as JSON in Firebase. Every piece of data has its own URL which can be used in Firebase's client libraries and as a REST endpoint. These URLs can also be entered into a browser to view the data and watch it update in real-time.

Reasons for favoring over GitHub Pages:

1. Realtime backend made easy
2. Fast and responsive

Instacart, 9GAG, and Twitch are some of the popular companies that use Firebase Firebase has a broader approval, being mentioned in 1215 company stacks & 4651developers stacks

Pros

1. Hosted by Google. Enough said.
2. Authentication, Cloud Messaging, and a whole lot of other handy services will be available to you.
3. A real-time database will be available to you, which can store 1 GB of data.
4. You'll also have access to a blob store, which can store another 1 GB of data.
5. Support for HTTPS. A free certificate will be provisioned for your custom domain within 24 hours.

Cons

1. Only 10 GB of data transfer is allowed per month. But this is not really a big problem, if you use a CDN or AMP.

\

2. Command-line interface only.
3. No in-built support for any static site generator.

Link to our GitHub repository: https://github.com/Jatintalreja0510/MPL-PWA-APP

Github Screenshot:

jatintalreja0510.github.io/MPL-PWA-APP/
E-commerce Store
Product 1
Description of product 1 goes here.
$19.99
Add to Cart
Product 2
Description of product 2 goes here.
$29.99
Add to Cart

MAD & PWA Lab Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	69
Name	Bhavesh Wadhwani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	14

Experiment 11

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

Theory :
Reference : https://www.semrush.com/blog/google-lighthouse/

Google Lighthouse :
Google Lighthouse is a tool that lets you audit your web application based on a number of parameters including (but not limited to) performance, based on a number of metrics, mobile compatibility, Progressive Web App (PWA) implementations, etc. All you have to do is run it on a page or pass it a URL, sit back for a couple of minutes and get a very elaborate report, not much short of one that a professional auditor would have compiled in about a week.
The best part is that you have to set up almost nothing to get started. Let's begin by looking at some of the top features and audit criteria used by Lighthouse.
Key Features and Audit Metrics

Google Lighthouse has the option of running the Audit for Desktop as well as mobile version of your page(s). The top metrics that will be measured in the Audit are:

Performance: This score is an aggregation of how the page fared in aspects such as (but not limited to) loading speed, time taken for loading for basic frame(s), displaying meaningful content to the user, etc. To a layman, this score is indicative of how decently the site performs, with a score of 100 meaning that you figure in the 98th percentile, 50 meaning that you figure in the 75th percentile and so on.
PWA Score (Mobile): Thanks to the rise of Service Workers, app manifests, etc., a lot of modern web applications are moving towards the PWA paradigm, where the objective is to make the application behave as close as possible to native mobile applications. Scoring points are based on the Baseline PWA checklist laid down by Google which includes Service Worker implementation(s), viewport handling, offline functionality, performance in script-disabled environments, etc.
Accessibility: As you might have guessed, this metric is a measure of how accessible your website is, across a plethora of accessibility features that can be implemented in your page (such as the 'aria-' attributes like aria-required, audio captions, button names, etc.). Unlike the other metrics though, Accessibility metrics score on a pass/fail basis i.e. if all possible elements of the page are not screen-reader friendly (HTML5 introduced features that would make pages easy to interpret for screen readers used by visually
challenged people like tag names, tags such as <section>, <article>, etc.), you get a 0 on that score. The aggregate of these scores is your Accessibility metric score.
Best Practices: As any developer would know, there are a number of practices that have been deemed 'best' based on empirical data. This metric is an aggregation of many such points, including but not limited to:Use of HTTPS
Avoiding the use of deprecated code elements like tags, directives, libraries, etc. Password input with paste-into disabled
Geo-Location and cookie usage alerts on load, etc.

Changes made to the code :

For theme color add a meta tag in index.html-

<meta name="theme-color" content="#4285f4">

For a maskable icon add "purpose": "any maskable" to the icons in manifest.json file For apple touch icon add the following meta tag in index.html-
<link rel="apple-touch-icon" href="">

Changes in manifest.json
{
"name": "flower shop website",
"short_na me": "flowers", "start_url": "index.htm l", "scope":
"./",
"theme_ color": "#ffd31d ",
"backgr ound_co lor": "#333",
"display ":
"standal one", "icons": [
{
"src": "icon-1.png", "size
s":
"192 x192 ",
"typ e":
"ima ge/p ng"

“purpose”:"any maskable"

},
{
"src": "icon-2.png", "size
s":
"512 x512 ",
"typ e":
"ima ge/p ng"
“purpose”:"any maskable"

}
]
}

Conclusion: Thus we successfully used google Lighthouse PWA Analysis Tool for testing the PWA functioning

MAD & PWA Lab Journal

Experiment No.	Assignment-1
Assignment 1 Questions	1. Flutter Overview:Explain the key features and advantages of using Flutter for mobile app development. Discuss how the Flutter framework differs from traditional approaches and why it has gained popularity in the developer community. 2. Widget Tree and Composition: Describe the concept of the widget tree in Flutter. Explain how widget composition is used to build complex user interfaces. Provide examples of commonly used widgets and their roles in creating a widget tree. 3. State Management in Flutter: Discuss the importance of state management in Flutter applications. Compare and contrast the different state management approaches available in Flutter, such as setState, Provider, and Riverpod. Provide scenarios where each approach is suitable. 4. Firebase Integration in Flutter: Explain the process of integrating Firebase with a Flutter application. Discuss the benefits of using Firebase as a backend solution. Highlight the Firebase services commonly used in Flutter development and provide a brief overview of how data synchronization is achieved.
Roll No.	69
Name	Bhavesh Wadhwani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	4

BHAVESH WADHWANI
D15A - 69

Date
Page

Assignment - 01
MAD and PWA LAB

Q1 Flutter Overview Explain the key features and advantages of using Flutter for mobile app development. Discuss how the flutter framework differ from traditional approaches and why it has gained popularity in the developer community.

Ans * Following are the key features of flutter :
1) Single codebase : Flutter enables developers to write code once and deploy it on both Android and IOS platforms reducing development time and effort.
2) Rich Widget library: Flutter provides a comprehensive set of customizable widgets that help in building expressive and interactive user interface.
3) Native performance :- Flutter compiles to native code ensuring high performance and smooth user experience.
4) Dart programming language : Flutter uses Dart as its programming language, which is designed for building scalable and high-performance applications.

* Advantages of using flutter:-
1) Faster developments with a single code base and instant reload, reducing development time and allowing for faster product releases.

Teacher's Sign.: _____

2) Cost - Effective: Developing with flutter can be cost effective since it elementes the need for separate develop-ment teams can handle both platforms.

3) Consistent UI across platforms:- Flutters widget library and design philosophy enables developers to create consistent UI's across different platforms, maintaining a unified brand identity.

* Differs from traditional approaches:

1) Native Development: unlike traditional approaches where separate code bases are maintained for Android and ios, flutter allows developers to use single codebase for both platforms

2) Web and desktop support: Flutter extends beyond mobile platforms, enabling developers to build applications for the web and desktop providing a more veratile solution.

3) Dart programming language: the use of Dart, a language designed especifically for flutter, destinguishes it from traditional frameworks that often use different languages for different platforms.

* Popularity in developer community:

1) Versatility: Flutter's ability to target multiple platforms from a single codebase appeals to developers looking for a versatile solution

Teacher's Sign.: _____

that extends beyond mobile app development.

2) Growing Ecosystem: The active and growing Flutter community, along with a rich ecosystem of packages and plugins has contributed to the framework popularity.

3) Ease of learning: Flutter's straight forward and expressive syntax, combined with comprehensive documentation makes it actively easy for developers to learn and adopt.

Q2 Widget tree and composition: Describe the concept of widget tree in flutter. Explain how widget compler user interfaces. Provideseramples of commonly used widgets and their roles in creating a widget tree.

Ans 1) In flutter, the widget tree is a hierarchical structure that represents the user interface of an application.

2) Each element in the tree is a widget, which is a light weight, immutable describes all of a part of the user interface.

3) Flutter follows a declarative approach, where the UI is expressed as a tree of widgets, and changes to the UI are made by creating a new widget tree.

Date
Page

* Widget Composition:

1) Container widget: The container widget is a basic building block that can contain other widgets. It is often used to define the size, padding, and decoration of a widget.

2) Row and Column: "Row" and "Column" widgets are used to arrange child widgets horizontally and vertically, respectively.

3) Stack widget: The 'Stack' widget allows widgets to be overlad on top of each other. It's useful for creating complex layouts.

4) Listview and Gridview widgets:- 'Listview' and 'Gridview' are used to create scroll-able lists and grids of widgets.

5) Textfield and Button widgets: "Textfield" is used for user input and various button widgets are used for user interaction.

6) Material Design Widgets: Flutter provides a wide range of material design widgets such as 'Card', 'Drawer' and 'Bottom Navigation Bar', to implement common degin patterns.

By combining these basic widgets, developers can create feature-rich user interfaces. The widget-tree allows for easy organization, reusability and maintainable of ui compone-nt making it a fundamental concept

Teacher's Sign.: _____

in flutter development.

Q3 State Management in flutter: Discuss the importance of state management in flutter application. Compare and contrast the different state management approches in flutter, such as set state, Provider, and Reverpod. Provide scenarios where each approch is suitable.

Ans 1) State management is crucial in flutter applications to handle and update the state of user interface.

2) In flutter, UI components can be ether stateful or statefless

3) Stateful widgets have mutable state, that can change during the eyetime of the widget, such as user interactions or data fetching.

4) Proper state management ensures that changes in the state are rejected in the UI, providing a responsive and dynamic user experience.

* State Management approach in flutter:

1) SetState:

Description: The 'setstate' method is a basic and built in mechanism for managing state in flutter. It belongs to 'statefulwidget' class,

and when called, it triggers a rebuild of the widget subtree.

Suitability: Suitable for small to medium-sized applications with a limited number of stateful widgets.

2) Provider:

Description :- Provider is a third party package used for state management in flutter. It follows the provides pattern and allows widgets to listen to changes in a provided object typically a data model or a service.

Suitability:- Suitable for medium to large-sized applications where a centralized state management solution is needed.

3) Riverpod:

Description: Riverpod is an extension of Provider and focuses on making the provider pattern more advanced and flexible approach

Suitability: Suitable for larger applications with complex state management requirement.

Teacher's Sign.: _____

Date
Page

Q4 Firebase Integration in flutter: Explain the process of Integrating firebase with a flutter application. Discuss the benefits of using firebase as a backend solution. Highlight the fire base service commonly used in flutter development and provide a brej overview of how data synchronization is achieved.

Ans Integrating firebase with flutter:

1) Create a firebase project
2) Add firebase to flutter project:- In the flutter project add the firebase: dependencies in the 'pubspec.yam'. file. Run 'flutter pub get' to install the dependencies.
3) Initialize firebase: Initialize firebase in your app by calling 'Firebase initialize. App()' in your app
4) Authentication for this the 'firebase_auth' package is used.

5) Fire base Database:. For a firestore Database, 'cloud-firestore' is used.

* Data Synchronization in firebase:

1) firebase achieves data synchronization through its real-time database (Firestore.).
2) when data changes in the data base the changes are immediately pushed to all connected clients.

Teacher's Sign.:

3) Clients can listen to specific documents on collections, and any changes trigger real-time UI.

4) Firebase's realtime data Synchronization simplifies the implemen tation of dynamic and responsive applications, making it a popular choice for flutter developers.

Teacher's Sign.: _____

MAD & PWA Lab Journal

Experiment No.	Assignment-2
Assignment 2 Questions	1. Define Progressive Web App (PWA) and explain its significance in modern web development.Discuss the key characteristics that differentiate PWAs from traditional mobile apps 2. Define responsive web design and explain its importance in the context of Progressive Web Apps.Compare and contrast responsive, fluid, and adaptive web design approaches. 3. Describe the lifecycle of Service Workers, including registration, installation, and activation phases. 4. Explain the use of IndexedDB in the Service Worker for data storage.
Roll No.	69
Name	Bhavesh Wadhwani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO4:Understand various PWA frameworks and their requirements LO5: Design and Develop a responsive User Interface by applying PWA Design techniques LO6:Develop and Analyze PWA Features and deploy it over app hosting solutions
Grade:	4

Assignment 2

1) Progressive Web App (PWA)

→ • A progressive web app (PWA) is a set of mobile web application development techniques that entails building apps that feel and look like native ones.

• Using a web stack, PWA combine a rich functionality and smooth user experience associated with native apps.

• Significance:

- Cross platform compatibility (can run on any device)
- Improved user experience (similar to native apps, with fast loading times, smooth animations and offline functionality)
- Offline functionality (allows users to access when offline)
- Push notifications (It can send push notification to users, keeping them engaged and informed even when the app is not actively in use, similar to native apps)
- Installation free (PWAs don't need to be downloaded and installed. These can be simply added to the home screen)

• Key characteristics that differentiate PWA:

- Cross-Platform compatibility
- No installation required
- Offline functionality
- Automatic updates
- Low device storage usage

• PWAs offer a modern approach to web development by combining the best features of web and mobile applications, providing a cross-platform, user-friendly, and efficient solution for delivering engaging experiences to users.

2) Responsive web design

→ • Responsive web design refers to a web design approach that enables your website content to "respond" to and adapt to

Sundaram
FOR EDUCATIONAL USE

Q2) Widget Tree and composition

Widget Tree is a hierarchial structure of UI elements represented by widgets, widgets are the building blocks of flutter application and the widget tree organizes them in a parent child relationship widget column involves combining simple widgets to create more complex UIs

Commonly widgets used include

A] Container
B] Stack
C] List View
D] Text
E] Image

Q3) State Management in Flutter

State management in flutter is crucial to handle changes in the application data and UI. Proper state management ensures that the UI stays in sync with the underlying data and contributes to a scalable and maintainable codebase

Some of the state management modules include

FOR EDUCATIONAL USE

SetState : Used for small to medium size applications where state is localized and doesn't need to be shared between multiple widgets

Provider : Well suited for medium to large-size applications where you need to share state across different parts of the app

Riverpod : Suitable for projects where you want to take advantage for advanced features.

(04) Firebase integration in Flutter

Integration Process

Step 1 : Create a Firebase Project in Firebase website

Step 2 : Add Firebase dependencies and Flutter fire packages to your local machine

Step 3 : Initialize Firebase in your Flutter app based on ~~setup~~ by calling Firebase. Firebase. initialize App in the main function

Step 4 : Use Firebase services based on your requirement like Firestore Database, Authentication, Storage a

FOR EDUCATIONAL USE

Benefits of Firebase

A] Real Time No-SQL Database
B] Authentication
C] Cloud Storage
D] Hosting