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.

Department of Information Technology

CERTIFICATE

This is to certify that **Harsh Gawali** 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

Project Title: Flutter-Signal Clone / PWA- React Ecommerce **Roll No. 19**

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.

Project Title: Flutter-Signal Clone / PWA- React Ecommerce **Roll No. 19**

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	15
2.	To design Flutter UI by including common widgets.	LO2	23/1	30/1	15
3.	To include icons, images, fonts in Flutter app	LO2	30/1	6/2	15
4.	To create an interactive Form using form widget	LO2	6/2	13/2	15
5.	To apply navigation, routing and gestures in Flutter App	LO2	13/2	20/2	15
6.	To Connect Flutter UI with fireBase database	LO3	20/2	5/3	15
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	15
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	15
9.	To implement Service worker events like fetch, sync and push for E-commerce PWA	LO5	19/3	26/3	15
10.	To study and implement deployment of Ecommerce PWA to GitHub Pages.	LO5	26/3	5/3	15
11.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.	LO6	5/3	2/2	15
12.	Assignment-1	LO1,LO2 ,LO3	2/2	5/2	5
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.	19
Name	Harsh Gawali
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	

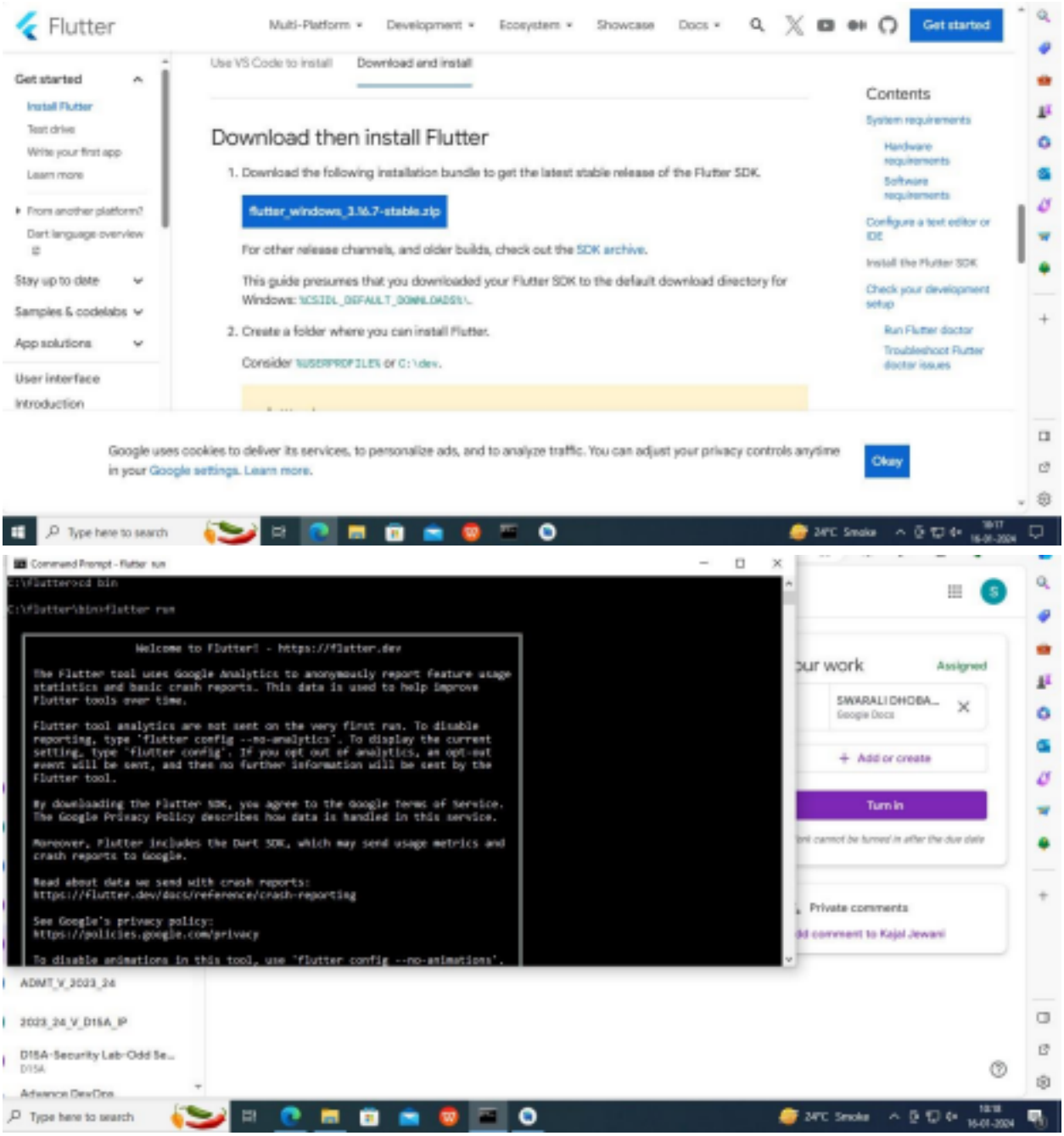
Project Title: Flutter-Signal Clone / PWA- React Ecommerce **Roll No. 19**

MPL Lab: Experiment No.1

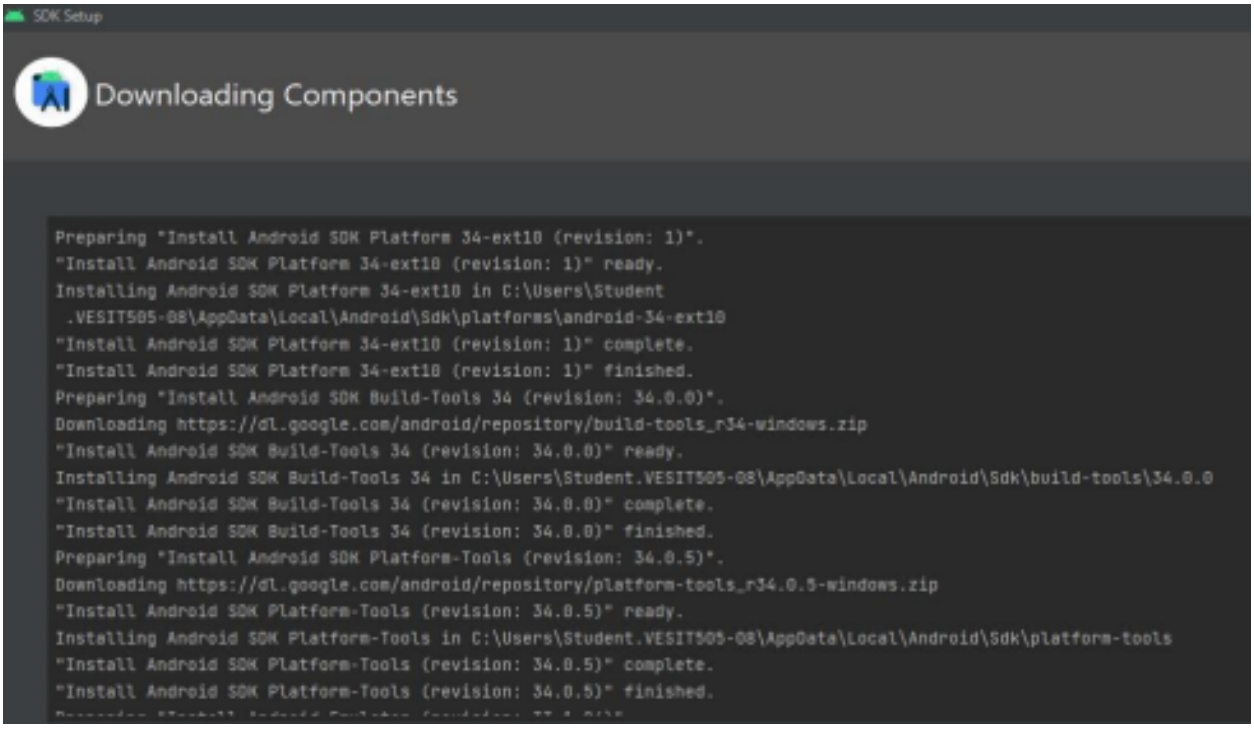
Name: Harsh Gawali_D15A_19

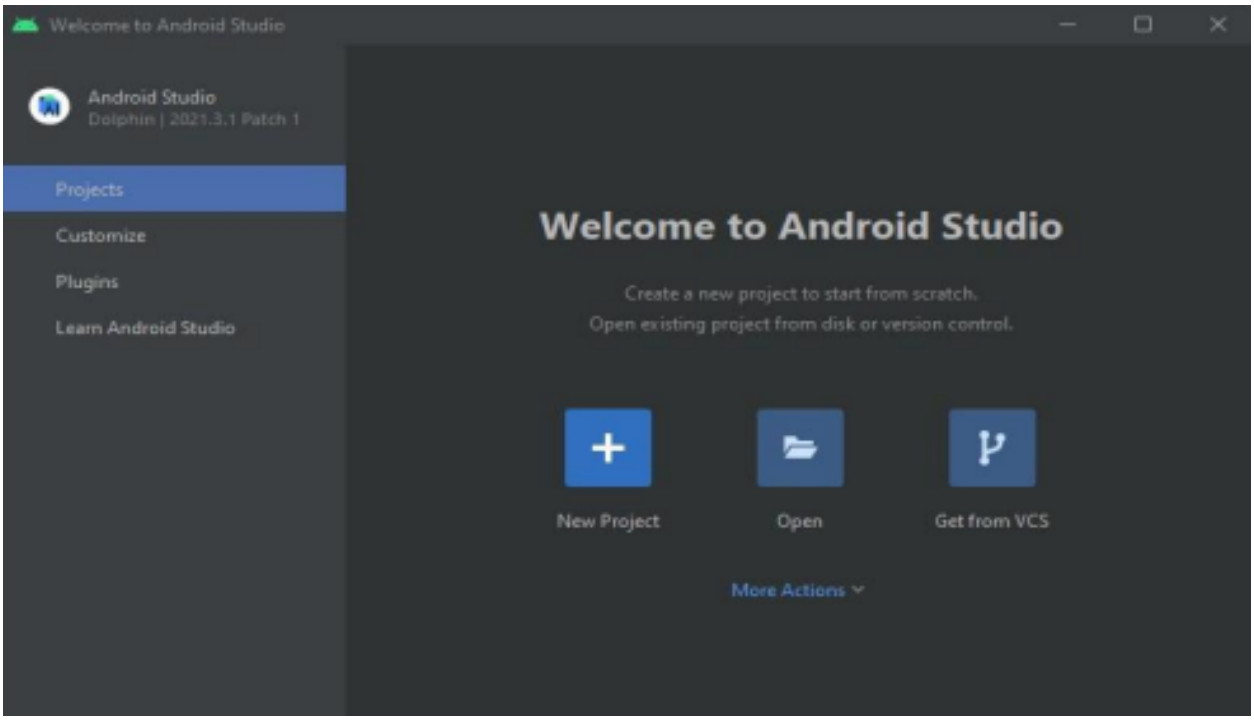
Aim: Installation and Configuration of Flutter Environment.

Flutter download and install:

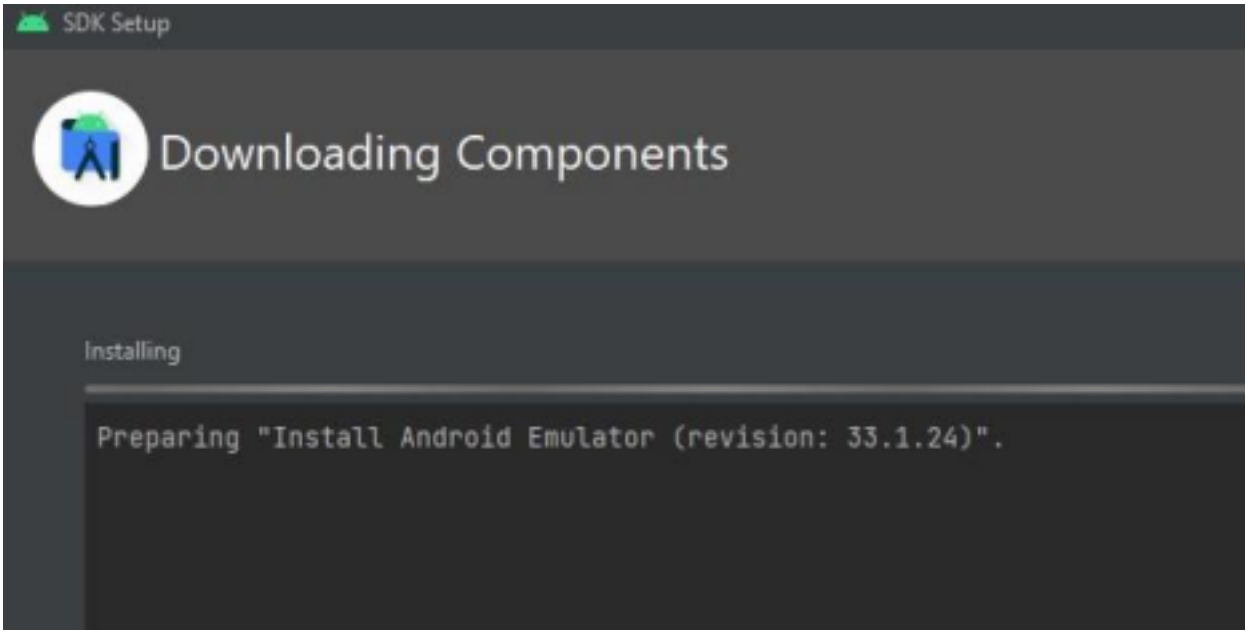


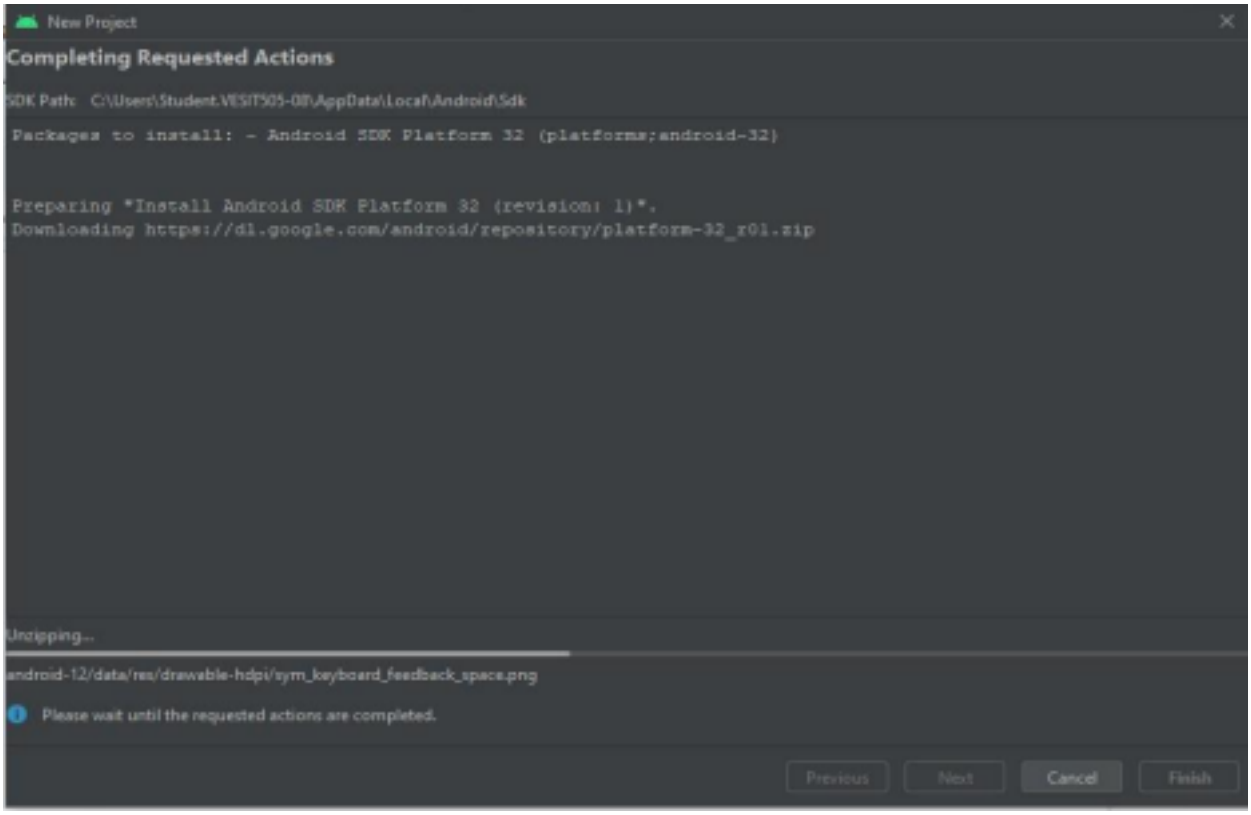
Android Studio installation:





Installation for Android Emulator:





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('My First
Flutter App'),
),
body: const Center(
child: Text('Hello World!'),
),
),
);}}
```

Output:

Conclusion: Flutter and Android Studio have been installed and configured.

MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	19
Name	Harsh Gawali
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

MAD LAB EXPT 2

Name: Harsh Gawali_D15A_19

Aim: To design Flutter UI by including common widgets.

Theory:

Flutter employs a reactive framework, allowing for fast development and expressive, flexible UI designs. Central to Flutter are widgets, which are the building blocks used to construct user interfaces.

Container: A versatile widget used to contain other widgets. It allows you to customize properties such as alignment, padding, margin, color, and more.

Row and Column: Widgets used to arrange child widgets horizontally (Row) or vertically (Column). They automatically size and position their children according to their properties.

Text: Widget used to display text with styling options like font size, color, weight, and alignment.

Image: Widget used to display images from various sources such as assets, network, or memory. It supports various image formats and provides options for resizing and scaling.

Icon: Widget used to display icons from the Material Icons or custom icon sets.

Output:

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

import 'package:whatsapp/common/utils/colours.dart';

//import 'package:whatsapp/common/extension/custom_theme_extension.dart'; //
Correct  import

import 'package:whatsapp/common/widgets/custom_elevated_button.dart';
import 'package:whatsapp/common/route/routes.dart'; // Correct
```

```
import import 'package:whatsapp/feature/auth/login_page.dart';

import 'package:whatsapp/feature/welcome/widgets/PrivacyAndTerms.dart'; // Correct

import //import 'dart:js';


import 'package:whatsapp/common/utils/colours.dart';

import '../../../common/extension/custon_theme_extension.dart';

import

'package:whatsapp/common/widgets/custom_elevated_button.dart';

//import 'lib/common/route/routes.dart';

import 'package:whatsapp/feature/welcome/widgets/PrivacyAndTerms.dart'; // Correct import


class WelcomePage extends StatelessWidget {

 const WelcomePage({super.key});


 navigateToLoginPage(context) {

 Navigator.of(context)

 .pushNamedAndRemoveUntil(Routes.login, (route) => false);

 }


 void showBottomSheet(BuildContext context) {

 showModalBottomSheet(
 context: context,
```

```
 builder: (context) {
 return Padding(
 padding: const EdgeInsets.symmetric(vertical: 10),
child: Column(
 mainAxisSize: MainAxisSize.min,
 children: [
 Container(
 height: 4,
 width: 35,
 decoration: BoxDecoration(
 color: context.theme.greyColor!.withOpacity(0.5),
borderRadius: BorderRadius.circular(10),  ),
 ),
 const SizedBox(
 height: 25,
 ),
 Row(
 children: [
 IconButton(
 onPressed: () => Navigator.of(context).pop(),
splashColor: Colors.transparent,
 splashRadius: 22,
```

```
 iconSize: 22,

 padding: EdgeInsets.zero,

 constraints: const BoxConstraints(minWidth: 40),  icon:

const Icon(

 Icons.close_outlined,

 color: Color.fromARGB(255, 255, 0, 0),  ),

 ),

 const SizedBox(

 width: 15,

 ),

 const Text(

 " App Language ",

 style: TextStyle(

 fontWeight: FontWeight.w600,

 ),

 ),

 ],
 ),

 const SizedBox(

 height: 15,

 ),

 Divider(
```

```
 color: context.theme.greyColor!.withOpacity(0.2),

thickness: 3,

 ),

 RadioListTile(

 value: true,

 groupValue: true,

 onChanged: (value) {},

 activeColor: Colours.greeDark,

 subtitle: Text(

 "(Phone's Language)",

 style: TextStyle(color: context.theme.greyColor),  ),

 ),

 RadioListTile(

 value: false,

 groupValue: true,
 onChanged: (value) {},

 activeColor: Colours.greeDark,

 title: const Text(" मराठी "),

 subtitle: Text(

 " India ",

 style: TextStyle(color: context.theme.greyColor),  ),

 ),
```

```
RadioListTile(

value: false,

groupValue: true,

onChanged: (value) {},

activeColor: Colours.greeDark,

title: const Text(" ગુજરાતી "),

subtitle: Text(

" India ",

style: TextStyle(color: context.theme.greyColor),  ),

)

],

),
);

/*

Container(

height: 300,

child: Column(

children: [

Text('Bottom Sheet Content'),  // You can add

more widgets as needed  ],

),

);
```

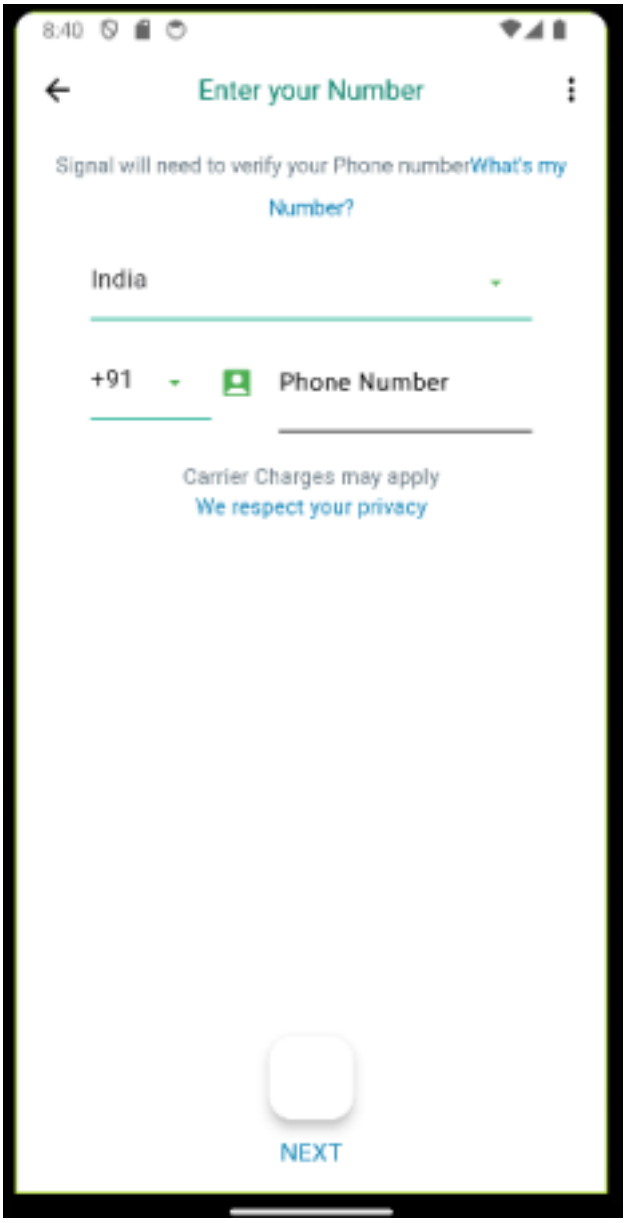
```
 */

 }, // <-- Add a comma here

 ); // <-- Add a semicolon here

 }



 @override

 Widget build(BuildContext context) {

return Scaffold(

 body: Column(

 children: [
 Expanded(

 child: Align(

 alignment: Alignment.bottomCenter,

 child: Padding(

 padding:

 const EdgeInsets.symmetric(horizontal: 50, vertical: 10),  child:

Image.asset(

 "assets/images/circle.png",

 color: Colors.green,

 //color: context.theme.circleImageColor,

 ),
```

```
//color: Colors.green,

//color: Theme.of(context).circleImageColor ?? Colors.green,  // ),

),

),

),

const SizedBox(

height: 43,

),
Expanded(

child: Column(

children: [

Center(

child: Center(

child: Text(

'Welcome to Signal',

style: TextStyle(

fontSize: 24,

fontWeight: FontWeight.w900,


//background theme ke hisaabse text color change hogaaa  color:

Theme.of(context).brightness == Brightness.dark  ?

ThemeData.dark().textTheme.headlineLarge!.color  :
```

```
ThemeData.light().textTheme.headlineLarge!.color,  ),

 ),

 ),

 ),

 const PrivacyAndTerms(),

 CustomElevatedButton(
 onPressed: () {

 Navigator.push(

 context,

 MaterialPageRoute(

 builder: (context) => const LoginPage(),  ),

 );

 },

 text: 'AGREE AND CONTINUE',

textStyle: TextStyle(),

 ),

 const SizedBox(

 height: 50,

 ),

 Material(

 color: Color(0xFF182235),

 borderRadius: BorderRadius.circular(22),  child:
```

```
InkWell(
 onTap: () {
 showBottomSheet(context);  },
 borderRadius: BorderRadius.circular(22),  splashFactory:
InkSplash.splashFactory,  highlightColor:
Color.fromARGB(255, 9, 20, 27),  child: Padding(
 padding: const EdgeInsets.symmetric(  horizontal:
13, vertical: 6),
 child: Row(
 mainAxisSize: MainAxisSize.min,  children:
const [
 Icon(
 Icons.language,
 color: Color(0xFF00A885),
 ),
 SizedBox(
 width: 20,
 ),
 Text(
 ' English ',
 style: TextStyle(color: Colors.white),  ),
 SizedBox(
```

```
 width: 20,
 ),
 Icon(
 Icons.keyboard_arrow_down,  color: Color.fromARGB(255,
110, 255, 226),  )
 ],
 ),
 ),
 ),
 )
 ],
 ),
 ),
 ],
 ),
 );
 }
}
```



Conclusion: Common widgets for text and button have been implemented for Flutter application.

Project Title: Flutter-Signal Clone / PWA- React Ecommerce **Roll No. 19**

MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	19
Name	Harsh Gawali
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Experiment 3: To include icons, images, fonts in Flutter app

Harsh Gawali Roll.no:19 D15A

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

We can split the Flutter widget into two categories:

Visible (Output and Input)

Invisible (Layout and Control)

Visible widget

The visible widgets are related to the user input and output data. Some of the important types of this widget are:

1. Text

A Text widget holds some text to display on the screen. We can align the text widget by using textAlign property, and style property allow the customization of Text that includes font, font weight, font style, letter spacing, color, and many more. We can use it as like below code snippets. new Text('Hello, ALL!', textAlign: TextAlign.center, style: new TextStyle(fontWeight: FontWeight.bold),

2. Button

This widget allows you to perform some action on click. Flutter does not allow you to use the Button widget directly; instead, it uses a type of buttons like a FlatButton and a RaisedButton. We can use it as like below code snippets

3. **Image**

This widget holds the image which can fetch it from multiple sources like from the asset folder or directly from the URL. It provides many constructors for loading image, which are given below:

- **Image:** It is a generic image loader, which is used by **ImageProvider**.
- **asset:** It load image from your project asset folder.
- **file:** It loads images from the system folder.

- **memory:** It load image from memory. - **network:** It loads images from the network.

To add an image in the project, you need first to create an assets folder where you keep your images and then add the below line in **pubspec.yaml** file.

Code:

```
import 'package:flutter/material.dart';
import 'package:whatsapp/common/utils/colours.dart';
//import 'package:whatsapp/common/extension/custom_theme_extension.dart'; //
Correct import
import 'package:whatsapp/common/widgets/custom_elevated_button.dart'; import
'package:whatsapp/common/route/routes.dart'; // Correct import import
'package:whatsapp/feature/auth/login_page.dart';
import 'package:whatsapp/feature/welcome/widgets/PrivacyAndTerms.dart'; //
Correct import
//import 'dart:js';
import 'package:whatsapp/common/utils/colours.dart';
import '../../../common/extension/custon_theme_extension.dart'; import
'package:whatsapp/common/widgets/custom_elevated_button.dart'; //import
'lib/common/route/routes.dart';
import 'package:whatsapp/feature/welcome/widgets/PrivacyAndTerms.dart'; //
Correct import
class WelcomePage extends StatelessWidget {
 const WelcomePage({super.key});
 navigateToLoginPage(context) {
 Navigator.of(context)
 .pushNamedAndRemoveUntil(Routes.login, (route) => false);  }
 void showBottomSheet(BuildContext context) {
 showModalBottomSheet(
 context: context,
 builder: (context) {
 return Padding(
 padding: const EdgeInsets.symmetric(vertical: 10),  child: Column(
 mainAxisSize: MainAxisSize.min,
 children: [
 Container(
 height: 4,
 width: 35,
 decoration: BoxDecoration(
 color: context.theme.greyColor!.withOpacity(0.5),  borderRadius:
BorderRadius.circular(10),  ),
 ),
 const SizedBox(
 height: 25,
 ),
 Row(
 children: [
 IconButton(
 onPressed: () => Navigator.of(context).pop(), splashColor: Colors.transparent,
splashRadius: 22,
iconSize: 22,
```

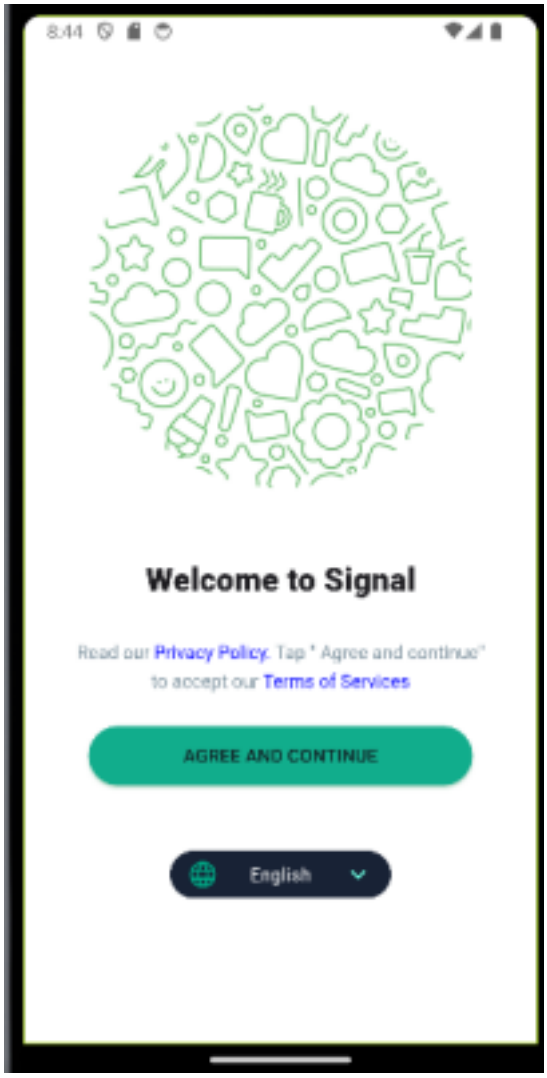
```
padding: EdgeInsets.zero,
constraints: const BoxConstraints(minWidth: 40),
 icon: const Icon(
 Icons.close_outlined,
color: Color.fromARGB(255, 255, 0, 0),
 ),
 ),
const SizedBox(
 width: 15,
 ),
const Text(
 " App Language ",
style: TextStyle(
 fontWeight: FontWeight.w600,
 ),
 ),
 ],
 ),
 const SizedBox(
 height: 15,
 ),
 Divider(
 color: context.theme.greyColor!.withOpacity(0.2),  thickness: 3,
 ),
 RadioListTile(
 value: true,
 groupValue: true,
 onChanged: (value) {},
 activeColor: Colours.greeDark,
 subtitle: Text(
 "(Phone's Language)",
style: TextStyle(color: context.theme.greyColor),
 ),
 ),
 RadioListTile(
 value: false,
 groupValue: true,
 onChanged: (value) {},
 activeColor: Colours.greeDark,
 title: const Text(" मराठी "),
 subtitle: Text(
 " India ",
style: TextStyle(color: context.theme.greyColor),
 ),
 ),
 RadioListTile(
 value: false,
 groupValue: true,
 onChanged: (value) {},
 activeColor: Colours.greeDark,
 title: const Text(" ગુજરાતી "),
 subtitle: Text(
 " India ",
style: TextStyle(color: context.theme.greyColor),
 ),
 )
 ],
```

```
 ),
 );
 /*
 Container(
 height: 300,
 child: Column(
 children: [
 Text('Bottom Sheet Content'),
 // You can add more widgets as needed
 ],
 ),
 );
 */
 }, // <-- Add a comma here
 ); // <-- Add a semicolon here
 }
 @override
 Widget build(BuildContext context) {
 return Scaffold(
 body: Column(
 children: [
 Expanded(
 child: Align(
 alignment: Alignment.bottomCenter,
 child: Padding(
 padding:
 const EdgeInsets.symmetric(horizontal: 50, vertical: 10),  child: Image.asset(
 "assets/images/circle.png",
color: Colors.green,
//color: context.theme.circleImageColor,
 ),
 //color: Colors.green,
 //color: Theme.of(context).circleImageColor ?? Colors.green,  // ),
 ),
 ),
 ),
 const SizedBox(
 height: 43,
 ),
 Expanded(
 child: Column(
 children: [
 Center(
 child: Center(
 child: Text(
 'Welcome to Signal',
style: TextStyle(
 fontSize: 24,
fontWeight: FontWeight.w900,
 //background theme ke hisaabse text color change  hogaaa
 color: Theme.of(context).brightness == Brightness.dark  ?
ThemeData.dark().textTheme.headlineLarge!.color  :
ThemeData.light().textTheme.headlineLarge!.color,
 ),
 ),
 ),
 ),
```

```
 const PrivacyAndTerms(),
 CustomElevatedButton(
 onPressed: () {
 Navigator.push(
 context,
MaterialPageRoute(
 builder: (context) => const LoginPage(),  ),
 );
 },
text: 'AGREE AND CONTINUE',
textStyle: TextStyle(),
 ),
 const SizedBox(
 height: 50,
 ),
 Material(
 color: Color(0xFF182235),
 borderRadius: BorderRadius.circular(22),
 child: InkWell(
 onTap: () {
 showBottomSheet(context);
 },
 borderRadius: BorderRadius.circular(22),  splashFactory:
InkSplash.splashFactory, highlightColor: Color.fromARGB(255, 9, 20, 27),
 child: Padding(
 padding: const EdgeInsets.symmetric(  horizontal: 13, vertical: 6),  child:
Row(
 mainAxisSize: MainAxisSize.min, children: const [
 Icon(
 Icons.language,
color: Color(0xFF00A885),
 ),
SizedBox(
 width: 20,
 ),
Text(
 ' English ',
style: TextStyle(color: Colors.white),
 ),
SizedBox(
 width: 20,
 ),
Icon(
 Icons.keyboard_arrow_down, color: Color.fromARGB(255, 110, 255, 226),
 )
 ],
 ),
 ),
 ),
 )
 ],
 ),
 ),
 ],
 ),
 );
 }
```

```
}
```

Output:



Conclusion:

In this experiment, we have succesfully imported and inerted image in the flutter and used font style to enter text and succesfully created button for it. All concept of image, font are implemented succesfully.

MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	19
Name	Harsh Gawali
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

MAD LAB EXPT 4

Name: Harsh Gawali_D15A_19

Aim: To make an interactive form using Widgets.

Theory:

Text Input Fields- Use the TextField widget to allow users to input text. You can customize it with properties like decoration, controller, keyboardType, validator, and onChanged callback.

Form Widget- Encapsulate your input fields within a Form widget. The Form widget manages the form state and provides methods for validation and submission.

Form Fields- Wrap each input field within a TextFormField widget. This widget integrates with the Form widget and automatically handles validation.

Validation-Implement validation logic to ensure that the user input meets certain criteria (e.g., required fields, valid email format). You can use the validator property of the TextFormField widget to specify validation functions.

State Management- Maintain the form's state using either StatefulWidget or state management solutions like Provider, Riverpod, or Bloc. When the user interacts with the form (e.g., typing in a text field), update the corresponding state.

Submit Button- Include a button (e.g., ElevatedButton or TextButton) to allow users to submit the form. Disable the button if the form is invalid.

Handling Form Submission-Define a function to handle form submission. This function should be called when the user taps the submit button. You can access the form data using the FormState object.

Output:

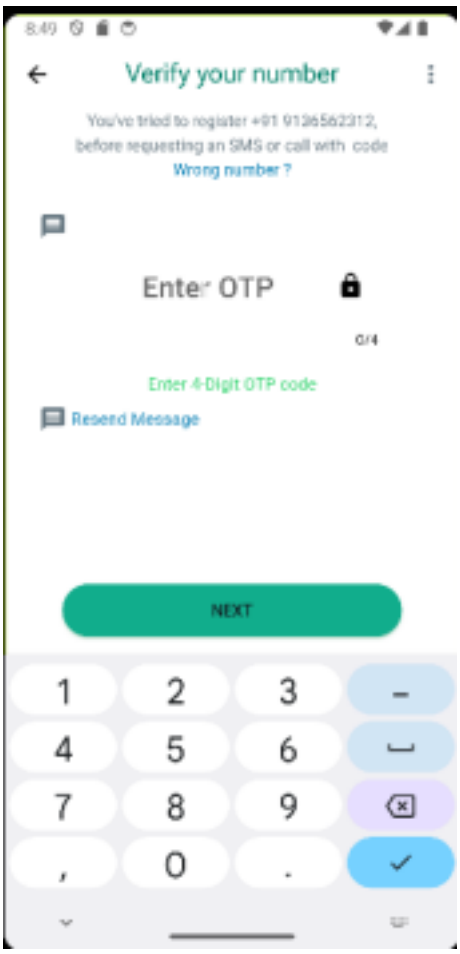
```
import 'dart:ffi';
import 'dart:io';

import 'package:flutter/material.dart';
import 'package:whatsapp/common/extension/custon_theme_extension.dart';
```

```
import 'package:whatsapp/common/widgets/custom_elevated_button.dart';
import 'package:whatsapp/feature/auth/widgets/Custom_Icon.dart'; import
'package:whatsapp/feature/auth/widgets/custom_text_field.dart'; import
'package:whatsapp/feature/welcome/pages/user_info_pages.dart';

class VerificationPage extends StatelessWidget {
 TextEditingController otpController = TextEditingController();
@override
 Widget build(BuildContext context) {
 return Scaffold(
 appBar: AppBar(
 backgroundColor: Theme.of(context).scaffoldBackgroundColor,
elevation: 11,
 title: Text(
 'Verify your number',
 style: TextStyle(
 color: context.theme.authAppbartextColor,
 ),
 ),
 centerTitle: true,
 actions: [
 CustomIconButton(
 onTap: () {},
 icon: Icons.more_vert,
 ),
 ],
 ),
 body: SingleChildScrollView(
 padding: const EdgeInsets.symmetric(horizontal: 30),  child:
Column(
 children: [
 Padding(
 padding: const EdgeInsets.symmetric(horizontal: 20),  child:
RichText(
 textAlign: TextAlign.center,
 text: TextSpan(
 style: TextStyle(
 color: context.theme.greyColor,
 height: 1.5,
 ),
                  children: [
 TextSpan(
 text:
 "You've tried to register +91 9136562312, before  requesting an SMS or call
with code ",
 ),
                    TextSpan(
 text: 'Wrong number ?',
                      style: TextStyle(
 color: context.theme.blueColor,
 ),
 ),
 ],
 ),
 ),
 ),
 const SizedBox(
```

```
 height: 20,
 ),
 Row(
 children: [
 Icon(Icons.message, color: context.theme.greyColor),  const
SizedBox(height: 40),
 ],
 ),
 Container(
 alignment: Alignment.center,
 padding: const EdgeInsets.symmetric(horizontal: 40),  child:
TextField(
 controller: otpController, // Use a TextEditingController  decoration:
InputDecoration(
 hintText: ' Enter OTP',
                  hintStyle: TextStyle(fontSize: 25),
 suffixIcon: Icon(Icons.lock),
 ),
 keyboardType: TextInputType.number,
 textAlign: TextAlign.center,
 style: TextStyle(fontSize: 25),
 maxLength: 4,
 ),
 ),
 const SizedBox(
 height: 20,
 ),
 Text(
 'Enter 4-Digit OTP code',
 style: TextStyle(color: context.theme.circleImageColor),  ),
 Row(
 children: [
 Icon(Icons.message, color: context.theme.greyColor),  const
SizedBox(height: 40),
 Text(
 ' Resend Message',
                   style: TextStyle(color: context.theme.blueColor),
 ),
 ],
 ),
 Divider(
 thickness: 0.2,
 color: context.theme.greyColor!.withOpacity(1),  endIndent:
2,
 ),
 ],
 ),
 ),
 floatingActionButtonLocation: FloatingActionButtonLocation.centerFloat,
floatingActionButton: CustomElevatedButton(
 onPressed: () {
 // Check if entered OTP is '9999'
 if (otpController.text == '9999') {
 // Navigate to the next page when the "NEXT" button is pressed
Navigator.push(
 context,
 MaterialPageRoute(
```

```
 builder: (context) => UserInfoPage(),
 ),
 );
 } else {
 // Show an error message or take any other action  // For now, print an
error message to the console  print('Invalid OTP. Please enter the
correct OTP (9999).');  }
 },
 text: 'NEXT',
 textStyle: TextStyle(color: context.theme.authAppbartextColor),  )
 );
 }
}
```

Output:



Conclusion: An interactive form using Widgets has been created in Flutter application.

Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	19
Name	Harsh Gawali
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

MAD LAB EXPT 5

Name: Harsh Gawali_D15A_19

Aim: To apply navigation,routing and gestures in Flutter.

Theory:

In Flutter, navigation, routing, and gestures are essential concepts for creating interactive and navigable user interfaces.

Navigation: Navigation refers to the process of moving between different screens or pages within a Flutter app. Flutter provides the Navigator widget for managing navigation and routing.

Routing:Routing is the mechanism used to define the paths or routes between different screens in your app. Each route typically corresponds to a different widget or screen in your app.

Gesture Detection:Gestures allow users to interact with the app by tapping, dragging, swiping, or performing other touch-based actions. Flutter provides various gesture detection widgets to handle user input.

Code:

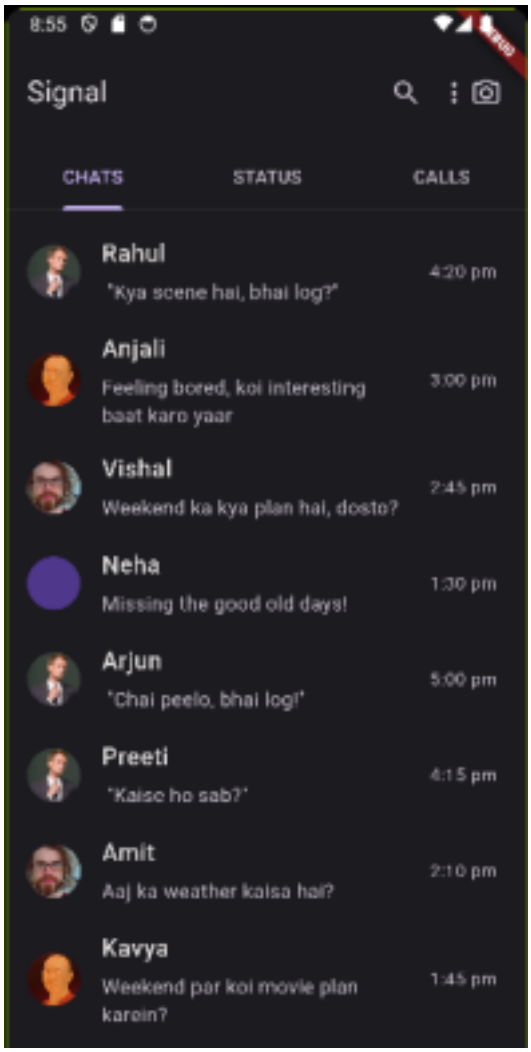
```
const info = [
 {
 'name': 'Rahul',
 'message': ' "Kya scene hai, bhai log?" ',
 'time': '4:20 pm',
 'profilePic':
 'https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTJWo
QWKz9IFVa8uH9qmESH57YQMcw30jnUMXqQ-eibRdpUo5w1VH6PGktKXLQWAa7FQff&s',
},
 {
 'name': 'Anjali',
 'message': 'Feeling bored, koi interesting baat karo yaar',
'time': '3:00 pm',
 'profilePic':
 'https://www.socialketchup.in/wp-content/uploads/2020/05/fi-vill-JOHN
DOE.jpg',
 },
 {
 'name': 'Vishal',
 'message': 'Weekend ka kya plan hai, dosto?',
 'time': '2:45 pm',
 'profilePic':
 'https://pbs.twimg.com/profile_images/1419974913260232732/Cy_CUavB.jpg',
 },
```

```
 {
 'name': 'Neha',
 'message': 'Missing the good old days!',
 'time': '1:30 pm',
 'profilePic':

'https://media.cntraveler.com/photos/60596b398f4452dac88c59f8/16:9/w_3999,h_2
249,c_limit/MtFuji-GettyImages-959111140.jpg',
 },
 {
 'name': 'Arjun',
 'message': ' "Chai peelo, bhai log!" ',
 'time': '5:00 pm',
 'profilePic':
 'https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTJWo
QWKz9IFVa8uH9qmESH57YQMcw30jnUMXqQ-eibRdpUo5w1VH6PGktKXLQWAa7FQff&s',
},
 {
 'name': 'Preeti',
 'message': ' "Kaise ho sab?" ',
 'time': '4:15 pm',
 'profilePic':
 'https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTJWo
QWKz9IFVa8uH9qmESH57YQMcw30jnUMXqQ-eibRdpUo5w1VH6PGktKXLQWAa7FQff&s',
},
 {
 'name': 'Amit',
 'message': 'Aaj ka weather kaisa hai?',
 'time': '2:10 pm',
 'profilePic':
 'https://pbs.twimg.com/profile_images/1419974913260232732/Cy_CUavB.jpg',  },
 {
 'name': 'Kavya',
 'message': 'Weekend par koi movie plan karein?',
 'time': '1:45 pm',
 'profilePic':
 'https://www.socialketchup.in/wp-content/uploads/2020/05/fi-vill-JOHN
DOE.jpg',
 },
 {
 'name': 'Rajat',
 'message': ' "Bhai, kuch interesting batao yaar!" ',
 'time': '3:30 pm',
 'profilePic':
 'https://example.com/rajat_profile_pic.jpg',
 },
 {
 'name': 'Shreya',
 'message': ' "Ghar ka kya scene hai?" ',
 'time': '4:45 pm',
 'profilePic':
 'https://example.com/shreya_profile_pic.jpg',
 },
 {
 'name': 'Vikram',
 'message': 'Bhai log, pubg khelte hai!',
 'time': '6:00 pm',
```

```
 'profilePic':
 'https://example.com/vikram_profile_pic.jpg',
 },
 {
 'name': 'Ananya',
 'message': ' "Koi naya joke sunao, yaar!" ',
 'time': '3:15 pm',
 'profilePic':
 'https://example.com/ananya_profile_pic.jpg',
 },
 {
 'name': 'Aryan',
 'message': 'Weekend par outing plan karo, guys!',
 'time': '2:30 pm',
 'profilePic':
 'https://example.com/aryan_profile_pic.jpg',
 },
 {
 'name': 'Kritika',
 'message': 'Boring lecture chal raha hai, koi distraction chahiye!',
'time': '1:15 pm',
 'profilePic':
 'https://example.com/kritika_profile_pic.jpg',
 },
 {
 'name': 'Sarthak',
 'message': ' "Kuch interesting discuss karte hai!" ',
'time': '4:50 pm',
 'profilePic':
 'https://example.com/sarthak_profile_pic.jpg',
 },
 {
 'name': 'Tanvi',
 'message': 'Bhai log, kaun kaun online hai?',
 'time': '3:10 pm',
 'profilePic':
 'https://example.com/tanvi_profile_pic.jpg',
 },
 {
 'name': 'Rohan',
 'message': 'Koi new music suggest karo, yaar!',
 'time': '2:00 pm',
 'profilePic':
 'https://example.com/rohan_profile_pic.jpg',
 },
 {
 'name': 'Shivani',
 'message': 'Missing the canteen wale maggi!',
 'time': '1:40 pm',
 'profilePic':
 'https://example.com/shivani_profile_pic.jpg',
 },
 {
 'name': 'Alok',
 'message': ' "Aaj kya special hai?" ',
 'time': '4:35 pm',
 'profilePic':
```

```
'https://example.com/alok_profile_pic.jpg',
},
{
'name': 'Nidhi',
'message': 'Weekend par koi trip plan karo, guys!',
'time': '5:15 pm',
'profilePic':
'https://example.com/nidhi_profile_pic.jpg',
},
];
```

Output:



Conclusion: Navigation,routing and gestures have been applied in Flutter Application.

Project Title: Flutter-Signal Clone / PWA- React Ecommerce **Roll No. 19**

MAD & PWA Lab
Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	19
Name	Harsh Gawali
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	

MAD LAB EXPT 6

Name: Harsh Gawali_D15A_19

Aim: To connect Flutter UI with Firebase.

Theory:

Step1:
Go to the Firebase Console (https://console.firebase.google.com/) and create a new project.Follow the instructions to add your app to the Firebase project. You'll need to provide your app's package name (Android) or bundle identifier (iOS).

Step 2:
Add the Firebase SDK dependencies to your Flutter app's pubspec.yaml file. These dependencies vary depending on the Firebase services you want to use (e.g., Firebase Authentication, Firestore, Realtime Database, Cloud Storage). Run flutter pub get to install the dependencies.

Step 3:
In your Flutter app, initialize Firebase by calling Firebase.initializeApp() in the main() function or at the entry point of your app.
This initialization step is crucial and should be done before accessing any Firebase services.

Step 4:
Once Firebase is initialized, you can start using Firebase services like Firestore (NoSQL database), Realtime Database (JSON database), Cloud Storage (file storage), Cloud Functions (serverless functions), etc.
You'll typically use Firebase APIs to read and write data, handle user authentication, and perform other tasks.

Step 5:
Use Firebase listeners to listen for real-time updates to your data. For example, in Firestore, you can set up listeners to receive updates whenever the data in a collection or document changes.

Step 6:
Implement error handling logic to handle exceptions and errors that may occur when

interacting with Firebase services.

Output:

```
// File generated by FlutterFire CLI.
// ignore_for_file: lines_longer_than_80_chars,
avoid_classes_with_only_static_members
import 'package:firebase_core/firebase_core.dart' show FirebaseOptions;
import 'package:flutter/foundation.dart'
 show defaultTargetPlatform, kIsWeb, TargetPlatform;

/// Default [FirebaseOptions] for use with your Firebase
apps. ///
/// Example:
/// ```dart
/// import 'firebase_options.dart';
/// // ...
/// await Firebase.initializeApp(
/// options: DefaultFirebaseOptions.currentPlatform,
/// );
/// ```
class DefaultFirebaseOptions {
 static FirebaseOptions get currentPlatform {
 if (kIsWeb) {
 throw UnsupportedError(
 'DefaultFirebaseOptions have not been configured for web - '  'you can
reconfigure this by running the FlutterFire CLI again.',  );
 }
 switch (defaultTargetPlatform) {
 case TargetPlatform.android:
 return android;
 case TargetPlatform.iOS:
 throw UnsupportedError(
 'DefaultFirebaseOptions have not been configured for ios - '  'you can
reconfigure this by running the FlutterFire CLI again.',  );
 case TargetPlatform.macOS:
 throw UnsupportedError(
 'DefaultFirebaseOptions have not been configured for macos - '  'you can
reconfigure this by running the FlutterFire CLI again.',  );
 case TargetPlatform.windows:
 throw UnsupportedError(
 'DefaultFirebaseOptions have not been configured for windows - '  'you can
reconfigure this by running the FlutterFire CLI again.',  );
 case TargetPlatform.linux:
 throw UnsupportedError(
 'DefaultFirebaseOptions have not been configured for linux - '  'you can
reconfigure this by running the FlutterFire CLI again.',  );
 default:
 throw UnsupportedError(
 'DefaultFirebaseOptions are not supported for this platform.',  );
 }
 }

 static const FirebaseOptions android = FirebaseOptions(
apiKey: 'AIzaSyAQ0Rqa0Ql_a3IKREiYIEocGr1BQSGhi14',
```

```
 appId: '1:860315971576:android:7b303c871e55a3bc28700d',
messagingSenderId: '860315971576',
 projectId: 'whatsappfire-a0ace',
 storageBucket: 'whatsappfire-a0ace.appspot.com',
 );
}
```

Connected with firebase for authentication.

<u>Conclusion: Firebase has been connected to the Flutter application for user authentication purposes.</u>

Project Title: Flutter-Signal Clone / PWA- React Ecommerce **Roll No. 19**

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.	19
Name	Harsh Gawali
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	

MAD and PWA Lab

Name: Harsh Gawali Class: D15A Roll no:19

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 Footer from './components/footer/Footer'; import Hero from
'./components/hero/Hero'; import Navbar from
'./components/navbar/Navbar'; import Newsletter from
'./components/newsletter/Newsletter'; import
PopularProperties from
'./components/popularProperties/PopularProperties'; import
Signin from './components/signin/Signin';
import Signup from './components/signup/Signup';
import Properties from
'./components/properties/Properties'; import
PropertyDetail from
'./components/propertyDetail/PropertyDetail'; import {
useSelector } from 'react-redux' import { Routes,
Route, Navigate, useLocation } from 'react-router-dom'
import { useEffect } from 'react'; import EditProperty
from './components/editProperty/EditProperty'; import
Yachts from './components/yachts/Yachts'; import
YachtDetails from
'./components/yachtDetails/YachtDetails'; import
CreateYacht from
'./components/createYacht/CreateYacht'; import
YachtEdit from
'./components/yachtEdit/YachtEdit'; import
MyProfile from
'./components/myProfile/MyProfile'; import
UpdateProfile from
'./components/updateProfile/UpdateProfile'; import
'./App.css'; import NotFound from
'./components/notFound/NotFound'; import Layout
from './components/Layout'; import Services from
```

```
'./components/Services/Services';


function App() {
  const { user } = useSelector((state) => state.auth)
  const url = useLocation().pathname

  useEffect(() => {
    url && window.scrollTo(0, 0)
  }, [url])

  return (
    <div>
      <Routes>
        <Route path='/' element={
          <Layout title="DormDine || Home">
            <Navbar />
            <Hero />
            <PopularProperties />
            <Services/>
            <Newsletter />
            <Footer />
          </Layout>
        } />
        <Route path='/signup' element={!user ? <Signup /> :
        <Navigate to='/' />} /> <Route path='/signin'
        element={!user ? <Signin /> : <Navigate to='/' />} />
        <Route path='/properties' element={
          <>
            <Navbar />
            <Properties />
            <Footer />
          </>
        } />
        <Route path='/yachts'
          element={user ? <>
            <Navbar />
            <Yachts />
```

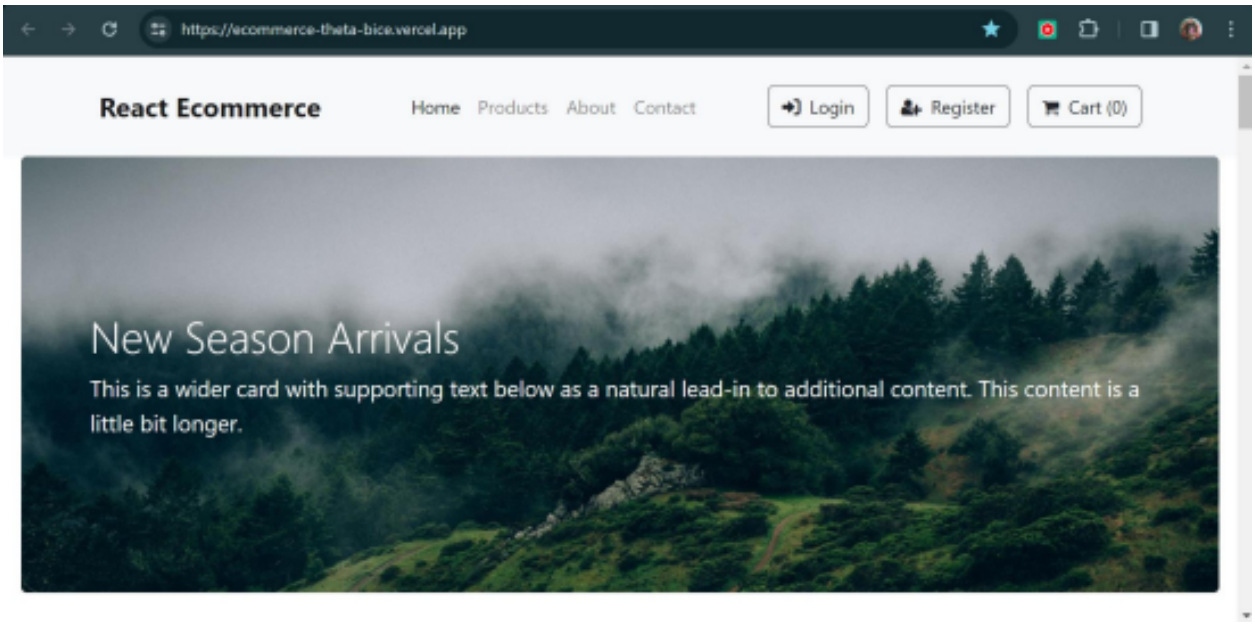
```
    <Footer />
   </>
   : <Navigate
  to='/signin' />} />
  <Route
  path='/yacht/:id'
  element={user ?
   <>
    <Navbar />
    <YachtDetails />
    <Footer />
   </>
   : <Navigate to='/signin' />} />
  <Route path='/create-yacht'
   element={user ? <>
    <Navbar />
    <CreateYacht />
    <Footer />
   </>
   : <Navigate to='/signin' />} />
  <Route path='/yacht-edit/:id'
   element={user ? <>
    <Navbar />
    <YachtEdit />
    <Footer />
   </>
   : <Navigate to='/signin' />} />
  <Route
   path='/propertyDetail/:id'
   element={ <>
    <Navbar />
    <PropertyDetail />
    <Footer />
   </>
  } />
  <Route
   path='/editproperty/:id'
   element={  user ?
    <>
```

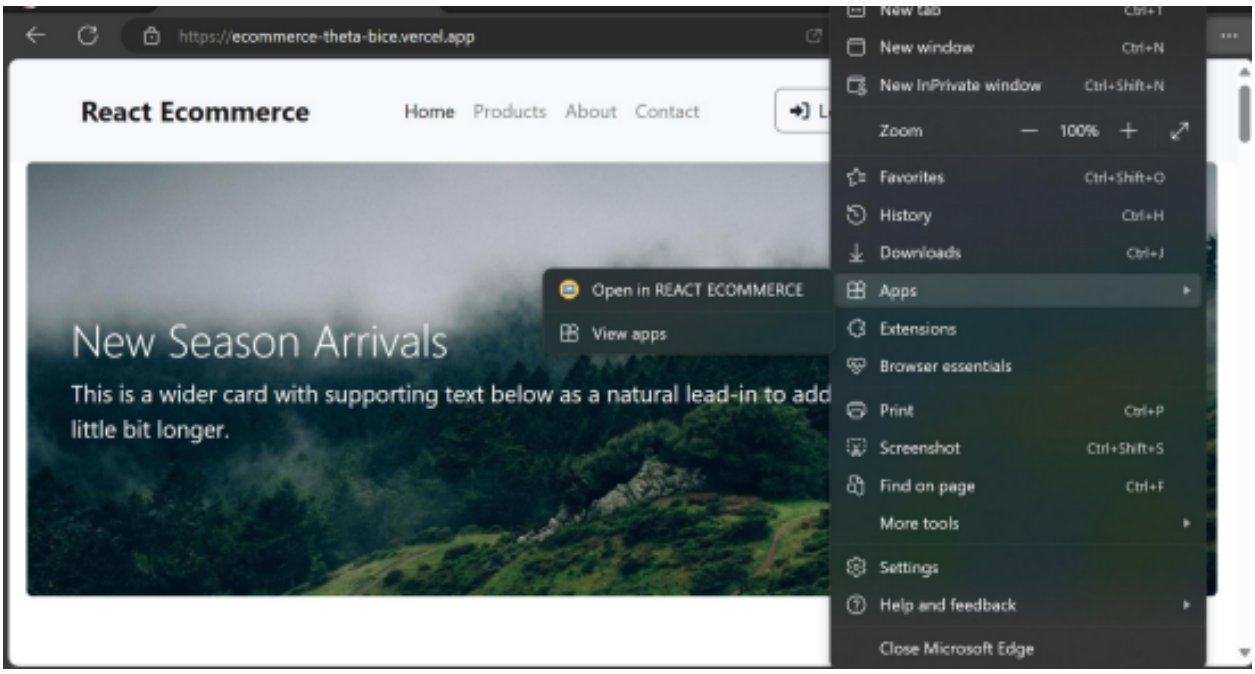
```
            <Navbar />
            <EditProperty />
            <Footer />
          </>
          : <Navigate to='/signin' />
        } />
        <Route path='/my-profile' element={
          user ?
          <>
            <Navbar />
            <MyProfile />
            <Footer />
          </>
          : <Navigate to='/signin' />
        } />
        <Route path='/update-profile' element={
          user ?
          <>
            <Navbar />
            <UpdateProfile />
            <Footer />
          </>
          : <Navigate to='/signin' />
        } />
        <Route path='*' element={
          <>
            <Navbar />
            <NotFound />
            <Footer />
          </>
        } />
      </Routes>
    </div>
  );
}

export default App;
```

Open folder in VS code and click go live at

bottom right corner Open your hosted site on

Microsoft Edge

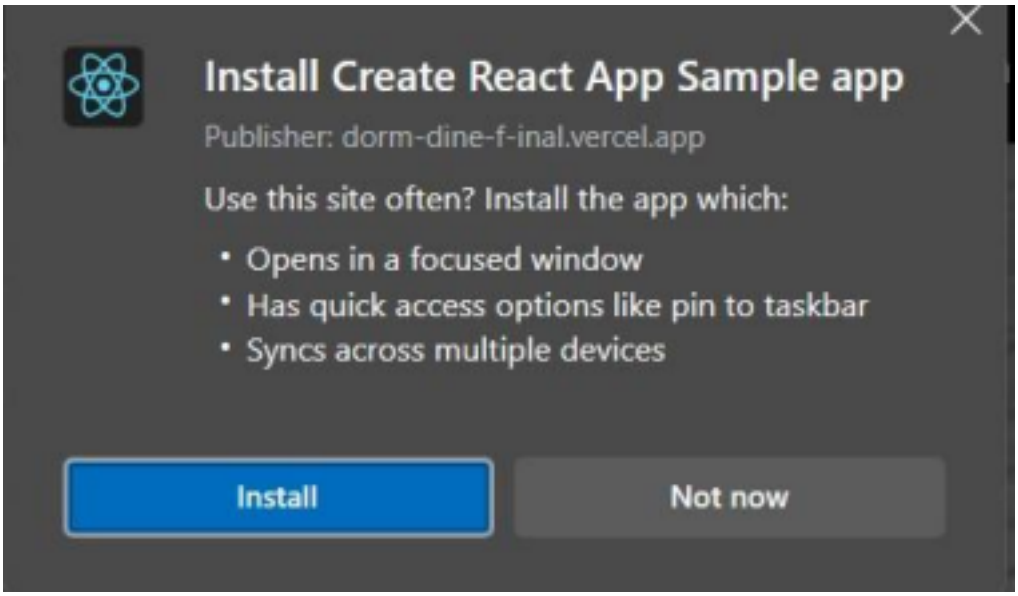




Click on 3 dots click on

Apps select install app
option

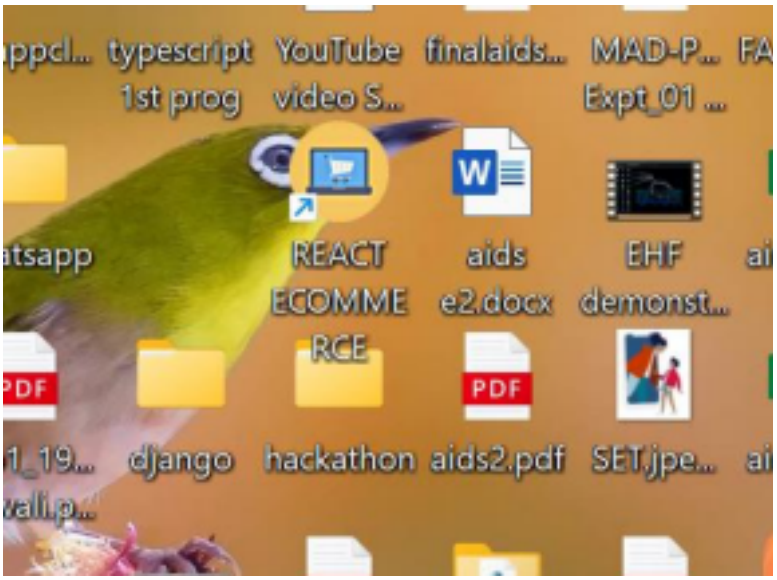
Click on Install



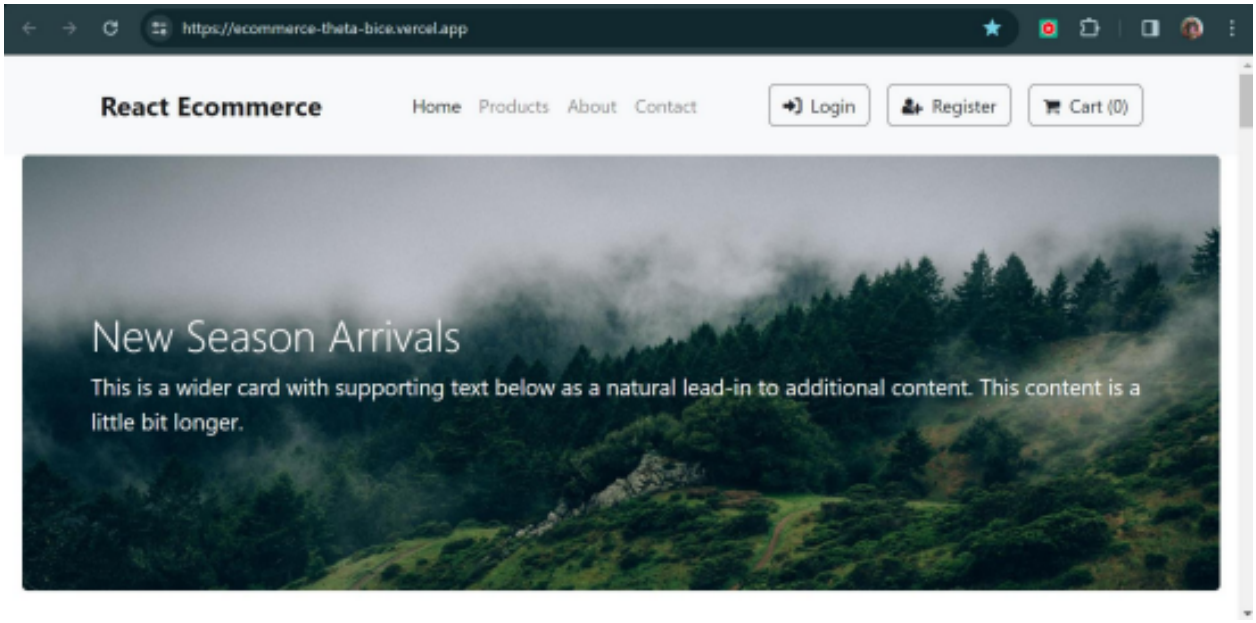
App icon will appear at the bottom

Output:

Desktop App Created Successfully.



Open Desktop App:



Conclusion:

In this experiment, we have successfully created a basic progressive web app of our web page 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.	19
Name	Harsh Gawali
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

MAD and PWA Lab

Name: Harsh Gawali Class: D15A Roll no:19 Experiment - 8

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.

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:

```
if ('serviceWorker' in navigator) {
navigator.serviceWorker.register('/service-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.

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

Installation

Once the browser registers a service worker, installation can be attempted. This occurs if the service worker is considered to be new by the browser, either because the site currently doesn't have a registered service worker, or because there is a byte difference between the new service worker and the previously installed one.
A service worker installation triggers an install event in the installing service worker. We can include an install event listener in the service worker to perform some task when the service worker installs. For instance, during the install, service workers can precache parts of a web app so that it loads instantly the next time a user opens it (see caching the application shell). So, after that first load, you're going to benefit from instant repeat loads and your time to interactivity is going to be even better in those cases. An example of an installation event listener looks like this:

service-worker.js

// Listen for install event, set callback self.addEventListener('install',
function(event) {
// Perform some task
});

Activation

Once a service worker has successfully installed, it transitions into the activation stage. If there are any open pages controlled by the previous service worker, the new service worker enters a waiting state. The new service worker only activates when there are no longer any pages loaded that are still using the old service worker. This ensures that only one version of the service worker is running at any given time.

When the new service worker activates, an activate event is triggered in the activating service worker. This event listener is a good place to clean up outdated caches (see the Offline Cookbook for an example).

Code:

index.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Amazon.com: Online Shopping for Electronics, Apparel, Computers, Books, DVDs &amp;amp;
  more</title>

  <link
  href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css"
  rel="stylesheet"> <link rel="stylesheet"
  href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+Pm
STsz/K68vbdEjh4u"  crossorigin="anonymous">
  <link rel="stylesheet" href="css/style.css"/>
</head>

<body>
  <a id="banner" href="#"></a>
  <!-- Main body -->

  <header>
    <a id="nav-top"></a>
    <nav id="nav-main">
```

```
  <div class="nav-left">
    <div class="nav-shop">
      <a class="nav-a" href="#">
        Departments
                  <i class="fa fa-caret-down" aria-hidden="true"></i>
      </a>
    </div>
  </div>
  <div class="nav-right">
    <a class="nav-a" href="#">
      <span>EN</span>
      <i class="fa fa-globe" aria-hidden="true"></i>
      <i class="fa fa-caret-down" aria-hidden="true"></i>
    </a>

    <a class="nav-a" href="#">
      <span>Hello. Sign in</span>
      Accounts &amp;amp; Lists
      <i class="fa fa-caret-down" aria-hidden="true"></i>
    </a>

    <a class="nav-a" href="#"> Orders
    </a>

    <a class="nav-a" href="#">
      Try Prime
      <i class="fa fa-caret-down" aria-hidden="true"></i>
    </a>

    <a class="nav-a cart" href="#">
      <span>0</span>
      Cart
    </a>
  </div>
  <div class="nav-fill">
    <ul>
      <li><a href="#">Your Amazon.com</a></li>
      <li><a href="#">Today's Deals</a></li>
      <li><a href="#">Gift Cards &amp;amp; Registry</a></li>
      <li><a href="#">Sell</a></li>
      <li><a href="#">Help</a></li>
    </ul>
  </div>
</nav>
```

```
  </header>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
integrity="sha384- Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa
" crossorigin="anonymous"></script> <script src="js/app.js"></script>
</body> </html>
```

main.css

```
html, body { margin: 0; font
    family: arial,sans-serif;
    min-width: 900px; line
    height: 14px; font-size:
     14px;
  }

  * {
    box-sizing: border-box;
  }

  a {
    color: #0066c0;
  }

  a:hover {
    color: #c45500;
  }

  #banner { background: #F6F6F6 url('../img/banner.jpg') no
    repeat top left; height: 55px; background-size: 1920px; min
    width: 1000px; display: block;
  }

 header {
   background-color: #232f3e;
 height: 99px; }
```

app.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

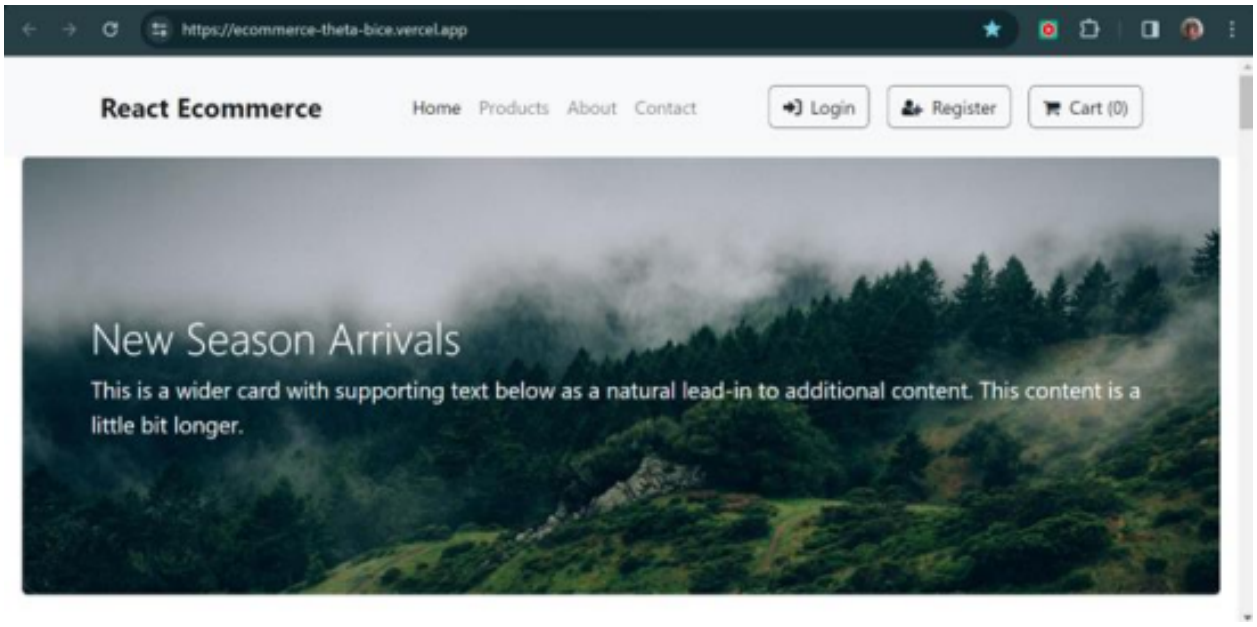
```
const cacheName = 'ecommerce-pwa-v1'; const
assetsToCache = [
    '/',
    '/index.html',
    '/main.css',
    '/app.js'
]

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);
                })
            );
        })
    );
});
```

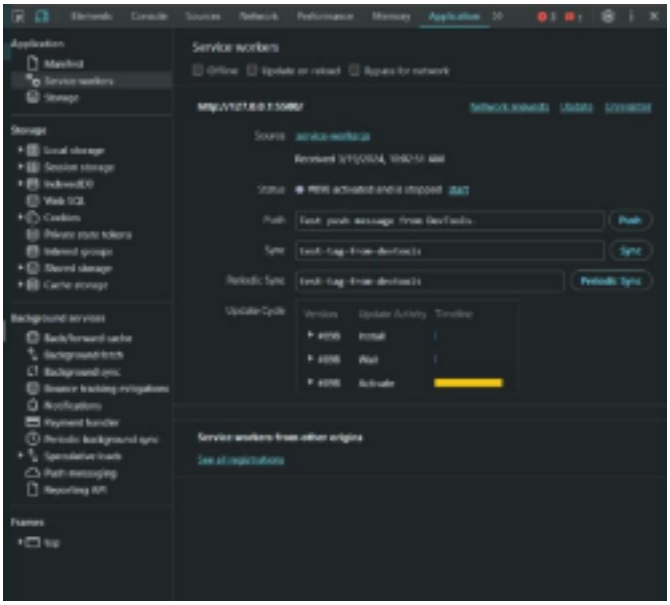
Steps for Execution

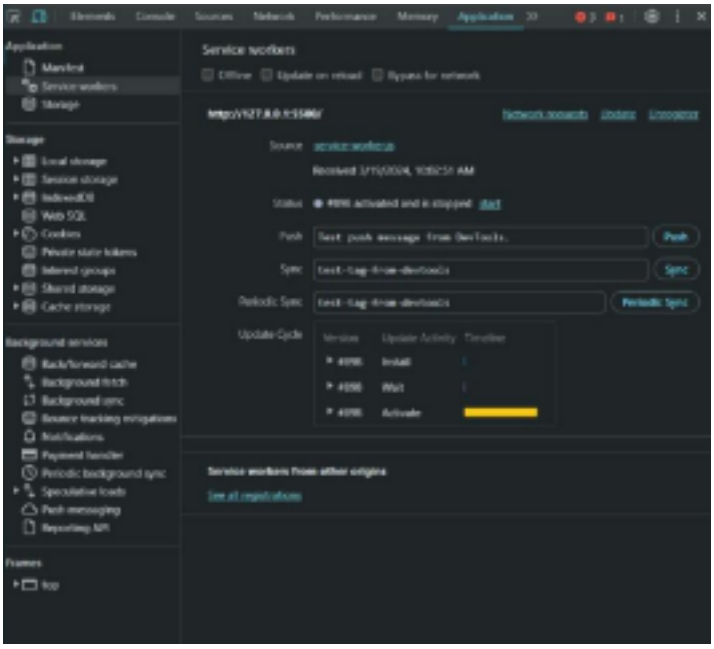
Create a folder and put all 4 files main.css , service-worker.js, app.js, index.html
open visual studio install extension Live server
open folder in visual studio open index.html on
bottom right corner click go Live it will open html
page in browser go to developer tools

Project Title: Flutter-Signal Clone / PWA- React Ecommerce **Roll No. 19**

Output:







Conclusion:

In this experiment, we have registered a service worker, and completed the install and activation process for a new service worker for the E-commerce PWA.

Project Title: Flutter-Signal Clone / PWA- React Ecommerce **Roll No. 19**

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.	19
Name	Harsh Gawali
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

MAD and PWA Lab

Name: Harsh Gawali Class: D15A Roll no:19 Experiment - 9

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 tra c of the page, manage push notifications and develop “o ine 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 tra c 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 di erent 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 the same (Static content is requested.), this is called “cacheFirst” but if you request a targeted external URL, this is called “networkFirst”.

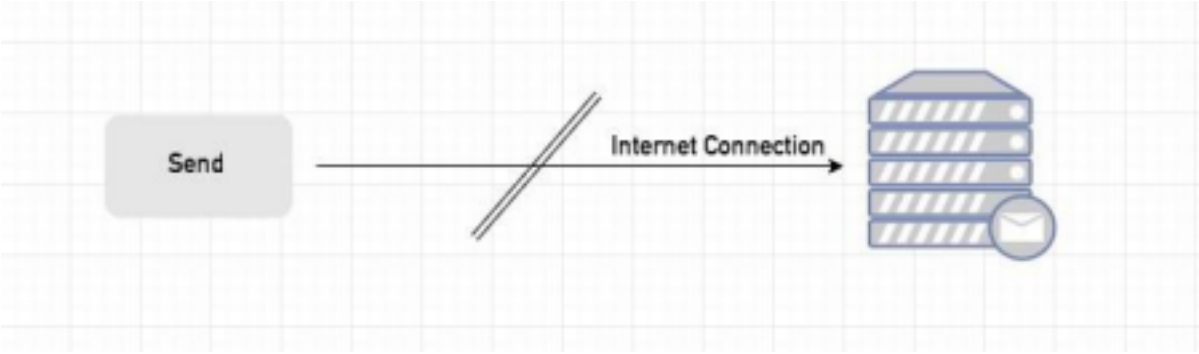
- **CacheFirst** - In this function, if the received request has cached before, the cached response is returned to the page. But if not, a new response requested from the network.
- **NetworkFirst** - In this function, firstly we can try getting an updated response from the network, if this process completed 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.

Sync Event

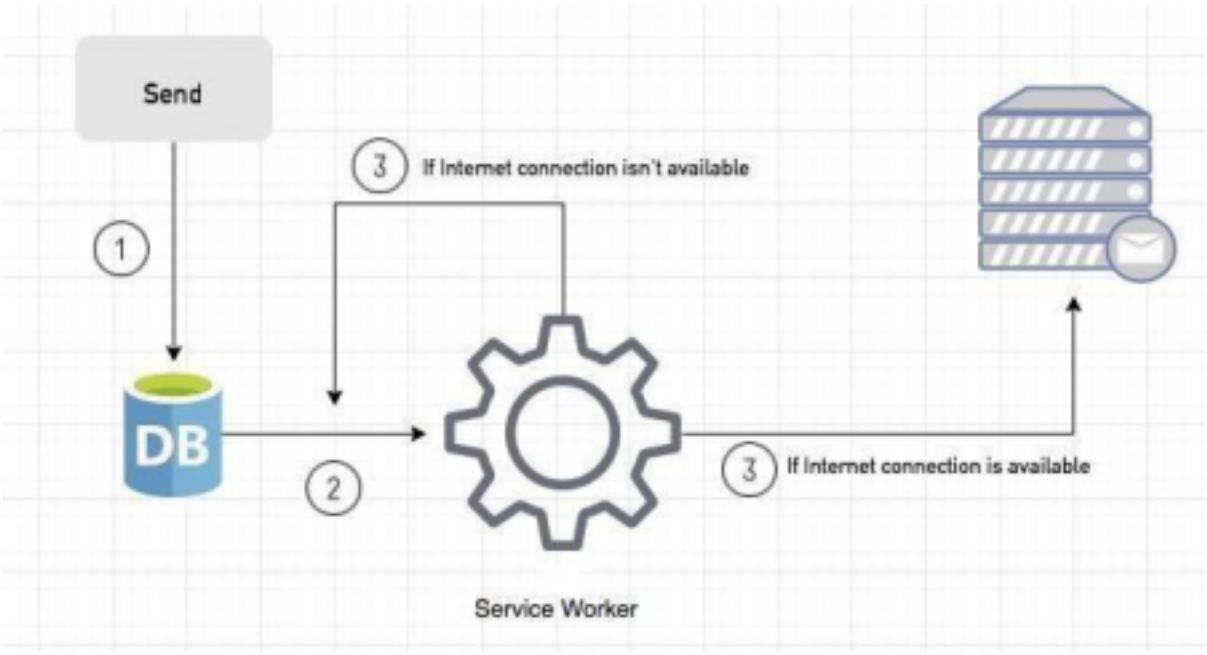
Background Sync is a Web API that is used to delay a process until the Internet connection is stable. We can adapt this definition to the real world; there is an e-mail client application that works on the browser and we want to send an email with this tool. Internet connection is broken while we are writing e-mail content and we didn’t realize it. When completing the writing, we click the send button.

Here is a job for the Background Sync.

The following view shows the classical process of sending email to us. If the Internet Connection is broken, we can’t send any content to Mail Server.



Here, you can create any scenario for yourself. A sample is in the following for this case.



1. When we click the “send” button, email content will be saved to IndexedDB.
2. Background Sync registration.
3. **If the Internet connection is available**, all email content will be read and sent to Mail Server. **If the Internet connection is unavailable**, the service worker waits until the connection is available even though the window is closed. When it is available, email content will be sent to Mail Server.

   You can see the working process within the following code block.

Push Event

This is the event that handles push notifications that are received from the server. You can apply any method with received data.

We can check in the following example.
“Notification.requestPermission();” is the necessary line to show notification to the user. If you don’t want to show any notification, you don’t need this line.

In the following code block is in sw.js file. You can handle push notifications with this event. In this example, I kept it simple. We send an object that has “method” and “message” properties. If the method value is “pushMessage”, we open the information notification with the “message” property

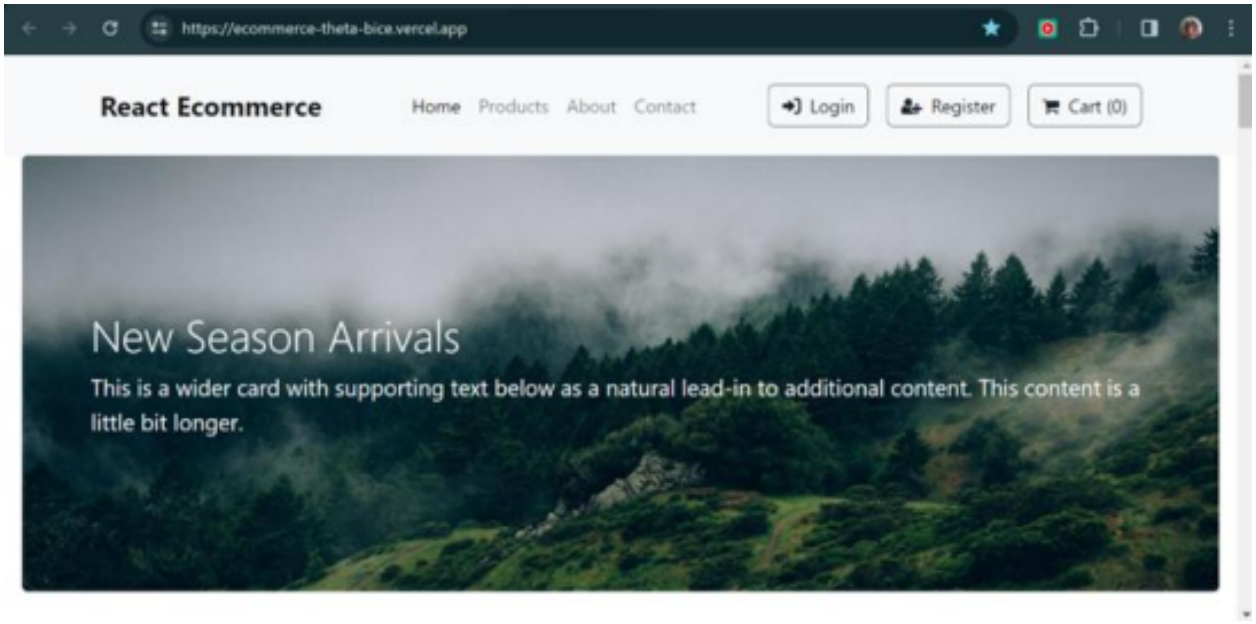
Code:

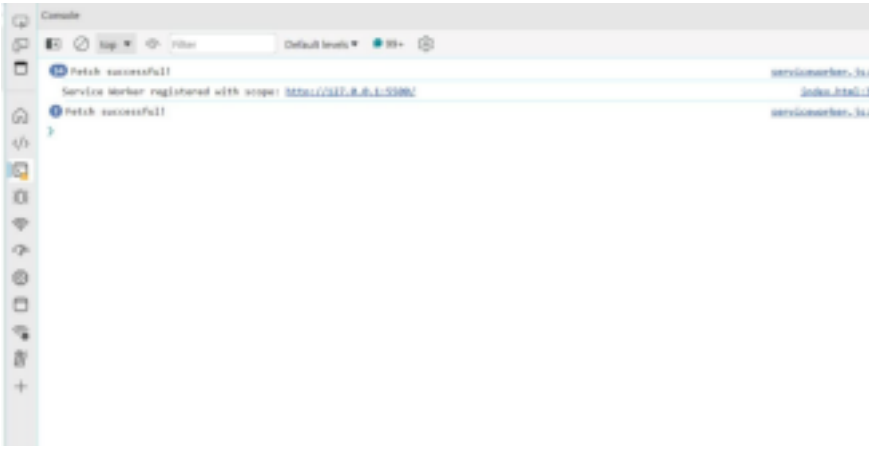
```
//serviceworker self.addEventListener("install",
function (event) {
 event.waitUntil(preLoad());
}); self.addEventListener("fetch", function
(event) {
 event.respondWith(
  checkResponse(event.request).catch(function () {
   console.log("Fetch from cache
  successful!"); return
  returnFromCache(event.request); })
 );
 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.showNotificati
   on("Ecommerce  website", {
      body: data.message,
     });
   }
  } catch (error) {
   console.error("Error
   parsing  push data:", error);
  }
 }
});
var preLoad = function () {
 return
  caches.open("offline").then(funct
  ion (cache)  { // caching index
  and important routes return
  cache.addAll([
```

```
      "/",
      "/index.html",
      "/about.html",
      "/blog.html",
      "/contact.html",
      "/services.html",
      "/img/slider-img4.jpg",
      "/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;
      });
    });
  });
};
```

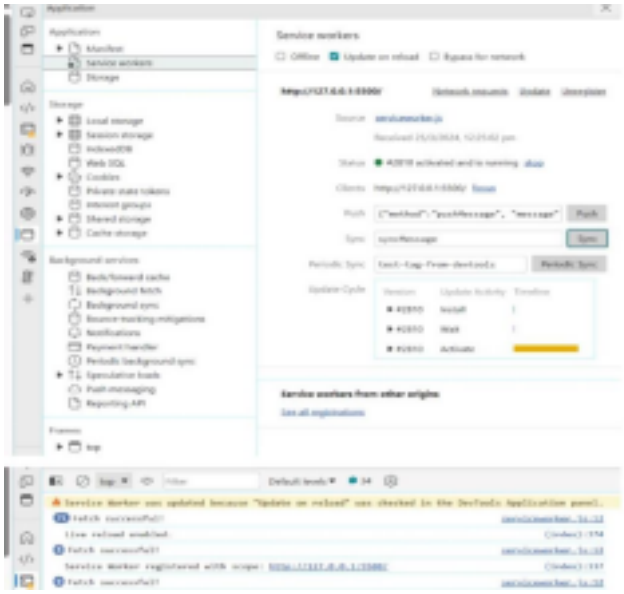
Output: Fetch:



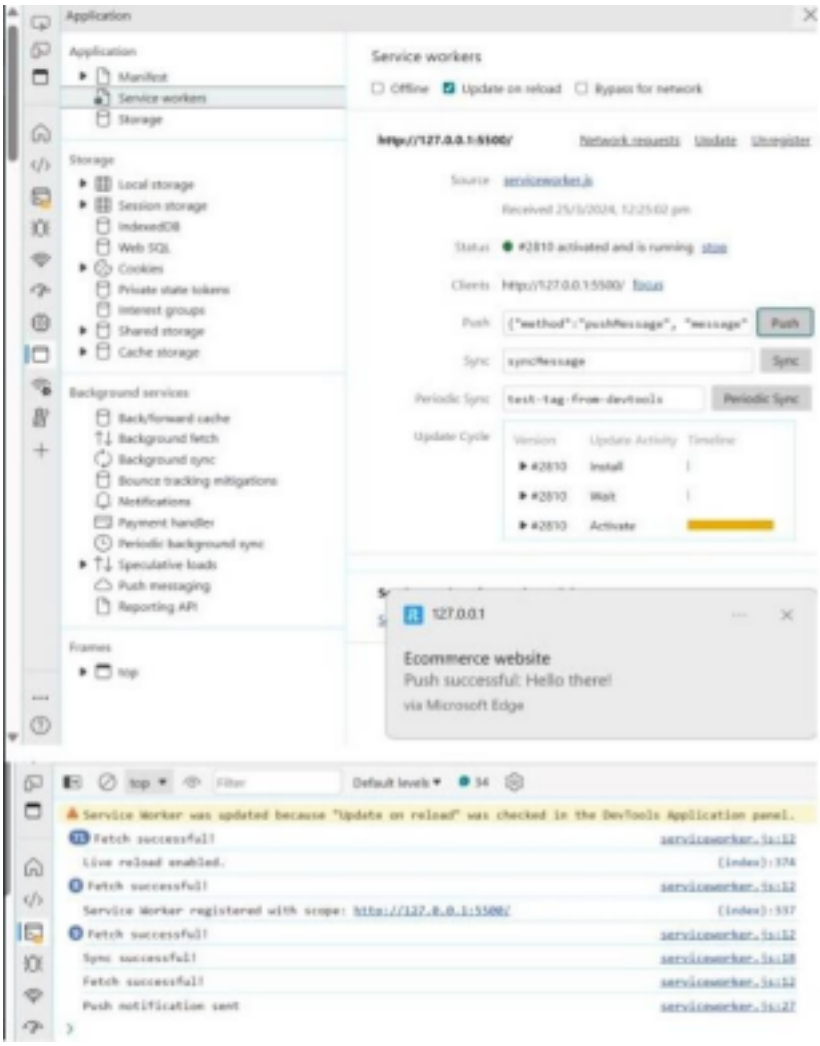


SYNC:

Project Title: Flutter-Signal Clone / PWA- React Ecommerce **Roll No. 19**



PUSH:



Conclusion :

Project Title: Flutter-Signal Clone / PWA- React Ecommerce Roll No. 19

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.	19
Name	Harsh Gawali
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

MAD and PWA Lab

Name: Harsh Gawali Class: D15A Roll no:19

Experiment - 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 directly from 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 & 4651 developers stacks

Pros

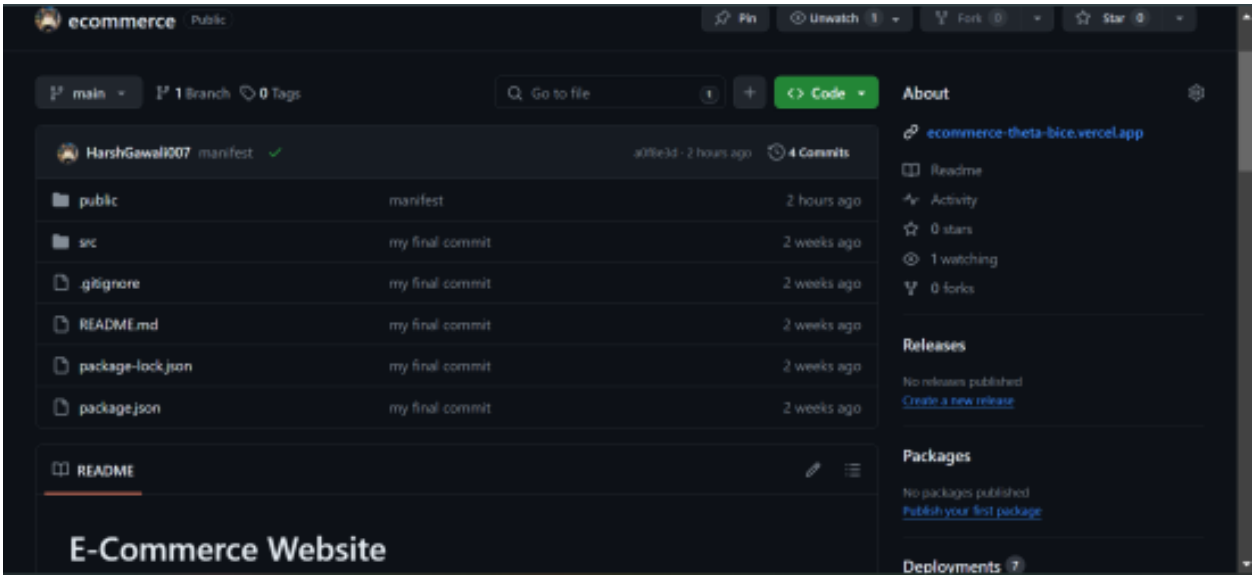
1. Hosted by Google. Enough said.
2. Authentication, Cloud Messaging, and a whole lot of other handy services will be available to you.

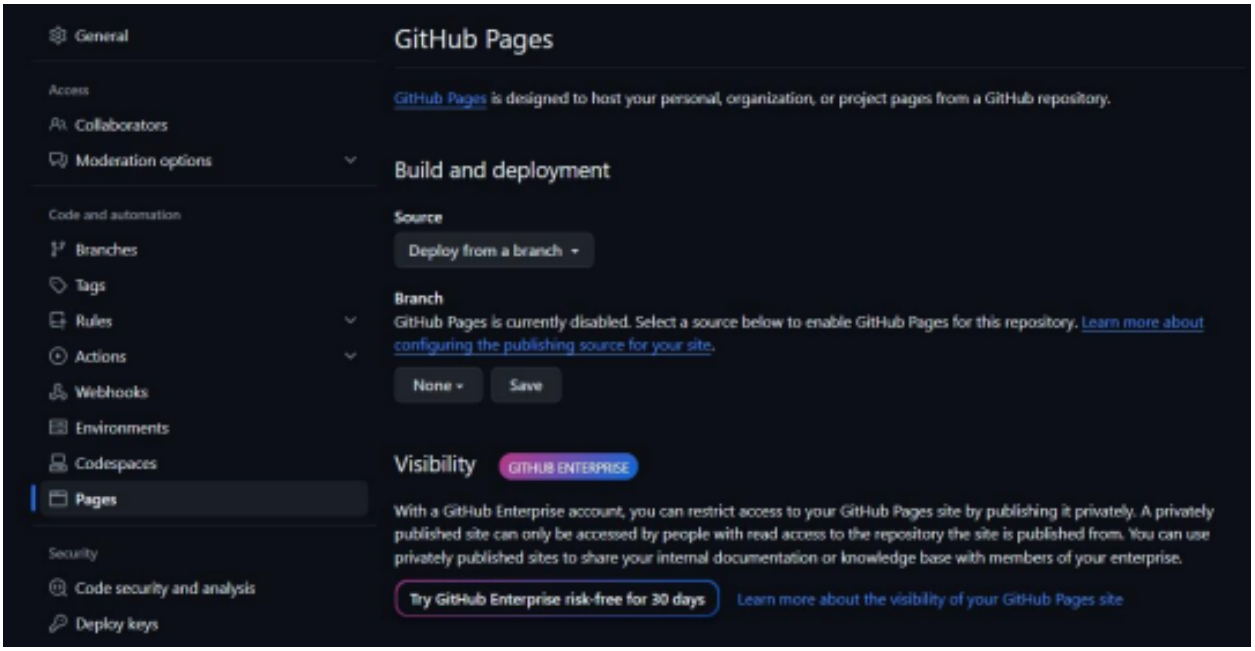
3. A real-time database will be available to you, which can store 1 GB of data.
4. You'll also have access to a blob store, which can store another 1 GB of data. 5. Support for HTTPS. A free certificate will be provisioned for your custom domain within 24 hours.

Cons

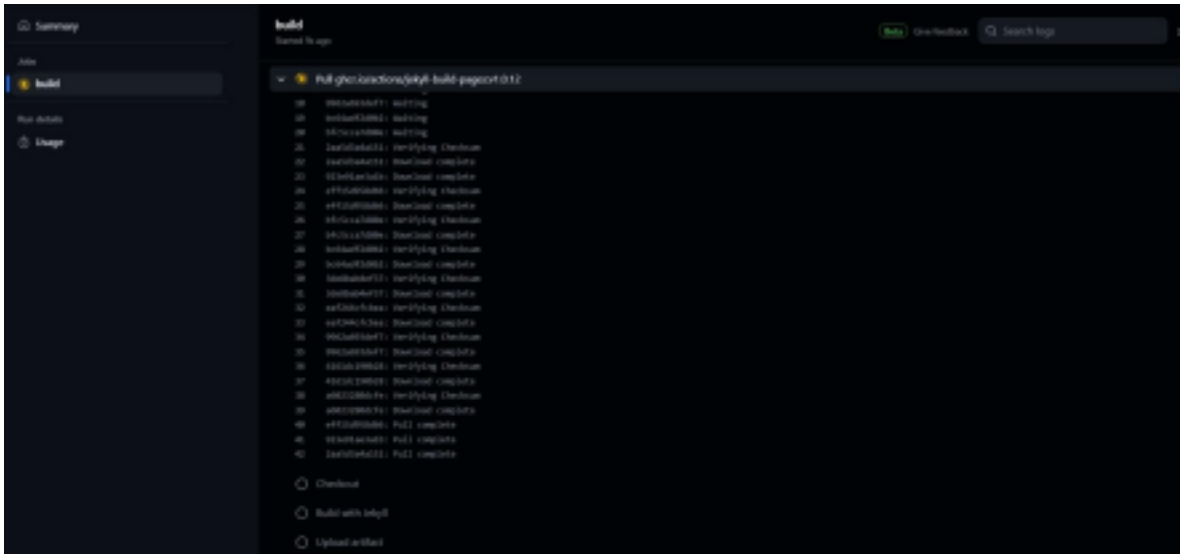
1. Only 10 GB of data transfer is allowed per month. But this is not really a big problem, if you use a CDN or AMP.
2. Command-line interface only.

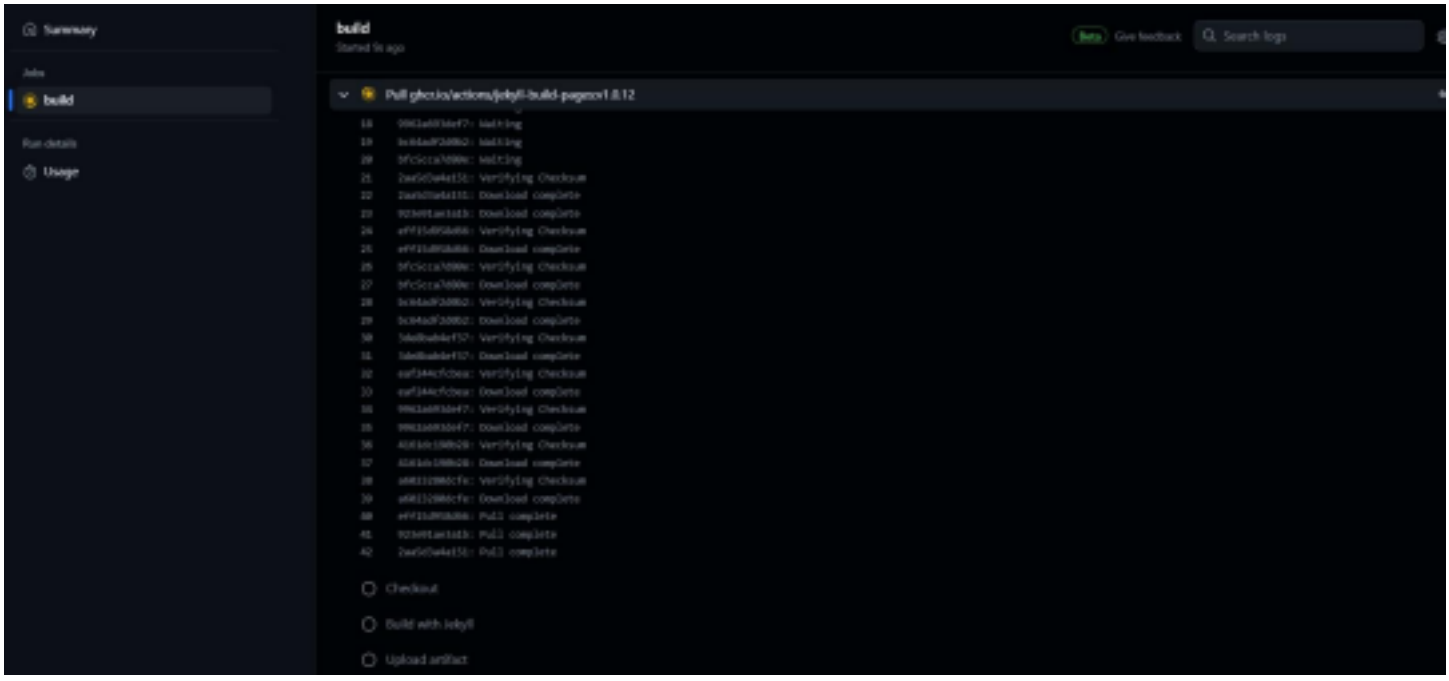
Implementation:





Project Title: Flutter-Signal Clone / PWA- React Ecommerce **Roll No. 19**





Link to our GitHub repository: https://github.com/HarshGawali007/ecommerce

Conclusion:

In this experiment, we have registered a service worker, and completed the install and activation process for a new service worker for the E-commerce PWA.

MAD & PWA Lab
Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	19
Name	Harsh Gawali
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	

MAD and PWA Lab

Name: Harsh Gawali Class: D15A Roll no:19

Experiment - 11

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

Theory:

Google Lighthouse

Lighthouse is an open-source, automated tool for improving the quality of web pages. You can run it against any web page, public or requiring authentication. It has audits for performance, accessibility, progressive web apps, SEO and more.

You can run Lighthouse in Chrome DevTools, from the command line, or as a Node module. You give Lighthouse a URL to audit, it runs a series of audits against the page, and then it generates a report on how well the page did. From there, use the failing audits as indicators on how to improve the page. Each audit has a reference doc explaining why the audit is important, as well as how to fix it.

Features of Lighthouse

Google Lighthouse gives a breakdown of your site into the accompanying metrics. Here is a brief explanationof each of the aforementioned metrics:

1. Performance

Performance is generally viewed as the most valuable metric given by the Google Lighthouse tool. Like the PageSpeed Insights, the Performance area of the Lighthouse report contains a few helpful metrics you can use to advance your site to climb Google's rankings. The Performance segment of the Lighthouse report joins

the Opportunities, Field Data, Lab Data, and Diagnostics metrics of the PageSpeed Insights tool.

A great example is the opportunities metric as it flags three types of render-blocking URL's namely stylesheets, scripts, and HTML imports. This merged perspective on performance metrics gives an exact and valuable analysis of your site's performance and any progressions you should make to expand your site's exhibition.

2. Accessibility

The first of the new regions of Google Lighthouse is the Accessibility metric. Basically what this metric does is feature potential chances to improve the availability and client experience of your mobile app or website.

Following the accessibility improvement report will guarantee that your clients can without much of a stretch explore and utilize your site. Just as guaranteeing that you have the most obvious opportunity with regards to positioning better on web search engines.

3. Best Practices

Another segment new to Google's analysis tools is the Best Practices metric. This region of the Lighthouse report doesn't carefully give execution related data. However, it will give you recommendations which can improve both your exhibition and client experience, particularly for mobile sites.

4. SEO

The latest and most dynamic of the highlights in Google's Lighthouse instrument is the SEO metric.

PageSpeed Insights doesn't offer this tool. This is why most web designers and SEO specialists prefer to utilize Google Lighthouse to analyze a website. The SEO metric gives fundamental tools to examine your page's streamlining for search engine results rankings. While there are numerous more factors which Lighthouse doesn't consider or quantify, the most essential focuses are secured.

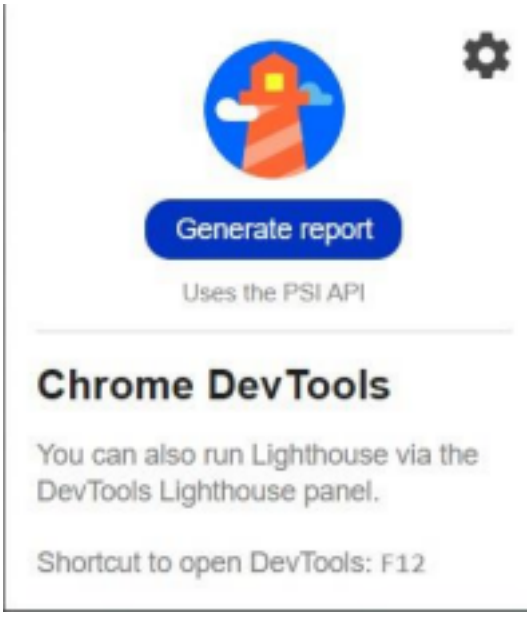
5. Progressive Web Applications

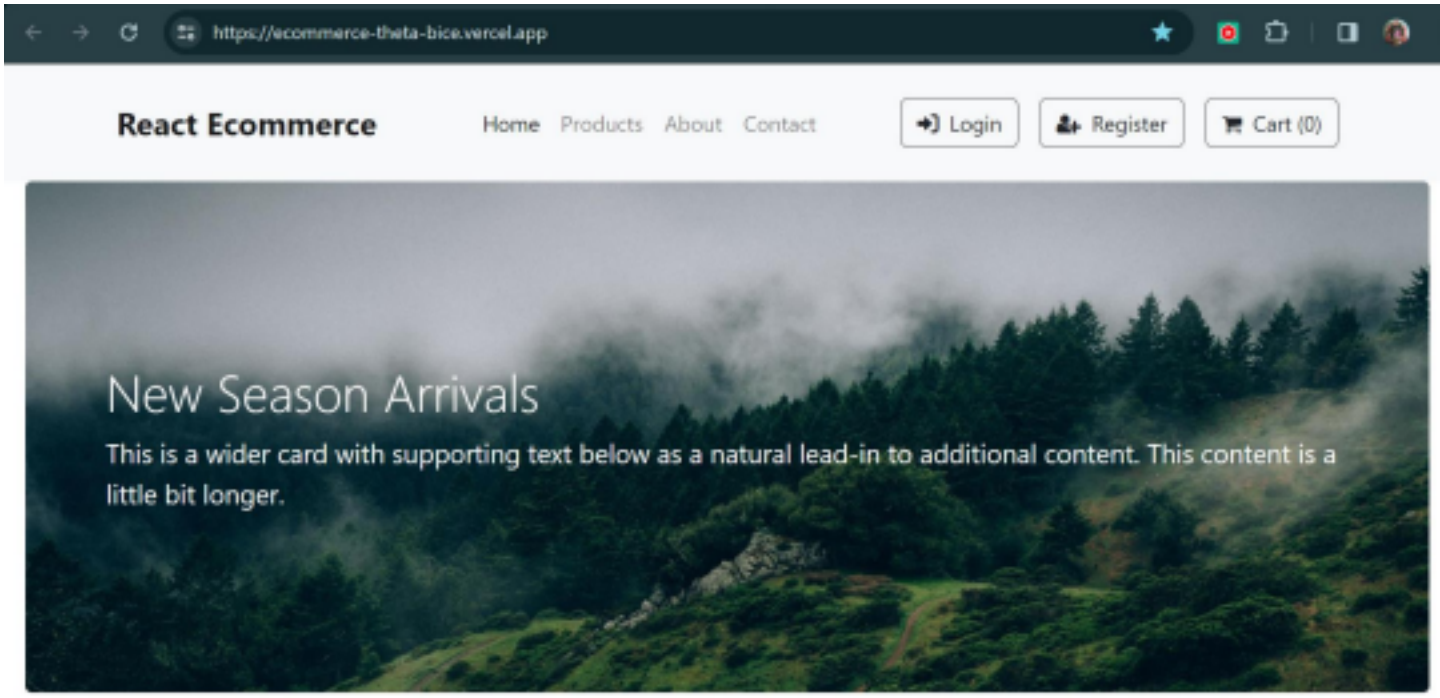
The Progressive Web App area is another of Google's most up to date execution measurements incorporated into its Lighthouse tool. While the meaning of a Progressive Web App (PWA) hasn't been especially clear, Google's documentation expresses that there are a few key variables which make a site a PWA. A great feature of this metric is registering service workers which allow you to enable push notifications on your web app **IMPLEMENTATION:**

Manifest.json :

```
{
 "short_name": "DormDine",
 "name": "DormDine Website",
 "icons": [
   {
    "src": "logo192.png",
    "type": "image/png",
    "sizes": "192x192",
    "purpose": "maskable"
   },
   {
    "src": "logo512.png",
    "type": "image/png",
    "sizes": "512x512"
   }
 ],
 "start_url": ".",
 "display": "standalone",
 "theme_color": "#000000",
 "back
ground
_color"
:
"#ffffff
" }
```

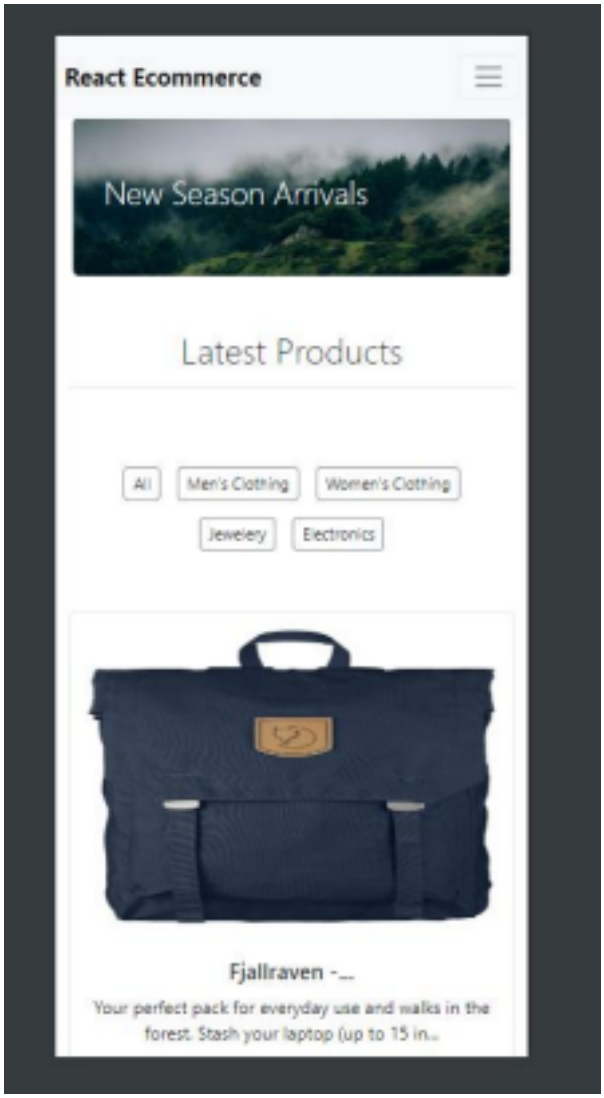
FOR DESKTOP DEVICE





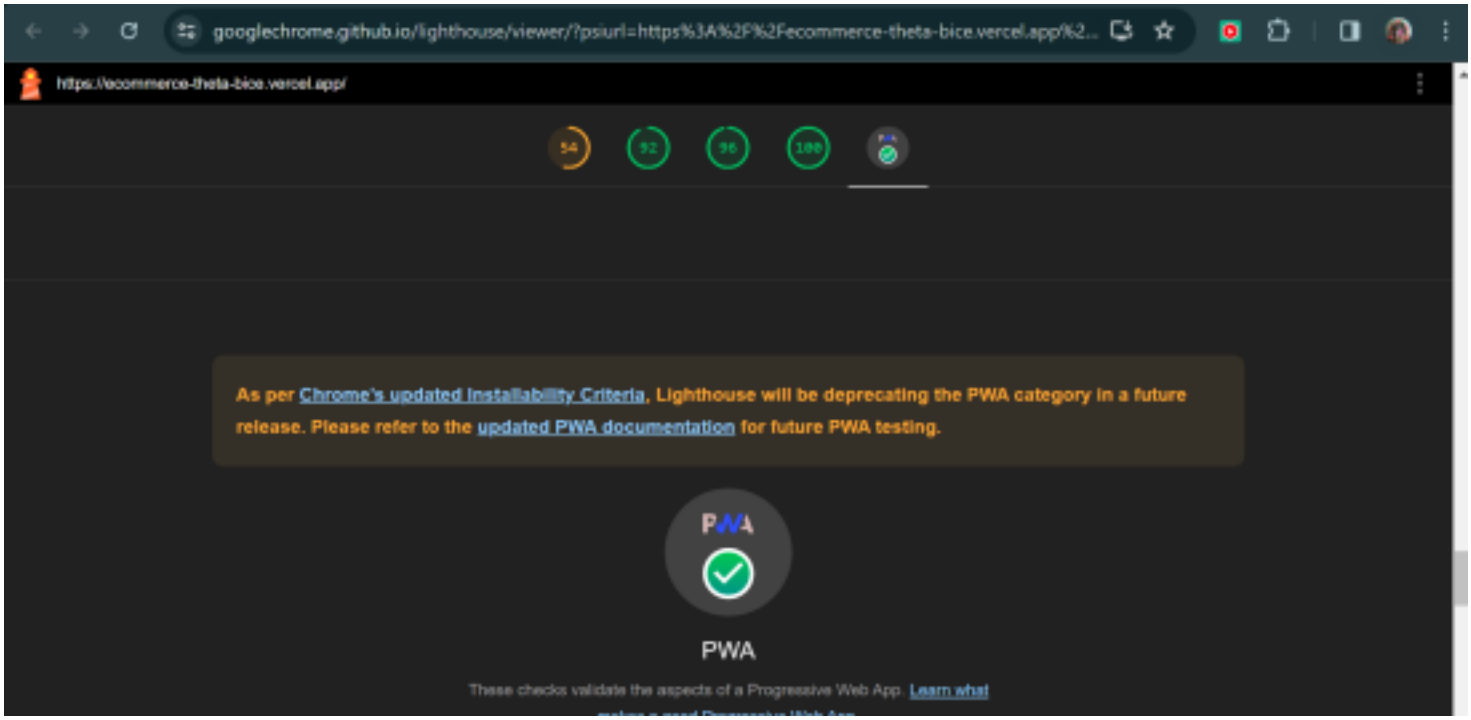
FOR MOBILE DEVICES

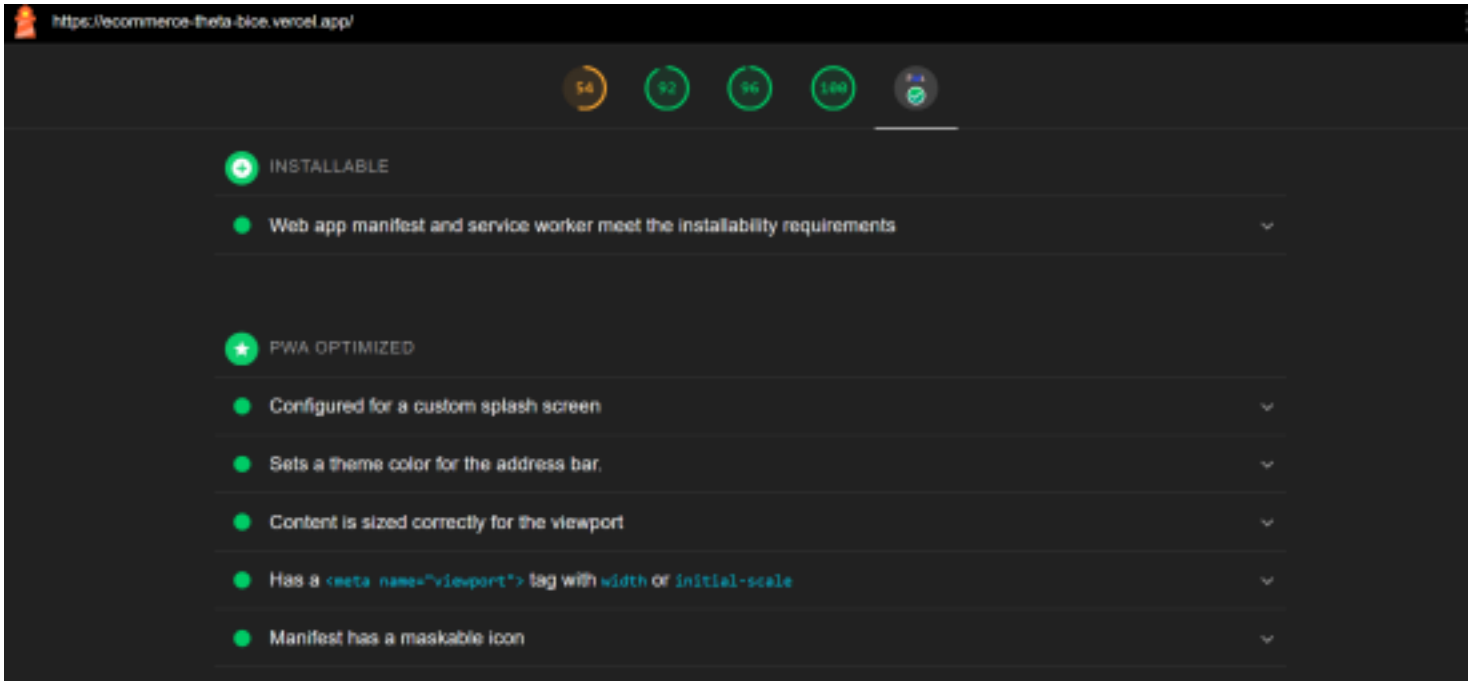
Project Title: Flutter-Signal Clone / PWA- React Ecommerce **Roll No. 19**



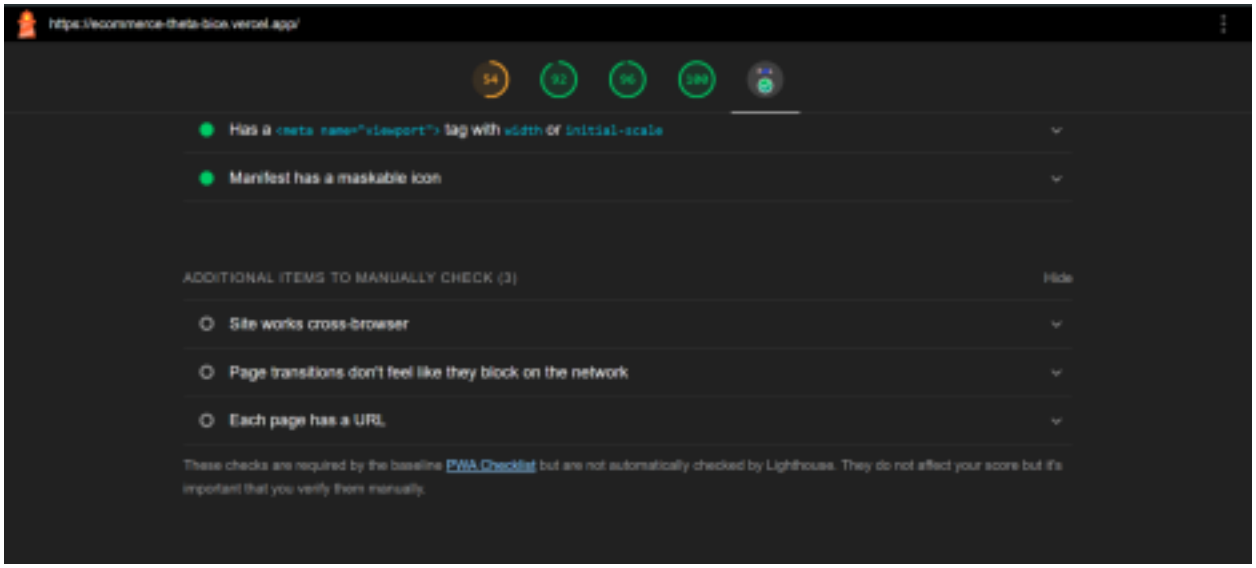
REPORT GENERATED:

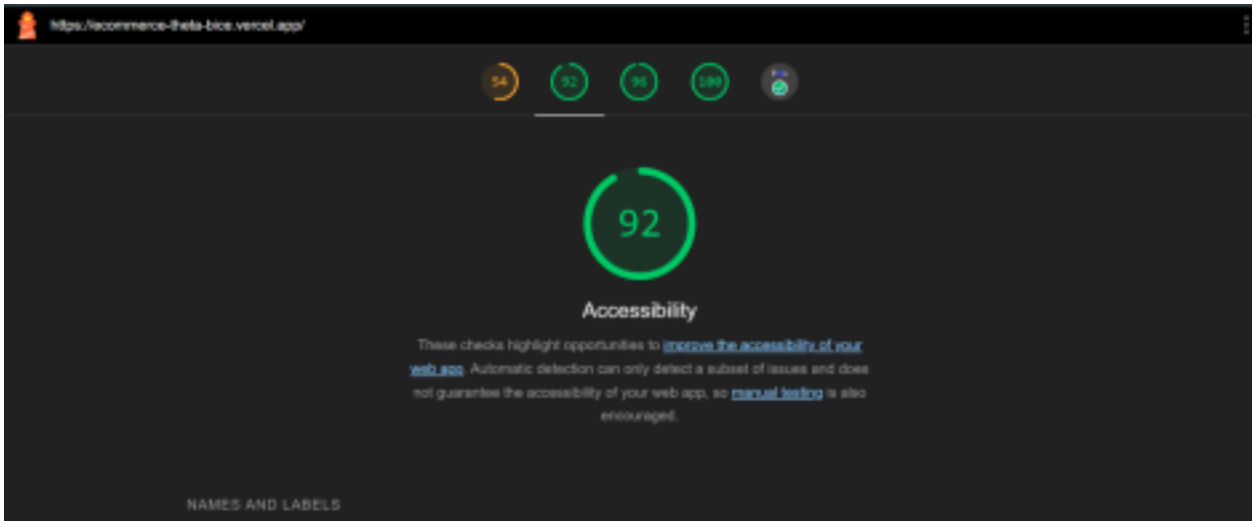
Project Title: Flutter-Signal Clone / PWA- React Ecommerce **Roll No. 19**





Project Title: Flutter-Signal Clone / PWA- React Ecommerce **Roll No. 19**





Conclusion:

In this experiment, we have 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.	19
Name	Harsh Gawali
Class	D15A
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:	

Sarvodnya Awaghad
D15A 04
Page No.
Date :

Assignment 1 flutter

Q i) Flutter Overview:- Explain key features and advantages of using flutter for mobile app development. Discuss how flutter framework differ from traditional approach and why it gained popularity

→ Key features:-

ⓐ Single codebase for multiple platforms:-
Flutter allows developer to write code and deploy it on both ios and android.

ⓑ Hot Reload:-
This enables developers to instantly see results of code changes they make.

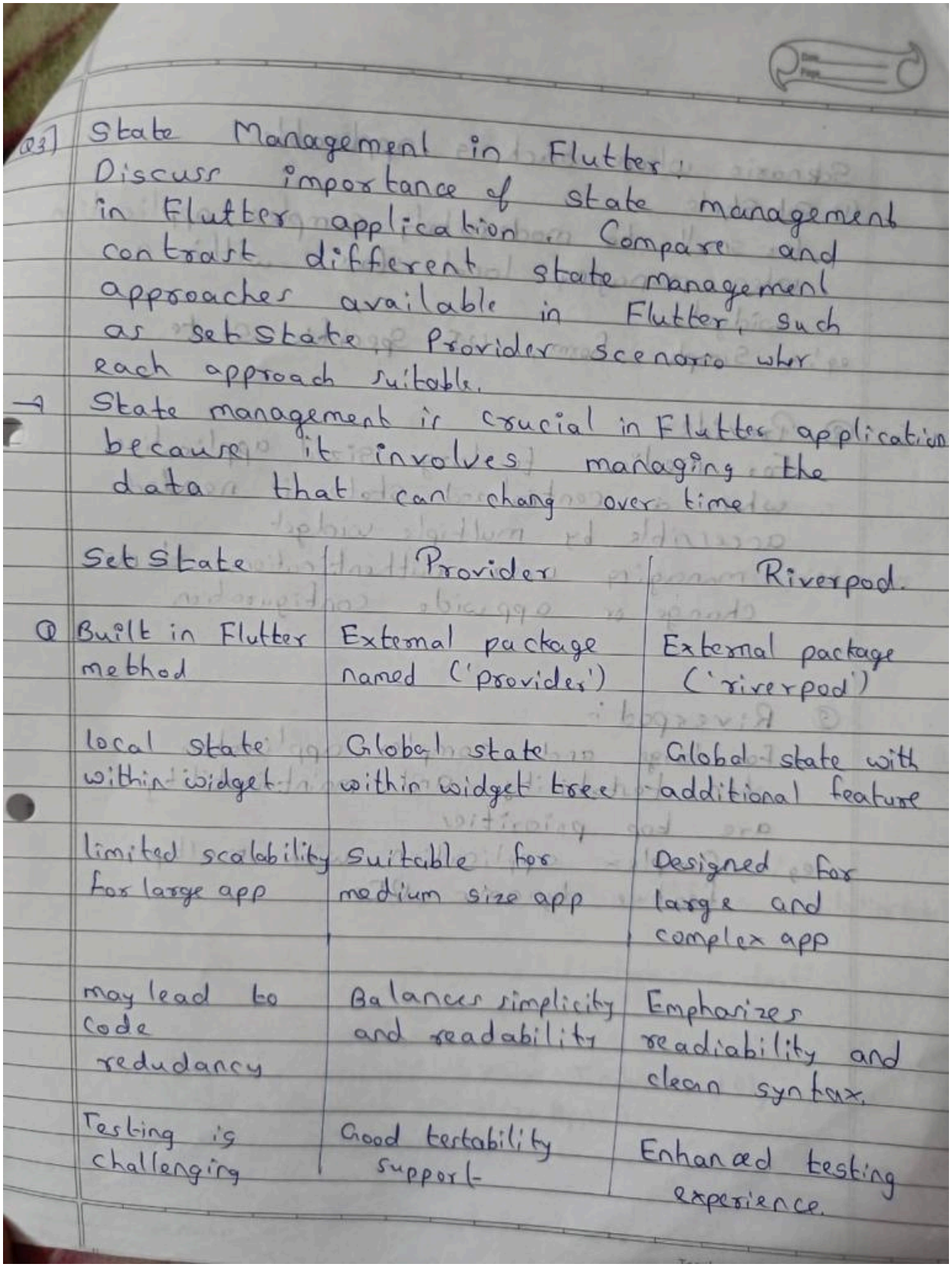
ⓒ Express UI:-
Developers have flexibility to create expressive and flexible UI.

ⓓ Integration with other tool.
Flutter easily integrate with other popular development tool and framework.

2. Advantages:-

ⓐ faster Development
Uses single codebase for multiple platform

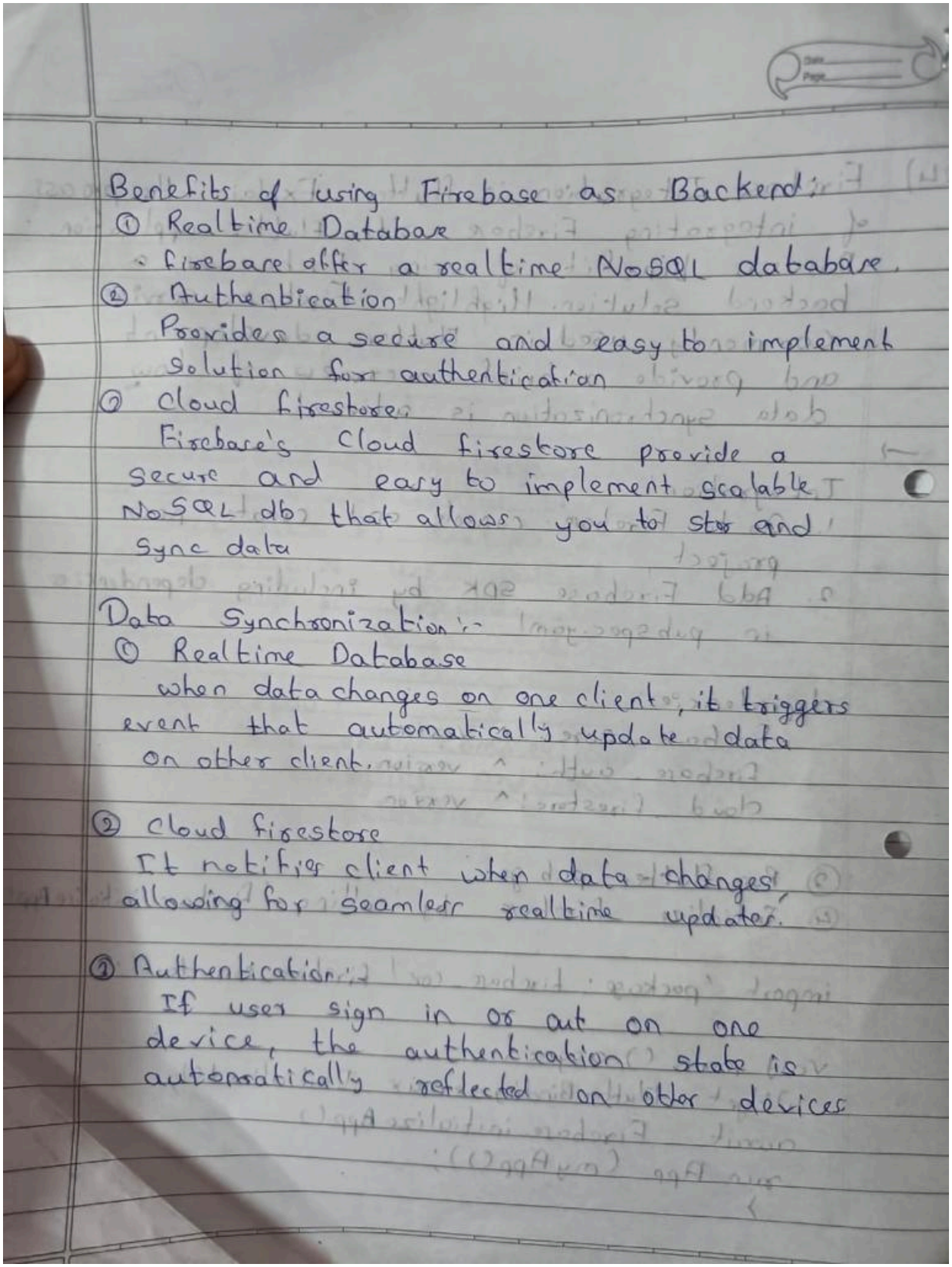
ⓑ Consistent UI
Widget provide a consistent look



Q3] State Management in Flutter:
Discuss importance of state management in Flutter application. Compare and contrast different state management approaches available in Flutter, such as SetState, Provider scenario wher each approach suitable.

→ State management is crucial in Flutter application because it involves managing the data that can chang over time

	SetState	Provider	Riverpod.
①	Built in Flutter method	External package named ('provider')	External package ('riverpod')
	local state within widget	Global state within widget tree	Global state with additional feature
	limited scalability for large app	Suitable for medium size app	Designed for large and complex app
	may lead to code redudancy	Balances simplicity and readability	Emphasizes readiability and clean syntax.
	Testing is challenging	Good testability support	Enhanced testing experience.



Benefits of using Firebase as Backend:

① Realtime Database
- firebase offer a realtime NoSQL database.

② Authentication
Provides a secure and easy to implement solution for authentication

③ Cloud firestore:
Firebase's Cloud firestore provide a secure and easy to implement scalable NoSQL db that allows you to store and sync data

Data Synchronization:-

① Realtime Database
when data changes on one client, it triggers event that automatically update data on other client.

② Cloud firestore
It notifies client when data changes, allowing for seamless realtime updates.

③ Authentication:
If user sign in or out on one device, the authentication state is automatically reflected on other devices

Q4) Firebase Integration in flutter. Explain the process of integrating Firebase with Flutter application. Discuss benefit of using Firebase as a backend solution. Highlight Firebase services commonly used in Flutter development and provide a brief overview of how data synchronization is achieved.

→

Integration:-

1. Go to Firebase console and create new project
2. Add Firebase SDK by including dependencies in pubspec.yaml

```
dependencies:
    firebase_core: ^ version
    firebase_auth: ^ version
    cloud_firestore: ^ version
```

3. Run flutter pub get
4. Initialize firebase by calling 'firebase.initialiseApp()

```
import 'package: firebase_core/ firebase_core.dart';

void main () async {
  Widget FlutterBinding.ensureInitialized();
  await Firebase.initializeApp()
  runApp (myApp());
  }
```

Teacher's Sign.:

Commonly used widget :-

(a) Container :-
A box model for padding, margin and decoration

(b) Column and Row
· layout widget for arranging children vertically or horizontally

(c) Stack
Overlapping widget allowing them to be layered on top of each other.

(d) List View :-
A scrollable grid of widget.

(e) Grid View :-
A scrollable grid of widget.

(f) Text field :-
An input field for user

and feel across different platform

© Cost Efficiency:
Developing and maintaining single codebase for both ios and android reduces development cost.

3) Differ from Traditional approach
@ Traditional approach uses a hierarchical structure for UI component, where as Flutter uses a widget based approach
ⓑ Hot Reload allows to see changes made instantly

Flutter popularity is driven by increased productivity, a growing community, cross platform and adoption by companies

Q2) Widget Tree and Composition: Describe the concept of widget tree in Flutter. Explain how widget composition is used to build complex, user interfaces. Provide eg of commonly used widget & their role in widget tree.

→

Widget Tree:

The widget Tree is a hierarchical structure of widget that defines the user interface

Every visual element, from simple component to complex layout, is represented by widget.

Widget are categorized in 2 types

ⓐ Stateless Widget

It is immutable and cannot change over time

ⓑ Stateful Widget

Widget that can change state over time.

Widget Composition

- Widget composition in Flutter involves combining multiple simple widget to create more complex and compound widget.

Teacher's Sign.: _____

Project Title: Flutter-Signal Clone / PWA- React Ecommerce **Roll No. 19**

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.	19
Name	Harsh Gawali
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:	

HARSH GAWALI
D15A
19.

Date
Page

PWA. Assignment: 2.

1. A Progressive web app is a website that looks and behaves as if it is a mobile app. PWA's are built to take advantage of native mobile device feature. It is built with regular web technologies like html, css & Javascript.

Its significance in modern web development is:

- Reduced development cost :- By using existing web skills and a single codebase. PWA are quicker & cheaper to develop.

- Wider Reach :- It works on any device with a modern browser, eliminating need for app store retrieval.

- Easy updates :- updates to PWA happens automatically in the background ensuring latest version.

- Improved user engagement :- features like push notifications and offline functionality enhance user experience.

	PWA	Tradational Mobile App
Technology	Web technology (HTML, CSS, Javascript)	Native code (Platform specific).
Installation	Installed from browser Add to home prompt	Download from App store.

Teacher's Sign.:

Accessible through search engines — Reliant on app store discovery

Automatic background update — Requires manual update.

2. Responsive web design is a web development approach that ensures a website adjusts its layout and functionality based on the device its being viewed on. This creates an optimal user experience for desktop, tablets, smartphones and any screen size.

Importance of PWA:-

Consistent user experience:-
Users expect a smooth experience regardless of the device they use. RWD ensures the PWA looks & functions well on all screens.

Improved Accessibility:-
RWD makes PWA accessible to a wider audience using devices with varying screen sizes.

Search Engine Optimisation:- Google & other search engines favor websites that offer a good user experience.

While RWD is a umbrella term, there can be confusion with other approaches.

Teacher's Sign.:

Fluid web design : A specific layout within PWD that uses percentages and relative units to define element sizes. This allows elements to expand and contract based on the screen size.

Adaptive web design :- This approach involves multiple fixed width layouts for different device categories. It requires more development work compared

3. A key players in this PWA universe is the service worker. The service worker is a Javascript file that runs on a seperate thread apart from the one usual website.

Three phases of lifecycle.

(a) Registration Phase
(b) Instanation Phase
(c) Activation Phase.

(a) Registration.
The 1st phase in the service worker's lifecycle is registering it to the browser. The registration can be done in 2 ways :-

Teacher's Sign.: _____

- You either specify a scope for a service worker has access to

- You leave it to the default globe scope of where the service worker file is present.

(b) Installation:-

Once the service worker is successfully registered it is not ready to be installed. The service worker script is downloaded to the browser and the browser will attempt to install the service worker.

(c) Activation:-

Once the installation phase is successful, the next phase is the activation phase. The service worker does not move into active state immediately. It can move into the activated state below.

- None of the pages use the service worker or that page.

- There is no other service workers active on that page.

Teacher's Sign.:

4. Indexed DB is a browser API for storing large amounts of structured data on the client side. It functions like a NoSQL database, allowing you to store key value pairs & organise data into objects with indexes for efficient retrieval.

Benefits of using Indexed DB with service workers.

- Offline data access - Improves user experience by allowing interaction with the PWA even without an internet connection.
- Faster load times:- Cached data retrival from indexed can be served much faster than fetching it
- Improved reliability - Reduces dependencies on a constant network connection making PWA more reliable.
- Security - Indexed DB provides a secure storage mechanism with sandbox.