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 **Shamaila Ansari** of **D15A** semester **VI,** have successfully completed necessary experiments in the **MAD & PWA Lab** under my supervision in **VES Institute of Technology** during the academic year **2023-2024.**

Lab Assistant

Subject Teacher

Mrs. Kajal Joseph

Principal

Head of Department

Dr. Mrs. Shalu Chopra

Name of the Course : MAD & PWA Lab

Course Code : ITL604

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

Faculty Incharge : Mrs. Kajal Joseph.

Lab Teachers **:** Mrs. Kajal Jewani.

Email **:** kajal.jewani@ves.ac.in

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

Program specific Outcomes

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

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

Project Title: Flutter-Myntra Clone /PWA- Fuzzy app **RollNo. 02**

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	2/4	15
11.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.	LO6	5/3	12/3	15
12.	Assignment-1	LO1,LO2 ,LO3	2/2	5/2	05
13.	Assignment-2	LO4,LO5 ,LO6	19/3	21/3	04

MAD & PWA Lab
Journal

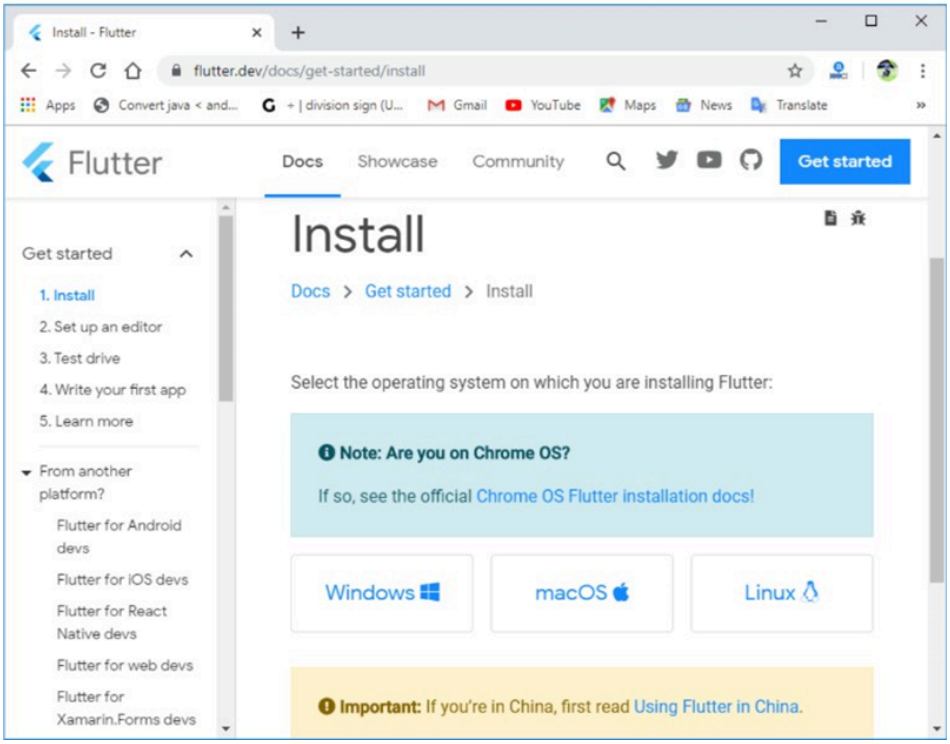
Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	02
Name	Shamaila Ansari
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	

Experiment No:01

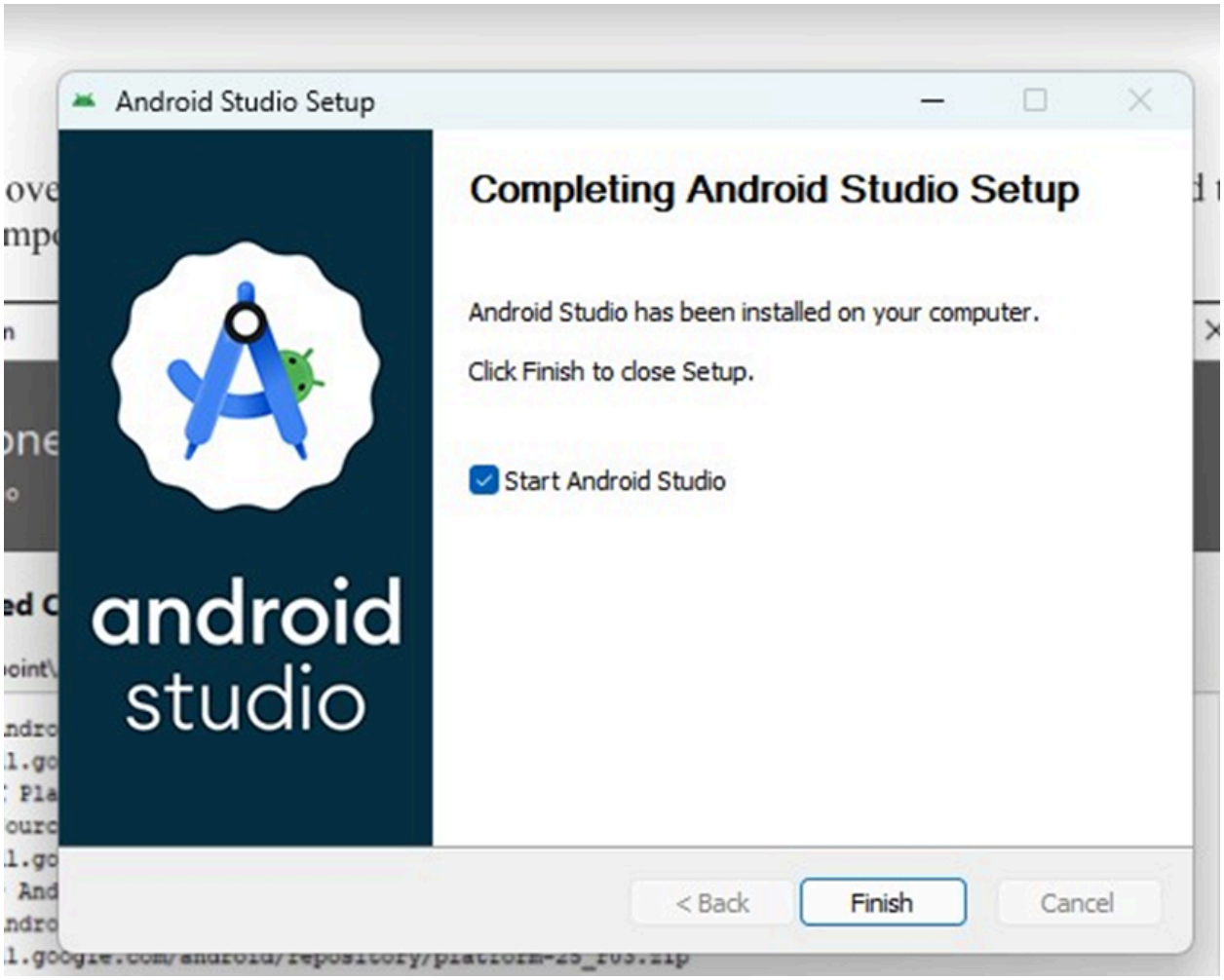
Aim: To install and configure the Flutter Environment

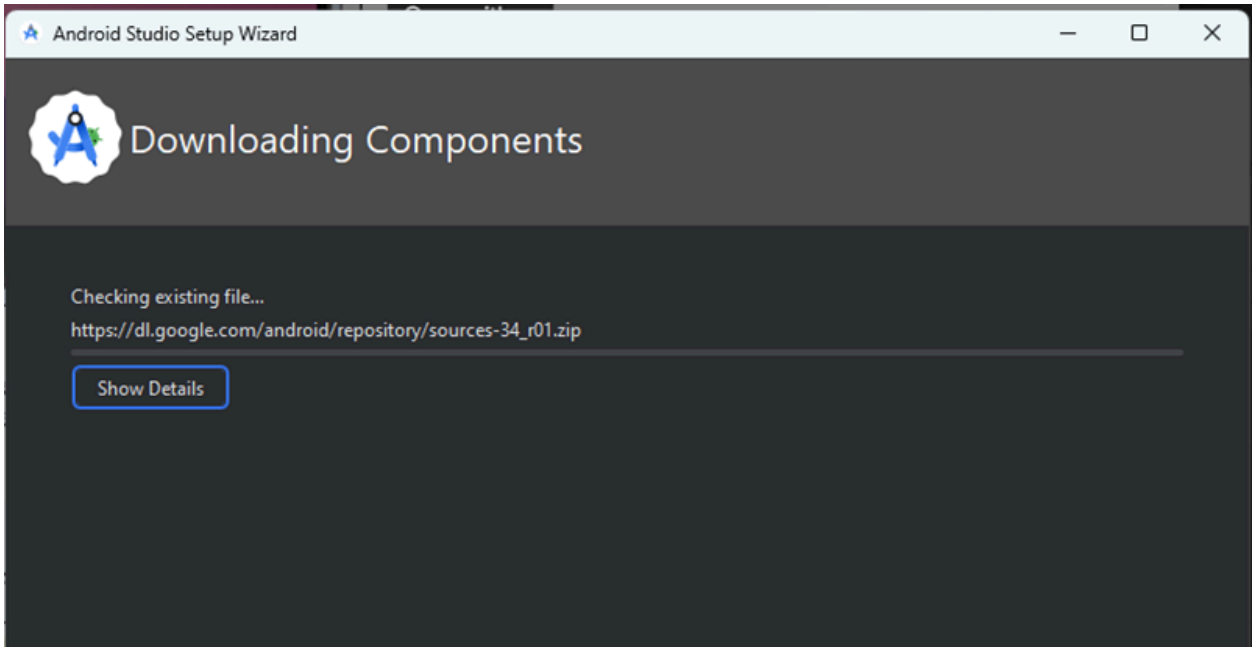
Install the Flutter SDK

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

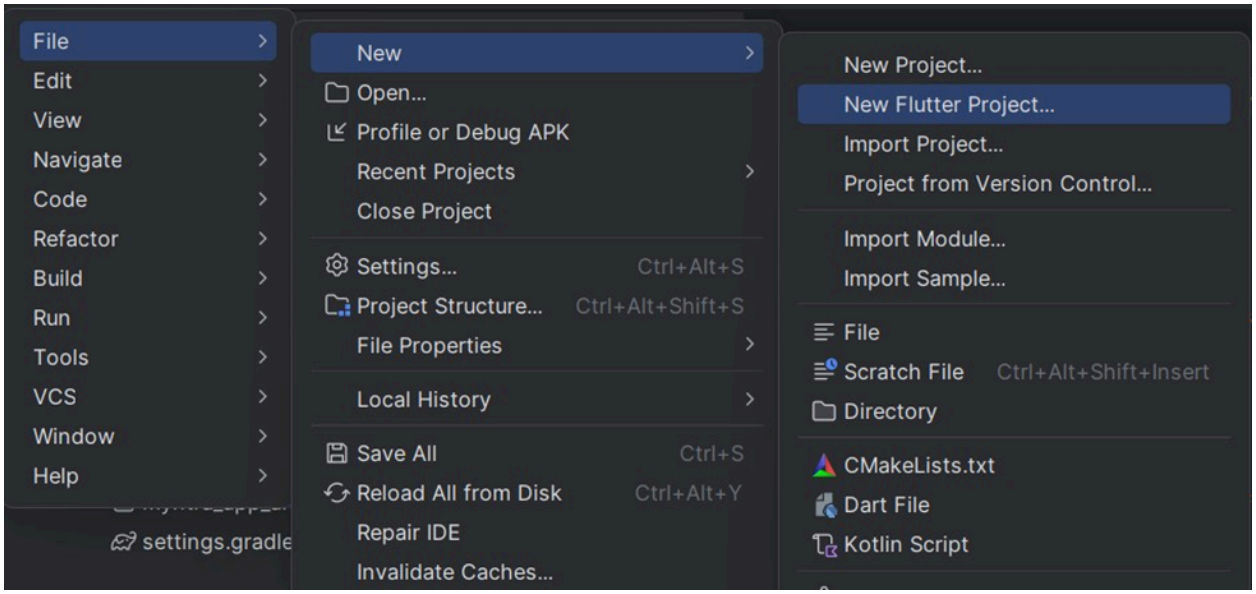


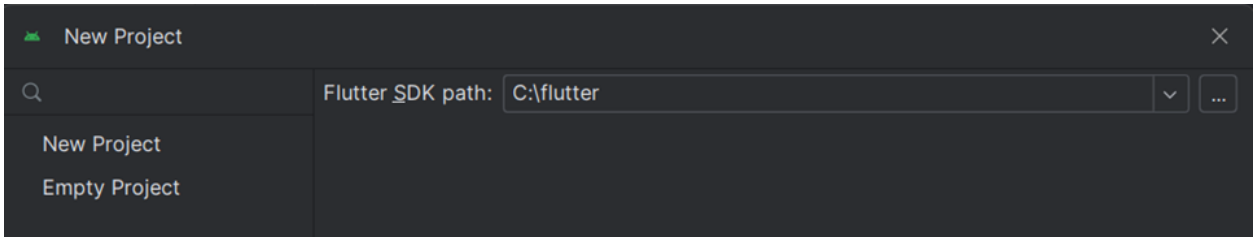
```
C:\Users\ansar>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.16.7, on Microsoft Windows [Version 10.0.22621.3007], locale en-IN)
[√] Windows Version (Installed version of Windows is version 10 or higher)
[√] Android toolchain - develop for Android devices (Android SDK version 34.0.0)
[√] Chrome - develop for the web
[X] Visual Studio - develop Windows apps
    X Visual Studio not installed; this is necessary to develop Windows apps.
      Download at https://visualstudio.microsoft.com/downloads/.
      Please install the "Desktop development with C++" workload, including all of its default components
[√] Android Studio (version 2023.1)
[√] VS Code (version 1.85.1)
[√] Connected device (3 available)
[√] Network resources

! Doctor found issues in 1 category.
```

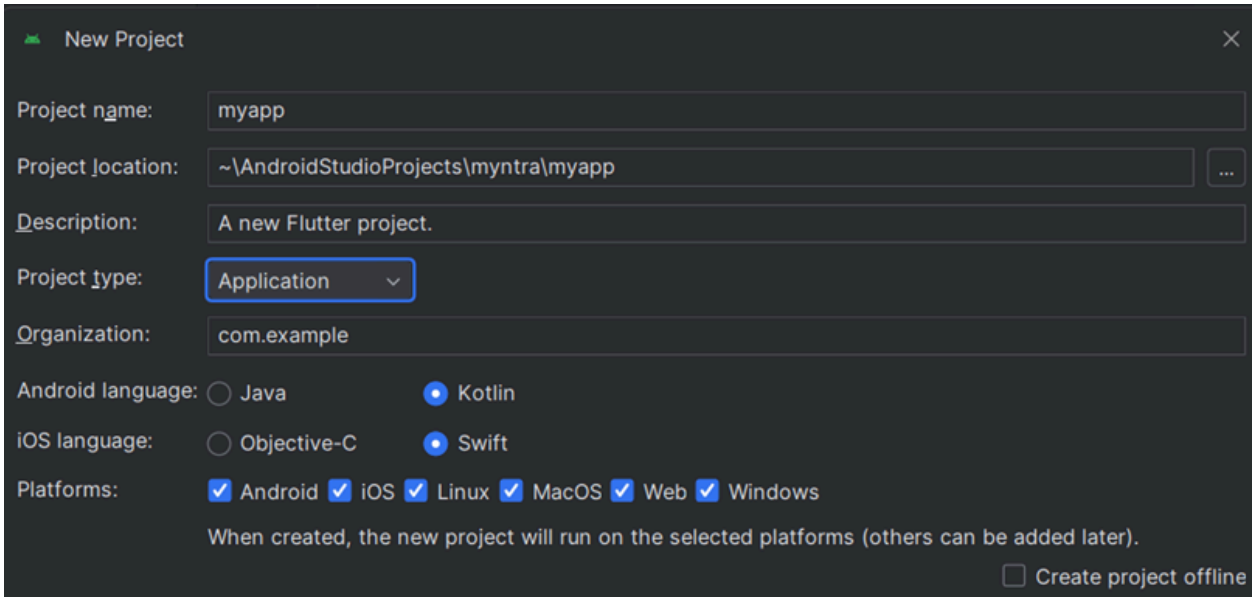

Android Studio Setup
Completing Android Studio Setup
Android Studio has been installed on your computer.
Click Finish to close Setup.
Start Android Studio
android studio
< Back
Finish
Cancel


Android Studio Setup Wizard
Downloading Components
Checking existing file...
https://dl.google.com/android/repository/sources-34_r01.zip
Show Details

Project Title: Flutter-Myntra Clone /PWA- Fuzzy app **RollNo. 02**



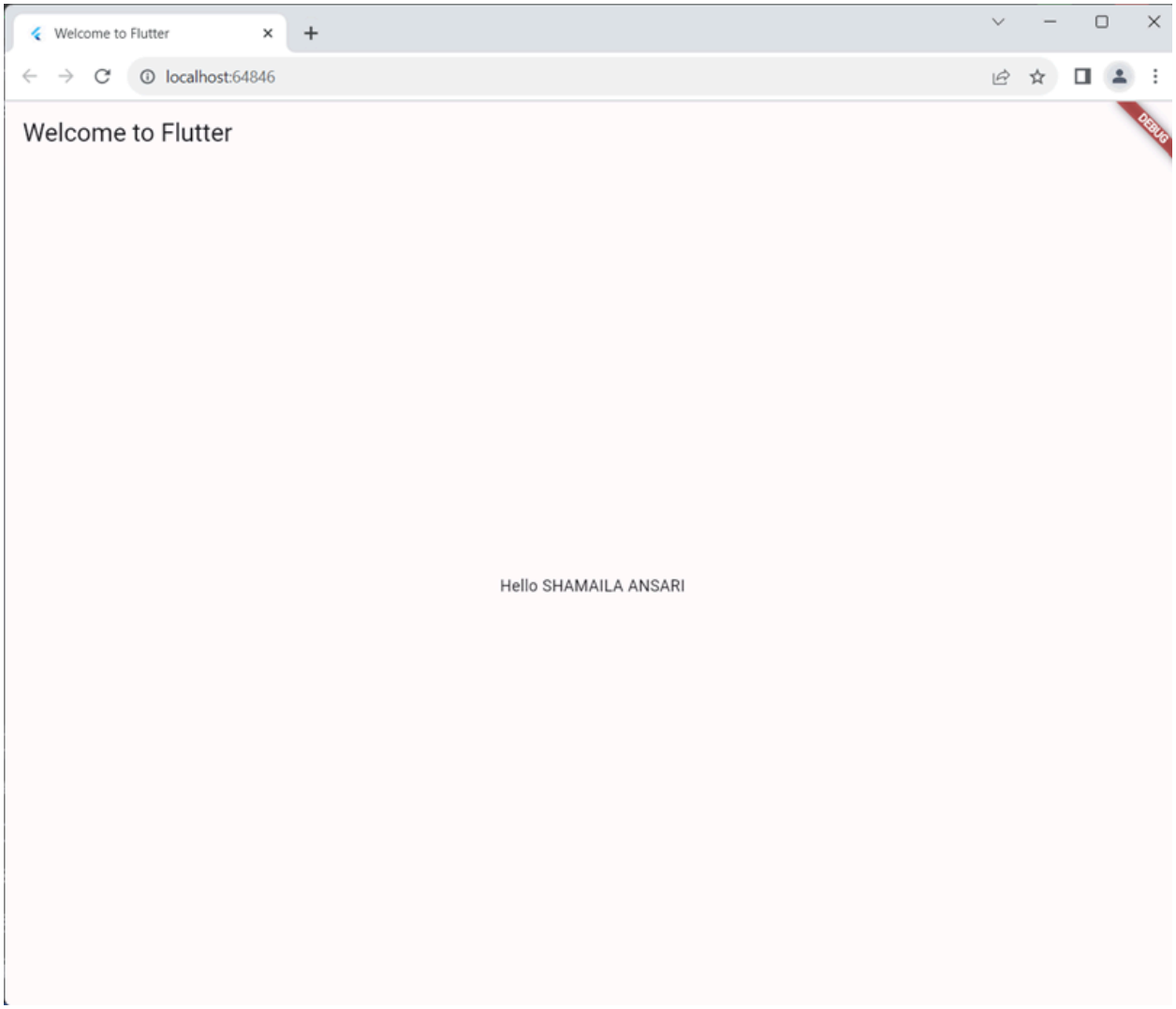


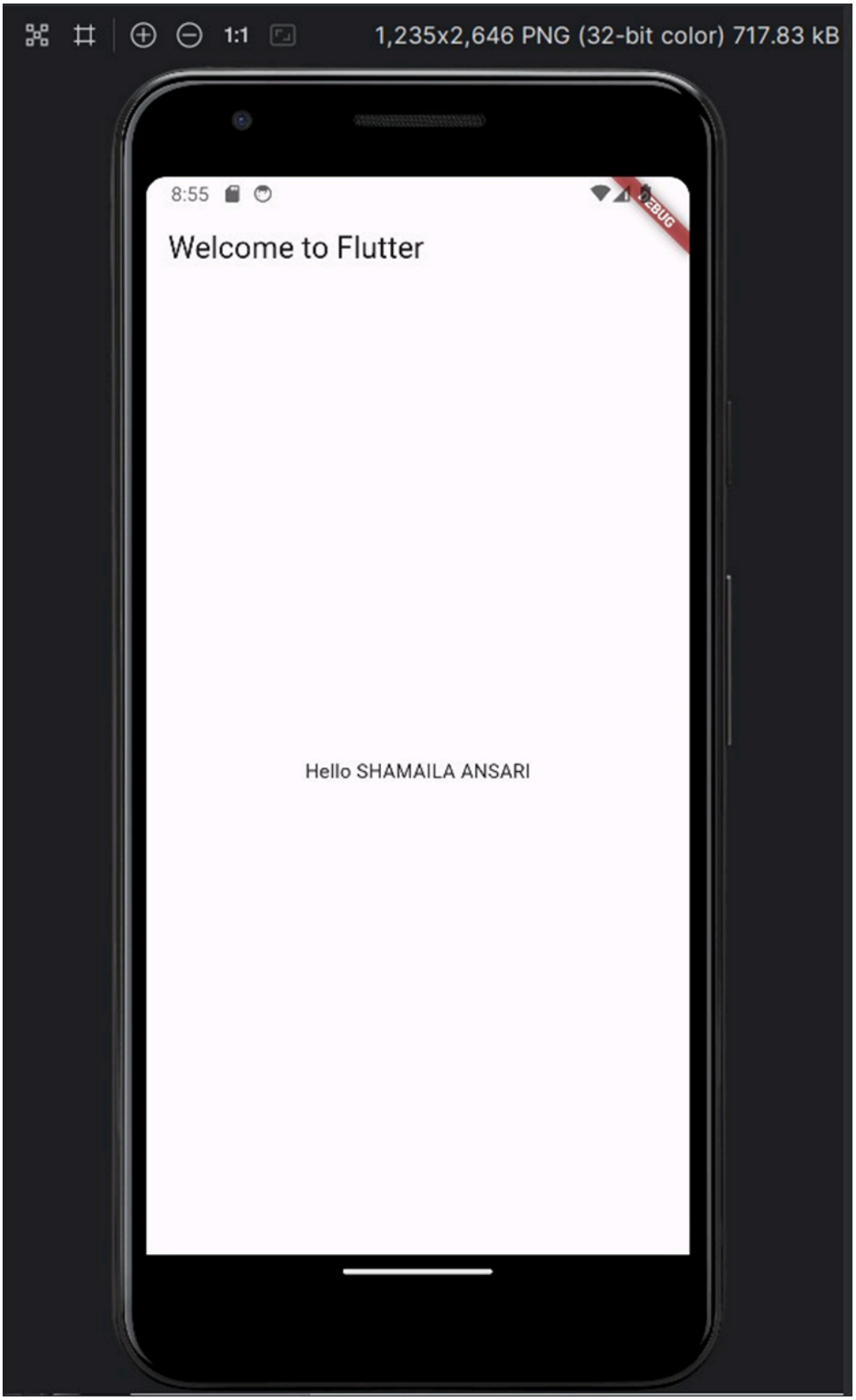


README.md main.dart

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

Project Title: Flutter-Myntra Clone /PWA- Fuzzy app **RollNo. 02**





Conclusion : Here we have successfully installed the Flutter app

MAD & PWA Lab
Journal

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

EXPERIMENT NO: 02

Aim :To design Flutter UI by including common widgets.

Theory :

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

```
//FlatButton
Example new
FlatButton(
 child: Text("Click
 here"), onPressed: () {
 // Do something here
 },
),
```

```
//RaisedButton
Example new
RaisedButton(
 child: Text("Click
 here"), elevation: 5.0,
 onPressed: () {
```

```
// Do something here


},
```

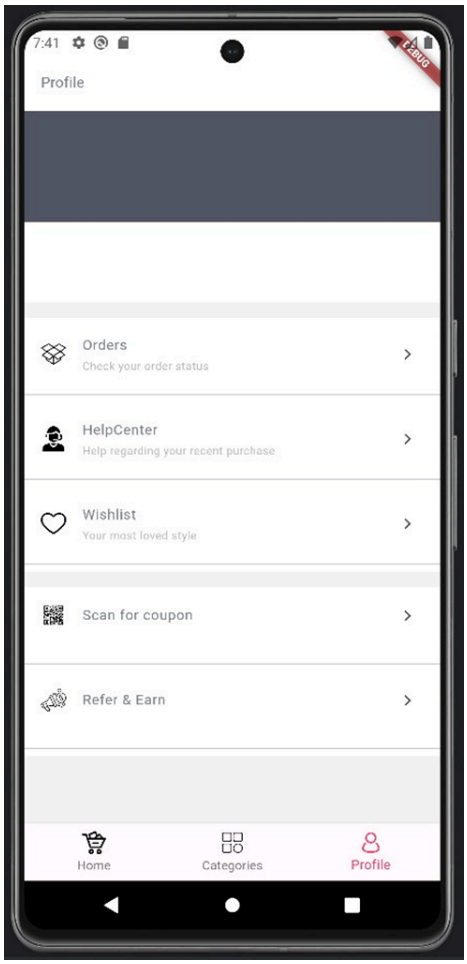
CODE:

```
import 'package:flutter/material.dart';
import 'package:myntra/foundation/profile_item/profile_item.dart';
import 'package:myntra/foundation/theme/colors.dart';


class ProfileWithoutLogin extends StatelessWidget {
 const ProfileWithoutLogin({Key? key}) : super(key:
 key); final double topContainerheight = 190;

@override

Widget build(BuildContext
 context){ return Column(
   children
    : [
    Contai
    ner(
    height:
    topContainerheight,
    child: Column(
        children
          : [
          Contai
          ner(
          height: topContainerheight* .58,
          color: AppColor.dummyBGColor,

        ),

        Container(
          height: topContainerheight* .42,
          color: AppColor.whiteColor,

        ),
```

```
    ],

  ),
),
  const
    SizedBox(
    height: 15,

),

  Container(
    color:AppColor.whiteColor,
    child: Column(
      children:
        const [
        ProfileItem(
        title:
        "Orders",
        subtitle: "Check your order
        status", assetName: "orders.png",

        isLast: false,

      ),
      ProfileItem(
        title: "HelpCenter",
        subtitle: "Help regarding your recent purchase",

        assetName:
        "help-desk.png", isLast:
        false,

      ),

        ProfileItem(
          title:
          "Wishlist",
        subtitle: "Your most loved
        style", assetName:
        "wishlist.png", isLast: false,

      ),



    ],
```

```
        )),
        const
          SizedBox(
          height: 15,

      ),

          Container(
            color:AppColor.whiteColo
            r, child: Column(
            children: const [
              ProfileItem(
              assetName:
              "qr.png",
              title: "Scan for coupon",
              isLast: false,

              ),

            ProfileItem(
              title: "Refer & Earn",
              assetName:
              "refer.png", isLast:
              false,

              ),

          ],
        )),
    ],
  );
}
```

Output:



Conclusion: Here, We have Successfully implemented a design Flutter UI by including common widgets

MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	02
Name	Shamaila Ansari
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

EXPERIMENT NO: 03

Aim: To Include icons,image and fonts in flutter app

Theory :

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:

o **Image:** It is a generic image loader, which is used by ImageProvider.

o **asset:** It load image from your project asset folder.
o **file:** It loads images from the system folder.
o **memory:** It load image from memory.
o **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.**

assets:

- assets/comp.jpg

CODE:

```
import
'package:flutter/material.dart';
void main() {

 runApp(const MyApp());

}
class MyApp extends StatelessWidget {
 const MyApp({Key? key}) : super(key:
 key); @override
```

```
  Widget build(BuildContext
   context) { return MaterialApp(
   title: 'Welcome to
   Flutter', home:
   Scaffold(

     appBar: AppBar(

       title: const Text('Welcome to Flutter'),
     ),
     body: Center(
       child: Image.asset('assets/comp.png'),
     ),
    ),
  );
 }
}
```

Pubspec.yml

```
assets:
 - assets/comp.png
```

OUTPUT: Device**(Emulator)**

Conclusion: Here, learned about images,fonts and icons in flutter app

MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	02
Name	Shamaila Ansari
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

EXPERIMENT NO: 04

Aim: To create an interactive Form using form widget

Theory :

Creating forms in Flutter involves several key steps, and understanding the theory behind it can help you build robust and user-friendly interfaces. Let's break it down into key components:

1. **State Management**:

· Forms often require managing the state of input fields. In Flutter, you can manage state using StatefulWidget or state management solutions like Provider, Bloc, Riverpod, etc.

· State management ensures that as users interact with the form (typing, selecting options, etc.), the UI updates to reflect the changes.

2. **Input Widgets**:

· Flutter provides various input widgets like TextField, TextFormField, DropdownButton, Checkbox, Radio, etc., to collect different types of user input.

· Each input widget has properties to customize its appearance, behavior, validation, and error handling.

3. **Validation**:

· Validating user input is crucial for data integrity and user experience. Flutter provides built-in validators like required, minLength, maxLength, email, etc., but you can also define custom validators.

· Validation ensures that users enter the correct format of data before submitting the form.

4. **Handling User Input**:

· As users interact with the form, you need to handle their input. You can do this by listening to events like onChanged, onSubmitted, onTap, etc., depending on the input widget.

· Update the state of the form based on user input to reflect changes in the UI.

5. **Submission**:

· After users fill out the form correctly, they usually submit it to perform some action (e.g., saving data, sending a request to a server, etc.).

· You can use buttons like RaisedButton, ElevatedButton, or even InkWell to handle form submission.

6. **Layout**:

· Proper layout and organization of form elements enhance usability. You can use Flutter's layout widgets like Column, Row, ListView, etc., along with containers like Padding, SizedBox, etc., to structure your form.

· Group related fields together, use labels and hints effectively, and consider the accessibility aspect of your form layout.

7. **Accessibility**:

· Make your form accessible to all users, including those with disabilities. Use semantic labels, provide textual alternatives to non-text elements, ensure proper tab navigation, etc.

· Accessibility features not only improve user experience but also help your app comply with accessibility standards and regulations.

CODE:

LoginScreen

```
import 'package:emart_app/consts/consts.dart'; import 'package:emart_app/consts/lists.dart';
import 'package:emart_app/views/auth_screen/signup_screen.dart'; import
'package:emart_app/widgets_common/applogo_widget.dart'; import
'package:emart_app/widgets_common/bg_widget.dart'; import
'package:emart_app/widgets_common/custom_textfield.dart'; import
'package:emart_app/widgets_common/our_button.dart'; import 'package:get/get.dart';
class LoginScreen extends StatelessWidget { const LoginScreen({super.key});
@override
Widget build(BuildContext context) { return bgWidget(
child:Scaffold( resizeToAvoidBottomInset: false,
body: Center( child: Column( children: [
(context.screenHeight*0.1).heightBox, applogoWidget(),
10.heightBox,
"Log in to $appname".text.fontFamily(bold).white.size(18).make(),
15.heightBox, Column( children: [
customTextField(hint:emailHint,title:email), customTextField(hint:passwordHint,title:password),
Align(
```

```
alignment: Alignment.centerRight,
child: TextButton(onPressed: () {}, child: forgetPass.text.make())), 5.heightBox,
//ourButton().box.width(context.screenWidth -50).make(), ourButton(color: buttonColor,title:
login,textColor: whiteColor,onPress:
(){}).box.width(context.screenWidth -50).make(), 5.heightBox,
createNewAccount.text.color(fontGrey).make(), 5.heightBox,
ourButton(color: bodyColor2,title: signup,textColor: whiteColor,onPress:

(){

Get.to(()=>SignupScreen());
}).box.width(context.screenWidth -50).make(),
10.heightBox, loginWith.text.color(fontGrey).make(), 5.heightBox,
Row(
mainAxisAlignment: MainAxisAlignment.center, children: List.generate(3, (index) =>
Padding(
padding: const EdgeInsets.all(8.0), child: CircleAvatar(
backgroundColor: lightGrey, radius: 25,
child: Image.asset(socialIconList[index], width: 30,),
),

)),
)
],
).box.white.rounded.padding(const EdgeInsets.all(16)). width(context.screenWidth -70).
shadowSm. make(),
],
),
),
));
}
}
```

Sign up Screen

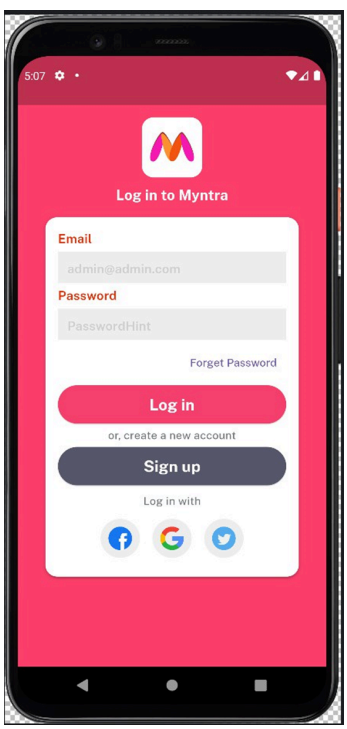
```
import 'package:emart_app/consts/consts.dart'; import 'package:emart_app/consts/lists.dart';
import 'package:emart_app/widgets_common/applogo_widget.dart'; import
'package:emart_app/widgets_common/bg_widget.dart'; import
'package:emart_app/widgets_common/custom_textfield.dart'; import
'package:emart_app/widgets_common/our_button.dart'; import 'package:get/get.dart';

class SignupScreen extends StatelessWidget { const SignupScreen({super.key});

@override @override
Widget build(BuildContext context) { return bgWidget(
```

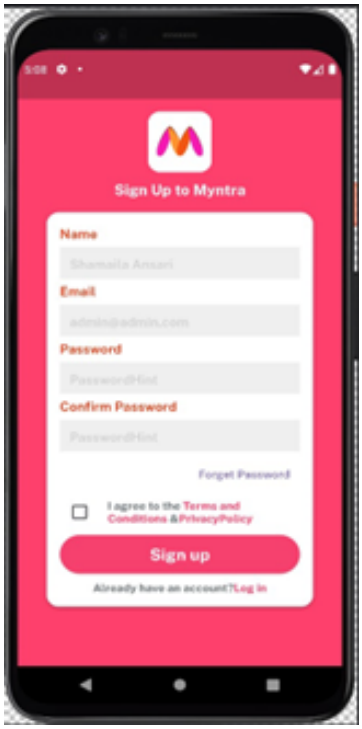
```
child:Scaffold( resizeToAvoidBottomInset: false, body: Center(
child: Column( children: [
(context.screenHeight*0.1).heightBox, applogoWidget(),
10.heightBox,
"Sign Up to $appname".text.fontFamily(bold).white.size(18).make(),
15.heightBox, Column( children: [
customTextField(hint:nameHint,title:name), customTextField(hint:emailHint,title:email),
customTextField(hint:passwordHint,title:password),
customTextField(hint:passwordHint,title:confirmPassword), Align(
alignment: Alignment.centerRight,
child: TextButton(onPressed: () {}, child: forgetPass.text.make())), Row(
children: [ Checkbox(
checkColor: buttonColor, value: false,

onChanged: (newValue) {},
),
10.widthBox, Expanded(
child: RichText(text: const TextSpan( children: [
TextSpan(
text: "I agree to the ", style: TextStyle( fontFamily: bold, color: fontGrey,
)
),
TextSpan(
text: termsAndCond, style: TextStyle( fontFamily: bold, color: buttonColor,
)
),
TextSpan( text: " &",
style: TextStyle( fontFamily: bold, color: fontGrey,
)
),
TextSpan(
text: privacyPolicy, style: TextStyle( fontFamily: bold, color: buttonColor,
)
),
]
)),
)

],
),
5.heightBox,
//ourButton().box.width(context.screenWidth -50).make(), ourButton(color: buttonColor,title:
signup,textColor:
```

```
whiteColor,onPress: (){}).box.width(context.screenWidth -50).make(),
10.heightBox, RichText(text: TextSpan( children: [
TextSpan(
text: alreadyHaveAccount,
style: TextStyle(fontFamily: bold,color: fontGrey),

),
TextSpan( text: login,
style: TextStyle(fontFamily: bold,color:buttonColor),

)
]
),
).onTap(() { Get.back();
}),


],
).box.white.rounded.padding(const EdgeInsets.all(16)). width(context.screenWidth -70).
shadowSm. make(),
],
),
),
));
}
}
```

OUTPUT: Device(Emulator)

LoginScreen **Signup Screen**





Conclusion: By understanding and implementing these theoretical concepts, you can create robust and user-friendly forms in Flutter that provide a seamless experience for your users.

MAD & PWA Lab

Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	02
Name	Shamaila Ansari
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

EXPERIMENT NO: 05

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

Theory :

Navigation and routing are some of the core concepts of all mobile application, which allows the user to move between different pages. We know that every mobile application contains several screens for displaying different types of information. For example, an app can have a screen that contains various products. When the user taps on that product, immediately it will display detailed information about that product.

In Flutter, the screens and pages are known as routes, and these routes are just a widget. In Android, a route is similar to an Activity, whereas, in iOS, it is equivalent to a ViewController.

In any mobile app, navigating to different pages defines the workflow of the application, and the way to handle the navigation is known as routing. Flutter provides a basic routing class MaterialPageRoute and two methods Navigator.push() and Navigator.pop() that shows how to navigate between two routes. The following steps are required to start navigation in your application.

Flutter provides a complete system for navigating between screens and handling deep links. Small applications without complex deep linking can use Navigator, while apps with specific deep linking and navigation requirements should also use the Router to correctly handle deep links on Android and iOS, and to stay in sync with the address bar when the app is running on the web.

CODE:

Home Screen

```
import 'package:myntra_clone/consts/consts.dart';
 import 'package:myntra_clone/consts/lists.dart';
 import 'package:myntra_clone/widgets_common/home_buttons.dart';

 class HomeScreen extends StatelessWidget {
   const HomeScreen({Key? key}): super(key: key);
```

```
@override
Widget build(BuildContext context) {
    return Container(
    padding: const EdgeInsets.all(12),
    color: lightGrey,
    width: context.screenWidth,
    height: context.screenHeight,
    child: SafeArea(
    child:Column(
     children: [
    Container(
            alignment: Alignment.center,
            height: 60,
            color: lightGrey,
            child: TextFormField(
            decoration: InputDecoration(
            border: InputBorder.none,
            suffixIcon: Icon(Icons.search),
            filled:true,
            fillColor: whiteColor,
            hintText: searchanything,
            hintStyle: TextStyle(color: textfieldGrey),
            ),
            ),
    ),
    10.heightBox,
    Expanded(
            child: SingleChildScrollView(
            physics: const BouncingScrollPhysics() ,
            child: Column(
        children:[
            //Swiper brands
            VxSwiper.builder(
            aspectRatio: 16/9,
            autoPlay: true,
            height: 150,
            enlargeCenterPage: true ,
            itemCount:slidersList.length,
            itemBuilder: (context,index){
            return  Image.asset(
            slidersList[index],
            fit: BoxFit.fill,
            ).box.rounded.clip(Clip.antiAlias).margin(const EdgeInsets.symmetric(horizontal:
8)).make();
```

```
                }),
                10.heightBox,
                //Deals button
                Row(
                mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                children: List.generate(2,
                        (index) => homeButtons(
                height: context.screenHeight*0.15,
                width: context.screenWidth /2.5,
                        icon: index == 0? icTodaysDeal:icFlashDeal,
                        title: index ==0?todayDeal:flashsale,
                )),
                ),
                // 2nd swipperSwiper brands
                10.heightBox,
                VxSwiper.builder(
                aspectRatio: 16/9,
                autoPlay: true,
                height: 150,
                enlargeCenterPage: true ,
                itemCount:secondSlidersList.length,
                itemBuilder: (context,index){
                return  Image.asset(
                   secondSlidersList[index],
                fit: BoxFit.fill,
                 ).box.rounded.clip(Clip.antiAlias).margin(const
EdgeInsets.symmetric(horizontal: 8)).make();

                }),
                10.heightBox,
                Row(
                mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                children: List.generate(3, (index) => homeButtons(
                height: context.screenHeight*0.15,
                 width: context.screenWidth /3.5,
                icon: index ==0? icTopCategories: index ==1? icBrands:icTopSeller,
                title: index == 0? topcategories: index == 1?brand:topSellers,


                )),
                ),
                10.heightBox,
                Align(
                alignment: Alignment.centerLeft,
                child:
featuredcategories.text.color(darkFontGrey).size(18).fontFamily(semibold).make())
```

```
                ]
                ),
                ),
        )
        //
        ],
        )
        ),
        );
  }
 }
```

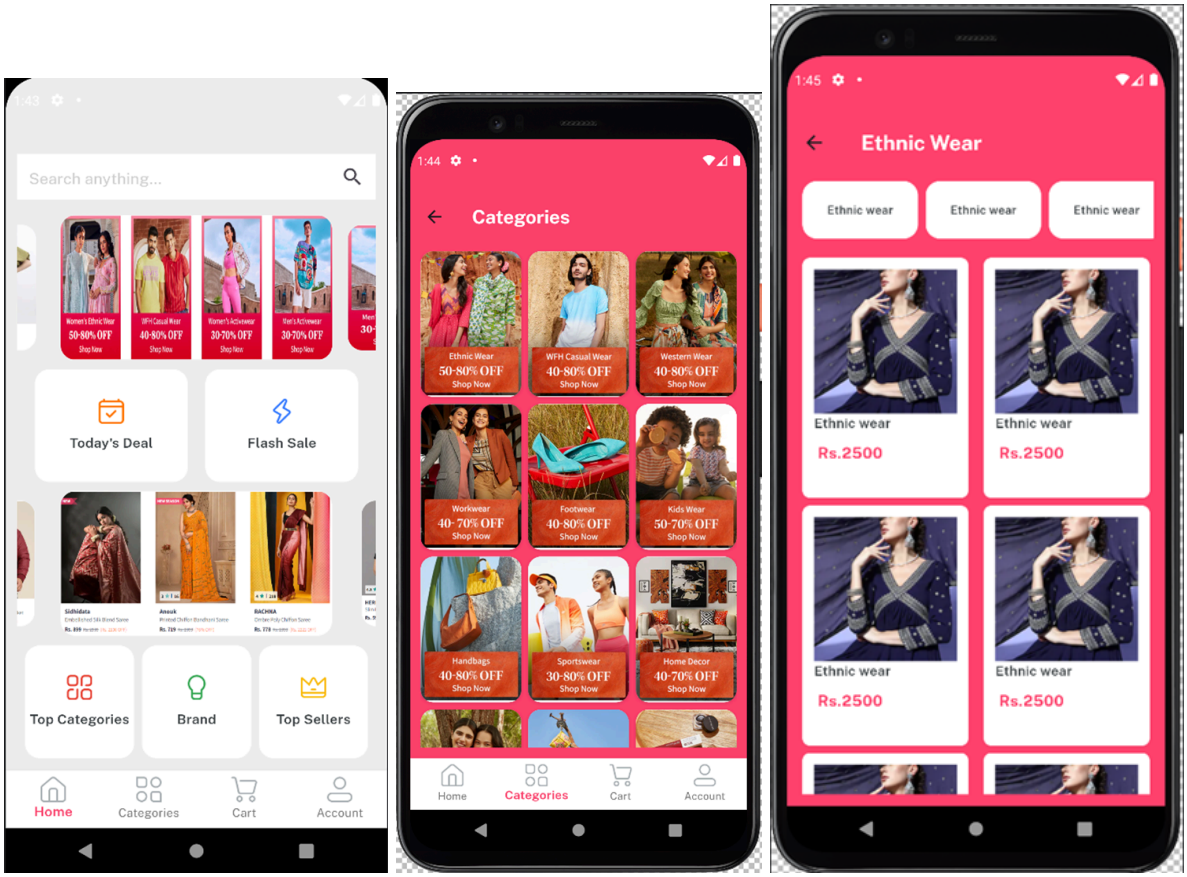
Category Screen

```
import 'package:myntra_clone/consts/consts.dart';
 import 'package:myntra_clone/consts/lists.dart';
 import 'package:myntra_clone/views/category_screen/category_details.dart';
 import 'package:myntra_clone/widgets_common/bg_widget.dart';
 import 'package:get/get.dart';
 import 'package:get/get_core/src/get_main.dart';

 class CategoryScreen extends StatelessWidget {
   const CategoryScreen({Key? key}): super(key: key);

   @override
   Widget build(BuildContext context) {
        return bgWidget(
        child: Scaffold(
        appBar: AppBar(
        title: categories.text.fontFamily(bold).white.make(),
        ),
         body: Container(
        padding: EdgeInsets.all(12),
        child: GridView.builder(
        shrinkWrap: true,
                itemCount: 12,
                gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(crossAxisCount:
3,mainAxisSpacing: 8,crossAxisSpacing: 8,mainAxisExtent: 170), itemBuilder:
(context,index){
        return Column(
            children:[Image.asset(categoriesImage[index]),
                // 10.heightBox,
            //categoriesList[index].text.color(darkFontGrey).align(TextAlign.center).make(),
                ],
           ).box.white.rounded.clip(Clip.antiAlias).outerShadow.make().onTap(() {
            Get.to(()=>CategoryDetails(title: categoriesList[index]));
        });
```

```
            }),
            ),
            )

            );
    }
  }
```

Output:

HomeScreen **Category Screen**



Conclusion:By understanding and implementing these theoretical concepts, you can create robust and user-friendly forms in Flutter that provide a seamless experience for your users.

MAD & PWA Lab
Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	02
Name	Shamaila Ansari
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:	

EXPERIMENT NO: 06

Aim: To Connect Flutter UI with firebase database

Theory:

Firebase is a Backend-as-a-Service (BaaS) app development platform that provides hosted backend services such as a realtime database, cloud storage, authentication, crash reporting, machine learning, remote configuration, and hosting for your static files.

CODE:

Auth_controller

```
import 'package:cloud_firestore/cloud_firestore.dart';
 import 'package:firebase_auth/firebase_auth.dart';
 import 'package:get/get.dart';
 import 'package:get/get_state_manager/src/simple/get_controllers.dart';
 import 'package:myntra_clone/consts/consts.dart';

 class AuthController extends GetxController{
   var isloading = false.obs;
 //text controller
 var emailController = TextEditingController();
 var passwordController = TextEditingController();


   //login method
   Future<UserCredential?>loginMethod({context}) async{
         UserCredential? userCredential;
         try{
         await auth.signInWithEmailAndPassword(email:emailController.text,password:
passwordController.text);

         } on FirebaseAuthException catch(e){
         VxToast.show(context,msg:e.toString());

         }
         return userCredential;

   }
   //Signup method
   Future<UserCredential?>signupMethod({email,password,context}) async{
         UserCredential? userCredential;
         try{
         await auth.createUserWithEmailAndPassword(email:email,password: password);

         } on FirebaseAuthException catch(e){
```

```
        VxToast.show(context,msg:e.toString());

        }
        return userCredential;

  }

  //Storing data
 storeUserData({name,password,email}) async{
        DocumentReference store =await
firestore.collection(userCollection).doc(currentUser!.uid);
        store.set({
        'name':name,
        'password':password,
        'email': email,
        'imageUrl': '',
        'id': currentUser!.uid,
        'cart_count': "00",
        'order_count': "00",
        'wishlist_count': "00",

        });
 }

 //signout method
 signoutMethod(context) async{
        try {
        await auth.signOut();

        }catch(e){
        VxToast.show(context,msg:e.toString());
        }
 }



 }
```

Edit_profile_screen

```
 import 'dart:io';

 import 'package:get/get.dart';
 import 'package:myntra_clone/controllers/profile_controller.dart';
 import 'package:myntra_clone/widgets_common/bg_widget.dart';
```

```
import 'package:myntra_clone/widgets_common/custom_textfield.dart';
import 'package:myntra_clone/widgets_common/our_button.dart';

import '../../consts/consts.dart';

class EditProfileScreen extends StatelessWidget {
  final dynamic data;
  const EditProfileScreen({Key ? key,this.data}):super(key:key);

  @override
  Widget build(BuildContext context) {
        var controller =  Get.find<ProfileController>();

        return bgWidget(
        child: Scaffold(
        resizeToAvoidBottomInset: false,
        appBar: AppBar(),
        body: Obx(()=> Column(
        mainAxisSize: MainAxisSize.min,
        children: [
                //Data Image Url and controller path is empty
        data['imageUrl'] == '' && controller.profileImgPath.isEmpty ?
Image.asset(imgProfile2,width: 100,fit:
BoxFit.cover).box.roundedFull.clip(Clip.antiAlias).make():
        data['imageUrl']  != '' && controller.profileImgPath.isEmpty
        ? Image.network(data['imageUrl'], width :100,fit: BoxFit.cover,
       ).box.roundedFull.clip(Clip.antiAlias).make():
         Image.file(File(controller.profileImgPath.value) ,
         width :100,
         fit: BoxFit.cover,
        ).box.roundedFull.clip(Clip.antiAlias).make(),


                10.heightBox,
                ourButton(color: buttonColor,onPress: () {
                controller.changeImage(context);
                },
                textColor: whiteColor,title: "Change"),
                const Divider(),
                20.heightBox,
                customTextField(
                controller: controller.nameController,
                hint: nameHint,title: name,isPass: false),
                customTextField(
                controller: controller.passController,
                hint: password,title: password,isPass: true),
```

```
                20.heightBox,
              controller.isloading.value? CircularProgressIndicator(
                valueColor: AlwaysStoppedAnimation(buttonColor) ,
                ):SizedBox(
               width:context.screenWidth-60,
                child: ourButton(color: buttonColor,onPress: () async{
                controller.isloading(true);
                await  controller.uploadProfileImage();
                await  controller.updateProfile(
                imgUrl: controller.profileImageLink,
                name: controller.nameController.text,
                password: controller.passController.text,
                );
                VxToast.show(context, msg: "Updated");

                },textColor: whiteColor,title: "Save")),
          ],
           ).box.white.shadowSm.padding(EdgeInsets.all(16)).margin(EdgeInsets.only(top: 50,left:
12,right: 12)).rounded.make(),
          ),
          ),

          );
  }
}
```

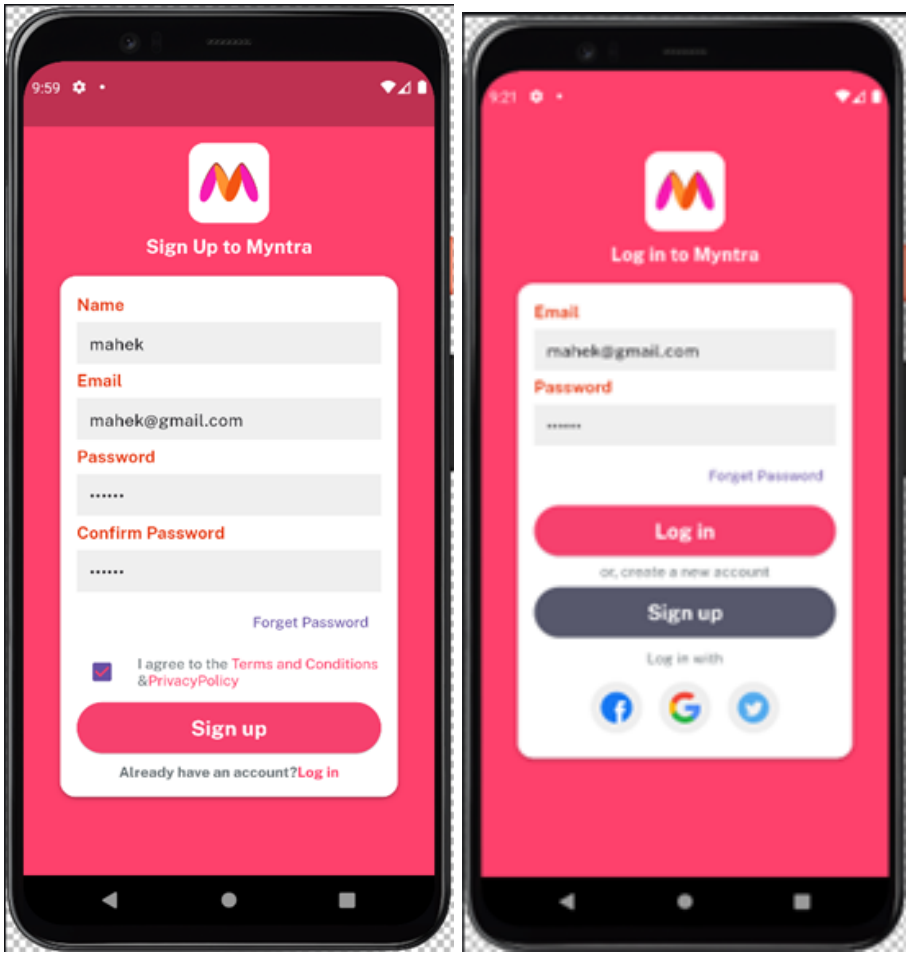
Profile_Controller

```
import 'dart:io';

import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_storage/firebase_storage.dart';
import 'package:flutter/services.dart';
import 'package:get/get.dart';
import 'package:image_picker/image_picker.dart';
import '../consts/consts.dart';
import 'package:path/path.dart';

class ProfileController extends GetxController{
  var profileImgPath = ''.obs;
  var profileImageLink = '';
  var isloading  = false.obs;
  var nameController = TextEditingController();
  var passController = TextEditingController();
  changeImage(context)async{
```

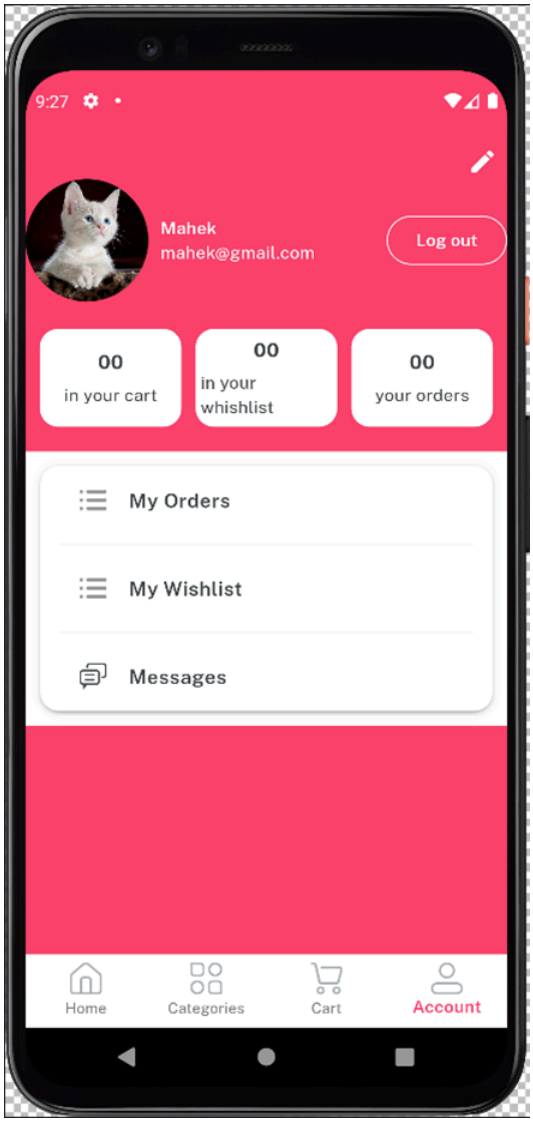
```
        try {
        final img = await ImagePicker().pickImage(
        source: ImageSource.gallery, imageQuality: 70);
        if(img ==null) return;
        profileImgPath.value = img.path;
        } on PlatformException catch(e){
        VxToast.show(context,msg : e.toString());

        }
  }
  uploadProfileImage() async {
        var filename = basename(profileImgPath.value);
        var destination = 'images/${currentUser!.uid}/$filename';
        Reference ref = FirebaseStorage.instance.ref().child(destination);
        await ref.putFile(File(profileImgPath.value));
        profileImageLink = await ref.getDownloadURL();
  }
  updateProfile({name,password,imgUrl}) async{
        var store = firestore.collection(userCollection).doc(currentUser!.uid);
   await  store.set({
        'name':name,
        'password':password,
        'imageUrl': imgUrl
        },SetOptions(merge: true));
   isloading(false);

  }
}
```

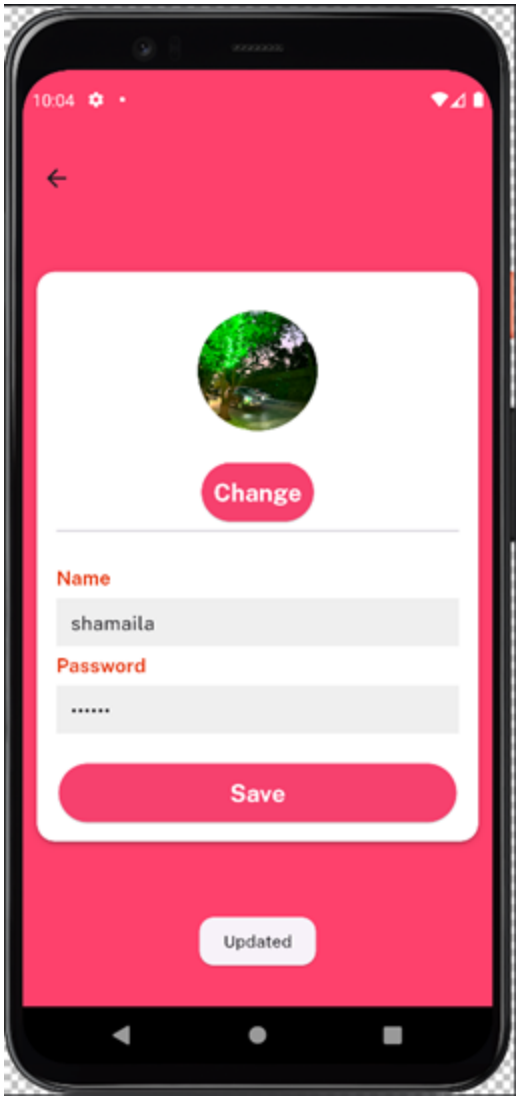
Output:

Sign up Screen **Login Screen**

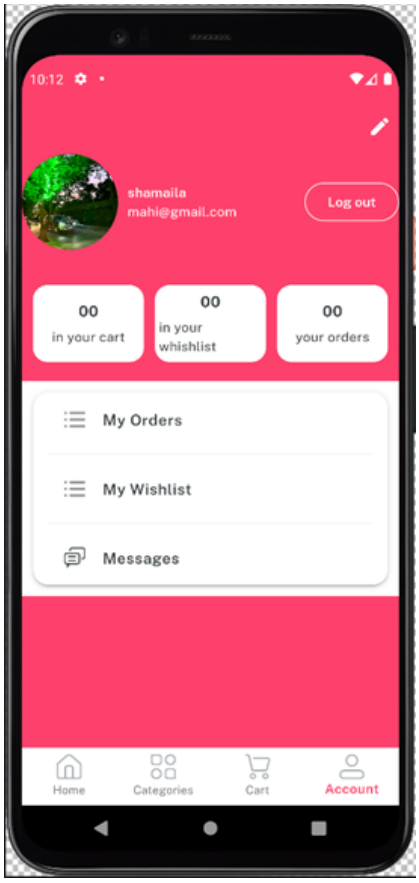


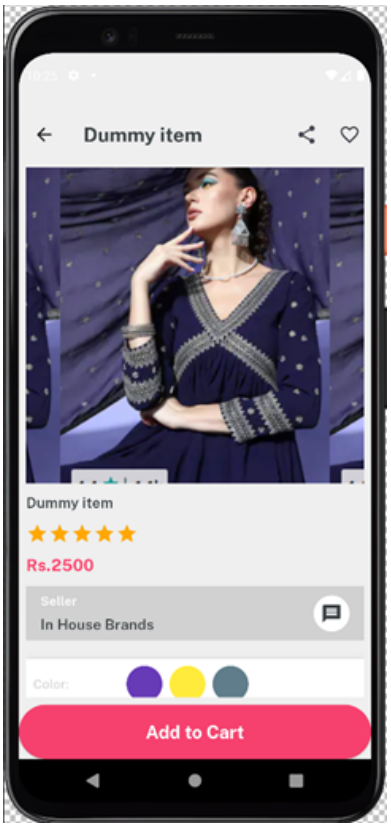
Edit Profile Screen **Update name**



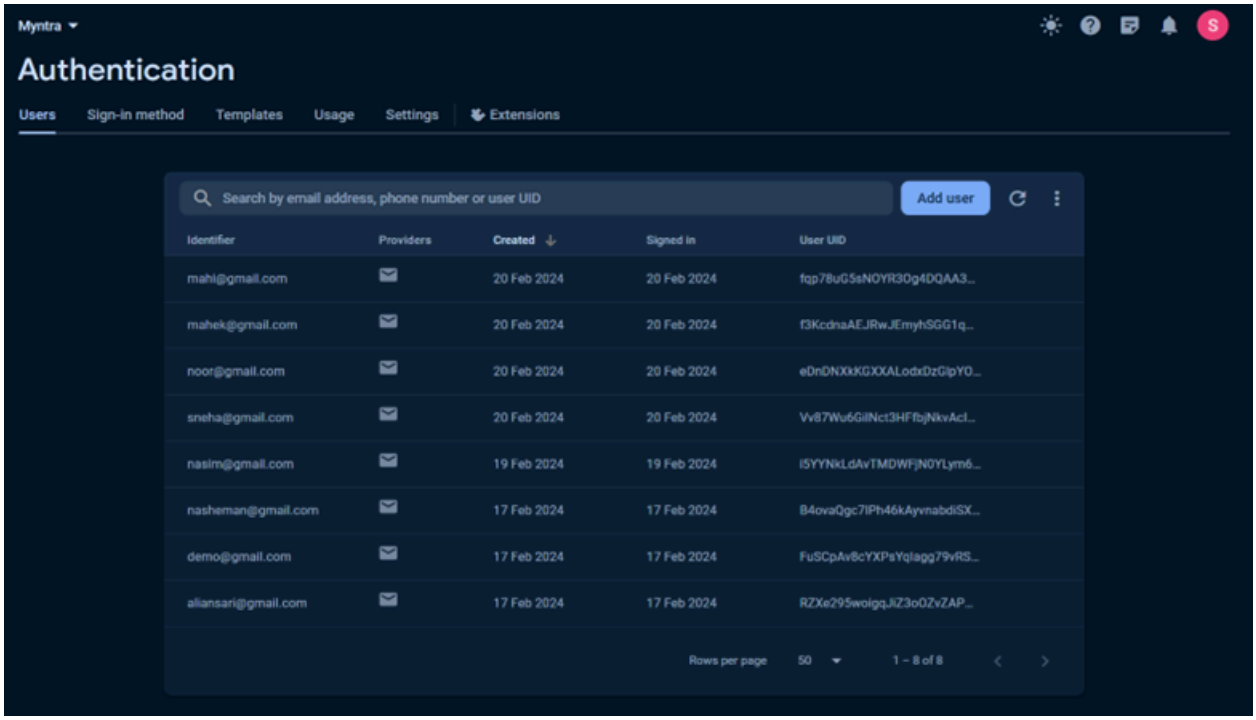


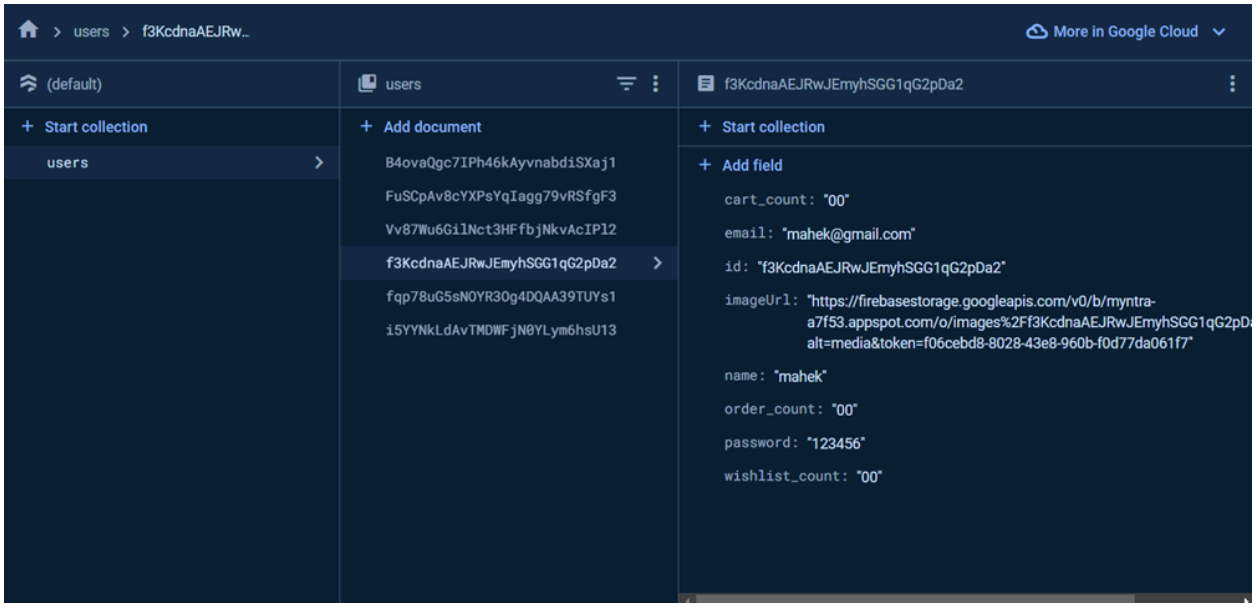
Account is updated **Category**

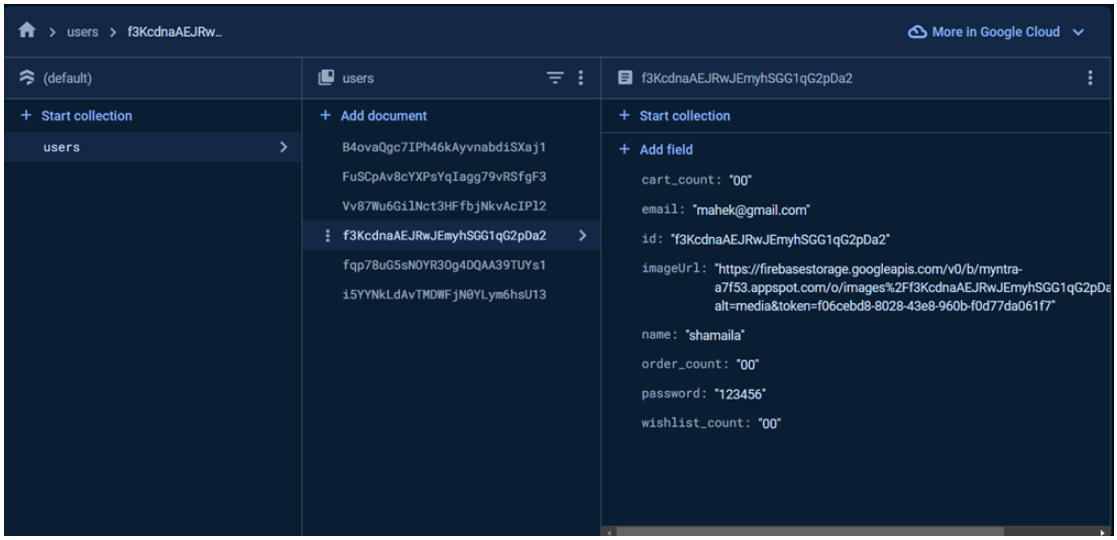


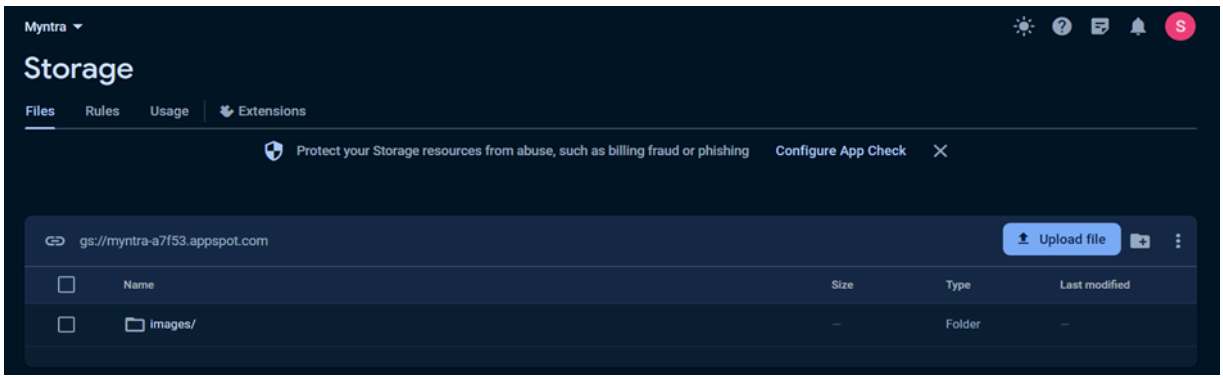


Firebase
Authentication


Myntra
Authentication
Users
Sign-in method
Templates
Usage
Settings
Extensions
Search by email address, phone number or user UID
Add user
Identifier
Providers
Created
Signed in
User UID
mahi@gmail.com 20 Feb 2024 20 Feb 2024 fqp78uG5sNOYR3Og4DQAA3...
mahek@gmail.com 20 Feb 2024 20 Feb 2024 f3KcdnaAEJRwJEmyhSGG1q...
noor@gmail.com 20 Feb 2024 20 Feb 2024 eDnDNXkKGXXALodxDzGlpYO...
sneha@gmail.com 20 Feb 2024 20 Feb 2024 Vv87Wu6GilNct3HFfbjNkvAcI...
nasim@gmail.com 19 Feb 2024 19 Feb 2024 i5YYNkLdAvTMDWFjN0YLym6...
nasheman@gmail.com 17 Feb 2024 17 Feb 2024 B4ovaQgc7IPh46kAyvnabdiSX...
demo@gmail.com 17 Feb 2024 17 Feb 2024 FuSCpAv8cYXPsYqIagg79vRS...
aliansari@gmail.com 17 Feb 2024 17 Feb 2024 RZXe295woiqqJiZ3oOZvZAP...
Rows per page 50
1 – 8 of 8


users > f3KcdnaAEJRw...
More in Google Cloud
(default)
users
f3KcdnaAEJRwJEmyhSGG1qG2pDa2
Start collection
users
Add document
B4ovaQgc7IPh46kAyvnabdiSXaj1
FuSCpAv8cYXPsYqIagg79vRSfgF3
Vv87Wu6GilNct3HFfbjNkvAcIPl2
f3KcdnaAEJRwJEmyhSGG1qG2pDa2
fqp78uG5sNOYR3Og4DQAA39TUYs1
i5YYNkLdAvTMDWFjN0YLym6hsU13
Start collection
Add field
cart_count: "00"
email: "mahek@gmail.com"
id: "f3KcdnaAEJRwJEmyhSGG1qG2pDa2"
imageUrl: "https://firebasestorage.googleapis.com/v0/b/myntra-a7f53.appspot.com/o/images%2Ff3KcdnaAEJRwJEmyhSGG1qG2pD
alt=media&token=f06cebd8-8028-43e8-960b-f0d77da061f7"
name: "mahek"
order_count: "00"
password: "123456"
wishlist_count: "00"





Conclusion: Here, Successfully Connected Flutter UI with Firebase database

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.	02
Name	Shamaila Ansari
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	

EXPERIMENT NO: 07

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.

6. Discoverable

PWAs reside in web browsers. This implies higher chances of optimizing them as per the Search Engine Optimization (SEO) criteria and improving the Google rankings like that in websites and other web apps.

7. Lower Development Cost

Progressive web apps can be installed on the user device like a native device, but it does not demand submission on an App Store. This makes it far more cost-effective than native mobile applications while offering the same set of functionalities.

Pros and cons of the Progressive Web
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.
Installable — They allow the user to “save” the apps that he considers most useful with the corresponding icon on the screen of his mobile terminal (home screen) without having to face all the steps and problems related to the use of the app store.

Linkable — Easily shared via URL without complex installations.

Offline — Once more it is about putting the user before everything, avoiding the usual error message in case of weak or no connection. The PWA are based on two particularities: first of all the ‘skeleton’ of the app, which recalls the page structure, even if its contents do not respond and its elements include the header, the page layout, as well as an illustration that signals that the page is loading.

Weaknesses refer to:

IOS support from version 11.3
onwards; Greater use of the
device battery;

Not all devices support the full range of PWA features (same speech for iOS
and Android operating systems);
It is not possible to establish a strong re-engagement for iOS users (URL scheme,
standard web notifications);
Support for offline execution is however limited;
Lack of presence on the stores (there is no possibility to acquire traffic from that
channel); There is no “body” of control (like the stores) and an approval process;
Limited access to some hardware components of the devices;
Little flexibility regarding “special” content for users (eg loyalty programs, loyalty, etc.)

CODE:

```
<!DOCTYPE html>

<html lang="en">

<head>

   <meta http-equiv="Content-Type" content="text/html;

charset=UTF-8" />

   <meta http-equiv="X-UA-Compatible" content="IE=edge" />

   <meta name="viewport" content="width=device-width,

initial-scale=1.0" />

   <meta name="description" content="fuzzy" />

   <meta name="keywords" content="fuzzy" />
```

```
  <meta name="author" content="fuzzy" />

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

  <link rel="icon" href="assets/images/logo/favicon.png"
type="image/x-icon" />

  <title>fuzzy</title>

  <link rel="apple-touch-icon" href="assets/images/logo/favicon.png"
/>

  <meta name="theme-color" content="#122636" />

  <meta name="apple-mobile-web-app-capable" content="yes" />

  <meta name="apple-mobile-web-app-status-bar-style"
content="black" />

  <meta name="apple-mobile-web-app-title" content="fuzzy" />

  <meta name="msapplication-TileImage"
content="assets/images/logo/favicon.png" />

  <meta name="msapplication-TileColor" content="#FFFFFF" />

  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
```

```
    <!--Google font-->
    <link rel="preconnect" href="https://fonts.googleapis.com/" />
    <link rel="preconnect" href="https://fonts.gstatic.com/" crossorigin />
    <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@100;200;300;400;500;600;700;800;900&amp;display=swap" rel="stylesheet" />

    <!-- iconsax css -->
    <link rel="stylesheet" type="text/css"
href="assets/css/vendors/iconsax.css" />

    <!-- bootstrap css -->
    <link rel="stylesheet" id="rtl-link" type="text/css"
href="assets/css/vendors/bootstrap.min.css" />
```

```
    <!-- swiper css -->
    <link rel="stylesheet" type="text/css"
href="assets/css/vendors/swiper-bundle.min.css" />

    <!-- Theme css -->
    <link rel="stylesheet" id="change-link" type="text/css"
href="assets/css/style.css" />
  </head>

  <body class="auth-body dark">
    <!-- onboarding section start -->
    <section class="section-b-space">
      <div class="swiper intro slider-1">
        <div class="swiper-wrapper">
```

```
        <div class="swiper-slide">

          <div class="theme-logo pb-3">

            <img class="img-fluid logo-img"
src="assets/images/logo/logo.png" alt="logo" />

          </div>

          <div class="onboarding-design">

            <img class="img-fluid design-img"
src="assets/images/onboarding/design1.png" alt="bg-design" />



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



            <img class="img-fluid vector1"
src="assets/images/onboarding/vector1.png" alt="v1" />

            <img class="img-fluid vector2"
src="assets/images/onboarding/vector2.png" alt="v2" />
```

```
            <img class="img-fluid vector3"
src="assets/images/onboarding/vector3.png" alt="v3" />
          </div>
          <div class="product-details">
            <h1>Office Furniture</h1>
            <span></span>
            <p>The best products that suits your lifestyle with visually
appealing designs.</p>
            <div class="redirate-btn">
              <a href="login.html" class="next-arrow">
                <i class="iconsax right-arrow" data-icon="arrow-right"></i>
              </a>
            </div>
          </div>
        </div>
```

```
        <div class="swiper-slide">
          <div class="theme-logo pb-3">
            <img class="img-fluid logo-img"
src="assets/images/logo/logo.png" alt="logo" />
          </div>
          <div class="onboarding-design">
            <img class="img-fluid design-img"
src="assets/images/onboarding/design1.png" alt="bg-design" />

            <img class="img-fluid slider-img2"
src="assets/images/onboarding/2.png" alt="slider-2" />

            <img class="img-fluid vector1"
src="assets/images/onboarding/vector1.png" alt="v1" />
```

```
            <img class="img-fluid vector2"
src="assets/images/onboarding/vector2.png" alt="v2" />

            <img class="img-fluid vector3"
src="assets/images/onboarding/vector3.png" alt="v3" />

          </div>

          <div class="product-details">

            <h1>Relaxing Furniture</h1>

            <span></span>

            <p>The best funiture that fits your House and
workspace.</p>

            <div class="redirate-btn">

              <a href="login.html" class="next-arrow">

                <i class="iconsax right-arrow" data-icon="arrow-right"></i>

              </a>

            </div>

          </div>
```

```
        </div>


        <div class="swiper-slide">

          <div class="theme-logo pb-3">

            <img class="img-fluid logo-img"
src="assets/images/logo/logo.png" alt="logo" />

          </div>

          <div class="onboarding-design">

            <img class="img-fluid design-img"
src="assets/images/onboarding/design1.png" alt="bg-design" />


            <img class="img-fluid slider-img3"
src="assets/images/onboarding/3.png" alt="slider-3" />


            <img class="img-fluid vector1"
src="assets/images/onboarding/vector1.png" alt="v1" />
```

```
            <img class="img-fluid vector2"
src="assets/images/onboarding/vector2.png" alt="v2" />

            <img class="img-fluid vector3"
src="assets/images/onboarding/vector3.png" alt="v3" />

            <div class="product-details">

              <h1>Home Decor</h1>

              <span></span>

              <p>The best payment method connects your money to
friends, family, brands, and experiences.</p>

              <div class="redirate-btn">

                <a href="login.html" class="next-arrow">

                  <i class="iconsax right-arrow"
data-icon="arrow-right"></i>

                </a>

              </div>

            </div>
```

```
          </div>
        </div>
      </div>
    </div>
  </section>


  <!-- onboarding section end -->


  <!-- pwa install app popup start -->
  <div class="offcanvas offcanvas-bottom addtohome-popup
theme-offcanvas" tabindex="-1" id="offcanvas">
    <button type="button" class="btn-close text-reset"
data-bs-dismiss="offcanvas" aria-label="Close"></button>
    <div class="offcanvas-body small">
      <div class="app-info">
```

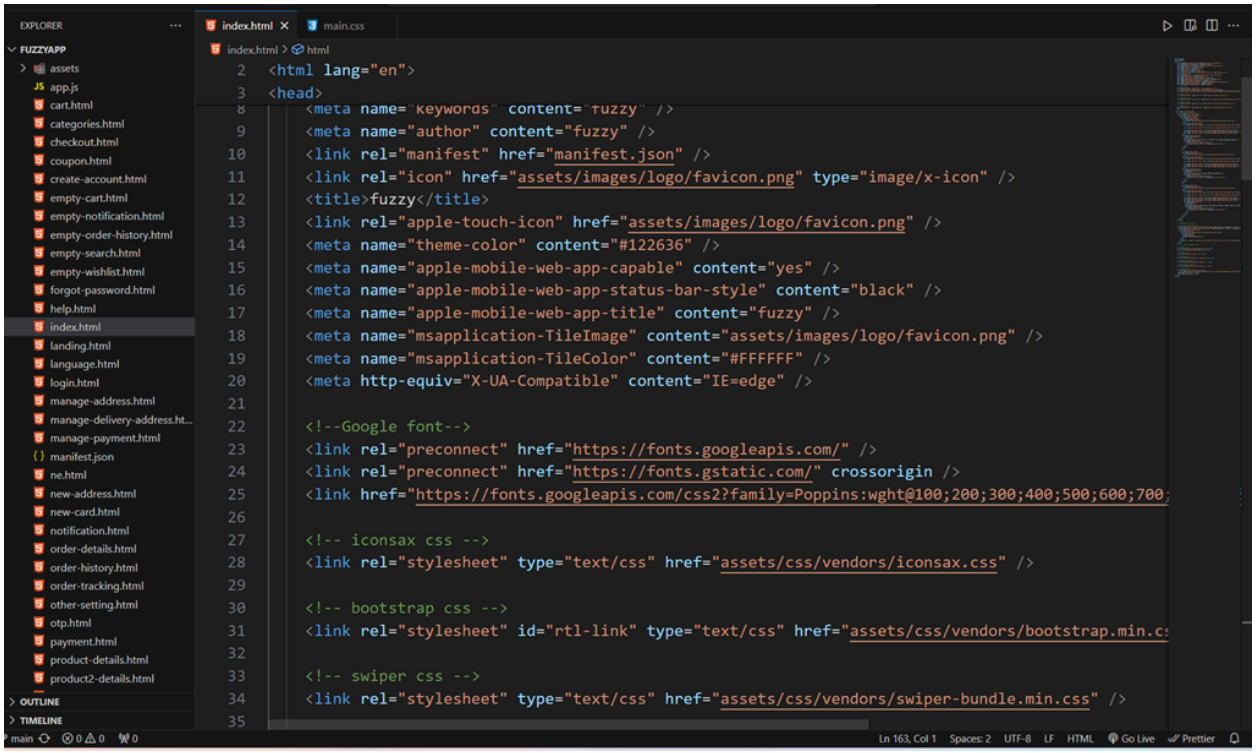
```
        <img src="assets/images/logo/48.png" class="img-fluid" alt="" />
        <div class="content">
          <h4>fuzzy app</h4>
          <a href="#">www.fuzzy-app.com</a>
        </div>
      </div>
      <a href="#!" class="btn theme-btn install-app btn-inline
home-screen-btn m-0" id="installapp">Add to Home Screen</a>
    </div>
  </div>
  <!-- pwa install app popup start -->


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

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

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

<!-- homescreen popup js -->
<script src="assets/js/homescreen-popup.js"></script>

<!-- PWA offcanvas popup js -->
<script src="assets/js/offcanvas-popup.js"></script>

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

```
    <script async
src="https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js
?client=ca-pub-56454510295440 50"
    crossorigin="anonymous"></script>
    <script src="/app.js"></script>
  </body>
</html>
```

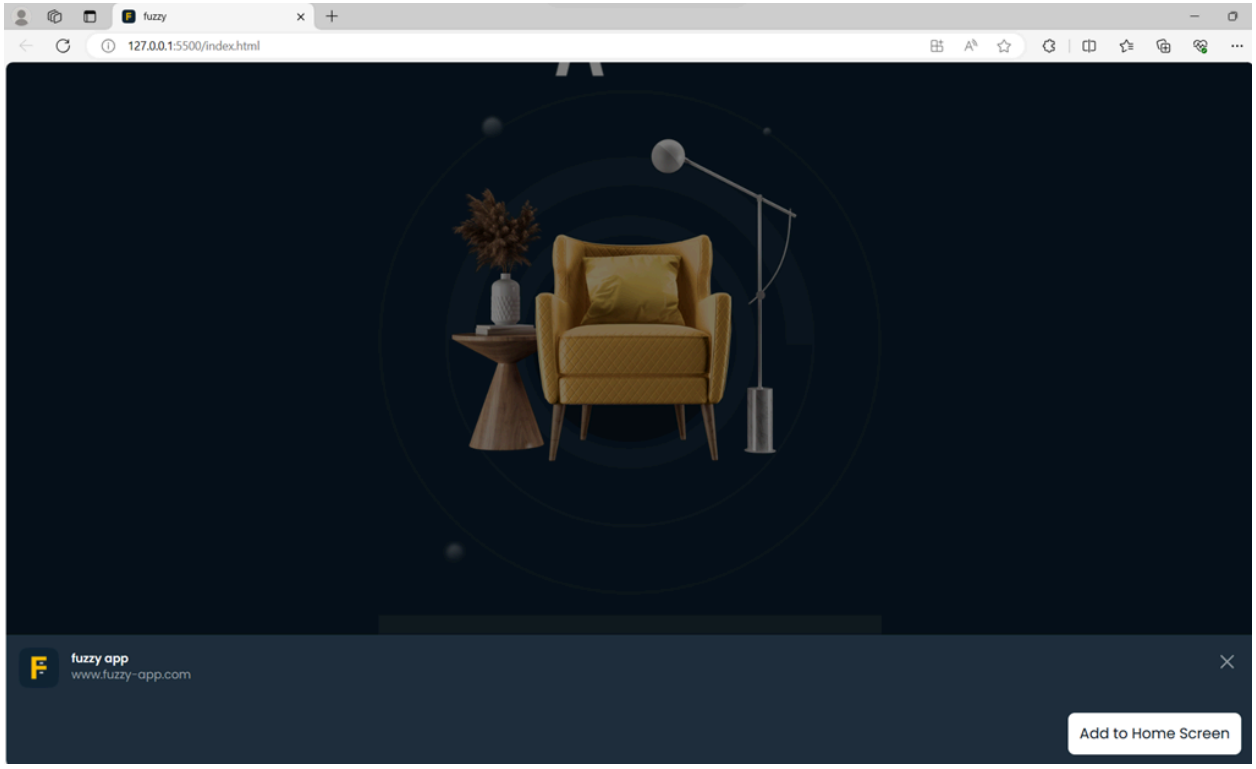
OUTPUT:

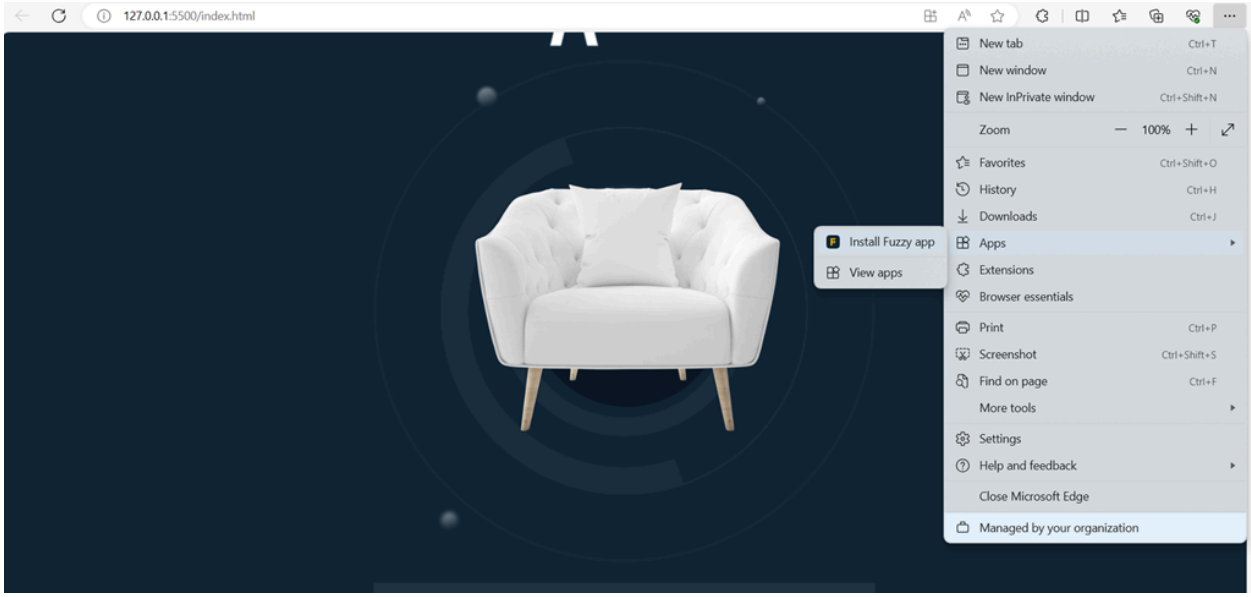
Open folder in VS code and click go live at bottom right corner

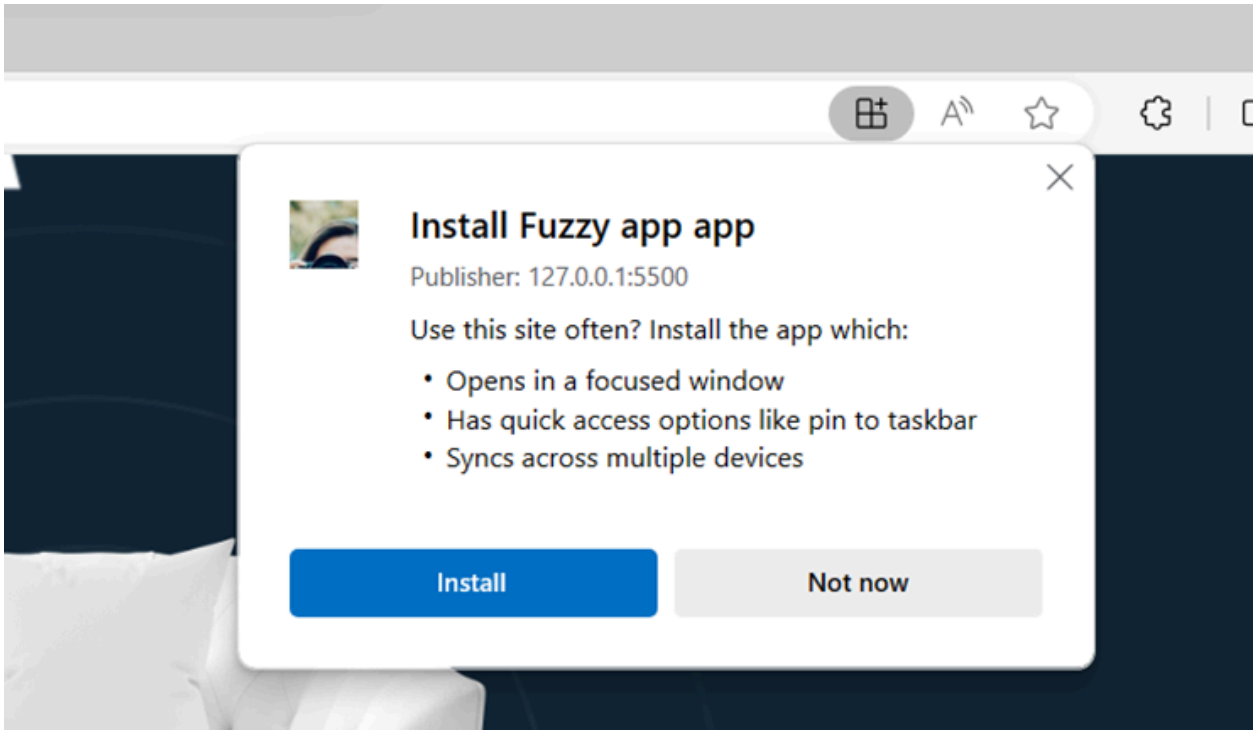


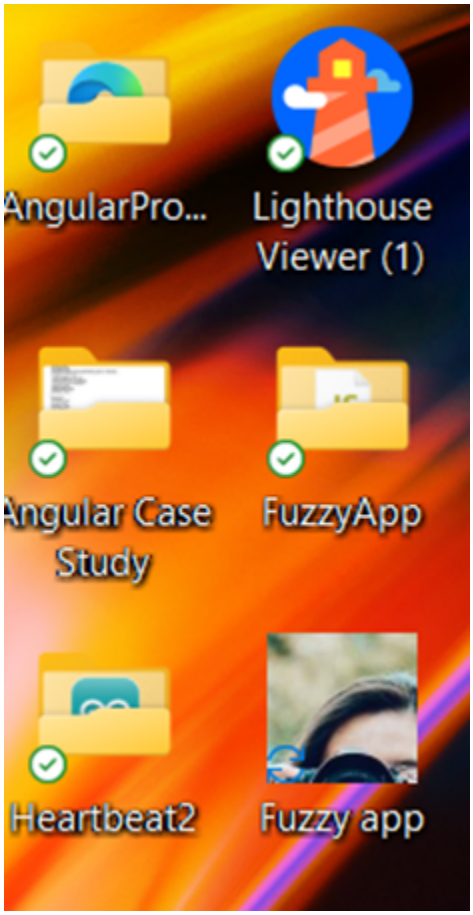
Open your hosted site on Microsoft Edge

Project Title: Flutter-Myntra Clone /PWA- Fuzzy app **RollNo. 02**



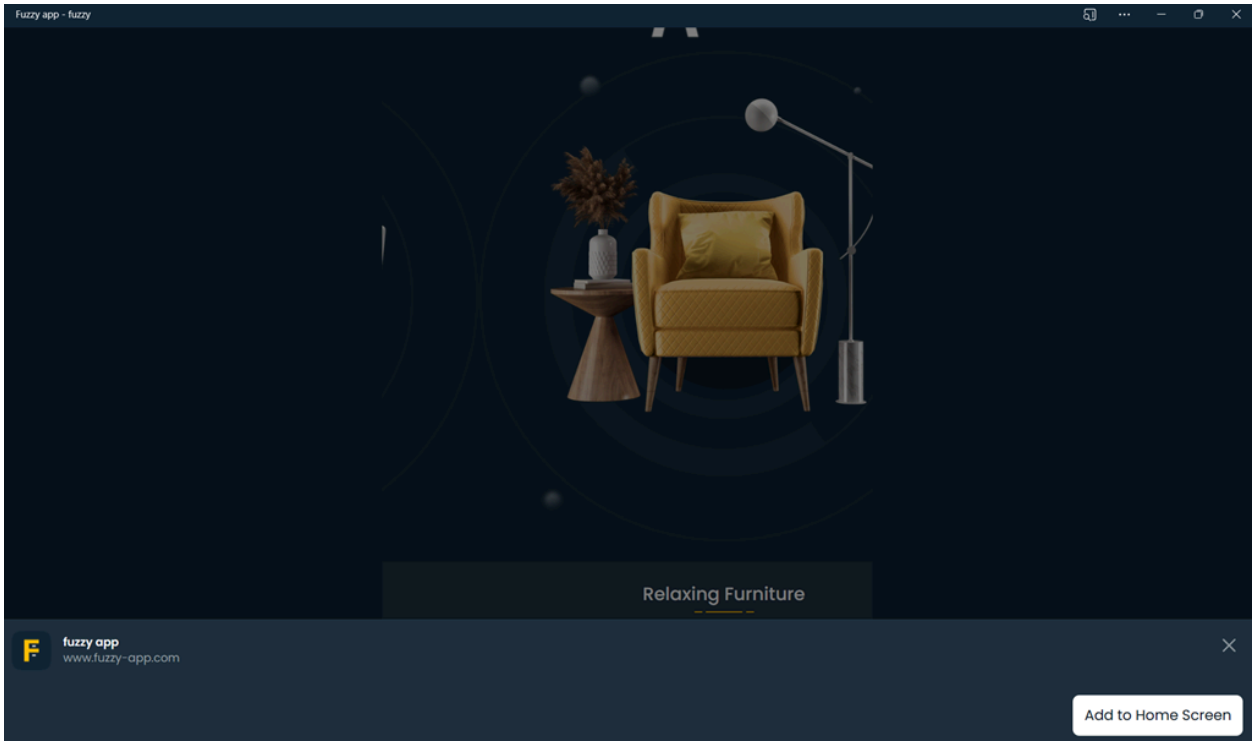


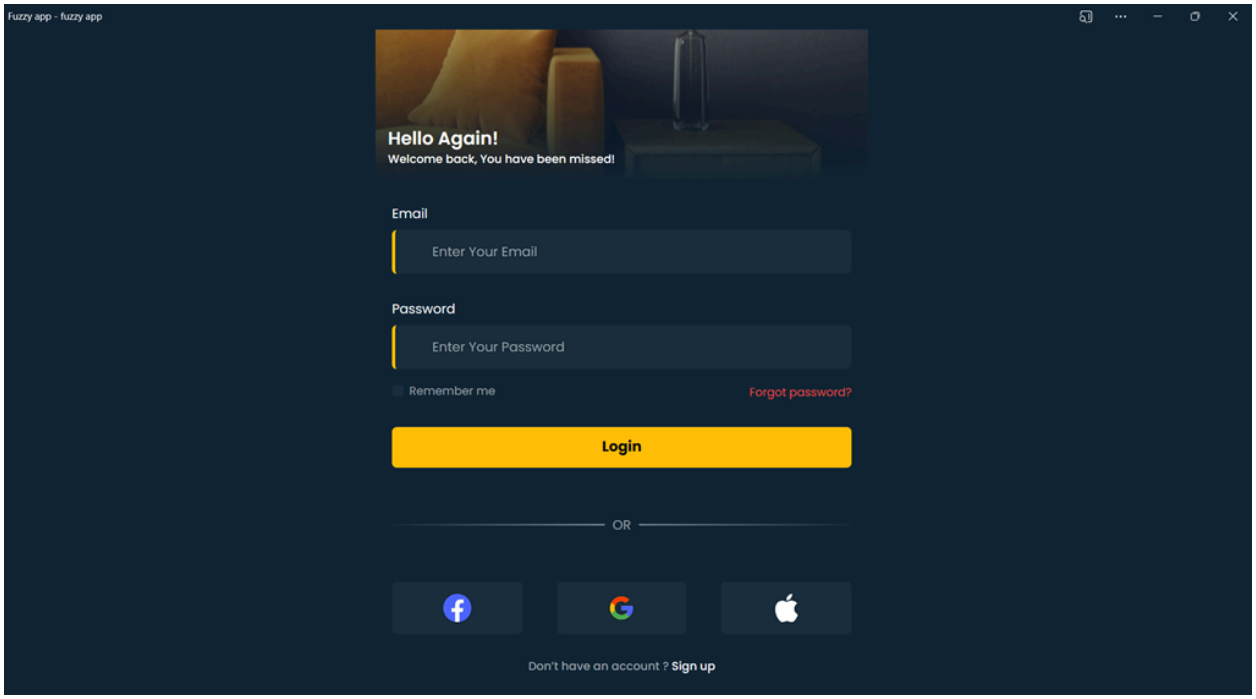

Install Fuzzy app app
Publisher: 127.0.0.1:5500
Use this site often? Install the app which:
• Opens in a focused window
• Has quick access options like pin to taskbar
• Syncs across multiple devices
Install
Not now


AngularPro...
Lighthouse Viewer (1)
Angular Case Study
FuzzyApp
Heartbeat2
Fuzzy app

Project Title: Flutter-Myntra Clone /PWA- Fuzzy app **RollNo. 02**

Open Desktop app





Conclusion: Here, 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.	02
Name	Shamaila Ansari
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

Experiment No:08

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

Theory:

Service Worker

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

Things to note about Service Worker:

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

What can we do with Service Workers?

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

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

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

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

What can't we do with Service Workers?

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

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

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

- Registration
- Installation
- Activation

Registration

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

main.js

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

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

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

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

main.js

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

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

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

Code

Index.html

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

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

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

```
</head>

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

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

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

Project Title: Flutter-Myntra Clone /PWA- Fuzzy app **RollNo. 02**

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

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

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

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

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

App.js

Project Title: Flutter-Myntra Clone /PWA- Fuzzy app **RollNo. 02**

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

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

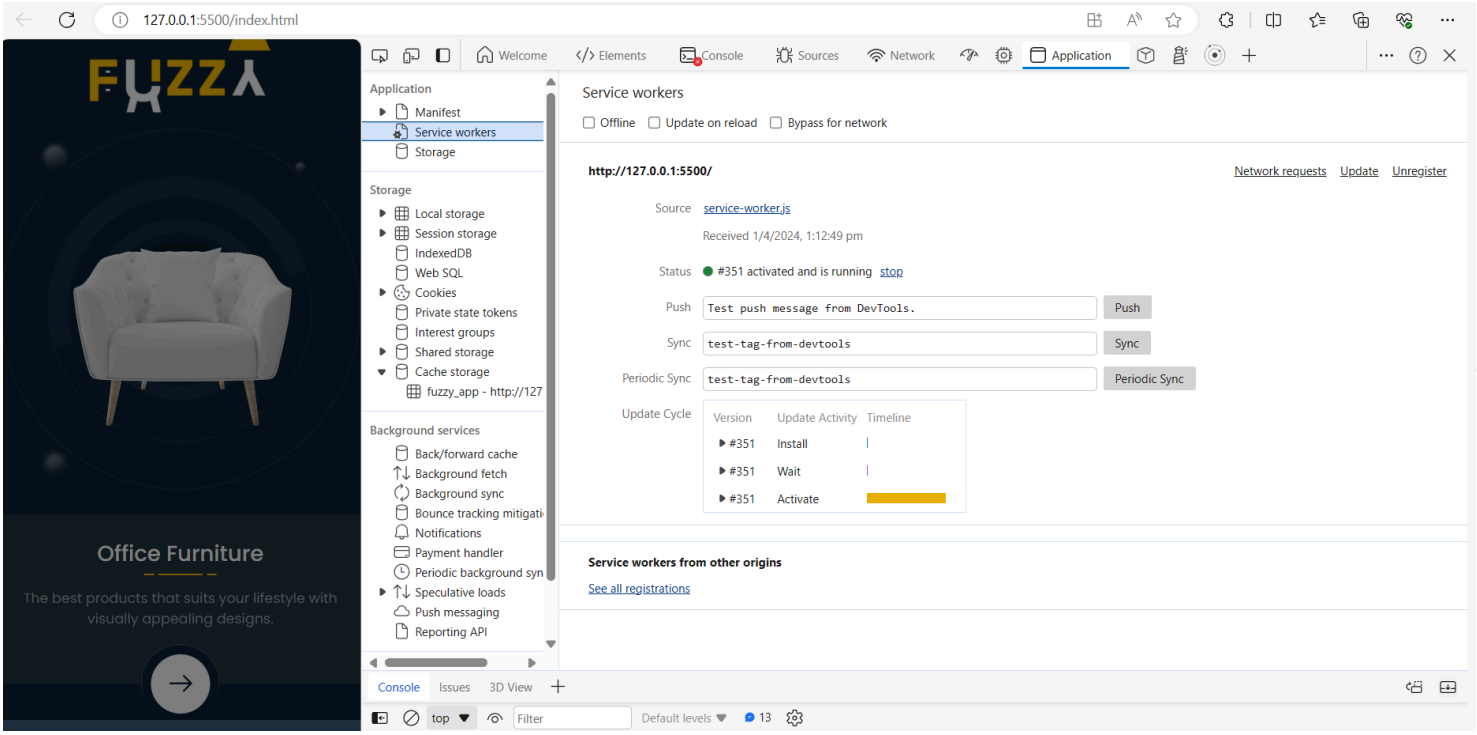
manifest.json M | app.js U | service-worker.js U | index.html M

service-worker.js > assetsToCache

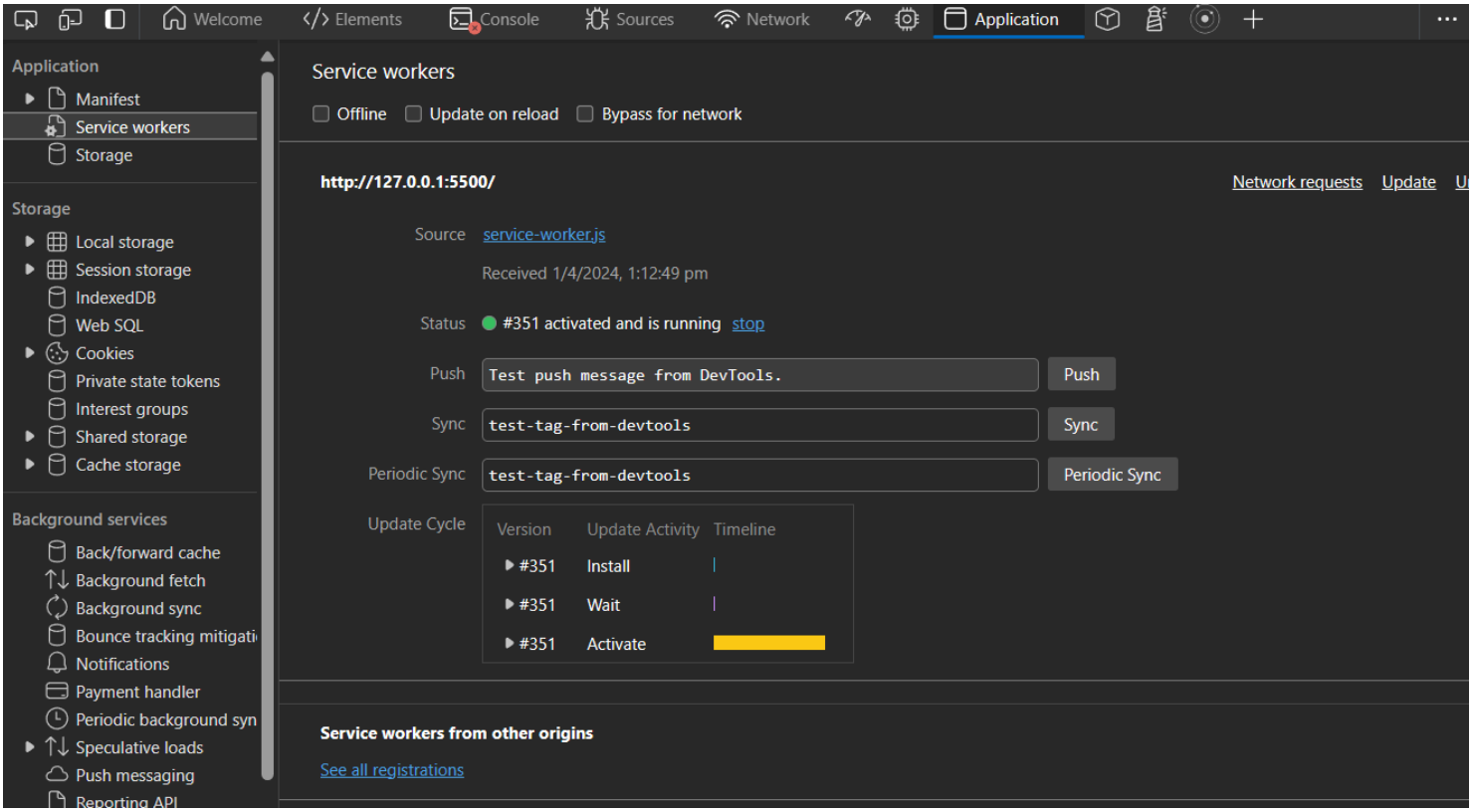
```
const cacheName = 'fuzzy_app';
const assetsToCache = [
'/',
'/index.html',
'/assets/css/style.css',
'/app.js'
]
self.addEventListener('install', event => {
event.waitUntil(
caches.open(cacheName)
.then(cache => {
return cache.addAll(assetsToCache);
})
);
});
self.addEventListener('activate', event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

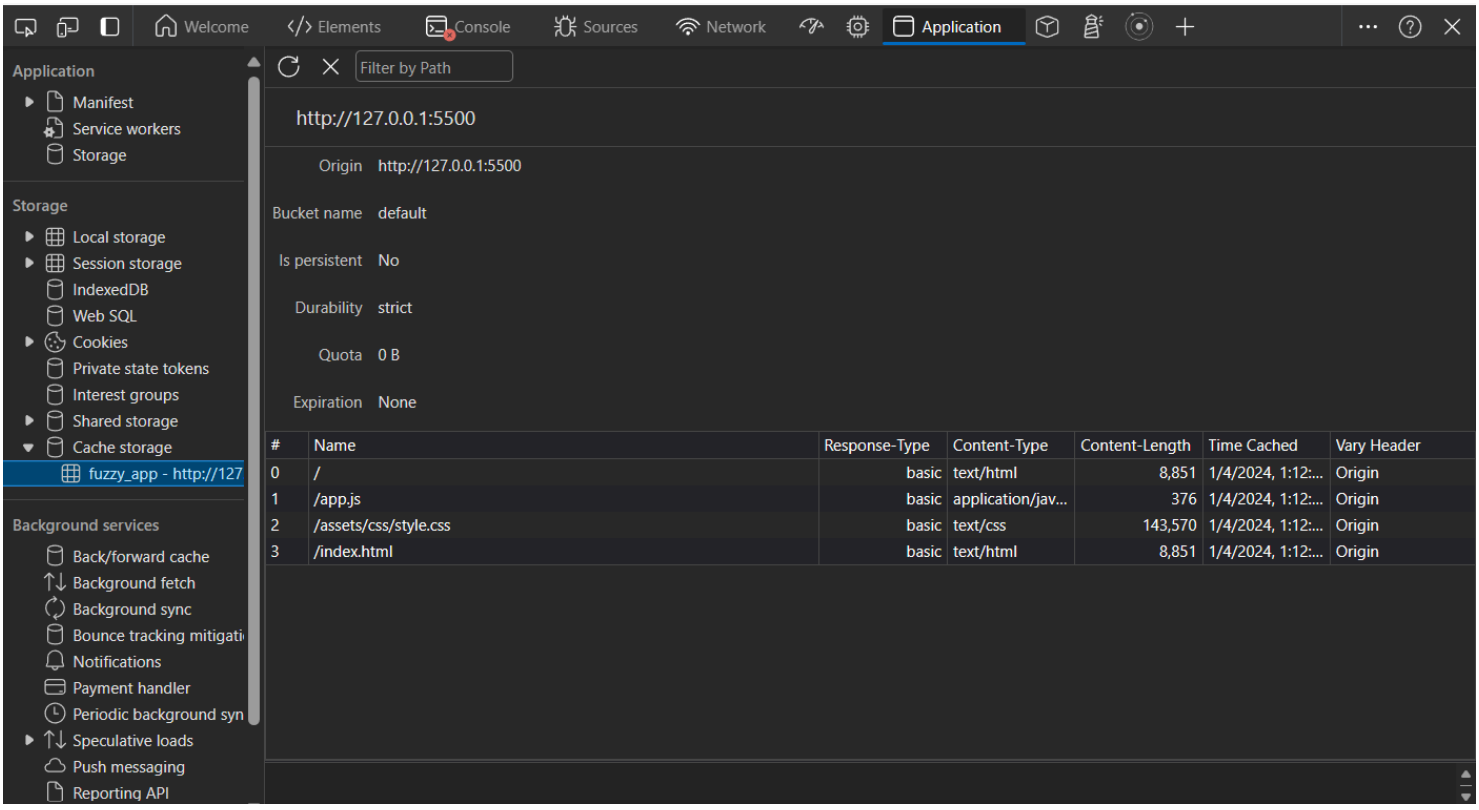
Project Title: Flutter-Myntra Clone /PWA- Fuzzy app **RollNo. 02**

Output:



Project Title: Flutter-Myntra Clone /PWA- Fuzzy app **RollNo. 02**





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 PW

MAD & PWA Lab
Journal

Experiment No.	09
Experiment Title.	To implement Service worker events like fetch, sync and push for E-commerce PWA
Roll No.	02
Name	Shamaila Ansari
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

Experiment No:09

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

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

Things to note about Service Worker:

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

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

Of course, you can use many different methods but you can find in the following example a “cache first” and “network first” approach. In this example, if the request’s and current location’s origin are 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.

Code:

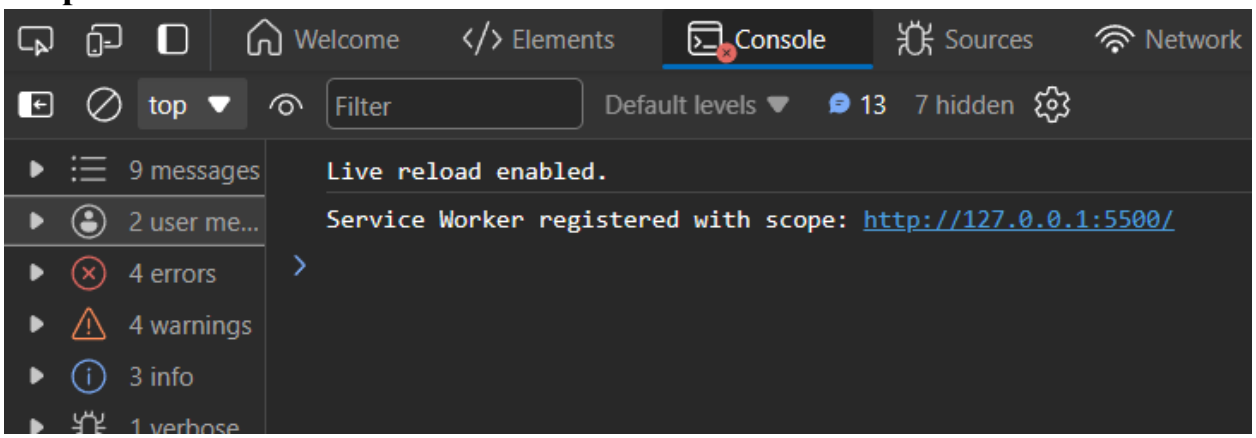
Service-Worker.js

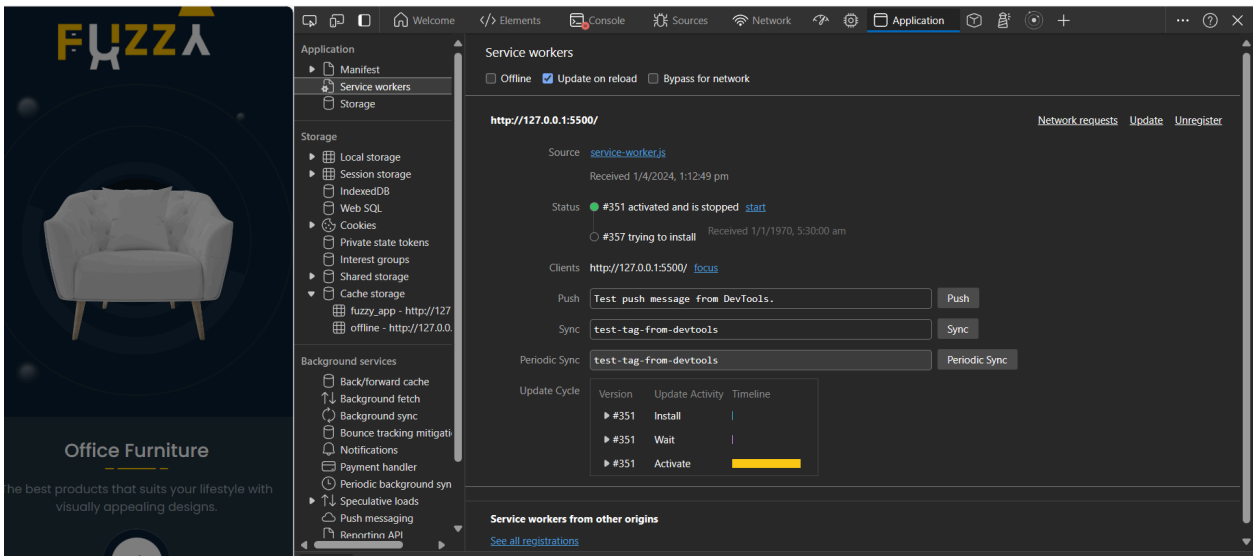
```
self.addEventListener("install", function (event) {
  event.waitUntil(preLoad());
  });
  self.addEventListener("fetch", function (event) {
  event.respondWith(
  checkResponse(event.request).catch(function () {
  console.log("Fetch from cache successful!");
  return returnFromCache(event.request);
  })
  );
  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.showNotification("Ecommerce website", {
  body: data.message,
  });
  }
  } catch (error) {
  console.error("Error parsing push data:", error);
  }
  }
  });
  var preLoad = function () {
  return caches.open("offline").then(function (cache) {
```

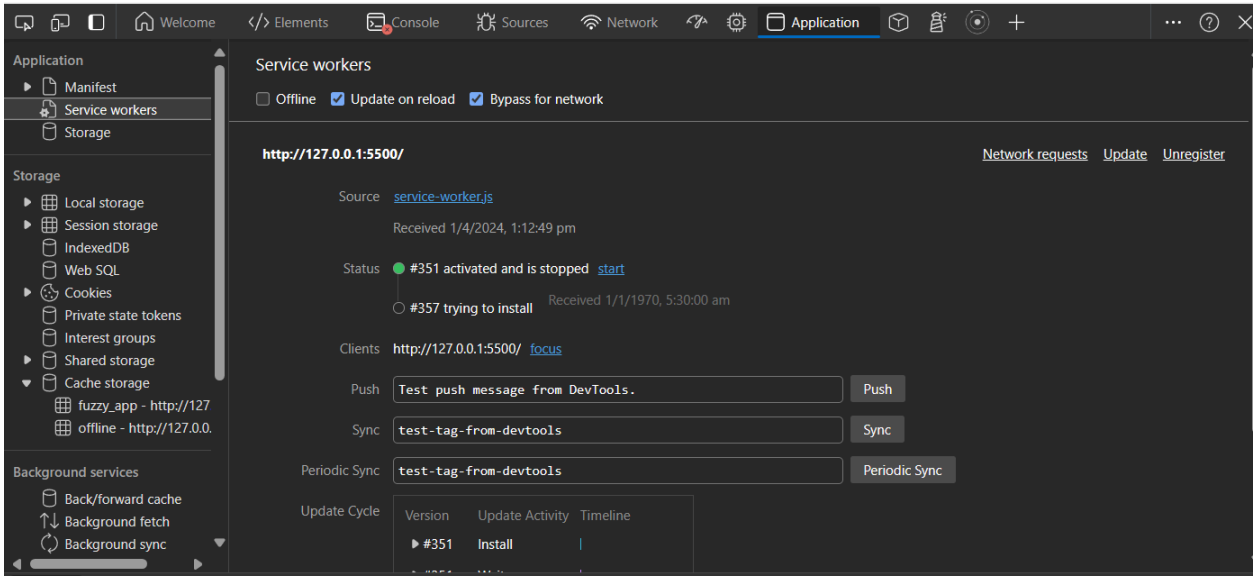
```
// caching index and important routes
return cache.addAll([
"/",
"/index.html",
"/landing.html",
"/profile.html",
"/shop.html",
"/cart.html",
"/css/main.css",
]);
});
};
var checkResponse = function (request) {
return new Promise(function (fulfill, reject) {
fetch(request)
.then(function (response)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

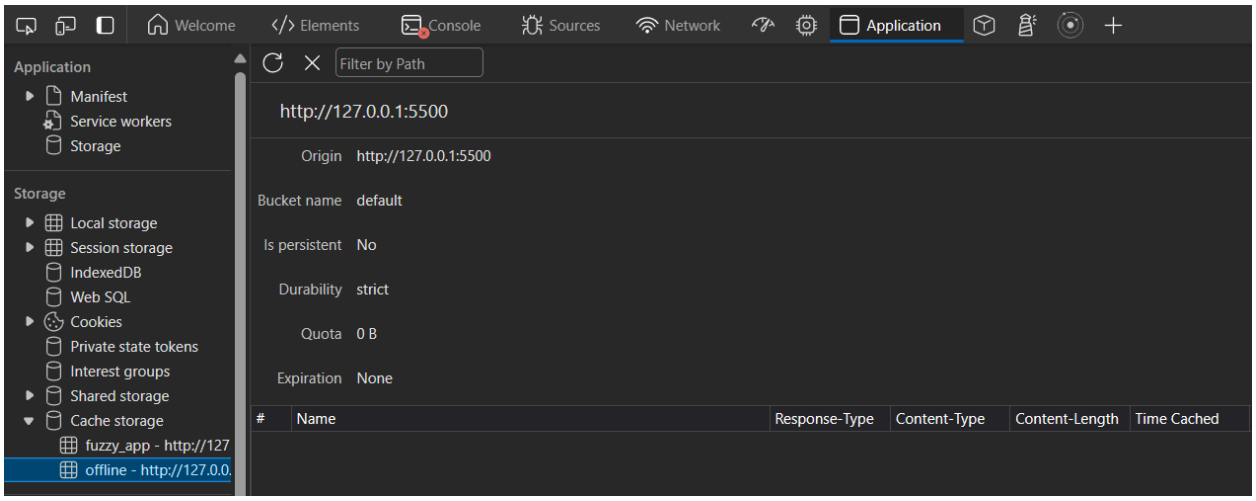
```
};
var addToCache = function (request) {
return caches.open("offline").then(function (cache) {
return fetch(request).then(function (response) {
return cache.put(request, response.clone()).then(function () {
return response;
});
});
});
};
```

Output









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

MAD & PWA Lab
Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	02
Name	Shamaila Ansari
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

Experiment No. 10

Aim:

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

Theory:

GitHub Pages

Public web pages are freely hosted and easily published. Public webpages hosted 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

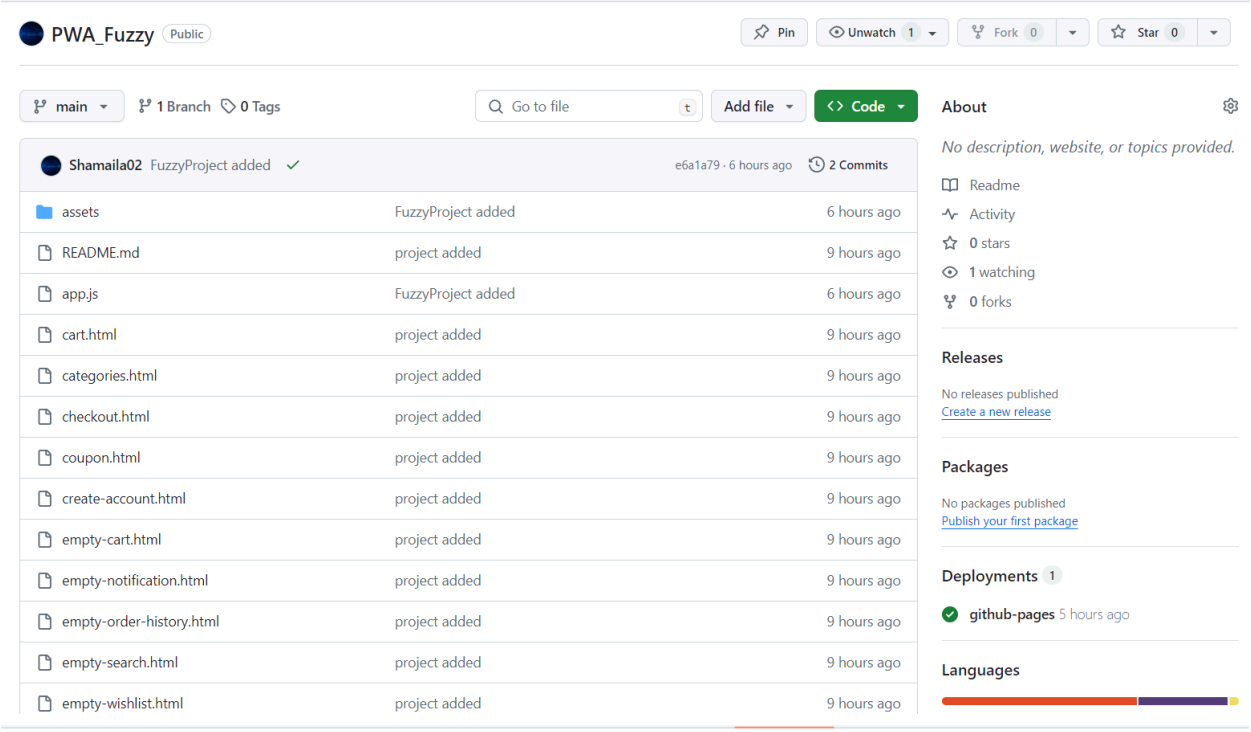
1. Only 10 GB of data transfer is allowed per month. But this is not really a big problem, if you use a CDN or AMP.
2. Command-line interface only.
3. No in-built support for any static site generator.

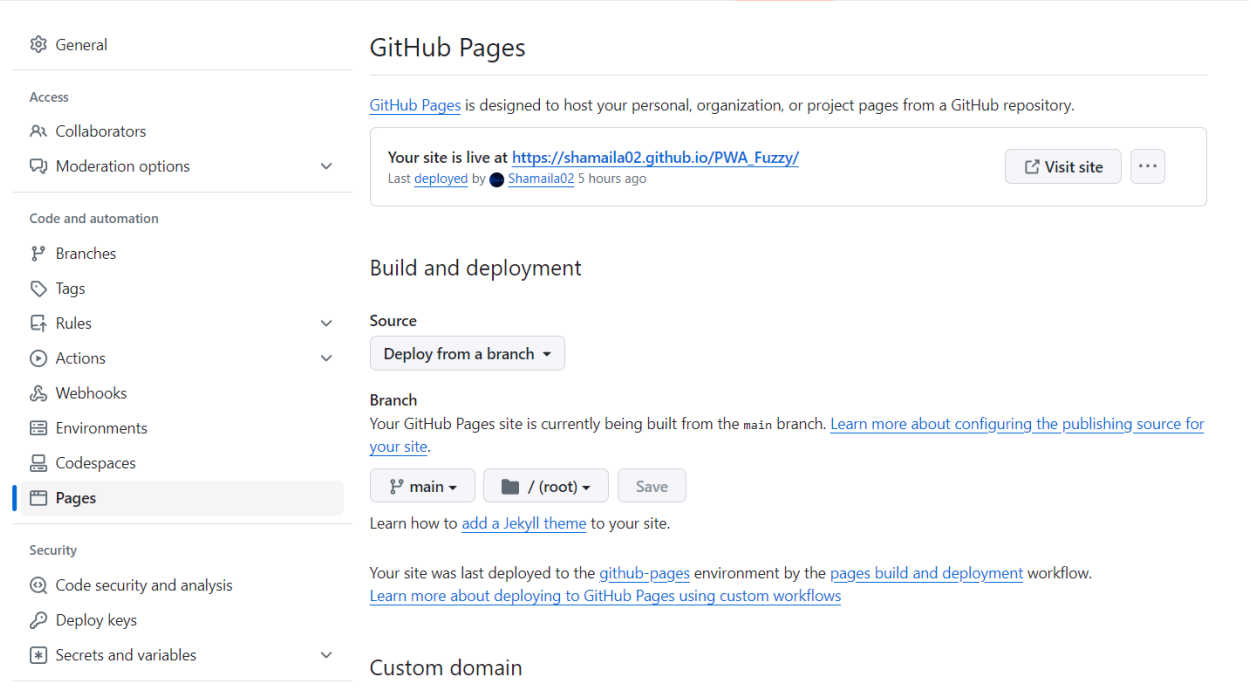
Link to our GitHub repository: https://github.com/Shamaila02/PWA_Fuzzy

Project Title: Flutter-Myntra Clone /PWA- Fuzzy app **RollNo. 02**

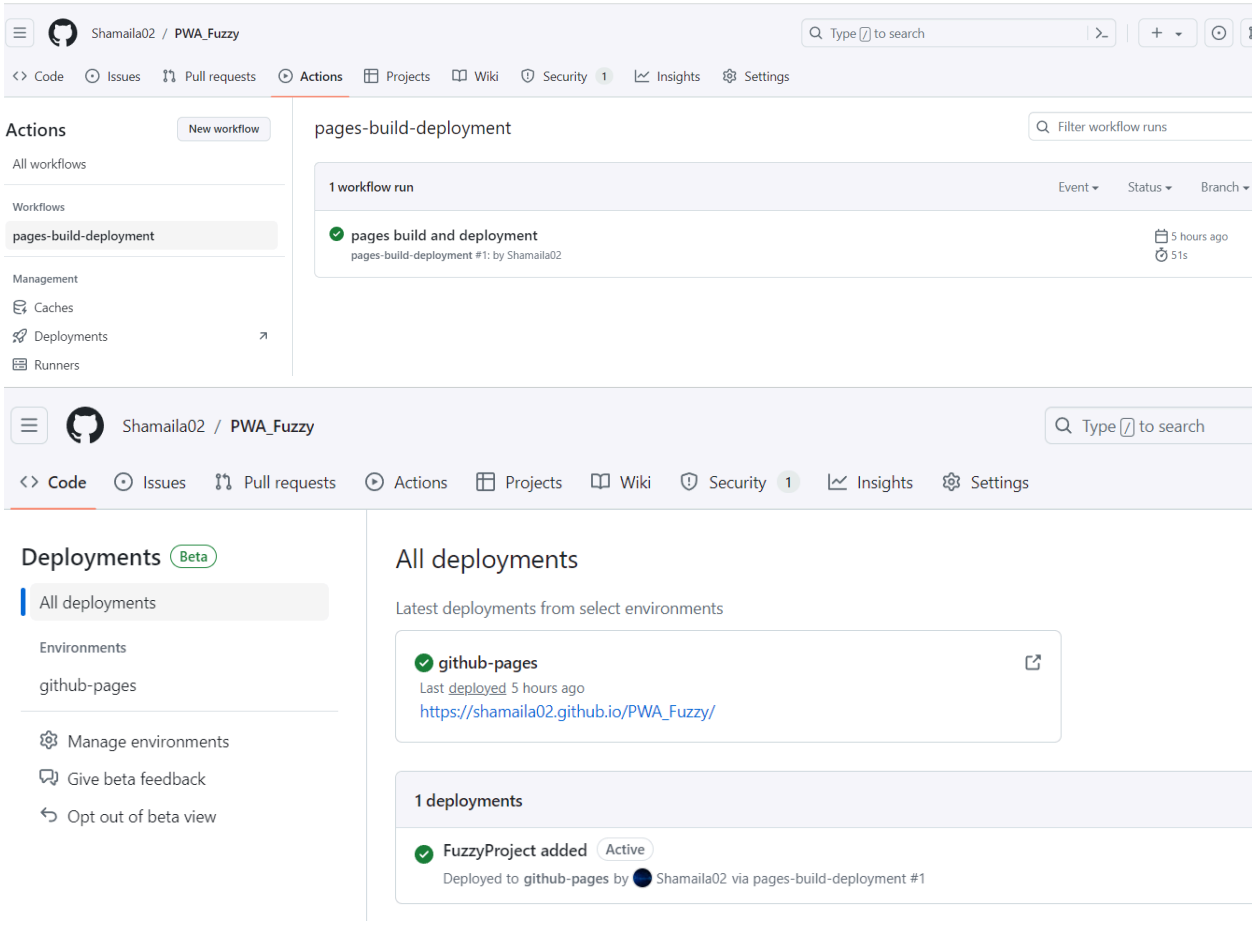
Hosted Link: https://shamaila02.github.io/PWA_Fuzzy/

Github Screenshot:





Project Title: Flutter-Myntra Clone /PWA- Fuzzy app **RollNo. 02**



Conclusion: In this experiment we have Successfully deployed the Ecommerce PWA to GitHub Pages.

MAD & PWA Lab
Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	02
Name	Shamaila Ansari
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	

Experiment 11

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

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

Google Lighthouse :

Google Lighthouse is a tool that lets you audit your web application based on a number of parameters including (but not limited to) performance, based on a number of metrics, mobile compatibility, Progressive Web App (PWA) implementations, etc. All you have to do is run it on a page or pass it a URL, sit back for a couple of minutes and get a very elaborate report, not much short of one that a professional auditor would have compiled in about a week.

The best part is that you have to set up almost nothing to get started. Let's begin by looking at some of the top features and audit criteria used by Lighthouse.

Key Features and Audit Metrics

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

1. **Performance:** This score is an aggregation of how the page fared in aspects such as (but not limited to) loading speed, time taken for loading for basic frame(s), displaying meaningful content to the user, etc. To a layman, this score is indicative of how decently the site performs, with a score of 100 meaning that you figure in the 98th percentile, 50 meaning that you figure in the 75th percentile and so on.
2. **PWA Score (Mobile):** Thanks to the rise of Service Workers, app manifests, etc., a lot of modern web applications are moving towards the PWA paradigm, where the objective is to make the application behave as close as possible to native mobile applications. Scoring

points are based on the Baseline PWA checklist laid down by Google which includes Service Worker implementation(s), viewport handling, offline functionality, performance in script-disabled environments, etc.

3. **Accessibility:** As you might have guessed, this metric is a measure of how accessible your website is, across a plethora of accessibility features that can be implemented in your page (such as the 'aria-' attributes like aria-required, audio captions, button names, etc.). Unlike the other metrics though, Accessibility metrics score on a pass/fail basis i.e. if all possible elements of the page are not screen-reader friendly (HTML5 introduced features that would make pages easy to interpret for screen readers used by visually challenged people like tag names, tags such as <section>, <article>, etc.), you get a 0 on that score. The aggregate of these scores is your Accessibility metric score.
4. **Best Practices:** As any developer would know, there are a number of practices that have been deemed 'best' based on empirical data. This metric is an aggregation of many such points, including but not limited to:Use of HTTPS

   Avoiding the use of deprecated code elements like tags, directives, libraries, etc. Password input with paste-into disabled

   Geo-Location and cookie usage alerts on load, etc.

Code&Implementation:

Index.html

manifest.json M | index.html

index.html > html > head

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

manifest.json M | style.css 5

assets > css > style.css > ...

```
body {
    font-family: "Poppins", sans-serif;
    max-width: 600px;
    width: 100%;
    margin: 0 auto;
    height: 100vh;
    background-color: rgba(var(--white), 1);
}
body::-webkit-scrollbar {
    width: 0;
}

h1 {
    font-weight: 600;
    font-size: 20px;
    line-height: 29px;
    margin-bottom: 0;
}

h2 {
    font-size: 16px;
    font-weight: 600;
    margin-bottom: 0;
}
```

Script.js

```
document.getElementById("loginForm").addEventListen
er("submit",(event)=>{
    event.preventDefault()
})

firebase.auth().onAuthStateChanged((user)=>{
    if(user){
        location.replace("/landing.html")
    }
})

function login(){
                            const       email       =
document.getElementById("email").value
                          const       password       =
document.getElementById("password").value

firebase.auth().signInWithEmailAndPassword(email,
password)
    .catch((error)=>{
          document.getElementById("error").innerHTML
= error.message
    })
}

function signUp(){
                        const      email_login      =
```

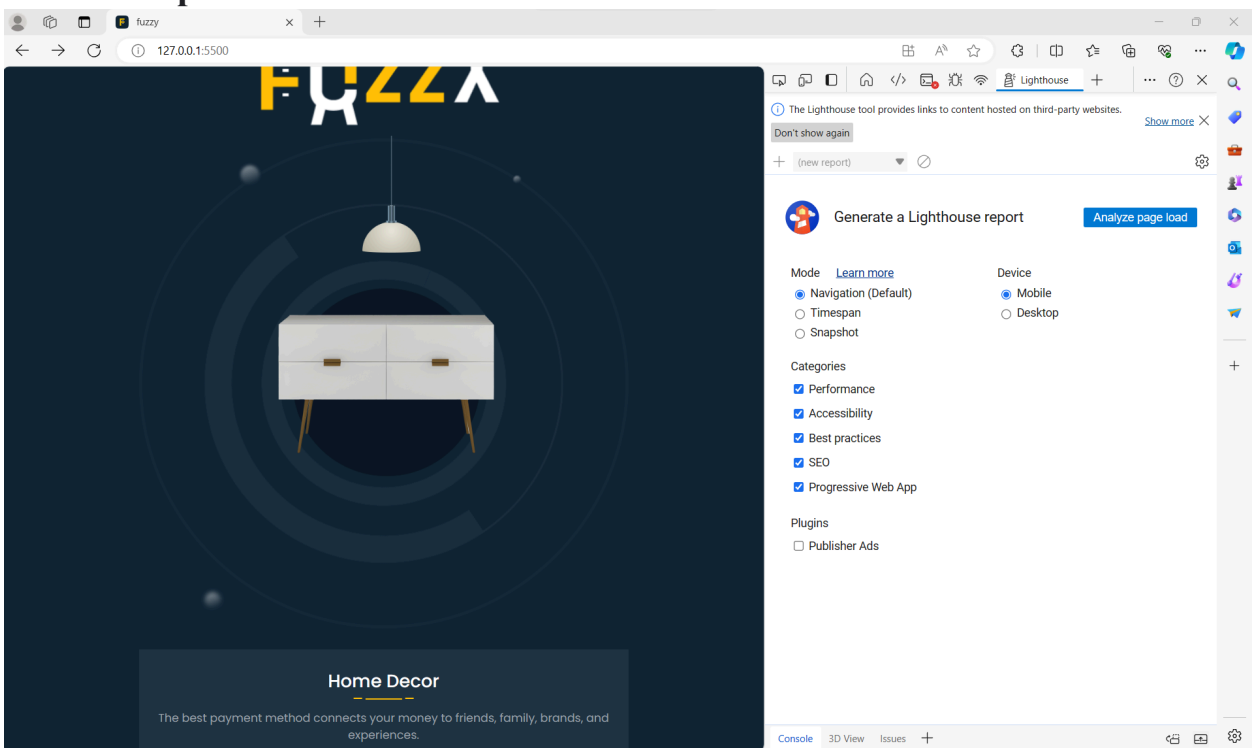
```
document.getElementById("email_login").value
                    const     password_login     =
document.getElementById("password_login").value

firebase.auth().createUserWithEmailAndPassword(emai
l, password)
    .catch((error) => {
          document.getElementById("error").innerHTML
= error.message
    });
}

function forgotPass(){
                           const       email       =
document.getElementById("email").value
    firebase.auth().sendPasswordResetEmail(email)
    .then(() => {
        alert("Reset link sent to your email id")
    })
    .catch((error) => {
          document.getElementById("error").innerHTML
= error.message
    });
}
```

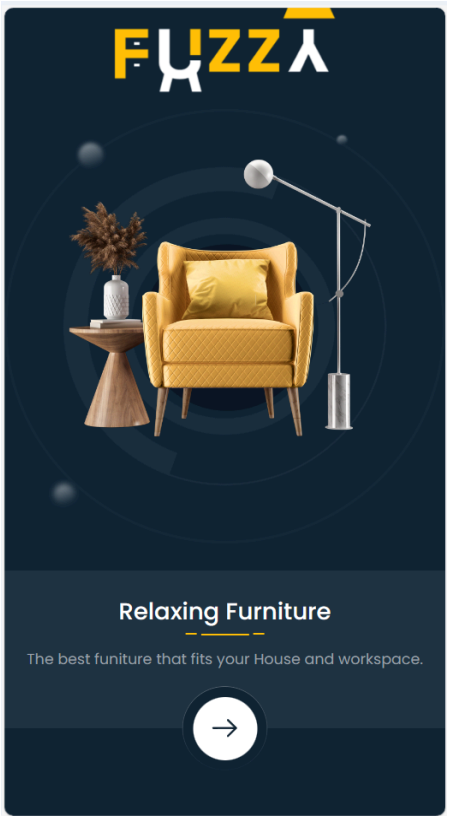
Manifest.json

```
{
  "name": "Fuzzy app",
  "short_name": "Fuzzy app",
```

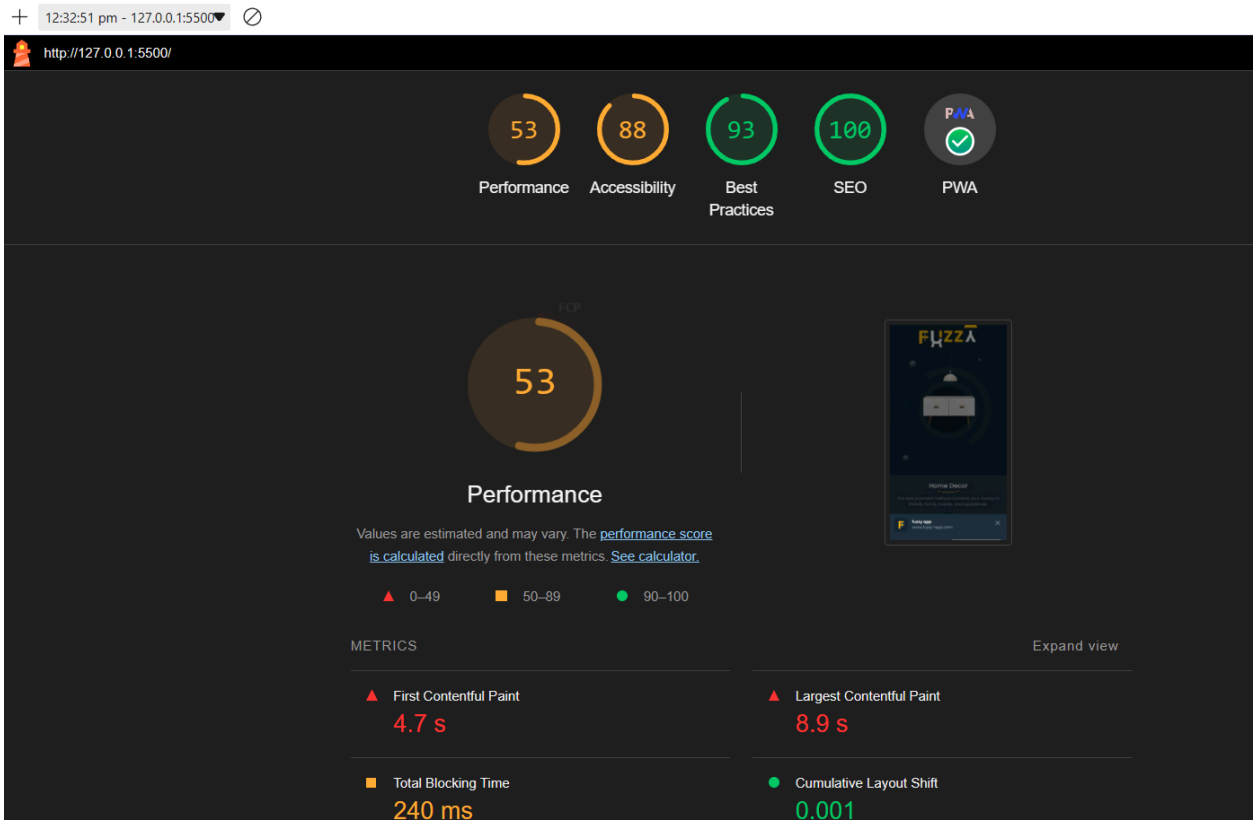
```
"lang": "en-US",
"start_url": ".",
"display": "standalone",
"background_color": "white",
"theme_color": "#122636",
"description": "fuzzy app",
"icons": [
  {
    "src": "assets/images/logo/512.png",
    "sizes": "144x144",
    "type": "image/png",
    "purpose": "any"
  },

  {
    "src": "assets/images/logo/512.png",
    "sizes": "512x512",
    "type": "image/png",
    "purpose": "any maskable"
  },
  {
    "src": "assets/images/icon.png",
    "sizes": "144x144",
    "type": "image/png",
    "purpose": "any maskable"
  }

]
```

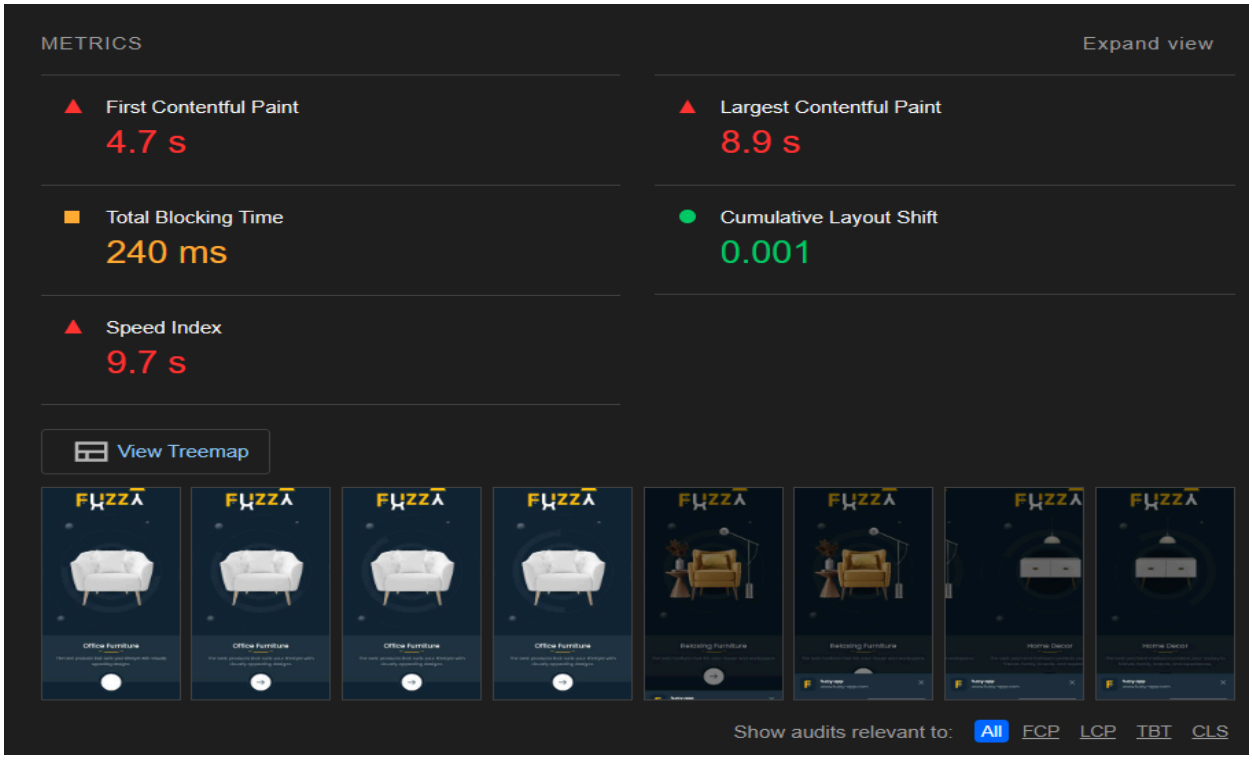
Output:
For Desktop Devices

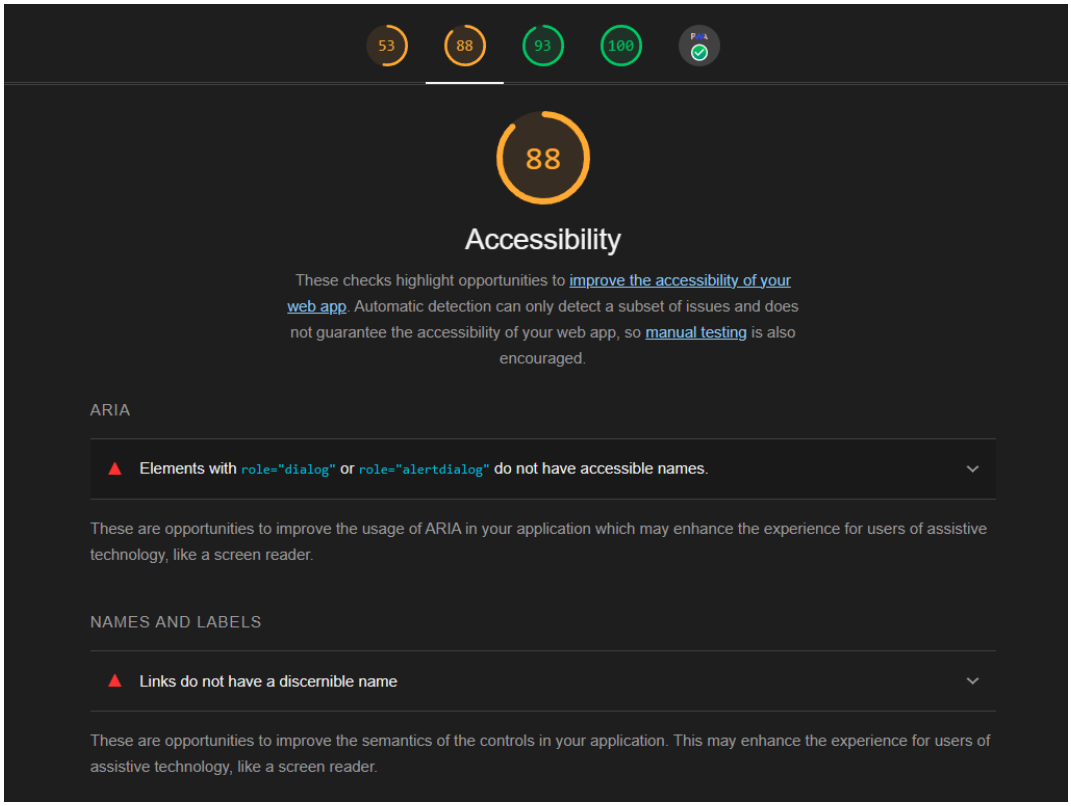


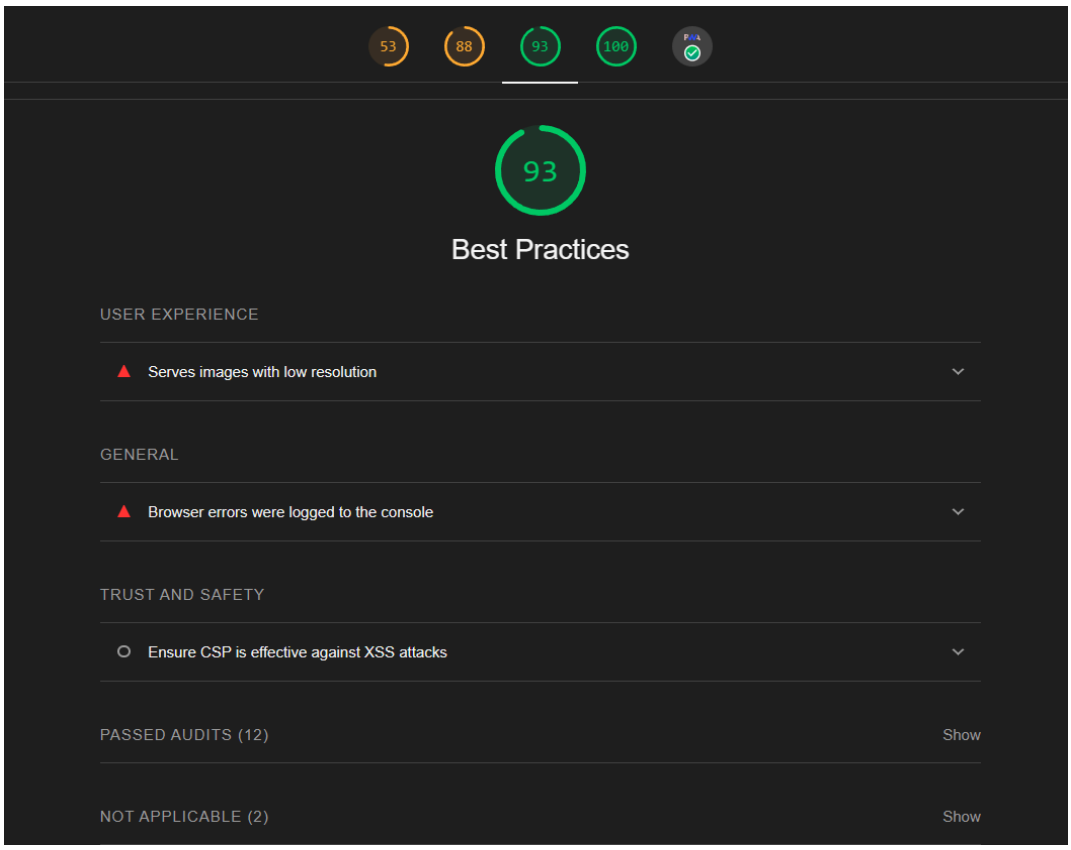


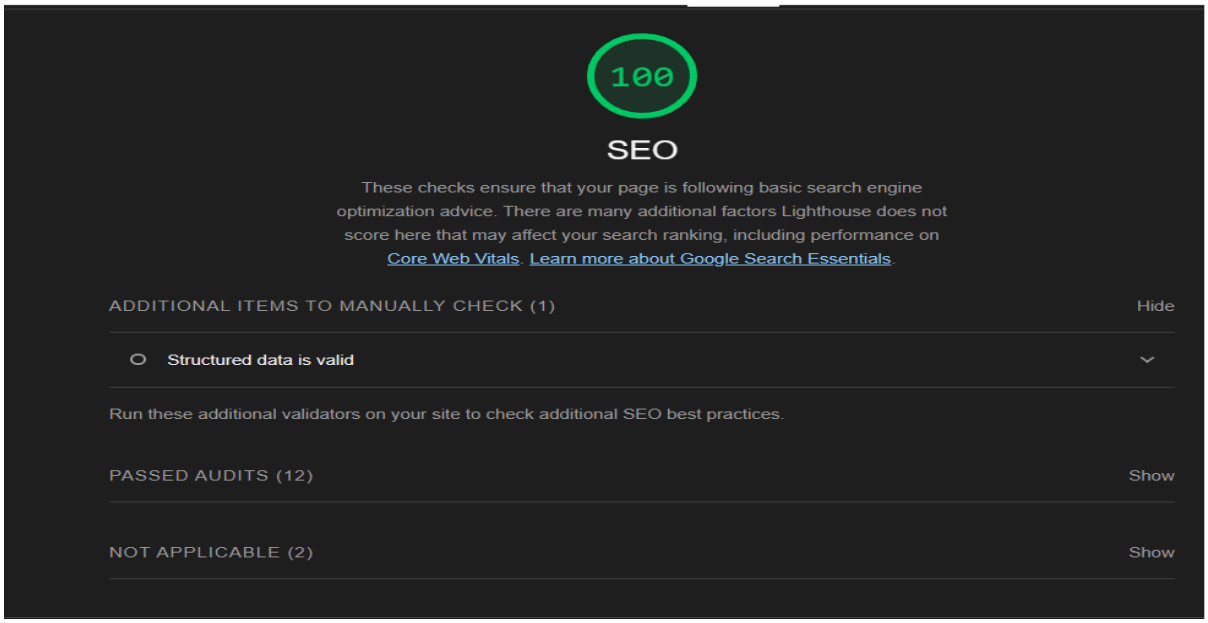
Project Title: Flutter-Myntra Clone /PWA- Fuzzy app **RollNo. 02**

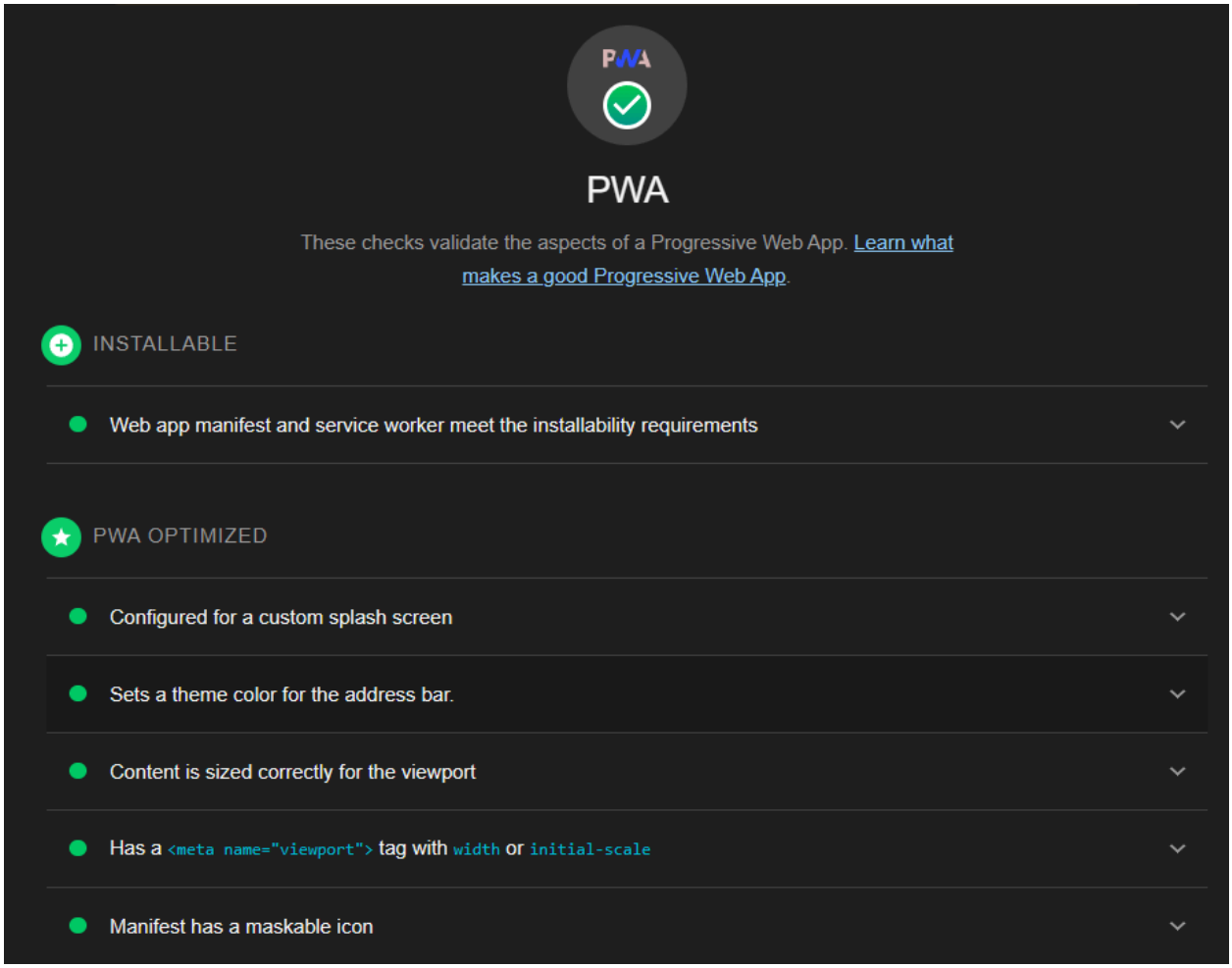


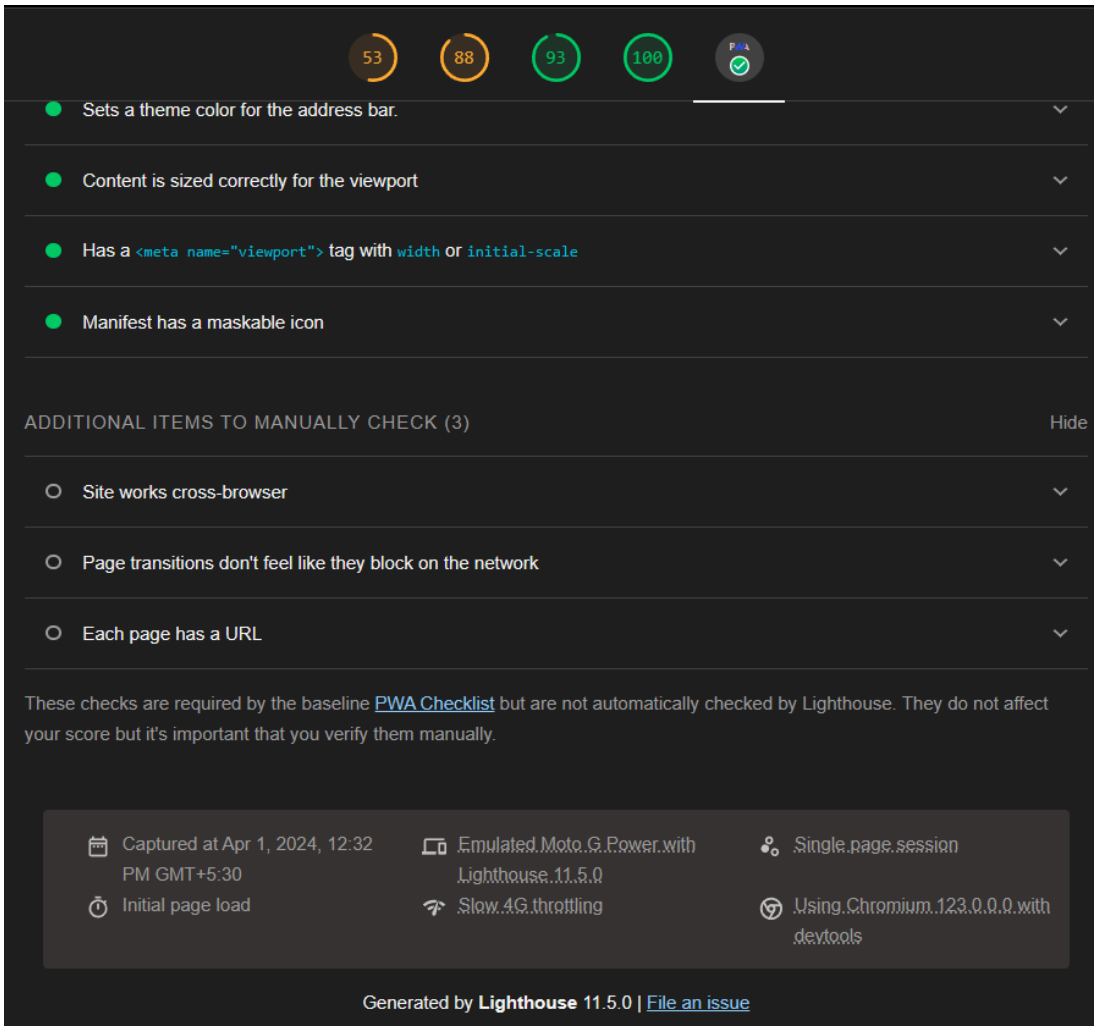



53
88
93
100
88
Accessibility
These checks highlight opportunities to improve the accessibility of your web app. Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so manual testing is also encouraged.
ARIA
Elements with role="dialog" or role="alertdialog" do not have accessible names.
These are opportunities to improve the usage of ARIA in your application which may enhance the experience for users of assistive technology, like a screen reader.
NAMES AND LABELS
Links do not have a discernible name
These are opportunities to improve the semantics of the controls in your application. This may enhance the experience for users of assistive technology, like a screen reader.


53
88
93
100
93
Best Practices
USER EXPERIENCE
Serves images with low resolution
GENERAL
Browser errors were logged to the console
TRUST AND SAFETY
Ensure CSP is effective against XSS attacks
PASSED AUDITS (12)
Show
NOT APPLICABLE (2)
Show


100
SEO
These checks ensure that your page is following basic search engine optimization advice. There are many additional factors Lighthouse does not score here that may affect your search ranking, including performance on Core Web Vitals. Learn more about Google Search Essentials.
ADDITIONAL ITEMS TO MANUALLY CHECK (1)
Hide
Structured data is valid
Run these additional validators on your site to check additional SEO best practices.
PASSED AUDITS (12)
Show
NOT APPLICABLE (2)
Show


PWA
These checks validate the aspects of a Progressive Web App. Learn what makes a good Progressive Web App.
INSTALLABLE
Web app manifest and service worker meet the installability requirements
PWA OPTIMIZED
Configured for a custom splash screen
Sets a theme color for the address bar.
Content is sized correctly for the viewport
Has a <meta name="viewport"> tag with width or initial-scale
Manifest has a maskable icon



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

MAD & PWA Lab
Journal

Experiment No.	Assignment-1
Assignment 1 Questions	1. Flutter Overview:Explain the key features and advantages of using Flutter for mobile app development. Discuss how the Flutter framework differs from traditional approaches and why it has gained popularity in the developer community. 2. Widget Tree and Composition: Describe the concept of the widget tree in Flutter. Explain how widget composition is used to build complex user interfaces. Provide examples of commonly used widgets and their roles in creating a widget tree. 3. State Management in Flutter: Discuss the importance of state management in Flutter applications. Compare and contrast the different state management approaches available in Flutter, such as setState, Provider, and Riverpod. Provide scenarios where each approach is suitable. 4. Firebase Integration in Flutter: Explain the process of integrating Firebase with a Flutter application. Discuss the benefits of using Firebase as a backend solution. Highlight the Firebase services commonly used in Flutter development and provide a brief overview of how data synchronization is achieved.
Roll No.	02
Name	Shamaila Ansari
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:	

Shamaila Ansari
D15A Roll No: 02

Assignment-1

1. Flutter Overview :-

Ans: Flutter is an open-source UI software development toolkit created by Google. It's designed to help developers build natively compiled applications for mobile, web, and desktop from a single codebase.

Key Features

1. Single codebase: Developers can write code once and deploy it on multiple platforms, such as iOS, Android, Web and desktop.

2. Widgets: Flutter uses a reactive framework where the user interface is built using widgets. widgets are UI components that can be customized and combined to create complex UIs.

3. Hot Reload: one of the standout features, Hot Reload allows developers to instantly see the effect of code changes without restarting the app. It speeds up the development process and makes it easier to experiment with different UI elements.

4. Expressive UI: Flutter offers a rich set of pre-designed widgets for unique designs.

5. performance: Flutter apps are compiled to native ARM code, providing high performance and smooth animations.

6. Dart programming language: Flutter uses dart as its programming language. Dart is designed for building modern web and mobile apps and has feature like strong typing and Just-In-Time (JIT) compilation

7. Community and Ecosystem: Flutter has a growing and active community, which means plenty of resources, packages, and plugins are availabl available

Flutter's ability to streamline the dvelopment process, provide a consistent UI across platforms, and offer feature like Hot Reload and high performance has made it stand out in comparison to traditional approaches. The framework addresses many pain points in cros

Chamila Ansari

platform development, making it an attractive choice for developers seeking efficiency, consistency, and a modern development experience.

Q.2. Widget Tree and Composition;

Ans: In flutter, the widget tree and Composition are fundamental concepts that contribute to building the user interface.

Widget Tree: In flutter, the UI is represented as a tree of widgets, where each widget is a part of the UI. widgets can be simple like text or buttons, or complex, like a custom-designed card or a list. The widget tree represents the hierarchy and structure of your UI.

Example:

```
//Root widget
MyApp (

home: Scaffold (
appBar: AppBar (
  title: Text ('My Flutter App'),

),
```

```
body: Center (
  child: Text ('Hello, Flutter!'),
),
),
);
```

In the above example:

'My App' is the root widget.

'Scaffold' is a container widget.

'AppBar' and 'Center' are child widgets of 'Scaffold'.

'Text' is a child widget of 'Center'.

The widget tree represents the hierarchical structure of your UI, making it easy to organize and understand the components.

- Widget Composition:

Widget composition in flutter refers to combining multiple widgets to create more complex and reusable UI elements. This is acheived by

nesting widgets inside each other or creating custom widgets that encapsulate specific functionality.

Example:

```
class MyButton extends StatelessWidget
{
  final String label;
  final VoidCallback onPressed;

  MyButton({required this.label, required
      this.onPressed});

  @override

  Widget build(BuildContext context)
  {
    return ElevatedButton(
      onPressed: onPressed,
      child: Text(label),

    );
  }
}
```

Now, you can use this custom button widget in your app:

```
MyApp (
  home: Scaffold (
  appBar: AppBar (
  title: Text ('My Flutter App'),
),

body: Center (
  child: MyButton (
  label: 'press Me',
  onPressed: () {
    print ('Button pressed!');

  },
),
),
);
```

- 'MyButton' is a custom widget composed of an 'ElevatedButton' and a 'Text' widget.
- 'MyButton' widget takes parameters like 'label' and 'onPressed', and making it versatile and reusable.
- The custom button widget is then used with in the widget tree.

understanding the widget tree and widget composition is crucial in flutter development, as it forms the basis for creating interactive and visually app

Q.3 State management in flutter:

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

- In flutter, UI components can be either stateful or stateless
- Stateful widgets have mutable state, that can change during the lifetime of the widget, such as user interactions or data fetching.

* State management approaches in flutter

1. setState:

- The set state method is a basic and built-in mechanism for managing state in flutter. It belongs to 'StatefulWidget' class and when called, it triggers a rebuild of the widget subtree.

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

2) Provider: provider is a third-party package used for state management in Flutter. It follows the provider pattern and allows widgets to listen to changes in a provided object, typically a data model or a service.

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

3. Riverpod - Riverpod is an extension of provider and focuses on making the provider pattern more robust and scalable. It provides a more advanced and flexible approach to dependency injection and state mangement.

- Suitable for longer applications with complex state management requirement.

Comparison:

1. setState:

Pros:

1. Simplicity: Easy to understand and implement
2. Built-in: No additional packages or dependencies required.

* Cons:

1. Limited Scalability Becomes challenging to manage in larger applications.
2. Global update: Can trigger unnecessary rebuilds of the entire widget subtree.

2) Provider:

Pros:

1. Centralized management.
2. Easy Integration

Cons:

1. learning curve
2. Not designed for complex scenarios.

3. Riverpod:

Pros:
1. Advanced features
2. Scalability

Cons:
1. Learning curve: More complex than basic provider

Q.4] Firebase Integration in Flutter:

Ans: Integrating firebase with Flutter:

1. Create a Firebase project.
2. Add Firebase to Flutter project: In the Flutter project add the Firebase dependencies in the 'pubspec.yaml' file Run 'Flutter pub get' to install the dependencies.
3. Initialize Firebase: Initialize Firebase in your app by calling 'Firebase.initializedApp()' in your 'main.dart' early entry point in your app.
4. Authentication: For this the firebase_auth package is used.

* Benefits of firebase as a Backend Solution:

1. Authentication: firebase authentication offers ready-to-use authentication methods, including email password, Social logins and phone authentication.

2. Cloud functions: serverless cloud functions allow running backend code without managing servers.

3. cloud storage: firebase cloud Stro Storage offers scalable and secure Cloud Storage for user genrated content like Images and files.

4. Hosting: firebase hosting allows dyploying web applications quickly, providing a global content delivery network (CDN) for better performance

* Firebase Servics commonly used in flutter.

1. Authentication (firebase_auth).
2. Firestore database ('cloud_firestore')
3. Firebase Storage ('firebase_storage')

4. Cloud functions: Serverless functions that can be triggered by events in firebase services or HTTP request

* Data Synchronization in firebases:

1. Firebase achieves data synchronization through its real-time database (FireStore).

2. When data changes in the database, the changes are immediately pushed to all connected clients.

3. Clients can listen to specific documents or collections, and any changes trigger real-time updates in the UI.

4. Firebase real-time data synchronization simplifies the implementation of dynamic and responsive applications making it a popular choice for flutter developer.

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.	02
Name	Shamaila Ansari
Class	D15A
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:	

Shamaila Ansari
D15A - 02

PWA Assignment - 02

Q1:

Ans A Progressive web App (PWA) is a type of web application that leverages modern web technologies to provide users with a native app-like experience directly through a web browser. PWAs are designed to be fast, reliable and engaging, offering features such as offline functionality, push notifications and access to device hardware like camera and geolocation.

Following are the key characteristics:

1. Cross-platform compatibility: PWAs are built using standard web technologies (HTML, CSS, Javascript) making them compatible with various devices and platforms including desktops, traditional mobile apps often require separate development efforts for different platforms (eg. iOS, android)
2. Offline functionality: PWAs can work offline or with a limited internet connection, thanks to features like service workers, which cache resources and enable offline access to previously visited content. This capability allows users to continue using the app even when they're offline, enhancing user experience.

3. Responsive Design:- PWAs are designed to be responsive, adapting seamlessly to different screen sizes and orientations. This ensures a consistent user experience across various devices, whether users access the apps on a desktop computer, smartphone or tablet.
4. Fast Peryormance: PWAs are optimized for speed and performance, allowing for quick loading times and smooth interactions. By minimizing network requests and efficiently caching resources, PWAs deliver a fast and responsive user experience, similar to native apps.
5. Engagement features: PWAs can engage users through features like push notifications, which enable real-time communication and re-engage users even when the app is not actively open. This helps increase user retention and keeps users informed about new content or updates.
6. Installation and Discoverability: PWAs can be installed directly from the web browser without the need for app distribution. This simplifies the installation process for user and improves app discoverability.

Q2

Ans 1) Responsive web design is an approach to web development that aims to create websites that provide an optimal viewing and interaction experience across a wide range of devices and screen sizes.

2) This is achieved by using flexible layouts, fluid grids, and media queries to adapt the design and content of the website based on the characteristics of the device and browser being used.

3) The importance of responsive web design in the context of Progressive Web Apps (PWAs) lies in its ability to ensure that PWAs are accessible and usable on various devices, including desktop computer, smartphones, tablets, and other mobile devices.

4) By implementing responsive design principles PWAs can dynamically adjust their layout, typography and user-friendly experience across different screen sizes and resolutions.

5) This helps enhance user engagement, improve accessibility and increases the reach of PWAs to a broader audience

Responsive Web design	Fluid web design	Adaptive web design
1. Adapts the layout and content dynamically based on the viewport size using flexible grids fluid layouts and media queries.	1. Utilizes flexible layouts and elements that scale smoothly with the viewport size, maintaining proportion and readability across different screen sizes.	1. Creates multiple versions of the website, each optimized for specific devices or screen sizes and serves the appropriate version based on device.
2. Ensures PWAs are accessible and usable across various devices, improving user experience and engagement.	2. Contributes to a consistent and visually appealing user experience by ensuring that content flows and adjusts smoothly across different devices.	2. Allows for greater control over the user experience on different devices by tailoring the design and content to specific device types or screen sizes.

Responsive web design	Fluid web design	Adaptive web design
3. Utilizes flexible grids, fluid layouts, and media queries to adjust styling based on viewport size.	3. Uses flexible widths percentages for layout and element sizes and scalable images and media to ensure smooth scalability.	3. Involves creating multiple fixed width layouts, server-side or client side detection methods, and targeted optimizations for specific devices or screen sizes.
4. One design adapts to various view port sizes through CSS media queries and flexible layouts	4. Content and layout elements smoothly scale with the view port size, maintaining proportions and readability.	4. Multiple versions of the website are created, each targeting specific devices or screen sizes and served accordingly.
5. Adapts layout and content based on viewport size ensuring a consistent user experience across devices.	5. Maintains readability and usability by scaling elements smoothly with the viewport size.	5. Offers tailored experience for different devices or screen size, potentially providing more optimized experience.

Q3.

Ans The lifecycle of service workers involves three main phases: registration, installation and activation. Service workers are type of Javascript code that runs in the background of a web application, enabling features like offline caching, handling push notifications and background sync.

1. Registration:-
   a) Registration is the process of telling the browser to start installing a service worker for a specific URL scope.
   b) Service worker registration typically occurs in the main Javascript file of a web application. Developers use the 'navigator.serviceWorker.register()' method to register the service worker file.
2. Installation:
   a) Installation occurs when the browser fetches and installs the service worker script for the first time.
   b) Installation is triggered immediately after registration if the service worker file is new or if there have been changes to the existing file.

c) During installation, the 'install' event is fired in the service worker script.

3. Activation:-

a) Activation occurs when the installed service worker becomes active and starts controlling pages within its scope.

b) After installation is complete, the service worker enters the 'waiting' state until all tabs using the old version of the service worker are ~~not~~ closed.

c) Once activated, the 'activate' event is fired. Developers can use the 'activate' event to clean up old caches, update databases, or perform other tasks related to the activation of the service worker.

Q4

Ans 1. IndexedDB is a client side storage mechanism provided by modern web browsers allowing web applications to store large amounts of structured data persistently.

2. In the context of service workers, IndexedDB is often used to cache data for offline access, manage background synchronization and store other application data needed for offline functionality.

Following are the steps of how IndexedDB is used in the service worker for data storage:-

1. Offline caching: i) Service workers can intercept network requests and cache responses using IndexedDB.
   ii) When a web application makes a request for a specific resources (eg. HTML, CSS, JS images), the service worker can store the response data in IndexedDB for future offline access.
   iii) This enables the web application to continue functioning even when the device is offline, serving cached resources from IndexedDB instead of making network requests.
2. Background sync: i) IndexedDB can be used to store data temporarily when the background sync events occurs.
   ii) For example, if a web application needs to synchronize data with a server while the device is offline, the service worker can queue the data in IndexedDB until a network connection becomes available.
   iii) Once the connection is restored, the service worker can retrieve the queued data from IndexedDB and synchronizes it with the server.

3. Data Storage: i) Service workers can utilize IndexedDB to store application data needed for offline functionality or other purposes.
   i) This data can include user preferences, or any other structured data required ~~for~~ by the web application.
4. Indexed DB API: i) Service workers interact with IndexedDB using the IndexedDB API, which provides methods for opening databases, creating objects stores, storing and retrieving data, and handling transactions.
   ii) Service Workers can use asynchronous IndexedDB operations to perform database operations without blocking the main threads, ensuring smooth performance and responsiveness.