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 **Mayank Pahuja** 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 **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:- Rapido Clone/ PWA-Tourly **Roll No.40**

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	17/01	24/01	11
2.	To design Flutter UI by including common widgets.	LO2	24/01	31/01	15
3.	To include icons, images, fonts in Flutter app	LO2	31/01	07/02	12
4.	To create an interactive Form using form widget	LO2	07/02	14/02	12
5.	To apply navigation, routing and gestures in Flutter App	LO2	14/02	21/02	12
6.	To Connect Flutter UI with fireBase database	LO3	21/02	06/03	13
7.	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.	LO4	06/03	29/03	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	13/03	29/03	15
9.	To implement Service worker events like fetch, sync and push for E-commerce PWA	LO5	20/03	29/03	15
10.	To study and implement deployment of Ecommerce PWA to GitHub Pages.	LO5	27/03	29/03	15
11.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.	LO6	27/03	29/03	15
12.	Assignment-1	LO1,LO2 ,LO3	28/01	05/02	05
13.	Assignment-2	LO4,LO5 ,LO6	14/03	21/03	04

MAD & PWA Lab
Journal

Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	40
Name	Mayank Pahuja
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	11

Experiment No: 01

Name:- Mayank Pahuja **Class:- D15-A Roll No:- 40**

Installation and Configuration of 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.

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

Step 3: When your download is complete, extract the **zip** file and place it in the desired installation folder or location, for example, C: /Flutter.

Step 4: To run the Flutter command in regular windows console, you need to update the system path to include the flutter bin directory. The following steps are required to do this:

Step 4.1: Go to MyComputer properties -> advanced tab -> environment variables. You will get the following screen.

Step 4.2: Now, select path -> click on edit. The following screen appears

Step 4.3: In the above window, click on New->write path of Flutter bin folder in variable value > ok -> ok -> ok.

Step 5: Now, run the $ **flutter** command in command prompt.

Now, run the $ **flutter doctor** command. This command checks for all the requirements of Flutter app development and displays a report of the status of your Flutter installation.

Step 6: When you run the above command, it will analyze the system and show its report, as shown in the below image. Here, you will find the details of all missing tools, which required to run Flutter as well as the development tools that are available but not connected with the device.

Step 7: Install the Android SDK. If the flutter doctor command does not find the Android SDK tool in your system, then you need first to install the Android Studio IDE. To install Android Studio IDE, do the following steps.

Step 7.1: Download the latest Android Studio executable or zip file from the official site.

Step 7.2: When the download is complete, open the **.exe** file and run it. You will get the following dialog box.

Step 7.3: Follow the steps of the installation wizard. Once the installation wizard completes, you will get the following screen.

Step 7.4: In the above screen, click Next-> Finish. Once the Finish button is clicked, you need to choose the 'Don't import Settings option' and click OK. It will start the Android Studio.

Step 7.5 run the $ **flutter doctor** command and Run flutter doctor --android-licenses command.

Step 8: Next, you need to set up an Android emulator. It is responsible for running and testing the Flutter application.

Step 8.1: To set an Android emulator, go to Android Studio > Tools > Android > AVD Manager and select Create Virtual Device. Or, go to Help->Find Action->Type Emulator in the search box. You will get the following screen.

Step 8.2: Choose your device definition and click on Next.

Step 8.3: Select the system image for the latest Android version and click on Next.

Step 8.4: Now, verify the all AVD configuration. If it is correct, click on Finish. The following screen appears.

Step 8.5: Last, click on the icon pointed into the red color rectangle. The Android emulator displayed as below screen.

Step 9: Now, install Flutter and Dart plugin for building Flutter application in Android Studio. These plugins provide a template to create a Flutter application, give an option to run and debug Flutter application in the Android Studio itself. Do the following steps to install these plugins.

Step 9.1: Open the Android Studio and then go to File->Settings->Plugins.

Step 9.2: Now, search the Flutter plugin. If found, select Flutter plugin and click install. When you click on install, it will ask you to install Dart plugin as below screen. Click yes to proceed.

Step 9.3: Restart the Android Studio.

Question:- Write Code to print the *hello your_name* in the output.

Code:-

```
import 'package:flutter/material.dart';
void main() {
  runApp(MyApp());
}
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
        return MaterialApp(
        home: Scaffold(
        appBar: AppBar(
        title: Text('Hello App'),
        ),
        body: Center(
        child: Text(
        'Hello Pahuja Mayank :)',
        style: TextStyle(fontSize: 24,
fontWeight: FontWeight.bold),
        ),
        ),
        ),
        );
  }
}
```

Output:-

MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	40
Name	Mayank Pahuja
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

Experiment No:- 02

Name:- Mayank Pahuja **Class:- D15-A RollNo:- 40**

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

Theory: In Flutter, widgets are the building blocks of the user interface, and several common widgets play crucial roles in creating engaging and interactive applications. Here's a brief overview of some fundamental Flutter widgets:

1. Container: The most basic building block, a container is a box model that can contain other widgets, allowing you to customize its dimensions, padding, and decoration.
2. Row and Column: These widgets help organize children widgets horizontally (Row) or vertically (Column), facilitating the creation of flexible and responsive layouts.
3. AppBar: AppBar is a material design widget providing a top app bar that typically includes the app's title, leading and trailing icons, and actions.
4. ListView: Used to create scrollable lists of widgets, ListView is versatile for displaying a large number of items efficiently.
5. TextField: Enables users to input text, providing a text editing interface with options for validation, styling, and interaction.
6. RaisedButton and FlatButton: These button widgets create interactive elements for users to trigger actions, with RaisedButton offering a raised appearance and FlatButton a flat design.
7. Image: The Image widget displays images from various sources, supporting both local and network images.
8. Scaffold: A top-level container for an app's visual elements, Scaffold provides a structure that includes an AppBar, body, and other optional features like drawers and bottom navigation.
9. Card: Representing a material design card, this widget displays information in a compact and visually appealing format, often used for grouping related content.
10. GestureDetector: Allows detection of various gestures like taps, drags, and long presses,enabling interactive responses to user input.

Code & Output:-

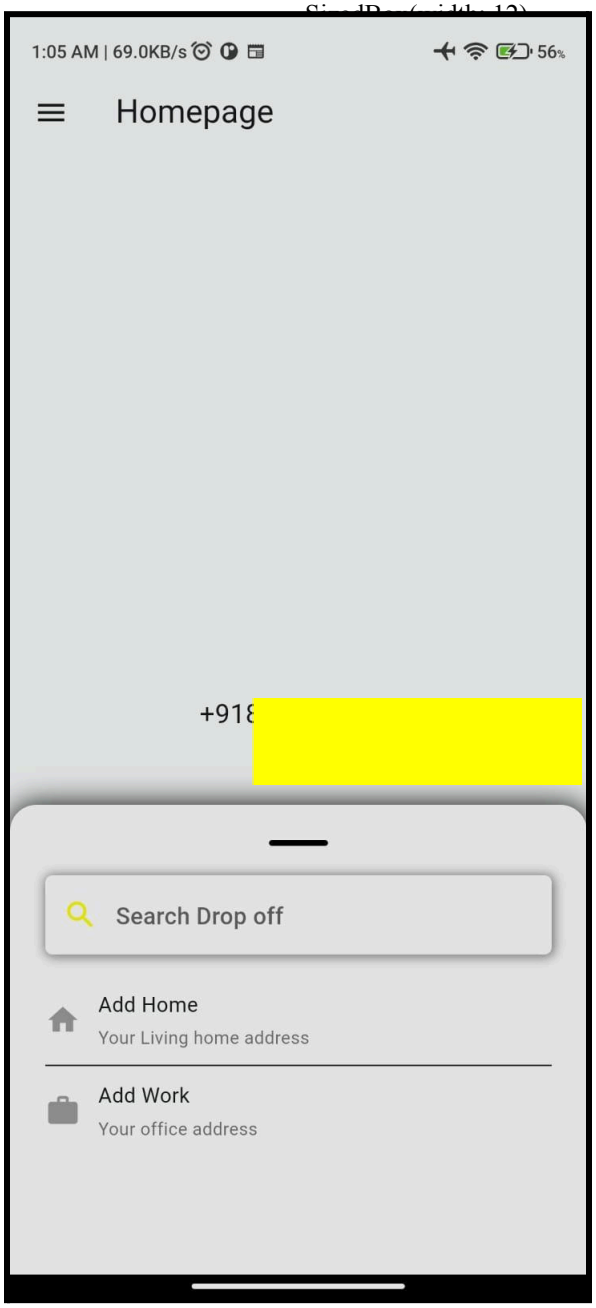
```
 ignore_for_file: prefer_const_constructors,
prefer_const_literals_to_create_immutables
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:rapido/AllWidgets/Divider.dart';
class Homepage extends StatefulWidget {
  const Homepage({Key? key}) : super(key: key);
  @override
  State<Homepage> createState() =>
HomePageState();
}

class HomePageState extends State<Homepage> {
  final User? user =
FirebaseAuth.instance.currentUser;
```

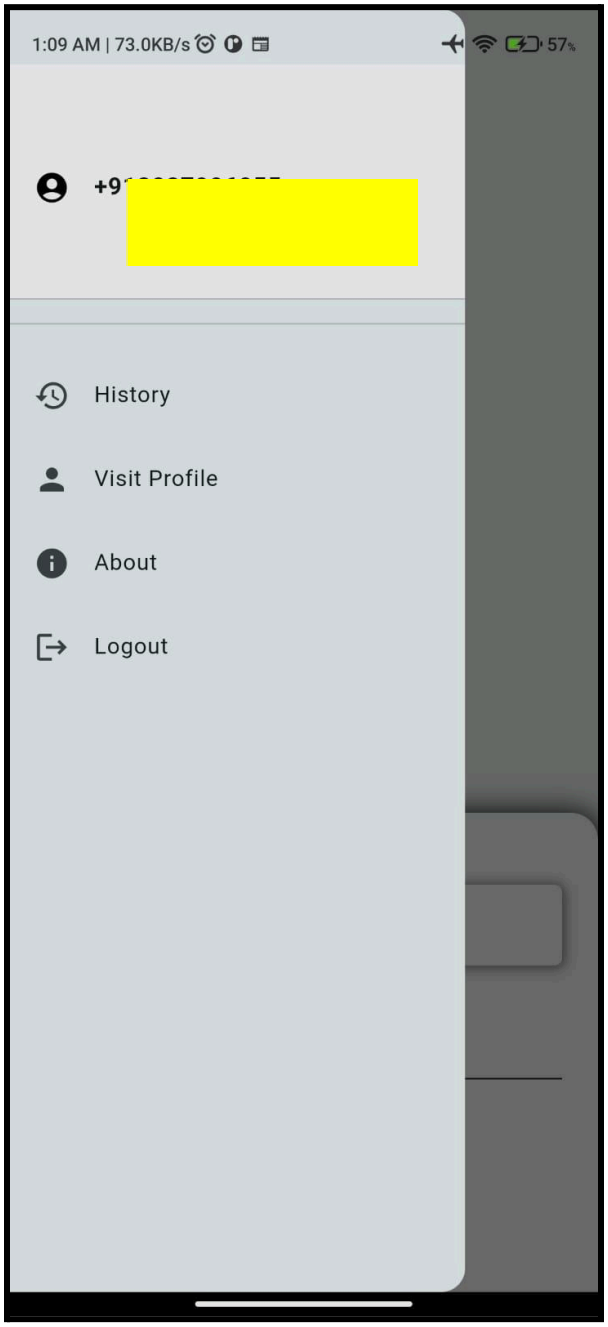
```
 GlobalKey<ScaffoldState> scaffoldKey =
GlobalKey<ScaffoldState>();
 Future<void> signOut() async {
  await FirebaseAuth.instance.signOut();
 }
 bool _isDragging = false;
 Offset _offset = Offset(0, 0);
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   key: scaffoldKey,
   appBar: AppBar(
    title: Text("Homepage"),
   ),
   drawer: Drawer(
    child: ListView(
     children: [
      GestureDetector(
       onTap: () {
        scaffoldKey.currentState?.openDrawer();
       },
       child: Container(
        height: 165.0,
        child: DrawerHeader(
         decoration: BoxDecoration(color:
Colors.white),
          child: Row(
           children: [
            Icon(Icons.account_circle, color:
Colors.black),
             SizedBox(width: 16.0),
             Column(
              mainAxisAlignment:
MainAxisAlignment.center,
               children: [
                Text(
                 "${user!.phoneNumber}",
                 style: TextStyle(
                  fontSize: 16,
                  fontWeight: FontWeight.bold,
                 ),
                ),

               ],
              ),
             ],
            ),
           ),
          ),
         ),
         Divider(),
         SizedBox(height: 12),
         ListTile(
          leading: Icon(Icons.history),
          title: Text("History", style:
TextStyle(fontSize: 15)),
          onTap: () {
           Navigator.pushNamed(context, '/history');
          },
         ),
         ListTile(
          leading: Icon(Icons.person),
          title: Text("Visit Profile", style:
TextStyle(fontSize: 15)),
          onTap: () {
           Navigator.pushNamed(context,
'/visit_profile');
          },
         ),
         ListTile(
          leading: Icon(Icons.info),
          title: Text("About", style:
TextStyle(fontSize: 15)),
          onTap: () {
           Navigator.pushNamed(context, '/about');
          },
         ),
         ListTile(
          leading: Icon(Icons.logout),
          title: Text("Logout", style:
TextStyle(fontSize: 15)),
          onTap: () {
           signOut();
          },
         ),
        ],
       ),
      ),
      body: Stack(
       children: [
        Center(
         child: Text(
          "${user!.phoneNumber}",
          style: TextStyle(fontSize: 18),
         ),
        ),
        Positioned(
         left: 0.0,
```

```
        right: 0.0,
        bottom: _isDragging ? _offset.dy : 0.0,
        child: GestureDetector(
         onPanUpdate: (details) {
          setState(() {
           _isDragging = true;
           _offset = Offset(0, _offset.dy +
details.delta.dy / 2); // Adjust the speed
          });
         },
         onPanEnd: (details) {
          setState(() {
           _isDragging = false;
          });
         },
         child: Container(
          height: 320,
          decoration: BoxDecoration(
           color: Colors.white,
           borderRadius: BorderRadius.only(
            topLeft: Radius.circular(18),
            topRight: Radius.circular(18),
           ),
           boxShadow: [
            BoxShadow(
             color: Colors.black,
             blurRadius: 16,
             spreadRadius: 0.6,
             offset: Offset(0.7, 0.7),
            ),
           ],
          ),
          child: Padding(
           padding: const
EdgeInsets.symmetric(horizontal: 24, vertical: 18),
           child: Column(
            crossAxisAlignment:
CrossAxisAlignment.start,
            children: [
             SizedBox(height: 6),
             Row(
              mainAxisAlignment:
MainAxisAlignment.center,
              children: [
               Container(
                height: 4,
                width: 40,
                decoration: BoxDecoration(
                 color: Colors.black,
                 borderRadius:
BorderRadius.circular(2),
                ),
               ),
              ],
             ),
             SizedBox(height: 20),
             Container(
              decoration: BoxDecoration(
               color: Colors.white,
               borderRadius:
BorderRadius.circular(5.0),
               boxShadow: [
                BoxShadow(
                 color: Colors.black54,
                 blurRadius: 6,
                 spreadRadius: 0.6,
                 offset: Offset(0.7, 0.7),
                ),
               ],
              ),
              child: TextFormField(
               onTap: () {

FocusScope.of(context).requestFocus(FocusNode());
               },
               decoration: InputDecoration(
                prefixIcon: Icon(Icons.search, color:
Colors.yellowAccent),
                hintText: "Search Drop off",
                border: InputBorder.none,
                contentPadding:
EdgeInsets.symmetric(vertical: 15, horizontal: 20),
               ),
              ),
             ),
             SizedBox(height: 24),
             Row(
              children: [
               Icon(Icons.home, color: Colors.grey),
               SizedBox(width: 12),
               Column(
                crossAxisAlignment:
CrossAxisAlignment.start,
                children: [
                 Text("Add Home"),
                 SizedBox(height: 4),
                 Text(
                  "Your Living home address",
```

```
                    style: TextStyle(color:
Colors.black54, fontSize: 12),
                  ),
                ],
              ),
            ],
          ),
          SizedBox(height: 10),
          DividerWidget(),
          SizedBox(height: 10),
          Row(
            children: [
              Icon(Icons.work, color: Colors.grey),
```

```
                Column(
                  crossAxisAlignment:
CrossAxisAlignment.start,
                  children: [
                    Text("Add Work"),
                    SizedBox(height: 4),
                    Text(
                      "Your office address",
                      style: TextStyle(color:
Colors.black54, fontSize: 12),
                    ),],),],),],),),),),),],),);}}
```





MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	40
Name	Mayank Pahuja
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	12

Experiment No:- 03

Name:- Mayank Pahuja **Class:- D15-A RollNo:- 40**

Aim: To include images and fonts in flutter app

Theory: Certainly! Here's a simplified process for adding images in Flutter:

1. I**mport Libraries:**

Ensure that you have the necessary libraries imported in your Dart file. For images, you'll typically use `dart:ui` and other relevant Flutter packages.

2. **Adding Local Images:**

- Place your local images in the `assets` folder.
- Declare the images in the `pubspec.yaml` file.

3. **Adding Network Images:**

- Use the `Image.network` widget for displaying images from the internet.

4. **Image Widget:**

- Create an `Image` widget and provide it with an `ImageProvider`.
- Use `AssetImage` for local images and `NetworkImage` for network images.

5. **ImageProvider:**

- Understand that `AssetImage` and `NetworkImage` are subclasses of the `ImageProvider` class.
- You can create custom `ImageProvider` if needed.

6. **CachedNetworkImage (Optional):**

- If you want to cache network images, consider using the `cached_network_image` package.

7. **Image Loading and Error Handling**:

- Customize the loading and error behavior using `loadingBuilder` and `errorBuilder` properties of the `Image` widget or other relevant widgets.

Remember, the actual implementation details might vary based on your specific use case and the packages you choose to use. The key is to understand the concepts of working with local and network images and the various widgets and packages available in Flutter for handling images.

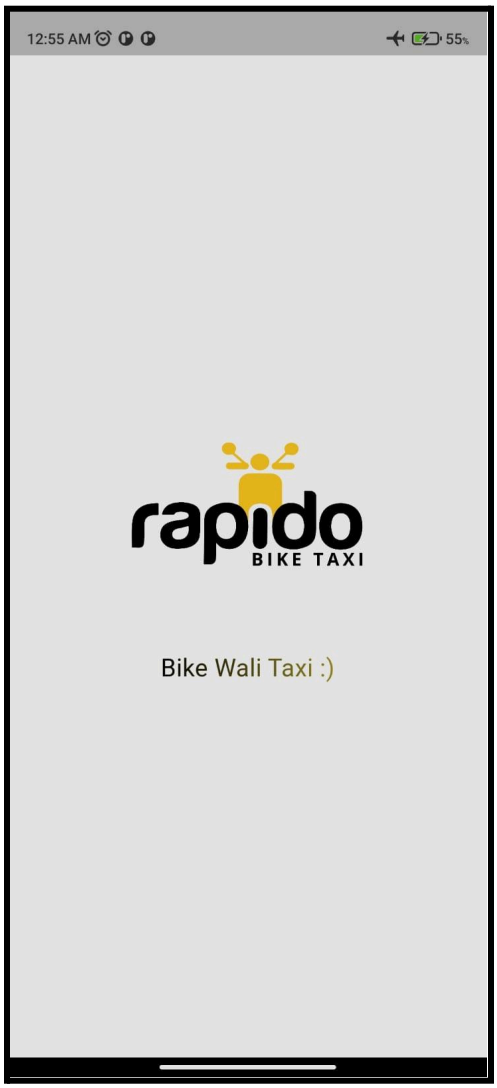
Code & Implementation:-

```
import
'package:animated_text_kit/animated_text_kit.dart';
import 'package:flutter/material.dart';
class SplashScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
const colorizeColors = [
    Colors.black,
    Colors.yellow,
];
const colorizeTextStyle = TextStyle(
  fontSize: 20.0,
  fontFamily: 'Horizon',
);
return Scaffold(
  backgroundColor: Colors.white,
  body: Center(
    child: Column(
```

```
        mainAxisSize: MainAxisSize.min,
        children: [
          SizedBox(
            width: 200, // Adjust width as needed
            height: 200, // Adjust height as needed
            child:
Image.asset('assets/images/logo1.png'),
          ),
          SizedBox(height: 20,),
          AnimatedTextKit(
            animatedTexts: [
              ColorizeAnimatedText(
                'Bike Wali Taxi :)',
                textStyle: colorizeTextStyle,
                colors: colorizeColors,
                textAlign: TextAlign.center,
              ),
            ],
            isRepeatingAnimation: true,
            onTap: () {
              print("Tap Event");
            },),] )),);}}
```

Pubspce.xml

```
# To add assets to your application, add an assets section, like this:
  assets:
    # - assets/images/logo.png
    - assets/images/logo1.png
    - assets/images/map.jpeg
```



MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	40
Name	Mayank Pahuja
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	12

Experiment No:- 04

Name:- Mayank Pahuja **Class:- D15-A RollNo:- 40**

Theory: In Flutter, a "Form" is a widget that represents a container for a collection of form fields.
It helps manage the state of the form and facilitates the validation and submission of user input. Here are some key concepts and theories about forms in Flutter:

1. Widget Hierarchy:

Forms in Flutter are composed of the `Form` widget, which contains a list of `FormField` widgets. Each `FormField` represents an individual input field like text fields, checkboxes, or dropdowns.

2. Form State:

The `Form` widget maintains the state of the form, including the current values of the form fields and their validation statuses. The form state is automatically managed by Flutter.

3. Validation:

Forms provide built-in validation through the `validator` property of each `FormField`. Validators are functions that determine whether the input is valid. The form's overall validity is determined by the validity of all its fields.

4. Form Submission:

Form submission is typically triggered by a button press. The `onPressed` callback of the button can call the `FormState.save()` method, which invokes the `onSaved` callback for each form field and then calls the `onFormSaved` callback.

5. GlobalKey<FormState>:

To interact with the form state, a `GlobalKey<FormState>` is commonly used. This key allows access to the form state and is used to validate and save the form.

6. Auto-validation:

Flutter provides automatic validation by calling the `validator` function whenever the user input changes. This allows for real-time feedback to the user about the validity of their input.

7. Form Submission Lifecycle:

The form submission process involves validation, saving, and then handling the saved data. Developers can customize this process by providing their own logic within the `onSaved` and `onFormSaved` callbacks.

8. Focus Management:

Forms handle the focus of input fields, making it easy to navigate through the form using keyboard input or programmatically setting focus on specific fields.

Code & Implementation:-

Login_screen.dart & Otp_screen.dart file code & output:-

```
import
'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:get/get_core/get_core.dart';
import
'package:get/get_navigation/get_navigation.dar
t';
import
'package:rapido/screens/otp_screen.dart';

class PhoneHome extends StatefulWidget {
  const PhoneHome({Key? key}) : super(key:
key);
  @override
  State<PhoneHome> createState() =>
PhoneHomeState();
}
class PhoneHomeState extends
State<PhoneHome> {
  TextEditingController
phoneNumberController =
TextEditingController();
  final _formKey = GlobalKey<FormState>();
  String counterText = '0';

  sendCode() async {
    try {
      await
FirebaseAuth.instance.verifyPhoneNumber(
        phoneNumber:
'+91${phoneNumberController.text}',
        verificationCompleted:
(PhoneAuthCredential credential) {},
        verificationFailed:
(FirebaseAuthException e) {
          Get.snackbar('Error Occurred', e.code);
        },
        codeSent: (String vid, int? token) {
          Get.to(OtpPage(vid: vid));
        },
        codeAutoRetrievalTimeout: (vid) {},
      );
    } on FirebaseAuthException catch (e) {
      Get.snackbar('Error Occurred', e.code);
    } catch (e) {
      Get.snackbar('Error Occurred',
e.toString());
    }
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Form(
          key: _formKey,
          child: Column(
            crossAxisAlignment:
CrossAxisAlignment.start,
            children: [
              SizedBox(height: 40),
              Padding(
                padding: const EdgeInsets.only(left:
8.0),
                child: Image.asset(
                  'assets/images/logo1.png',
                  width: 90,
                  height: 70,
                ),
              ),
              SizedBox(height: 12),
              Text(
                'What\'s your number?',
                style: TextStyle(fontSize: 20,
fontWeight: FontWeight.bold),
              ),
              SizedBox(height: 10),
              Text(
                'Enter your phone number to
proceed',
                style: TextStyle(color: Colors.grey),
              ),
              SizedBox(height: 10),
              TextFormField(
                maxLength: 10,
                keyboardType: TextInputType.phone,
                controller: phoneNumberController,
                decoration: InputDecoration(
                  prefix: Text("+91 ") ,
                  prefixStyle: TextStyle(fontSize: 15),
                  prefixIcon: Icon(Icons.phone),
                  counterText: '$counterText / 10',
                  counterStyle: TextStyle(fontSize: 10),
                  labelText: 'Number',
```

```
            hintText: 'Enter your phone
number',
            hintStyle: TextStyle(fontSize: 10,
color: Colors.grey),
           ),
           onChanged: (value) {
            setState(() {
              counterText =
value.length.toString();
            });
           },
           validator: (value) {
            if (value!.isEmpty) {
              return 'Please enter your phone
number';
            } else if (value.length != 10) {
              return 'Phone number must be 10
digits';
            }
            return null;
           },
          ),
```

```
          SizedBox(height: 20),
          Spacer(), // Move the button to the
bottom of the screen
          ElevatedButton(
           onPressed: (counterText == '10')
              ? () {
                if
(_formKey.currentState!.validate()) {
                  sendCode();
                }
               }
              : null, // Disable button if phone
number is not 10 digits
            child: Padding(
             padding: const
EdgeInsets.symmetric(horizontal: 90),
              child: Text(
               'Received OTP',
               style: TextStyle(
                fontSize: 18.0,
               ),),),),],),),),); }}
```

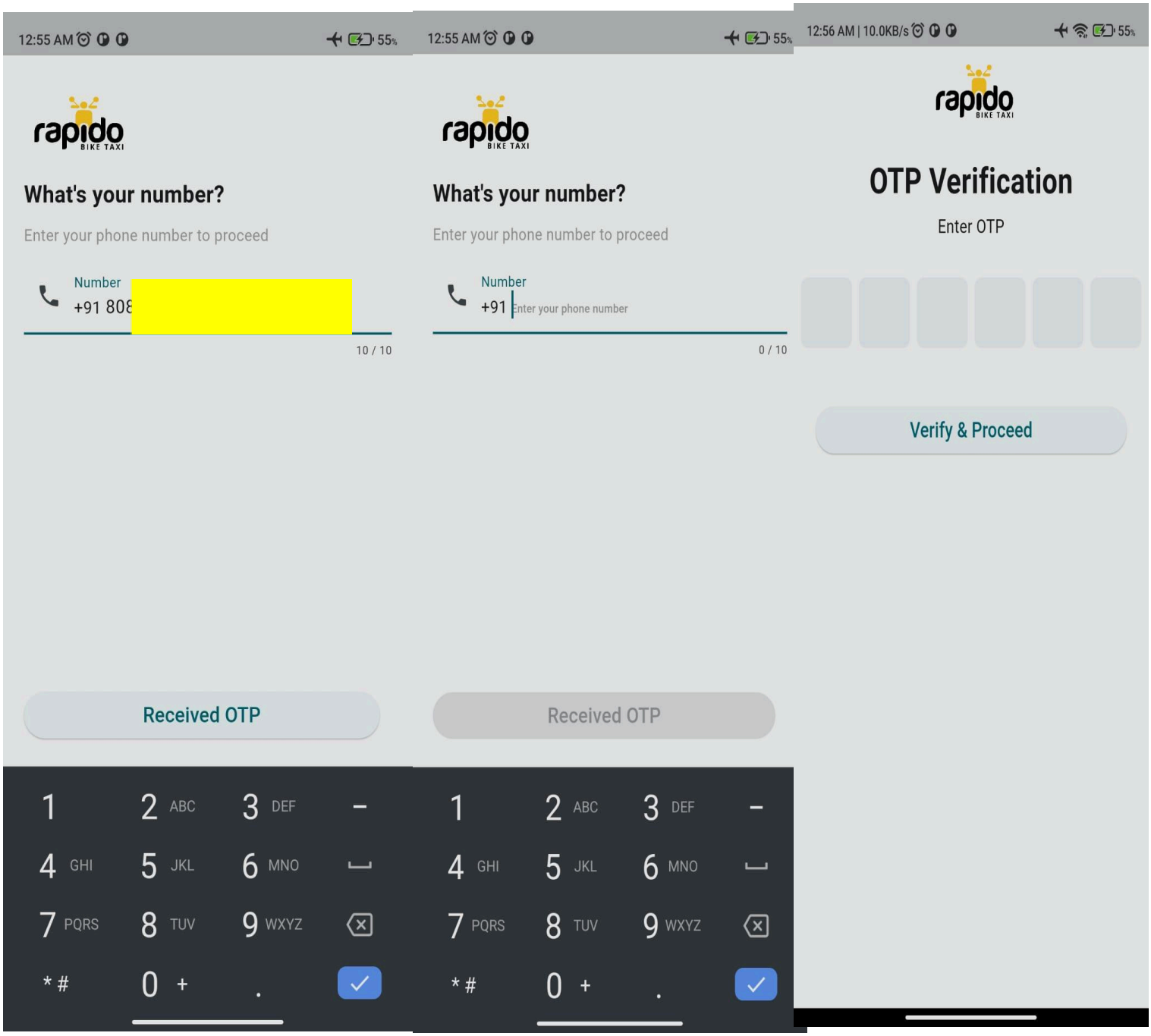
#Otpscreen Code:-

```
// ignore_for_file: prefer_const_constructors

import
'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:pinput/pinput.dart';
import 'package:rapido/wrapper.dart';
class OtpPage extends StatefulWidget {
 final String vid;
 const OtpPage({super.key, required this.vid});
 @override
 State<OtpPage> createState() =>
OtpPageState();
}
class OtpPageState extends State<OtpPage> {
 var code = '';
 signIn() async {
  PhoneAuthCredential credential =
PhoneAuthProvider.credential(
```

```
   verificationId: widget.vid,
   smsCode: code,
  );
  try {
   await
FirebaseAuth.instance.signInWithCredential(cr
edential).then((value) {
     Get.offAll(() => Wrapper());
   });
  } on FirebaseAuthException catch (e) {
   Get.snackbar('Error Occurred', e.code);
  } catch (e) {
   Get.snackbar('Error Occurred',
e.toString());
  }
 }
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   resizeToAvoidBottomInset: true,
   body: SingleChildScrollView(
```

```
      child: Column(
        crossAxisAlignment:
CrossAxisAlignment.center,
        children: [
          SizedBox(height: 40),
          Padding(
            padding: const EdgeInsets.only(left:
8.0),
            child: Image.asset(
              'assets/images/logo1.png',
              width: 90,
              height: 70,
            ),
          ),
          SizedBox(height: 20),
          Center(
            child: Text(
              "OTP Verification",
              style: TextStyle(fontSize: 30,
fontWeight: FontWeight.bold),
            ),
          ),
          Padding(
            padding: const
EdgeInsets.symmetric(horizontal: 25, vertical:
6),
            child: Text(
              "Enter OTP",
              textAlign: TextAlign.center,
              style: TextStyle(fontSize: 16),
            ),
          ),
          SizedBox(height: 20),
          textcode(),
          SizedBox(height: 40),
          button(),
        ],
      ),
    ),
  );
 }
 Widget button() {
  return Center(
    child: ElevatedButton(
      onPressed: () {
        signIn();
      },
      child: Padding(
        padding: const
EdgeInsets.symmetric(horizontal: 80),
        child: Text(
          'Verify & Proceed',
          style: TextStyle(
            fontSize: 18,
          ),
        ),
      ),
    ),
  );
 }
 Widget textcode() {
  return Center(
    child: Padding(
      padding: const EdgeInsets.all(6.0),
      child: Pinput(
        length: 6,
        onChanged: (value) {
          setState(() {
            code = value;
          });
        },
      ),
    ),
  );
 }
}
```

- **Until User can't enter the 10 Digit Phone number the Otp Authentication & login to the app is not to be done.**



MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	40
Name	Mayank Pahuja
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	12

Experiment No:- 05

Name:- Mayank Pahuja **Class:- D15-A RollNo:- 40**

Aim: To apply routing in Flutter Application

Theory:

In Flutter, routing refers to the navigation system that allows users to move between different screens or pages within an app. Flutter uses a widget-based approach for navigation, where each screen is represented by a widget. The `Navigator` class manages the stack of routes and facilitates transitions between them.

Routes are typically defined using the `MaterialPageRoute` class, providing a seamless and platform-aware transition between screens. Developers can use the `Navigator` to push new routes onto the stack or pop existing routes off it. Named routes help in easily identifying and navigating to specific screens.

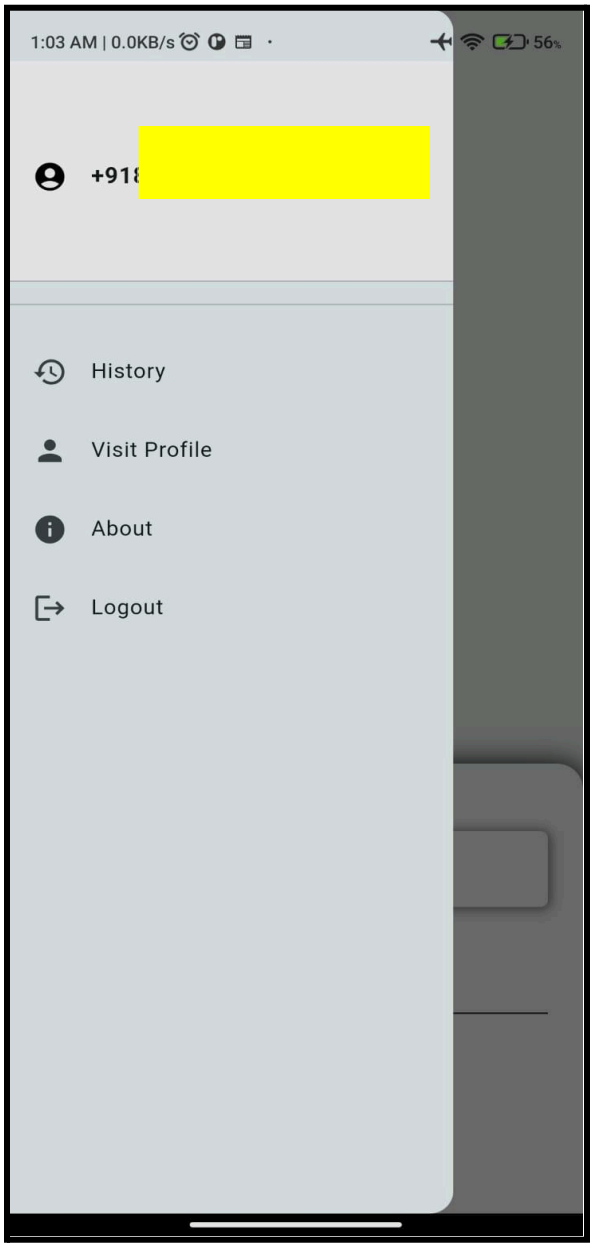
Flutter also supports route arguments, allowing developers to pass data between screens. Additionally, the `Navigator` provides a flexible set of transitions, such as slide, fade, or custom animations, enhancing the user experience during navigation.
Overall, Flutter's routing system provides a structured and intuitive way to handle navigation within mobile and web applications**.**

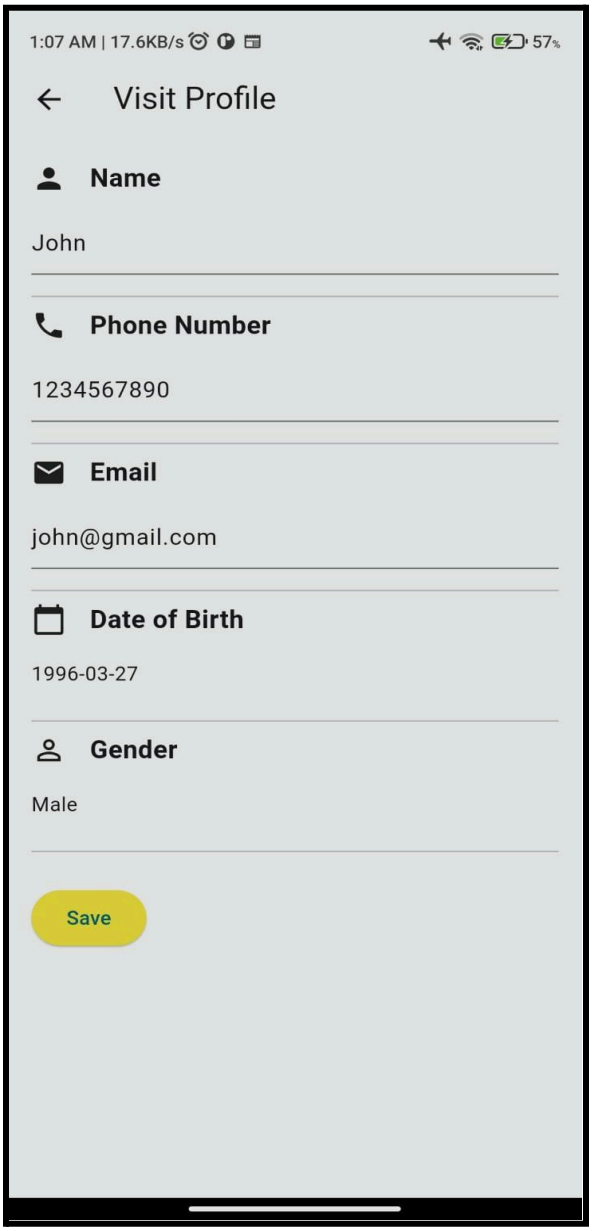
Code & Implementation:

```
 ignore_for_file: prefer_const_constructors,
prefer_const_literals_to_create_immutables
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:rapido/AllWidgets/Divider.dart';
class Homepage extends StatefulWidget {
  const Homepage({Key? key}) : super(key: key);
  @override
  State<Homepage> createState() =>
HomePageState();
}

class HomePageState extends State<Homepage> {
  final User? user =
FirebaseAuth.instance.currentUser;
  GlobalKey<ScaffoldState> scaffoldKey =
GlobalKey<ScaffoldState>();
  Future<void> signOut() async {
    await FirebaseAuth.instance.signOut();
  }
  bool _isDragging = false;
Offset _offset = Offset(0, 0);
@override
Widget build(BuildContext context) {
  return Scaffold(
    key: scaffoldKey,
    appBar: AppBar(
      title: Text("Homepage"),
    ),
    drawer: Drawer(
      child: ListView(
        children: [
          GestureDetector(
            onTap: () {
              scaffoldKey.currentState?.openDrawer();
            },
            child: Container(
              height: 165.0,
              child: DrawerHeader(
                decoration: BoxDecoration(color:
Colors.white),
                child: Row(
```

```
                children: [
                  Icon(Icons.account_circle, color:
Colors.black),
                  SizedBox(width: 16.0),
                  Column(
                    mainAxisAlignment:
MainAxisAlignment.center,
                    children: [
                      Text(
                        "${user!.phoneNumber}",
                        style: TextStyle(
                          fontSize: 16,
                          fontWeight: FontWeight.bold,
                        ),
                      ),

                    ],
                  ),
                ],
              ),
            ),
          ),
        ),
        Divider(),
        SizedBox(height: 12),
        ListTile(
          leading: Icon(Icons.history),
          title: Text("History", style:
TextStyle(fontSize: 15)),
          onTap: () {
            Navigator.pushNamed(context, '/history');
          },
        ),
        ListTile(
          leading: Icon(Icons.person),
          title: Text("Visit Profile", style:
TextStyle(fontSize: 15)),
          onTap: () {
            Navigator.pushNamed(context,
'/visit_profile');
          },
        ),
        ListTile(
          leading: Icon(Icons.info),
          title: Text("About", style:
TextStyle(fontSize: 15)),
          onTap: () {
            Navigator.pushNamed(context, '/about');
          },
        ),
        ListTile(
          leading: Icon(Icons.logout),
          title: Text("Logout", style:
TextStyle(fontSize: 15)),
          onTap: () {
            signOut();
          },
        ),
      ],
    ),
  ),
  body: Stack(
    children: [
      Center(
        child: Text(
          "${user!.phoneNumber}",
          style: TextStyle(fontSize: 18),
        ),
      ),
      Positioned(
        left: 0.0,
        right: 0.0,
        bottom: _isDragging ? _offset.dy : 0.0,
        child: GestureDetector(
          onPanUpdate: (details) {
            setState(() {
              _isDragging = true;
              _offset = Offset(0, _offset.dy +
details.delta.dy / 2); // Adjust the speed
            });
          },
          onPanEnd: (details) {
            setState(() {
              _isDragging = false;
            });
          },
          child: Container(
            height: 320,
            decoration: BoxDecoration(
              color: Colors.white,
              borderRadius: BorderRadius.only(
                topLeft: Radius.circular(18),
                topRight: Radius.circular(18),
              ),
              boxShadow: [
                BoxShadow(
                  color: Colors.black,
                  blurRadius: 16,
```

```
                spreadRadius: 0.6,
                offset: Offset(0.7, 0.7),
              ),
            ],
          ),
          child: Padding(
            padding: const
EdgeInsets.symmetric(horizontal: 24, vertical: 18),
            child: Column(
              crossAxisAlignment:
CrossAxisAlignment.start,
              children: [
                SizedBox(height: 6),
                Row(
                  mainAxisAlignment:
MainAxisAlignment.center,
                  children: [
                    Container(
                      height: 4,
                      width: 40,
                      decoration: BoxDecoration(
                        color: Colors.black,
                        borderRadius:
BorderRadius.circular(2),
                      ),
                    ),
                  ],
                ),
                SizedBox(height: 20),
                Container(
                  decoration: BoxDecoration(
                    color: Colors.white,
                    borderRadius:
BorderRadius.circular(5.0),
                    boxShadow: [
                      BoxShadow(
                        color: Colors.black54,
                        blurRadius: 6,
                        spreadRadius: 0.6,
                        offset: Offset(0.7, 0.7),
                      ),
                    ],
                  ),
                  child: TextFormField(
                    onTap: () {

FocusScope.of(context).requestFocus(FocusNode());
                    },
                    decoration: InputDecoration(
                      prefixIcon: Icon(Icons.search, color:
Colors.yellowAccent),
                      hintText: "Search Drop off",
                      border: InputBorder.none,
                      contentPadding:
EdgeInsets.symmetric(vertical: 15, horizontal: 20),
                    ),
                  ),
                ),
                SizedBox(height: 24),
                Row(
                  children: [
                    Icon(Icons.home, color: Colors.grey),
                    SizedBox(width: 12),
                    Column(
                      crossAxisAlignment:
CrossAxisAlignment.start,
                      children: [
                        Text("Add Home"),
                        SizedBox(height: 4),
                        Text(
                          "Your Living home address",
                          style: TextStyle(color:
Colors.black54, fontSize: 12),
                        ),
                      ],
                    ),
                  ],
                ),
                SizedBox(height: 10),
                DividerWidget(),
                SizedBox(height: 10),
                Row(
                  children: [
                    Icon(Icons.work, color: Colors.grey),
                    SizedBox(width: 12),
                    Column(
                      crossAxisAlignment:
CrossAxisAlignment.start,
                      children: [
                        Text("Add Work"),
                        SizedBox(height: 4),
                        Text(
                          "Your office address",
                          style: TextStyle(color:
Colors.black54, fontSize: 12),
                        ),],),],),],),),),),),],),);}}
```


1:03 AM | 0.0KB/s
56%
+91
History
Visit Profile
About
Logout


1:07 AM | 17.6KB/s
57%
Visit Profile
Name
John
Phone Number
1234567890
Email
john@gmail.com
Date of Birth
1996-03-27
Gender
Male
Save

MAD & PWA Lab
Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	40
Name	Mayank Pahuja
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:	13

Experiment No:- 06

Name:- Mayank Pahuja **Class:- D15-A RollNo:- 40**

Aim: To Connect Flutter UI with Firebase

Theory:

FlutterFire is a set of Flutter plugins that enable Flutter developers to integrate their applications with various Firebase services. Firebase is a comprehensive mobile and web application development platform provided by Google. FlutterFire is specifically designed to provide Flutter developers with a seamless way to interact with Firebase services.

Key features of FlutterFire include:

1. **Firebase Authentication:** FlutterFire provides plugins to easily integrate Firebase Authentication, allowing developers to implement user sign-up, sign-in, and password recovery features in their Flutter applications. Firebase supports various authentication methods, including email/password, Google Sign-In, Facebook Sign-In, and more.
2. **Cloud Firestore and Realtime Database:** FlutterFire supports both Cloud Firestore and Firebase Realtime Database, enabling developers to store and retrieve data in real-time. Firestore is a NoSQL document database, while Realtime Database is a JSON-based database.
3. **Cloud Functions:** Developers can deploy serverless functions using Cloud Functions for Firebase, and FlutterFire allows Flutter apps to trigger and interact with these functions.
4. **Cloud Storage:** FlutterFire supports Firebase Cloud Storage, allowing developers to upload, download, and manage files in the cloud. This is useful for handling user-generated content, such as images or videos.
5. **Firebase Cloud Messaging (FCM):** FCM enables developers to send push notifications to their Flutter applications. FlutterFire provides plugins for integrating FCM and handling push notifications.
6. **Firebase Performance Monitoring:** Developers can monitor the performance of their Flutter applications using Firebase Performance Monitoring. This includes measuring app startup time, screen rendering, and network performance.
7. **Firebase Analytics:** FlutterFire includes plugins for integrating Firebase Analytics, enabling developers to gain insights into user behavior and app usage.
8. **Firebase Remote Config:** FlutterFire supports Firebase Remote Config, allowing developers to remotely configure app behavior without publishing updates. This is useful for A/B testing and feature toggling.
9. **Firebase Crashlytics:** FlutterFire includes support for Firebase Crashlytics, providing real-time crash reporting to help developers identify and fix issues quickly.
10. **Firebase AdMob:** FlutterFire includes AdMob plugins for integrating advertisements into Flutter applications using Firebase AdMob.

Code & Implementation:-

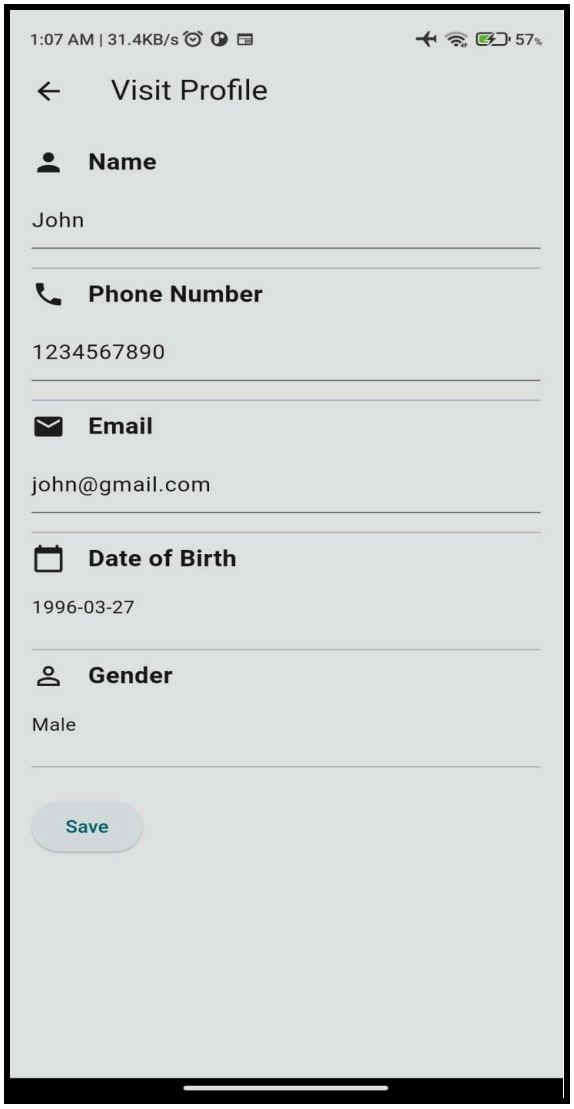
Main.dart file for connecting the firebase

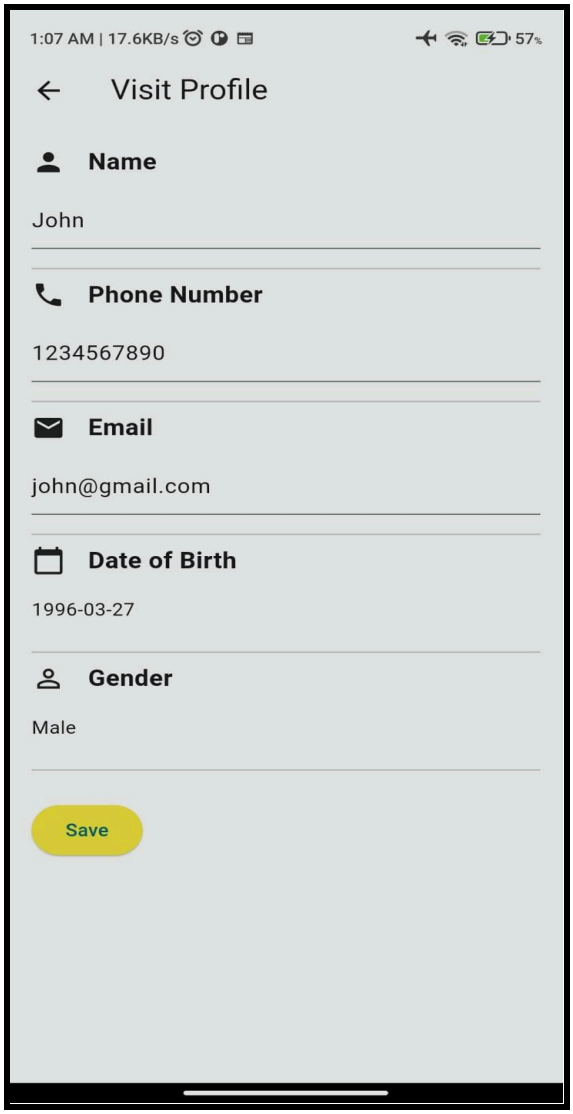
```
import 'dart:io';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import
'package:get/get_navigation/src/root/get_material_ap
p.dart';
import 'package:rapido/screens/about_screen.dart';
import 'package:rapido/screens/history_screen.dart';
import 'package:rapido/screens/splash_screen.dart';
import
'package:rapido/screens/visit_profile_screen.dart';
import 'package:rapido/wrapper.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  Platform.isAndroid
      ? await Firebase.initializeApp(
          options: const FirebaseOptions(
            apiKey:
"AIzaSyDjMB0YuZP3SiJRxpsxy1pBGyBW-OxuNX
k",
            appId:
"1:179713960522:android:391199b92da9ce114fd4b3
",
            messagingSenderId: "179713960522",
            projectId: "rapido-clone-78a08",
            storageBucket:
"rapido-clone-78a08.appspot.com",
          ),
        )
      : await Firebase.initializeApp();
  runApp(const MyApp());
}
```

Store user data from visited_profile.dart

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

class VisitProfileScreen extends StatefulWidget {
  const VisitProfileScreen({Key? key}) : super(key:
key);
  @override
  _VisitProfileScreenState createState() =>
_VisitProfileScreenState();
}
class _VisitProfileScreenState extends
State<VisitProfileScreen> {
  TextEditingController nameController =
TextEditingController();
  TextEditingController phoneNumberController =
TextEditingController();
  TextEditingController emailController =
TextEditingController();

  String name = '';
  String phoneNumber = '';
  String email = '';
  String gender = ''; // Default gender
  DateTime? dob; // Date of Birth
  bool isDataSaved = false; // To track if data is saved

  final DatabaseReference _userRef =

FirebaseDatabase.instance.reference().child('users');

  @override
  void initState() {
    super.initState();
    // Set initial values to controllers
    nameController.text = name;
    phoneNumberController.text = phoneNumber;
    emailController.text = email;
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Visit Profile'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment:
CrossAxisAlignment.start,
          children: [
            _buildProfileItem(Icons.person, 'Name',
nameController),
            _buildProfileItem(Icons.phone, 'Phone
Number', phoneNumberController),
            _buildProfileItem(Icons.email, 'Email',
emailController),
```

```
          _buildProfileItem(
              Icons.calendar_today,
              'Date of Birth',
              null,
              text: dob != null ?
dob.toString().substring(0, 10) : 'Select Date',
              onTap: () => _selectDate(context),
            ),

          _buildProfileItem(
            Icons.person_outline,
            'Gender',
            null,
            text: gender.isNotEmpty ? gender : 'Select
Gender',
            onTap: () => _showGenderDialog(context),
          ),

          SizedBox(height: 16),
          ElevatedButton(
            onPressed: () {
              _saveUserData();
            },
            style: ElevatedButton.styleFrom(
              backgroundColor: isDataSaved ?
Colors.yellow : null,
            ),
            child: Text('Save'),
          ),
        ],
      ),
    ),
  );
 }
 void _updateUserInfo(String fieldName, String
newValue) {
  setState(() {
   // Update local state based on field name
   switch (fieldName) {
     case 'Name':
       name = newValue;
       break;
     case 'Phone Number':
       phoneNumber = newValue;
       break;
     case 'Email':
       email = newValue;
       break;
     case 'Gender':
       gender = newValue;
       break;
     default:
       break;
   }
  });
}

void _saveUserData() {
  _userRef.child('userId').set({
    'name': name,
    'phoneNumber': phoneNumber,
    'email': email,
    'gender': gender,
    // Add other fields as needed
  }).then((_) {
    setState(() {
      isDataSaved = true;
    });
  });
}
```


1:07 AM | 31.4KB/s
57%
Visit Profile
Name
John
Phone Number
1234567890
Email
john@gmail.com
Date of Birth
1996-03-27
Gender
Male
Save


1:07 AM | 17.6KB/s
57%
Visit Profile
Name
John
Phone Number
1234567890
Email
john@gmail.com
Date of Birth
1996-03-27
Gender
Male
Save


Rapido-Clone
Realtime Database
Data
Rules
Backups
Usage
Extensions
Protect your Realtime Database resou
https://rapido-clone-78a08-default-rtdb.firebaseio.com > users
users
userId
email: "john@gmail.com"
gender: "Male"
name: "John"
phoneNumber: "1234567890"

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.	40
Name	Mayank Pahuja
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	15

PWA Lab
PRACTICAL 7

Name : Mayank Pahuja **Class:D15A Roll no: 40**

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 improve 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.

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.
- 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.

Implementation:-

Step 01:- Create the manifest.json file & added the app details

```
{
   "name": "Tourly",
   "short_name": "Traveling",
   "start_url": "./index.html",
   "display": "standalone",
   "background_color": "#ffffff",
   "theme_color": "#3367D6",
   "description": "Travel for all Destinations with our Tourly app.",
   "icons": [
     {
      "src": "readme-images/desktop.png",
      "sizes": "1296x736",
      "type": "image/png"
     },
     {
      "src": "readme-images/project-logo.png",
      "sizes": "217x76",
      "type": "image/png"
     }
   ]
}
```

Step 02:- Created the serviceworker.js file and added the installing & fetching urls logic

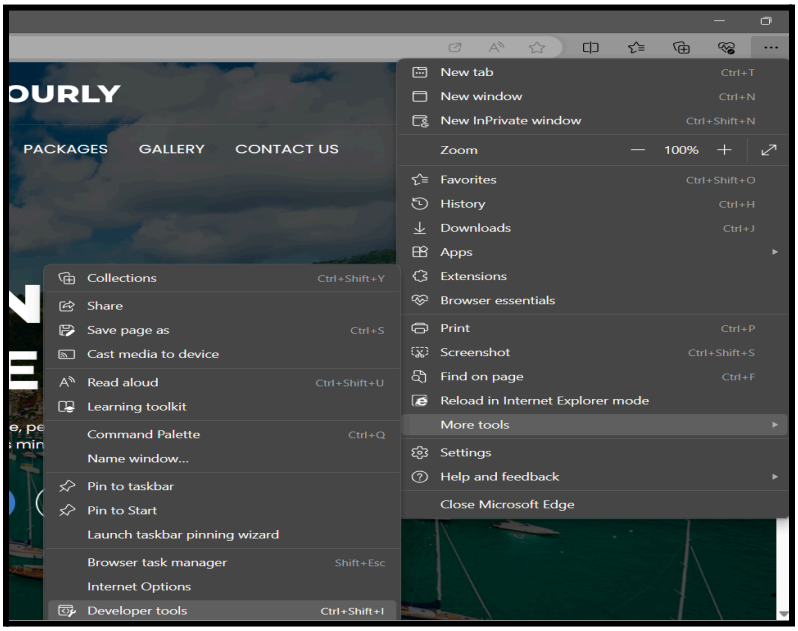
```
// Service worker code

const CACHE_NAME = 'ecommerce-pwa-cache-v1';
const urlsToCache = [
  '/',
  '/index.html',
  'assets/css/style.css',
  '../assets/js/script.js',
  // Add more URLs of assets to cache as needed
];
// Install service worker
self.addEventListener('install', event => {
  // Perform install steps
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then(cache => {
        console.log('Opened cache');
        return cache.addAll(urlsToCache);
      })
  );
});
// Fetch assets from cache or network
self.addEventListener('fetch', event => {
  event.respondWith(
    caches.match(event.request)
      .then(response => {
        // Cache hit - return response
```

```
    if (response) {
      return response;
    }
    // Clone the request
    const fetchRequest = event.request.clone();

    return fetch(fetchRequest).then(
      response => {
        // Check if we received a valid response
        if (!response || response.status !== 200 || response.type !== 'basic') {
          return response;
        }
        // Clone the response
        const responseToCache = response.clone();

        caches.open(CACHE_NAME)
          .then(cache => {
            cache.put(event.request, responseToCache);
          });
        return response;
      });}));});
```

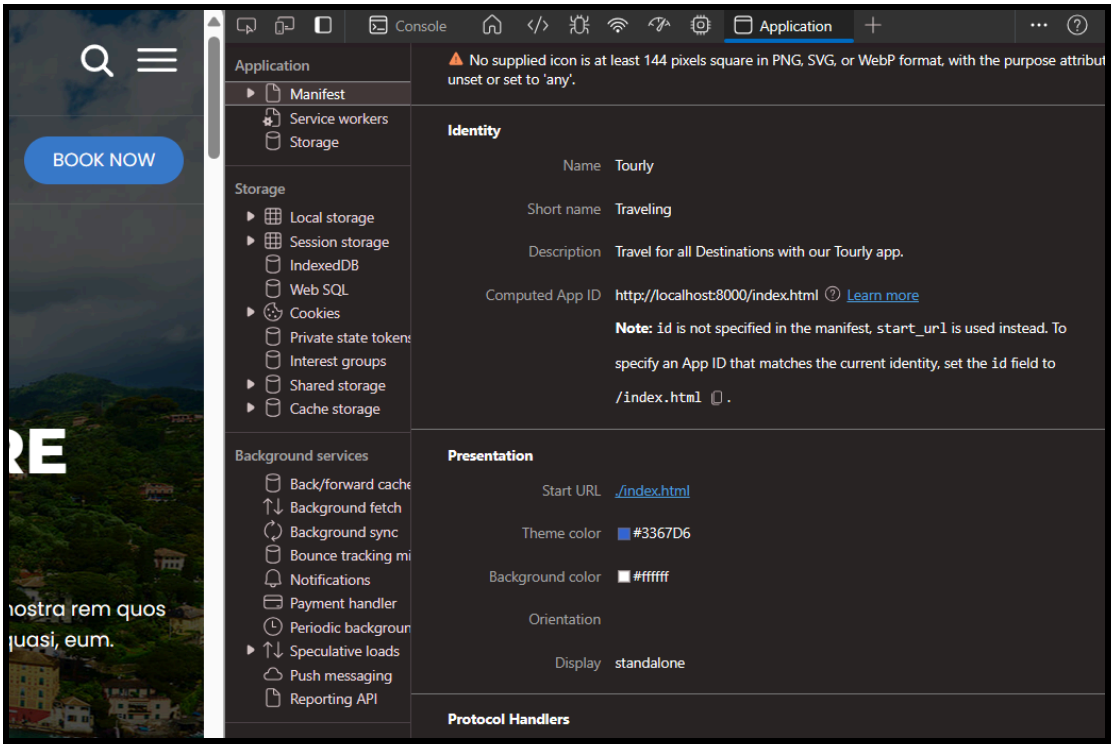
Step 03:- link the manifest & write serviceworker script for detecting,registeration,loading & error handling of serviceworker.

```
<link rel="manifest" href="/manifest.json">
<script>
  if ('serviceWorker' in navigator) {
    window.addEventListener('load', () => {
      navigator.serviceWorker.register('/serviceworker.js')
        .then(registration => {
          console.log('Service Worker registered with scope:', registration.scope);
        })
        .catch(error => {
          console.error('Service Worker registration failed:', error);
        });});}
</script>
```

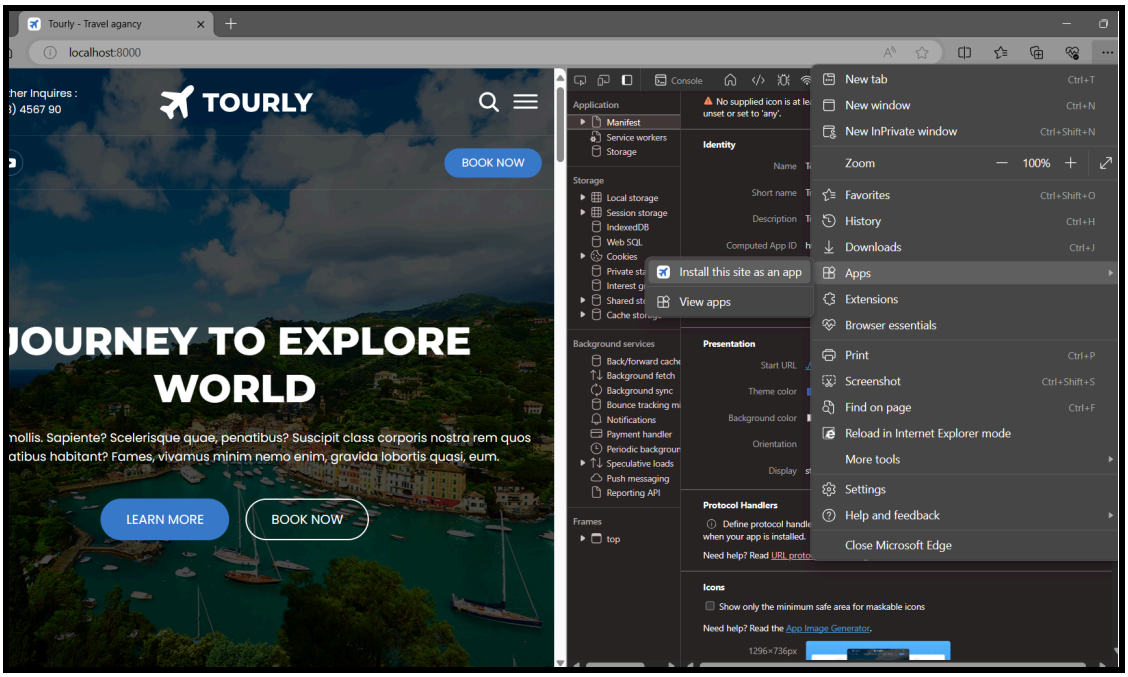
On Chrome/Edge:- host the index.html on http server:-

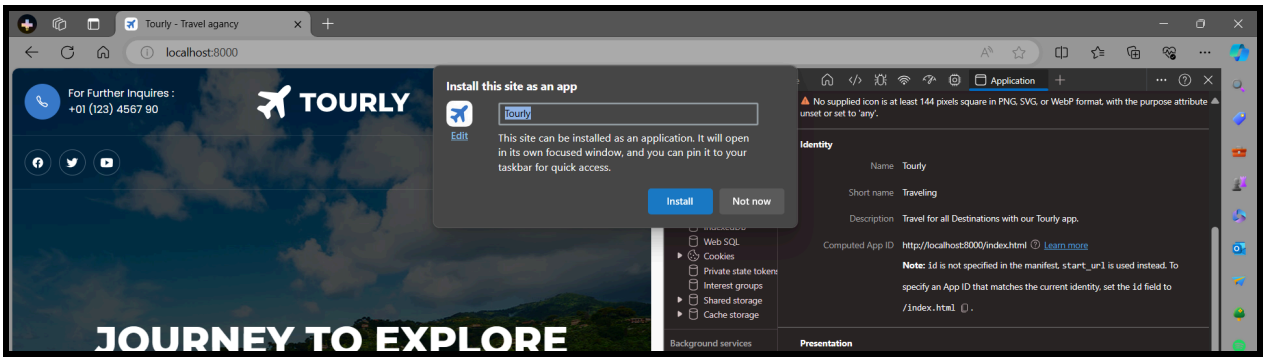


Go to Application section where you can see the manifest.json file if link correctly

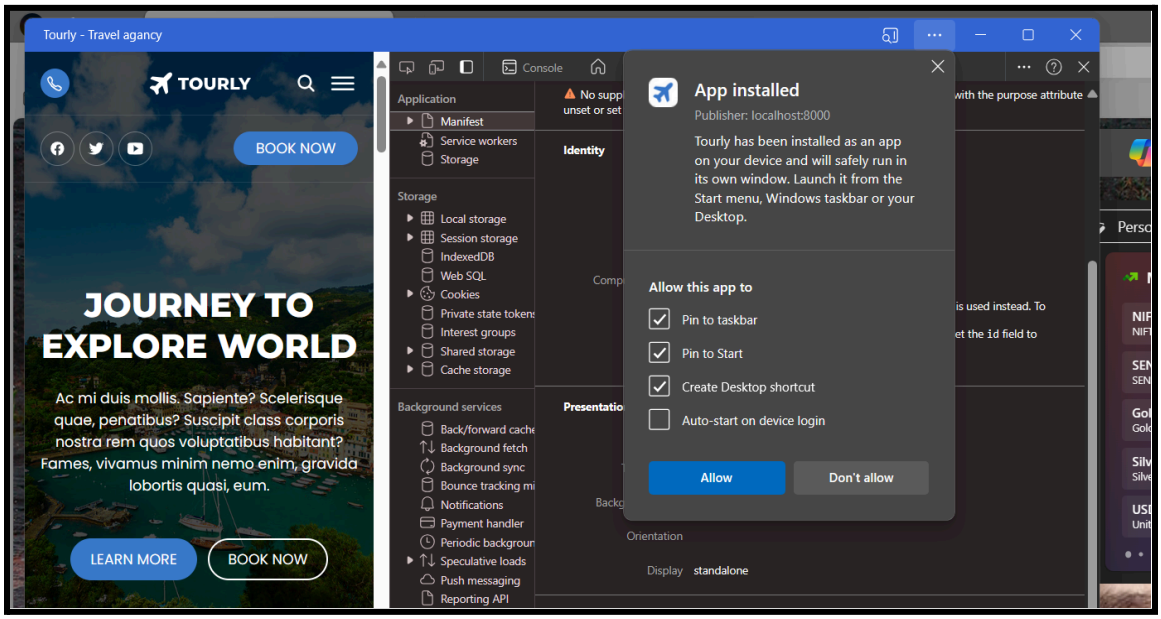


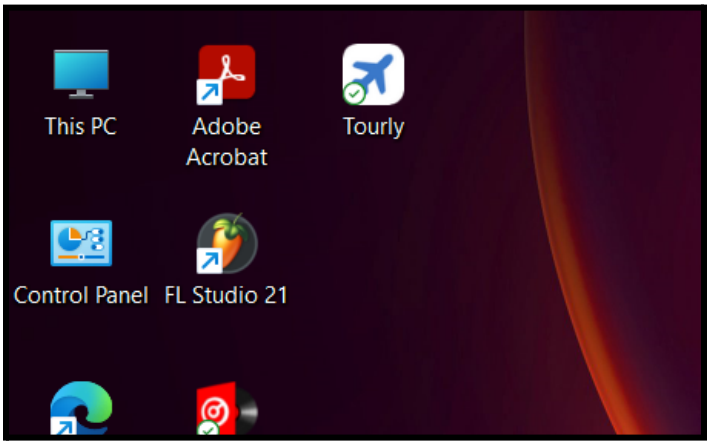
Then Click on three dots→ Apps → Install this site as an app→ install





Then click on allow & your website can be used as an app.





Conclusion:

- Thus “add to homescreen feature was successfully implemented”
- We created a shortcut icon to open our websites app locally
- We also added an image to the app icon

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.	40
Name	Mayank Pahuja
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

PWA Lab
PRACTICAL 8

Name: Mayank Pahuja **Class:D15A Roll no: 40**

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

Service Worker Cycle
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.

Installation
Once the browser registers a service worker, installation can be attempted. This occurs if the service worker is considered to be new by the browser, either because the site currently doesn't have a registered service worker, or because there is a byte difference between the new service worker and the previously installed one.

A service worker installation triggers an install event in the installing service worker. We can include an install event listener in the service worker to perform some task when the service worker installs. For instance, during the install, service workers can precache parts of a web app so that it loads instantly the next time a user opens it (see caching the application shell). So, after that first load, you're going to benefit from instant repeat loads and your time to interactivity is going to be even better in those cases.

Activation
Once a service worker has successfully installed, it transitions into the activation stage. If there are any open pages controlled by the previous service worker, the new service worker enters a waiting state. The new service worker only activates when there are no longer any pages loaded that are still using the old service worker. This ensures that only one version of the service worker is running at any given time.
When the new service worker activates, an activate event is triggered in the activating service worker. This event listener is a good place to clean up outdated caches

Code & Implementation

New-service-worker code:-

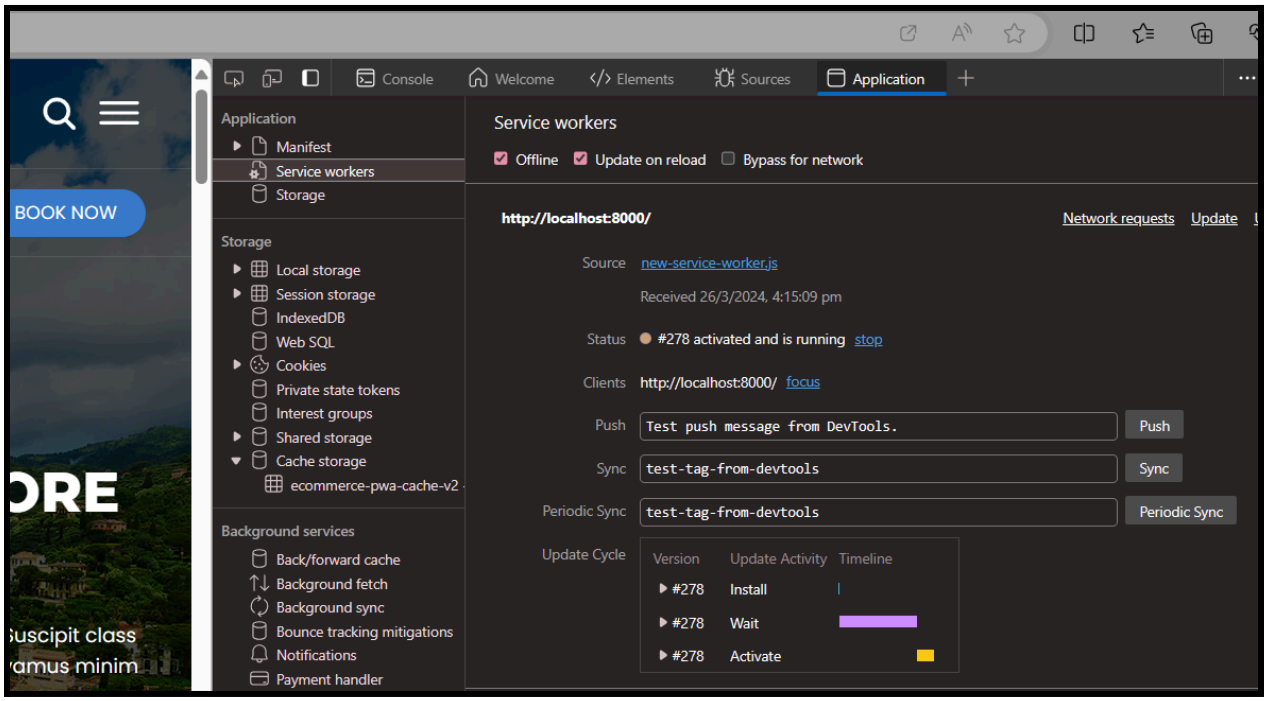
```
const CACHE_NAME = 'ecommerce-pwa-cache-v2';
const urlsToCache = [
    '/',
    '/index.html',
    'assets/css/style.css',
    '/assets/js/script.js',
    'assets/images',
  // Add more URLs of assets to cache as needed
];
self.addEventListener('install', event => {
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then(cache => {
        console.log('Opened cache');
        return cache.addAll(urlsToCache);
      }));});
self.addEventListener('activate', event => {
  event.waitUntil(
    caches.keys().then(cacheNames => {
      return Promise.all(
        cacheNames.filter(cacheName => {
          return cacheName !== CACHE_NAME;
        }).map(cacheName => {
          return caches.delete(cacheName);
        }));}));});
self.addEventListener('fetch', event => {
  event.respondWith(
    caches.match(event.request)
      .then(response => {
        return response || fetch(event.request);
      }));});
```

Linking the new service worker in html code

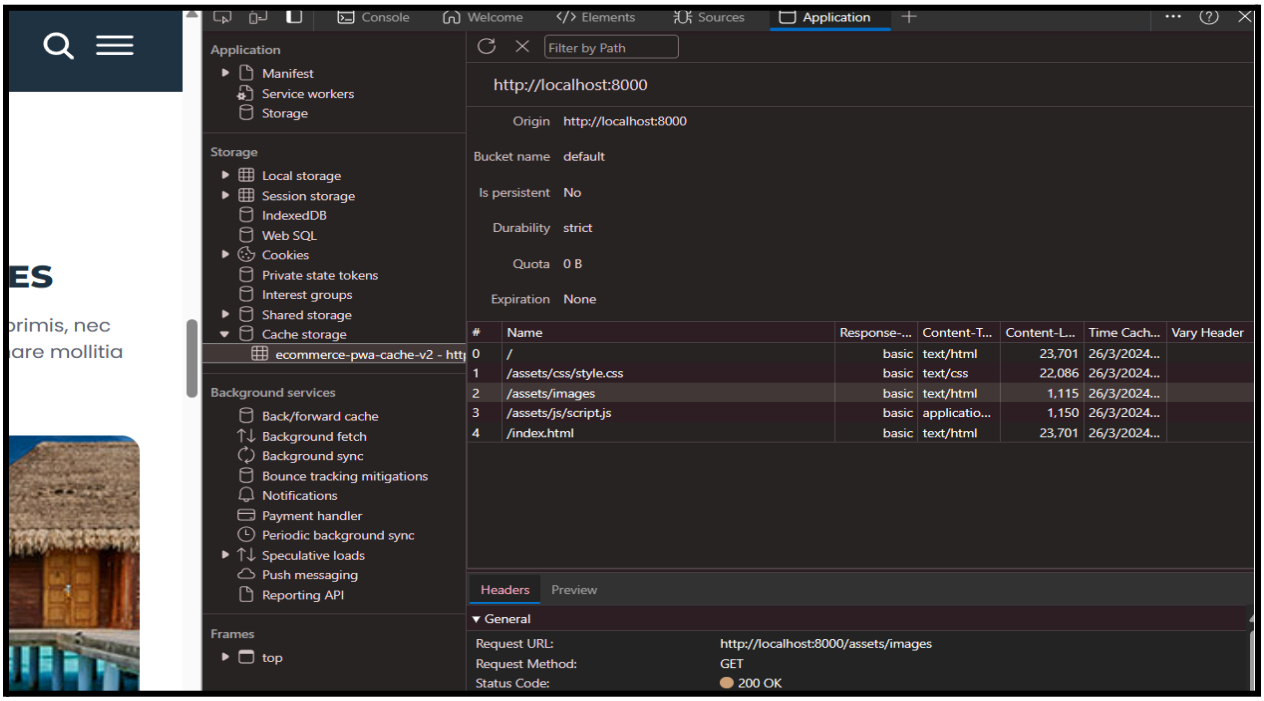
```
<script>
    if ('serviceWorker' in navigator) {
      window.addEventListener('load', () => {
        navigator.serviceWorker.register('/new-service-worker.js')
          .then(registration => {
            console.log('New Service Worker registered with scope:', registration.scope);
          })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

Output:-

Application → Service worker → we can see the new -service-worker.js is there



Cache Storage - we can see the name of the newly created service worker with status 200 ok



Conclusion: Registered a service worker, and completed the installation and activation process for a new service worker. Also checked the Cache Storage.

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.	40
Name	Mayank Pahuja
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

PWA Lab
PRACTICAL 9

Name: Mayank Pahuja **Class:D15A Roll no: 40**

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.

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”.

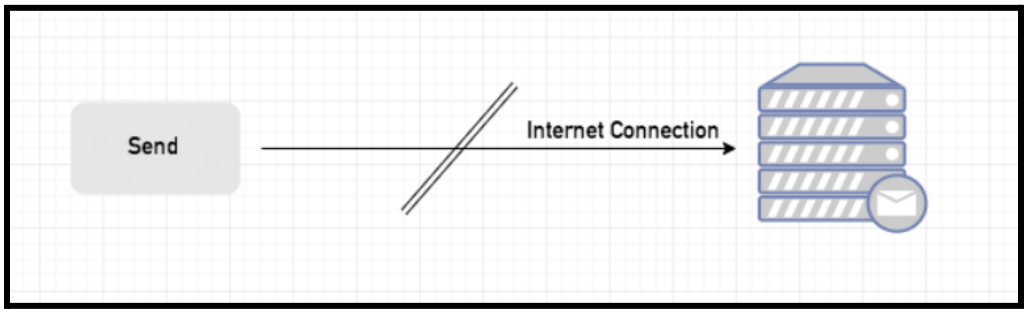
- CacheFirst - In this function, if the received request has cached before, the cached response is returned to the page. But if not, a new response requested from the network.
- NetworkFirst - In this function, firstly we can try getting an updated response from the network, if this process completed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

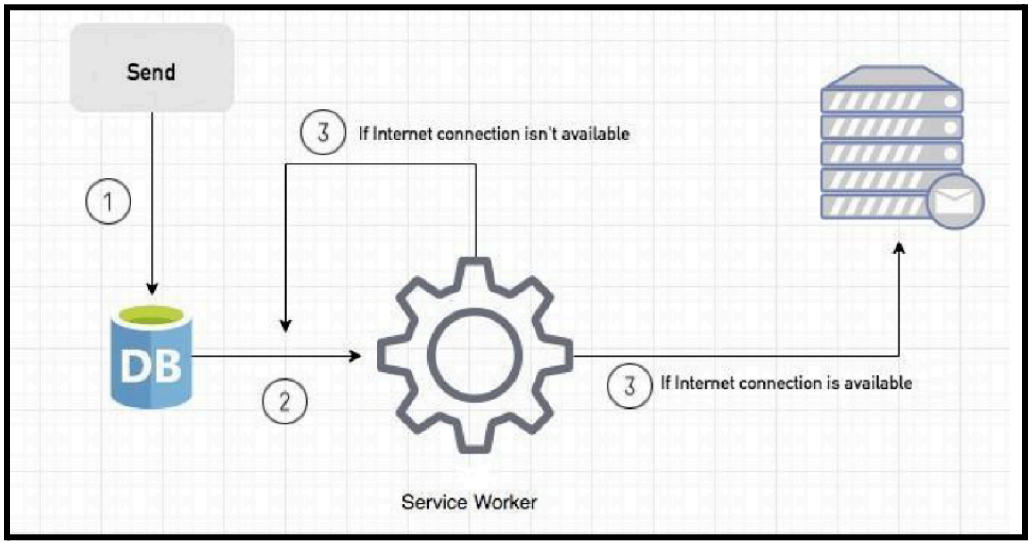
Sync Event

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

Here is a job for the Background Sync.

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





1. When we click the "send" button, email content will be saved to IndexedDB.
2. Background Sync registration.
3. If the Internet connection is available, all email content will be read and sent to Mail Server.

If the Internet connection is unavailable, the service worker waits until the connection is available even though the window is closed. When it is available, email content will be sent to Mail Server.

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

Event Listener for Background Sync Registration

Event Listener for sw.js

Push Event This is the event that handles push notifications that are received from the server. You can apply any method with received data. We can check in the following example.
"Notification.requestPermission();" is the necessary line to show notification to the user. If you don't want to show any notification, you don't need this line.
In the following code block is in sw.js file. You can handle push notifications with this event. In this example, I kept it simple. We send an object that has "method" and "message" properties. If the method value is "pushMessage", we open the information notification with the "message" property.

Code & Implementation:-

Service worker code:-

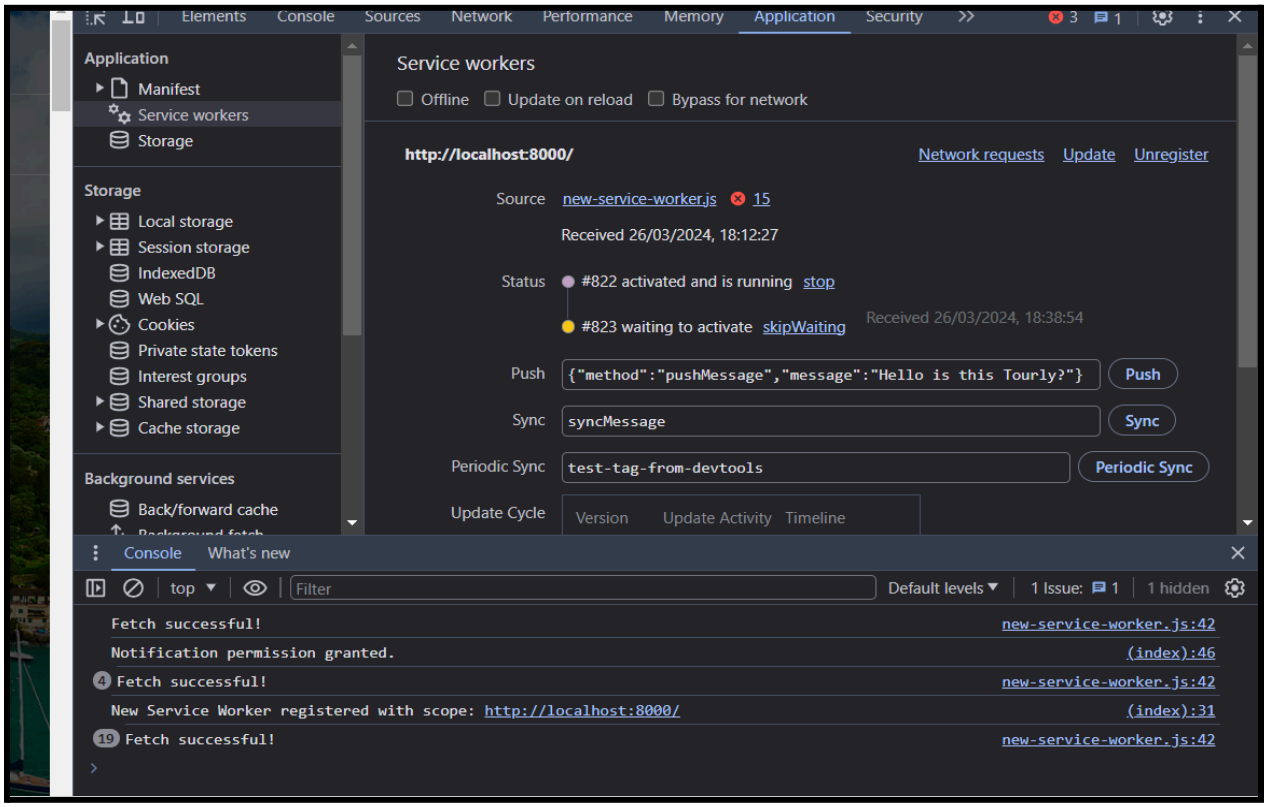
```
const CACHE_NAME = 'ecommerce-pwa-cache-v2';
const urlsToCache = [
   '/',
   '/index.html',
   '/assets/css/style.css',
   '/assets/js/script.js',
   '/assets/images',
   // Add more URLs of assets to cache as needed
];
self.addEventListener('install', event => {
   event.waitUntil(
      caches.open(CACHE_NAME)
         .then(cache => {
            console.log('Opened cache');
            return cache.addAll(urlsToCache);
         }));});
self.addEventListener('activate', event => {
   event.waitUntil(
      caches.keys().then(cacheNames => {
         return Promise.all(
            cacheNames.filter(cacheName => {
               return cacheName !== CACHE_NAME;
            }).map(cacheName => {
               return caches.delete(cacheName);
            }));}) );});
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.request));
});
self.addEventListener("sync", (event) => {
   if (event.tag === "syncMessage") {
      console.log("Sync successful!");
   }});
// Push event listener
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);
}}});
function checkResponse(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);
            });});}
function returnFromCache(request) {
    return caches.open(CACHE_NAME).then(function (cache) {
        return cache.match(request).then(function (matching) {
            if (!matching || matching.status == 404) {
                return cache.match("offline.html");
            } else {
                return matching;
            }});});}
function addToCache(request) {
    return caches.open(CACHE_NAME).then(function (cache) {
        return fetch(request).then(function (response) {
            return cache.put(request, response.clone()).then(function () {
                return response;});});});}
```

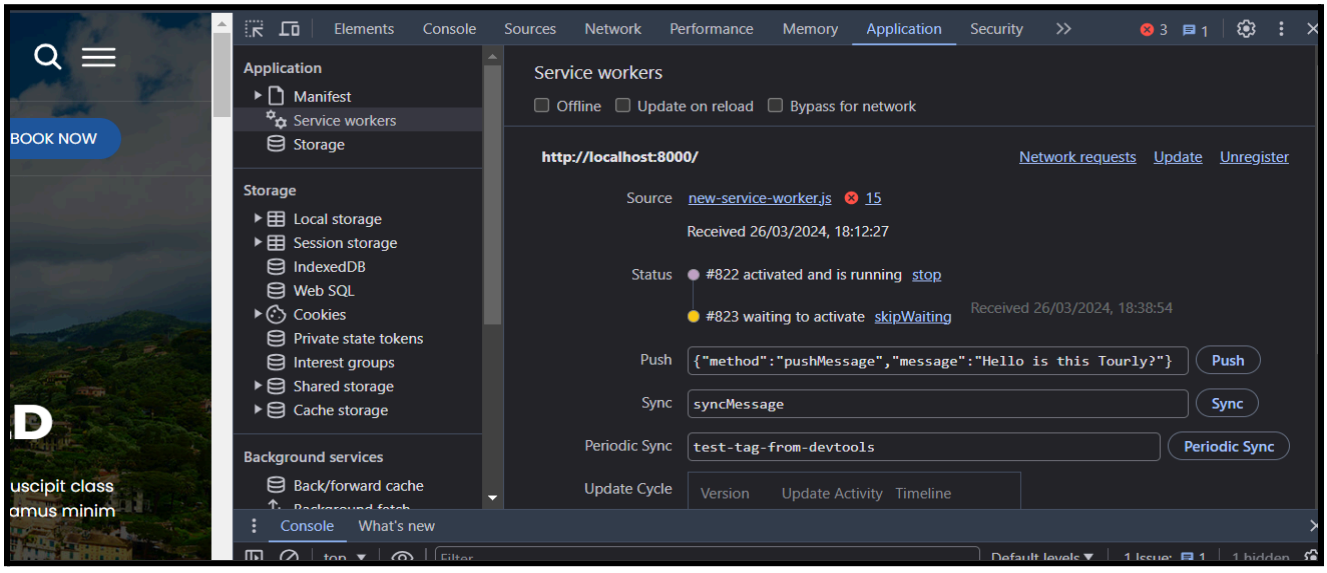
Html code:-

```
<script>
    if ('serviceWorker' in navigator) {
      window.addEventListener('load', () => {
        navigator.serviceWorker.register('/new-service-worker.js')
          .then(registration => {
            console.log('New Service Worker registered with scope:', registration.scope);
          })
          .catch(error => {
            console.error('New Service Worker registration failed:', error);
          });}); }
  </script>
<script>
  // Check if the browser supports notifications
  if ('Notification' in window) {
      // Request permission for notifications
      Notification.requestPermission().then(function (permission) {
          if (permission === 'granted') {
              console.log('Notification permission granted.');
          } else {
              console.warn('Notification permission denied.');
          }});}</script>
```

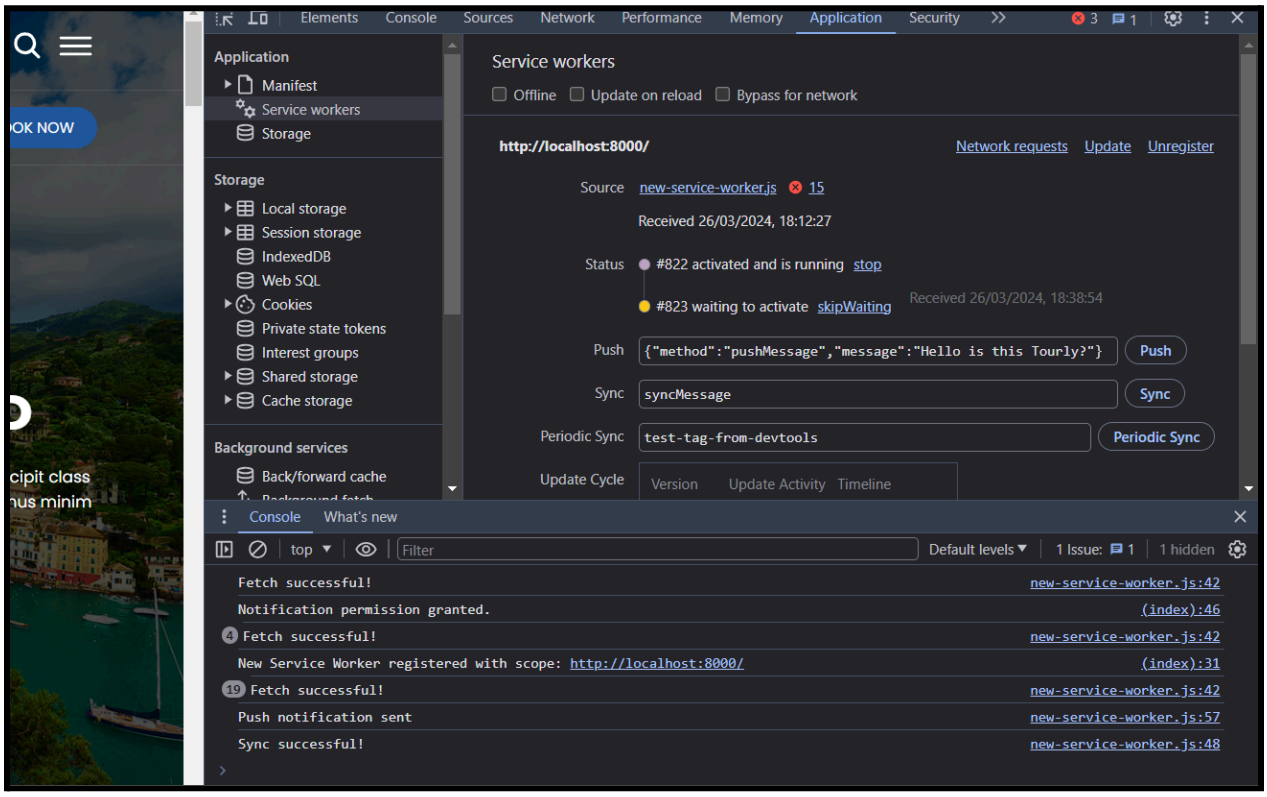
Here we can see that the fetching is done successfully.



As we click on push it will send the notification on chrome or any other browser



Now we have fetch,push, sync the message successfully.



Conclusion:

In this experiment, we have successfully implemented service worker events like fetch, sync and push for my Twiggy 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.	40
Name	Mayank Pahuja
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

PWA Lab

PRACTICAL 10

Name: Mayank Pahuja **Class:D15A Roll no: 40**

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

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:

- Blogging with Jekyll
- Custom URL
- Automatic Page Generator

Reasons for favoring this over Firebase:

- Free to use
- Right out of github
- 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

- Very familiar interface if you are already using GitHub for your projects.
- Easy to set up. Just push your static website to the gh-pages branch and your website is ready.
- Supports Jekyll out of the box.
- 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

- The code of your website will be public, unless you pay for a private repository.
- Currently, there is no support for HTTPS for custom domains. It's probably coming soon though.
- 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:

- 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.

- 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.
- 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:

- Realtime backend made easy
- 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

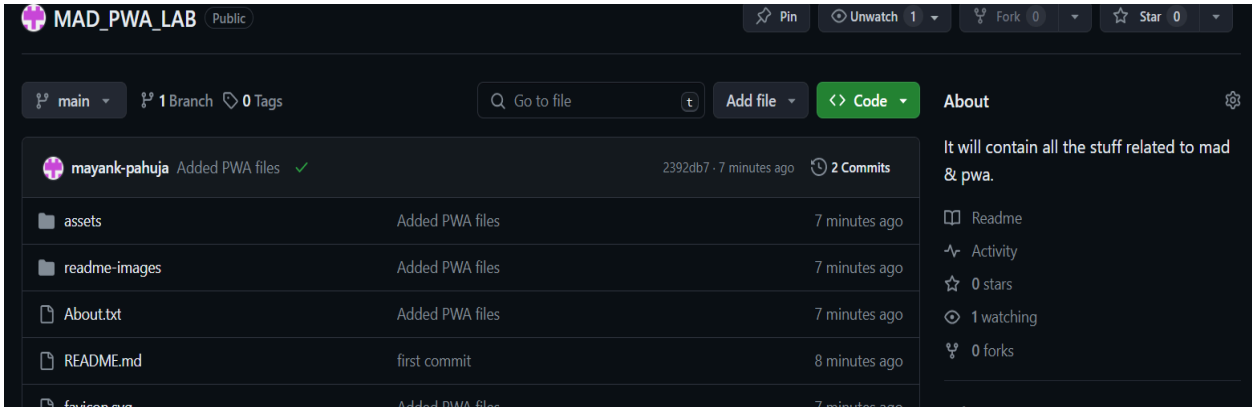
- Hosted by Google. Enough said.
- Authentication, Cloud Messaging, and a whole lot of other handy services will be available to you.
- A real-time database will be available to you, which can store 1 GB of data.
- You'll also have access to a blob store, which can store another 1 GB of data.
- Support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

Cons

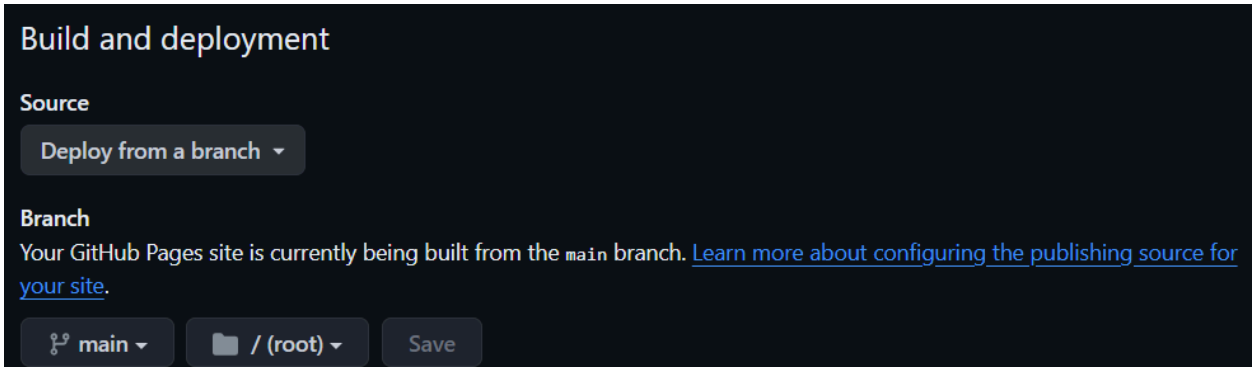
- Only 10 GB of data transfer is allowed per month. But this is not really a big problem if you use a CDN or AMP.
- Command-line interface only.
- No in-built support for any static site generator.

Code & Implementation:-

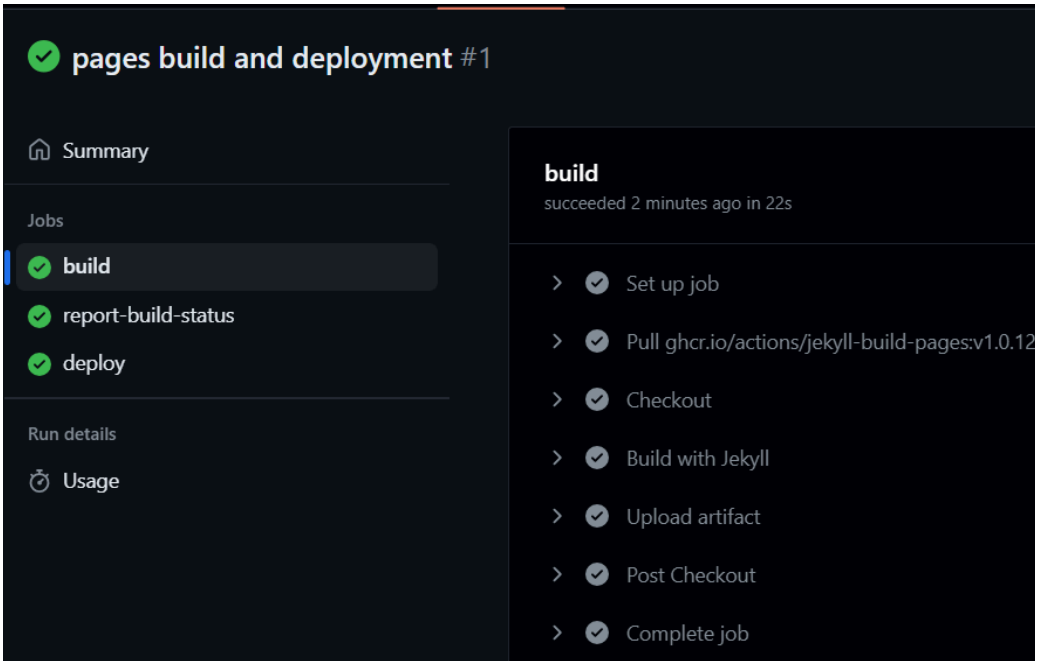
Step 01:- Go to git repository add your pwa project in that repo



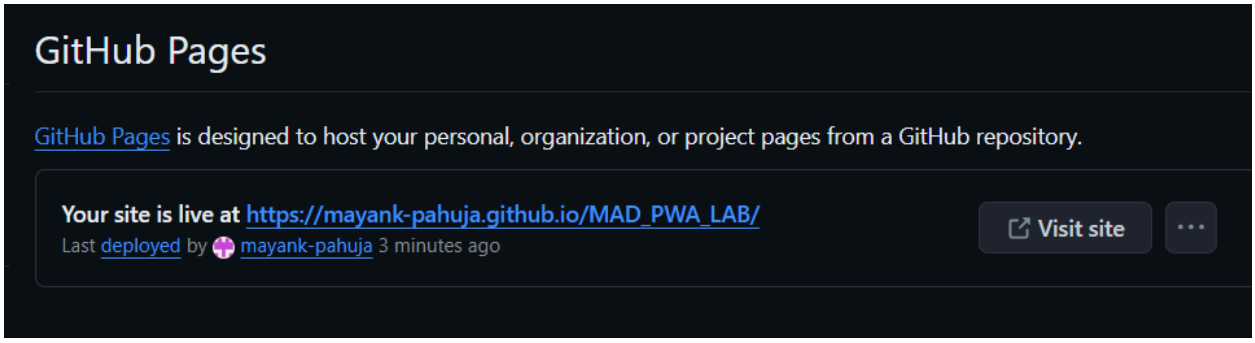
Step02:- Now Settings→Pages→ choose branch as a root

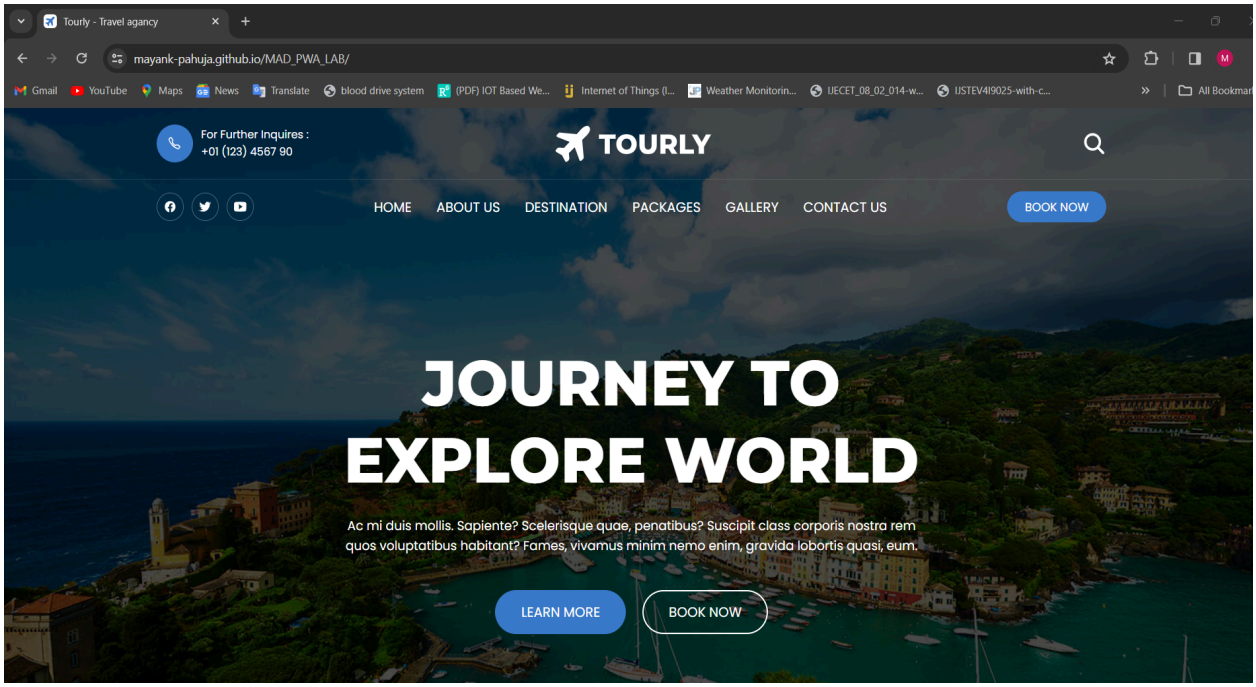


Step03:- Wait until it deploy your project pages & all.



Step04:- Go to settings → pages→ there you see the deployed project link, click on it enjoy.





Conclusion:

Successfully hosted website on 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.	40
Name	Mayank Pahuja
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	15

PWA Lab
PRACTICAL 11

Name: Mayank Pahuja **Class:D15A Roll no: 40**

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

Theory:

Google Lighthouse :

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

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

Key Features and Audit Metrics

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

Performance: This score is an aggregation of how the page fared in aspects such as (but not limited to) loading speed, time taken for loading for basic frame(s), displaying meaningful content to the user, etc. To a layman, this score is indicative of how decently the site performs, with a score of 100 meaning that you figure in the 98th percentile, 50 meaning that you figure in the 75th percentile and so on.

PWA Score (**Mobile**): Thanks to the rise of Service Workers, app manifests, etc., a lot of modern web applications are moving towards the PWA paradigm, where the objective is to make the application behave as close as possible to native mobile applications. Scoring points are based on the Baseline PWA checklist laid down by Google which includes Service Worker implementation(s), viewport handling, offline functionality, performance in script-disabled environments, etc.

Accessibility: As you might have guessed, this metric is a measure of how accessible your website is, across a plethora of accessibility features that can be implemented in your page (such as the 'aria-' attributes like aria-required, audio captions, button names, etc.). Unlike the other metrics though, Accessibility metrics score on a pass/fail basis i.e. if all possible elements of the page are not screen-reader friendly (HTML5 introduced features that would make pages easy to interpret for screen readers used by visually challenged people like tag names, tags such as <section>, <article>, etc.), you get a 0 on that score. The aggregate of these scores is your Accessibility metric score.

Best Practices: As any developer would know, there are a number of practices that have been deemed 'best' based on empirical data. This metric is an aggregation of many such

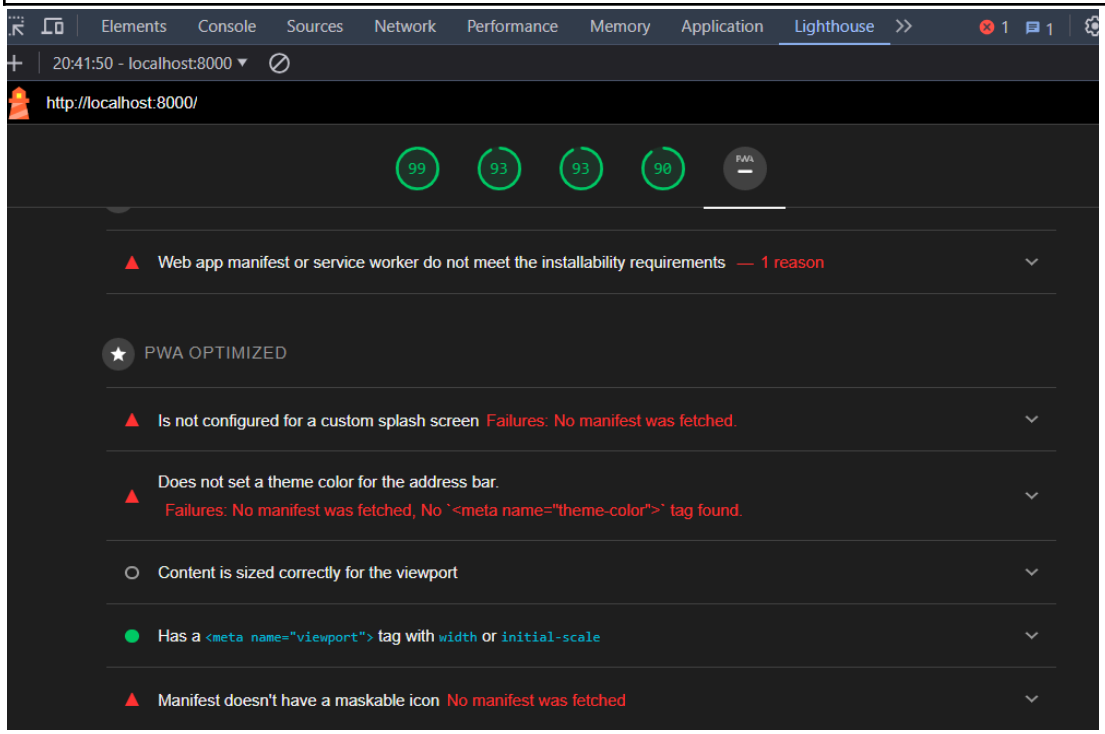
points, including but not limited to:Use of HTTPS
Avoiding the use of deprecated code elements like tags, directives, libraries, etc. Password input with paste-into disabled Geo-Location and cookie usage alerts on load, etc.

Code & Implementation

As we can see that there is the issue in manifest file

original code

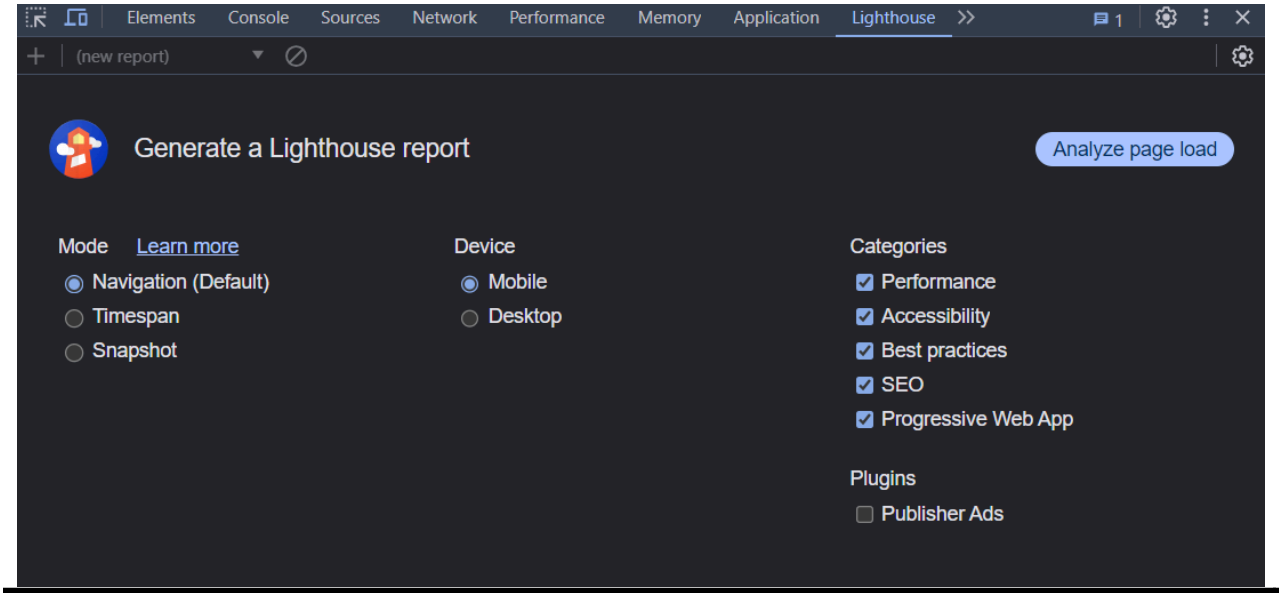
```
{
  "name": "Tourly",
  "short_name": "Traveling",
  "start_url": "./index.html",
  "display": "standalone",
  "background_color": "#ffffff",
  "theme_color": "#3367D6",
  "description": "Travel for all Destinations with our Tourly app.",
}
```

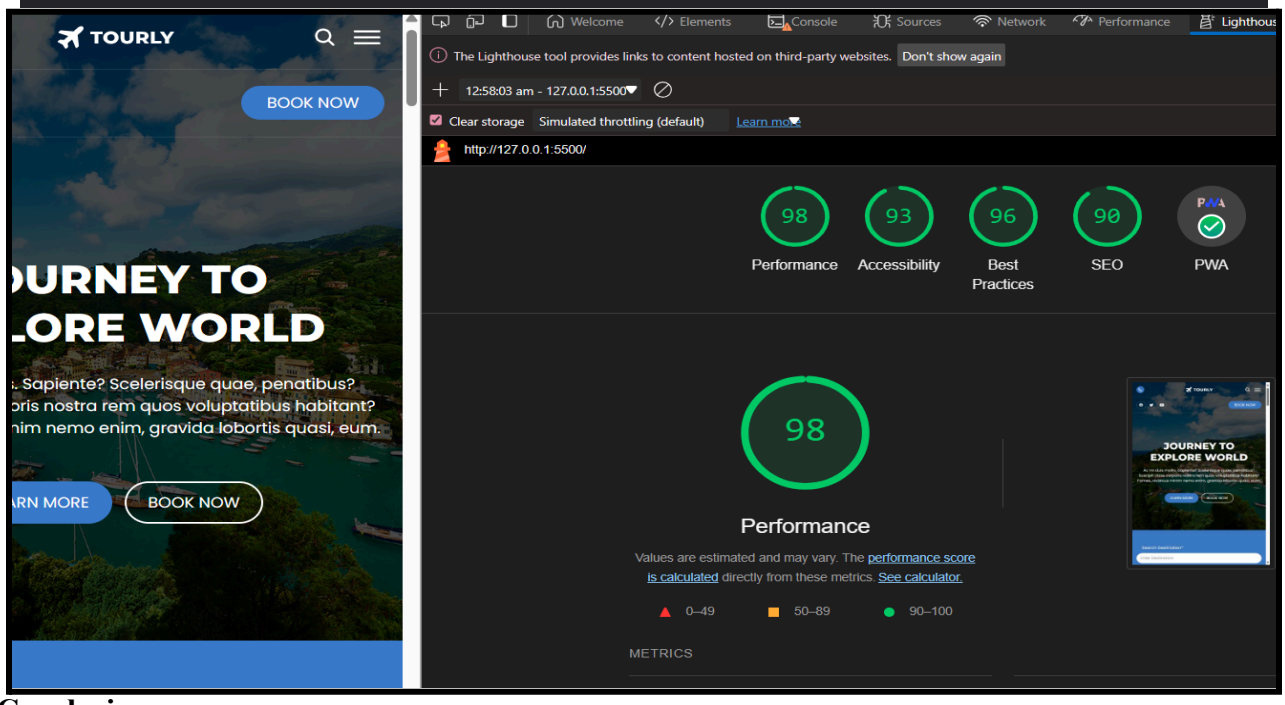


After Changes:-

```
{
 "name": "Tourly",
 "short_name": "Traveling",
 "start_url": "./index.html",
 "display": "standalone",
 "background_color": "#ffffff",
 "theme_color": "white",
 "description": "Travel for all Destinations with our Tourly app.",
 "icons": [{
```

```
  "src": "readme-images/desktop.png",
  "sizes": "196x196",
  "type": "readme-images/th.jpeg",
  "purpose": "maskable"
},{
  "src": "readme-images/project-logo-1.png",
  "sizes": "512x512",
  "type": "image/png"}]}
```





Conclusion:

We analyzed the website with the help of lighthouse tool and found some issues with the website Hence we made changes in the manifest.json file.

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.	40
Name	Mayank Pahuja
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:	5

Name:- mayank Paliya
RollNO:- 40, class:- D15-A

MAD PWA Lab Assignment - 01

Q1) ⇒ Flutter Overview:-

- Flutter is an open source UI software development toolkit created by Google for building natively compiled application for Mobile user, web & desktop users.
- Key features of flutter:-

1) Single codebase:- flutter offers a single codebase that can be used to create application to multiple platform.

2) Hot Reload:- Developers can see the change made in the code instantly reflected in the app.

3) High Performance:- Developers can see the change made in the code instantly reflected in the app.

4) Access to native features:- Flutter provides seamless access to native features and APIs following developers to integrate device specific functionalities.

Q2) ⇒ Widget Tree and Composition

Widget Tree is a hierarchical structure of UI elements developers represent by widget. Widgets are building blocks of flutter application & the widgets tree organizes them in a parent child realtionship. Widget colum involes combining simple widgets to create more complex UIs.

Commonly widgets used includes:-
1) Container 2) Stack 3) List view 4) Image 5) Text 6) Label.

Q5) ⇒ State Management in flutter.

- State Management in Flutter is crucial to handle changes in the application data & UI, proper state Management ensures that the UI stay in sync with underlying data and contributes to a scalable & maintainable codebase.

- SetState :- used for small to medium size application where state is localized & doesn't need to be shared between multiple widget.

- Provider :- well suited for medium to large size application where you need to share state across different parts of app.

- Riverpod :- Suitable for project where we want to take advantage for advanced features.

Q4) ⇒ Firebase integration in flutter

Step 01:- Create application/project in firebase website.

Step 02:- Add firebase dependencies and flutter fire Packages to our Local Machine.

Step 03:- Initialize Firebase in your flutter app by calling firebase Initialize App in the main function.

Step 04:- Use firebase services based on your requirements like Firestore Database, Authentication, Storage, etc.

* Benefits of Firebase :-

1) Real Time NO-SQL Database.
2) Authentication.
3) Cloud Storage.
4) Hosting.

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.	40
Name	Mayank Pahuja
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:	4

Name:- Mayank Pahuja
Roll No:- 40
Batch :- B3

PWA Assignment - 02 (1)

Q1) Define Progressive web App (PWA) and explain its significance in modern web development. Discuss the key characteristics that differentiate PWA's from traditional mobile Appl.

⇒ A Progressive Web App (PWA) is a website that looks and behave as if it is a mobile app. PWAs are block to take advantages of Native Mobile devices features It is built with Regular web technologies "Html, CSS & J.S."

- Reduced development cost - By using existing web skills and a single codebase. PWA are quicker and cheaper to develop.
- Wider Reach :- It works on any devices with a modern browser. Eliminating need for app store approval
- Easy Update :- Updates in PWA happen automatically in background.
- Improved user engagement :- Features like push notifications and offline functionality enhance user experience.

PWA	Traditional Mobile App
- Web Technologies (HTML, CSS, JS	• Native code (platform specific).
• Installed from browser "Add to home" prompt	• Download from app store
• Accessible through Search engines	• Reliant on an app store discovery.
• Automatic background update	• Requires Manual updates.

Q3.

Q2) Define Responsive web design and explain its important in the context of Progressive Web apps. Compare and contrast responsive fluid & adaptive web design approach.

⇒ • Responsive web design (RWD) is web development approach that ensures a website adjusts its layouts and functionality based on the device it being viewed on.

1] Importance for PWA:-

consists user experience. User ^smooth experience regardless of the devices they use, It ensures PWA looks and functional ^on all Stats.

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

3) Search Engine optimization:- Google and other search engines favour websites that offers a good user experience on all devices.

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

• Responsive web design (RWA):- RWD uses a single codebase that adapts to different devices.

• Fluids web Design:- A specific layout within RWA that uses Percentages and relative units to define elements size that allows elements to expand and contract ~~on~~ based ^on screen size.

• Adaptive web Design:- This approach involves multiple fixed width layouts for different devices category. It requires more development work compare to RWD

Ans

(2)

Q.3. Describe the lifecycle of service workers, including registration installation & activation phase.

→
- A Key Player in this PWA universe is the "Services Worker". The Services Worker is a Javascript file that runs on a seperate thread apart from the one in which your real website Javascript file run
- Three phases of lifecycle:-
  1) Registration phase.
  2) Installation phase.
  3) Activation phase.

1) Registration Phase:- This done in two ways ⇒
- You either specify is scope for a Services Worker after
- You either leave to default global sope where servey worker

2) Installation:- Once the Service Worker successfully registered, it is not ~~read~~ ready to installed. The Service worker script is downloaded to the browser and browser will install few situation service worker installation:-
- A New Service Worker file.
- A modified Service Worker file.

3) Activation:- once the Installation phase is successfully the Service worker does not in activation state. It could only help
1) None of the Pages with the Service Worker and are closed.
2) There is no other Service Worker active in that page.

Q4) Explain the use of IndexedDB with service workers for data Storage.

⇒ • IndexedDB is a browser API for storing large amount of structure data on the Client-Slide. It functions like NOSQL database, allowing you to store key value and organize object store with indexes.

• Benefits of using IndexedDB with service workers.

1) Offline data access - Improves user experience by allowing interaction with PWA even without an internet connection.

2) Faster load times - Cached data retrieved from IndexedDB can be served much faster than fetching it from the server again.

3) Improved reliability - Reduces dependency on a constant Network connection making the PWA more reliable.

4) Security - Intermediate provides a secure storage mechanism within the browsers sandbox environments.

5) Data management - PWA should have mechanisms to manage storage space and avoid excessive data accumulation in IndexedDB.