Department of Information Technology

CERTIFICATE

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

Lab Assistant

Subject Teacher

Mrs. Kajal Joseph

Principal

Head of Department

Dr. Mrs. Shalu Chopra

Name of the Course : MAD & PWA Lab

Course Code : ITL604

Year/Sem/Class : D15A/D15B

A.Y.: 23-24

Faculty Incharge : Mrs. Kajal Joseph.

Lab Teachers : Mrs. Kajal Jewani.

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

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

Program specific Outcomes

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

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

Lab Objectives:

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

Lab Outcomes:

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

Index

Sr. No	Experiment Title	LO	DOP	DOS	Grade
1.	To install and configure the Flutter Environment	LO1	17/01	24/01	15
2.	To design Flutter UI by including common widgets.	LO2	24/01	31/01	13
3.	To include icons, images, fonts in Flutter app	LO2	31/01	07/02	15
4.	To create an interactive Form using form widget	LO2	07/02	14/02	15
5.	To apply navigation, routing and gestures in Flutter App	LO2	14/02	21/02	14
6.	To Connect Flutter UI with fireBase database	LO3	21/02	06/03	14
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	28/03	05/02	03
13.	Assignment-2	LO4,LO5 ,LO6	14/03	21/03	05

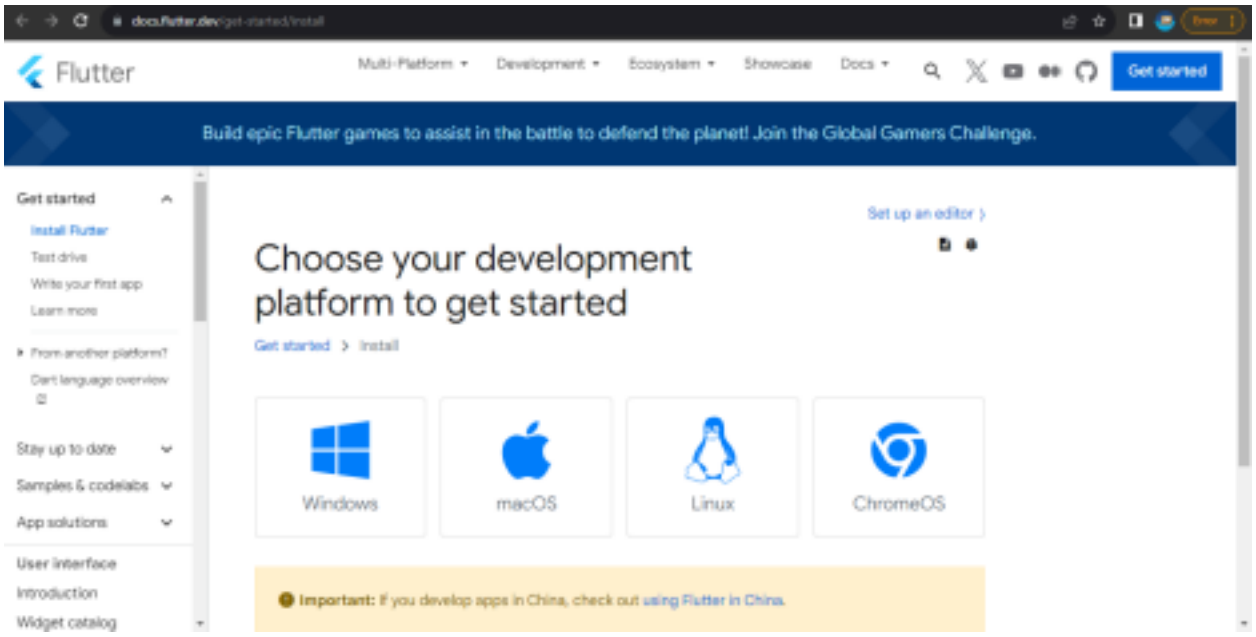
MAD & PWA Lab
Journal

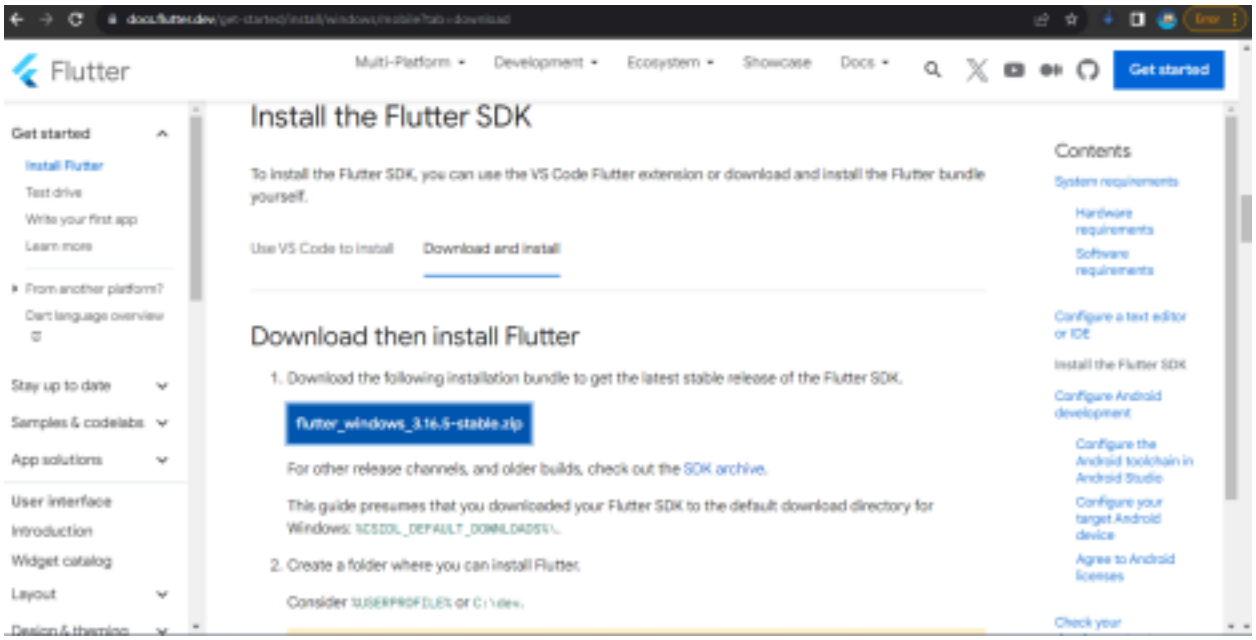
Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	41
Name	Nilanchala Panda
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	15

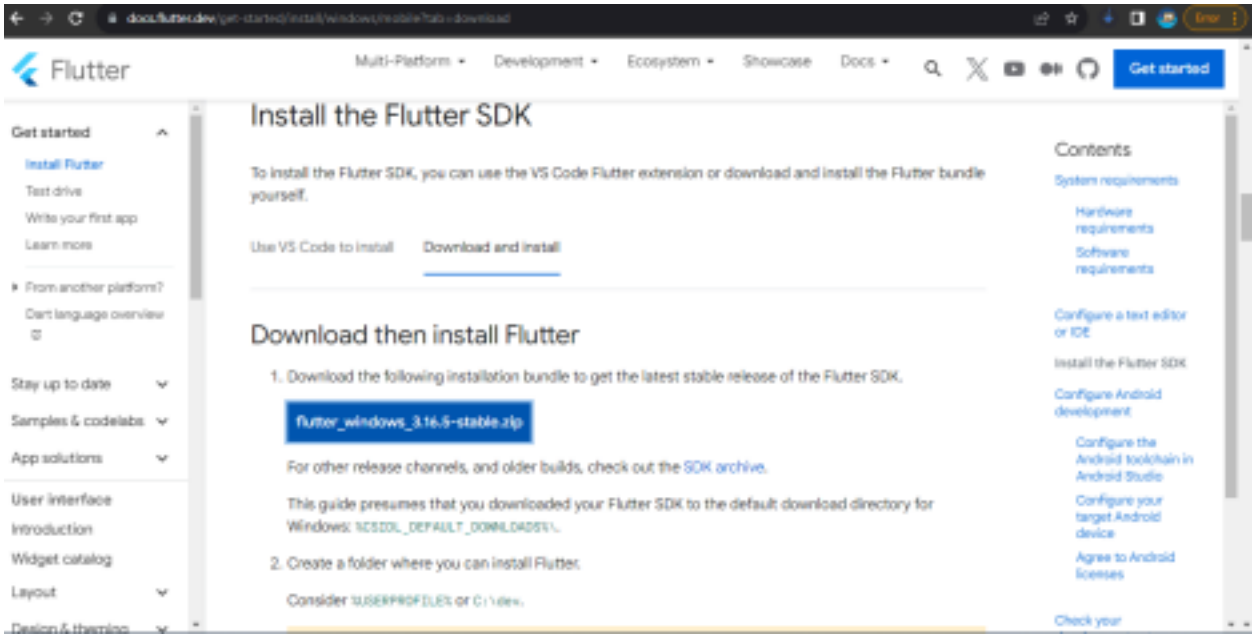
Aim: To write a Hello World Program on Flutter

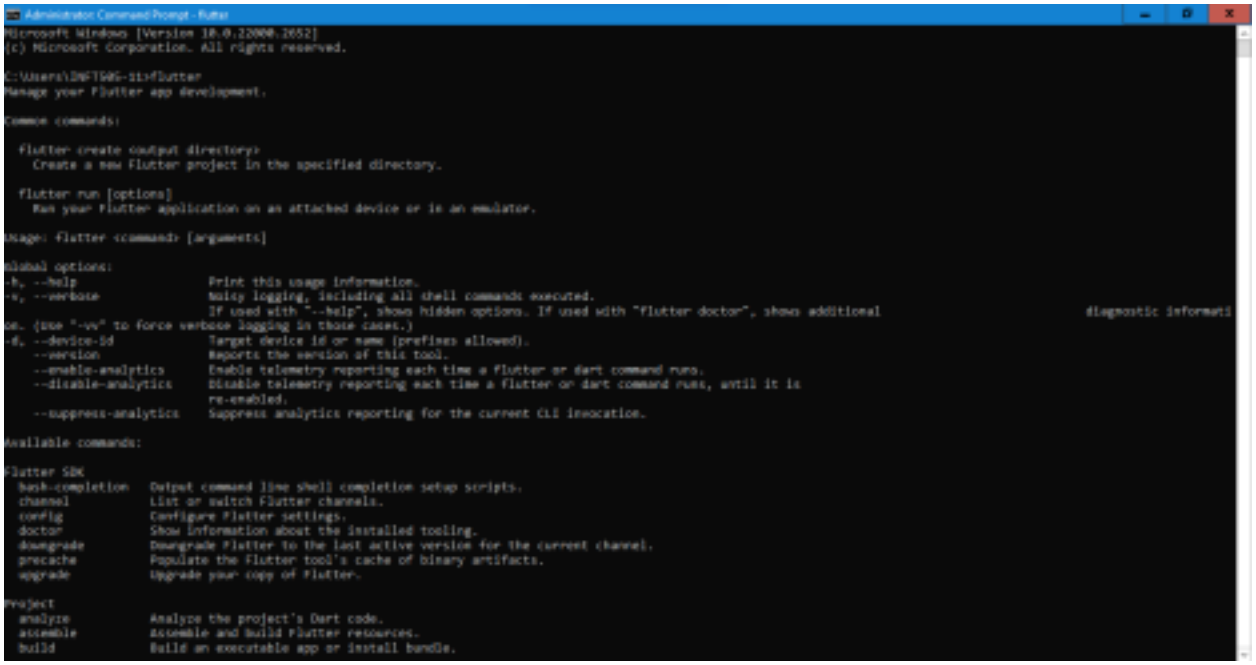
Pre Requisites: Android Studio Hedgehog, Visual Studio Code and Flutter package.I highly recommend watching this video, for Setting up your Flutter Environment and your Virtual Device: https://youtu.be/ZSWfgxrxN0M?feature=shared

INSTALLATION :






docs.flutter.dev/get-started/install/windows/desktop?tab=download
Flutter
Multi-Platform
Development
Ecosystem
Showcase
Docs
Get started
Get started
Install Flutter
Test drive
Write your first app
Learn more
From another platform?
Dart language overview
Stay up to date
Samples & codelabs
App solutions
User interface
Introduction
Widget catalog
Layout
Design & theming
Install the Flutter SDK
To install the Flutter SDK, you can use the VS Code Flutter extension or download and install the Flutter bundle yourself.
Use VS Code to install
Download and install
Download then install Flutter
1. Download the following installation bundle to get the latest stable release of the Flutter SDK.
flutter_windows_3.16.5-stable.zip
For other release channels, and older builds, check out the SDK archive.
This guide presumes that you downloaded your Flutter SDK to the default download directory for Windows: %USERPROFILE%\Downloads\.
2. Create a folder where you can install Flutter.
Consider %USERPROFILE% or C:\dev.
Contents
System requirements
Hardware requirements
Software requirements
Configure a text editor or IDE
Install the Flutter SDK
Configure Android development
Configure the Android toolchain in Android Studio
Configure your target Android device
Agree to Android licenses
Check your

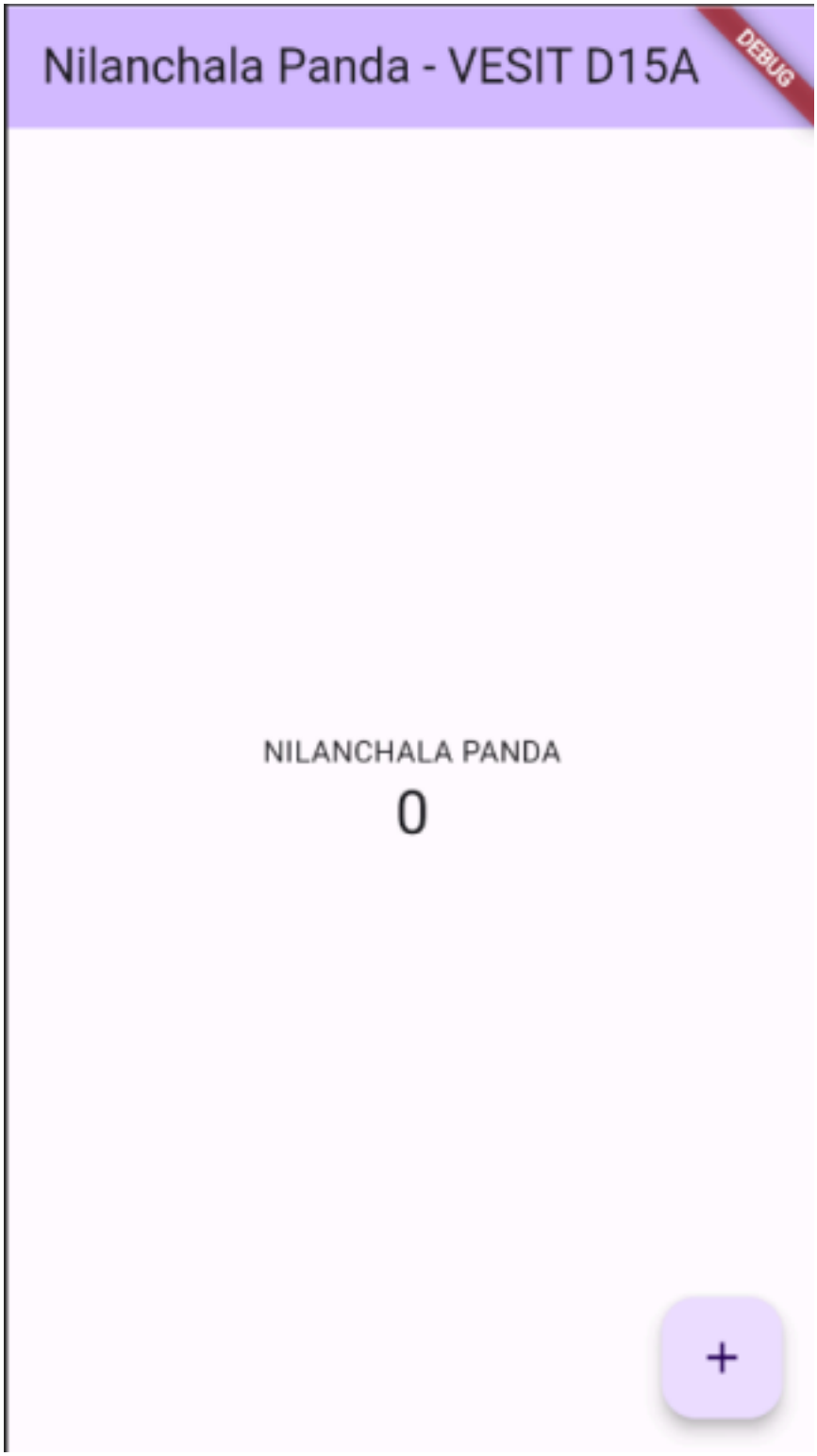

Administrator: Command Prompt - flutter
Microsoft Windows [Version 10.0.22000.2652]
(c) Microsoft Corporation. All rights reserved.
C:\Users\DNFT006-1>flutter
Manage your Flutter app development.
Common commands:
flutter create <output directory>
Create a new Flutter project in the specified directory.
flutter run [options]
Run your Flutter application on an attached device or in an emulator.
Usage: flutter <command> [arguments]
Global options:
-h, --help
-v, --verbose
-d, --device-id
--version
--enable-analytics
--disable-analytics
--suppress-analytics
Available commands:
Flutter SDK
bash-completion
channel
config
doctor
downgrade
precache
upgrade
Project
analyze
assemble
build

```
C:\Users\INFT505-11>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.16.5, on Microsoft Windows [Version 10.0.22000.2652], locale en-US)
[√] Windows Version (Installed version of Windows is version 10 or higher)
[!] Android toolchain - develop for Android devices (Android SDK version 33.0.1)
    ! Some Android licenses not accepted. To resolve this, run: flutter doctor --android-licenses
[√] Chrome - develop for the web
[X] Visual Studio - develop Windows apps
    X Visual Studio not installed; this is necessary to develop Windows apps.
      Download at https://visualstudio.microsoft.com/downloads/.
      Please install the "Desktop development with C++" workload, including all of its default components
[√] Android Studio (version 2021.3)
[√] VS Code (version 1.76.0)
[√] Connected device (3 available)
[√] Network resources

! Doctor found issues in 2 categories.

C:\Users\INFT505-11>
```

Code:

```dart
  void _incrementCounter() {
    setState(() {
      _counter = _counter + 2;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Theme.of(context).colorScheme.inversePrimary,
        title: Text(widget.title),
      ), // AppBar
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            const Text(
              'NILANCHALA PANDA',
            ), // Text
            Text(
              '$_counter',
              style: Theme.of(context).textTheme.headlineMedium,
            ), // Text
          ], // <Widget>[]
        ), // Column
      ), // Center
      floatingActionButton: FloatingActionButton(
        onPressed: _incrementCounter,
        tooltip: 'Increment',
        child: const Icon(Icons.add),
      ), // This trailing comma makes auto-formatting nicer for build methods. // FloatingActionButton
    ); // Scaffold
  }
}
```

Output:



Conclusion: Hence We ran a simple program on running a simple text,on flutter, running on a virtual device

MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	41
Name	Nilanchala Panda
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	13

Experiment 2

Name: Nilanchala Panda Div .: D15A Roll No .: 41 Batch: B

Aim: To design Flutter UI by including common widgets.

Theory:

Widgets: Each element on a screen of the Flutter app is a widget. The view of the screen
completely depends upon the choice and sequence of the widgets used to build the apps. And the structure of the code of an app is a tree of widgets. **Category of Widgets:**
There are mainly 14 categories in which the flutter widgets are divided. They are mainly
segregated on the basis of the functionality they provide in a flutter application. 1. Accessibility: These are the set of widgets that make a flutter app more easily accessible.
2. Animation and Motion: These widgets add animation to other widgets. 3. Assets, Images, and Icons: These widgets take charge of assets such as display images and show icons.
4. Async: These provide async functionality in the flutter application. 5. Basics: These are the bundle of widgets that are absolutely necessary for the development of any flutter application.
6. Cupertino: These are the iOS designed widgets.
7. Input: This set of widgets provides input functionality in a flutter application. 8. Interaction Models: These widgets are here to manage touch events and route users to
different views in the application.
9. Layout: This bundle of widgets helps in placing the other widgets on the screen as needed.
10. Material Components: This is a set of widgets that mainly follow material design by Google.
11. Painting and effects: This is the set of widgets that apply visual changes to their child
widgets without changing their layout or shape.
12. Scrolling: This provides scrollability of to a set of other widgets that are not scrollable by
default.
13. Styling: This deals with the theme, responsiveness, and sizing of the app.
14. Text: This displays text.
Description of few of the widgets are as follows:

● Scaffold – Implements the basic material design visual layout structure. ●
App-Bar – To create a bar at the top of the screen.
● Text - To write anything on the screen.
● Container – To contain any widget.
● Center – To provide center alignment to other widgets.

CODE :

```
import 'package:dunzo/widget/widget_support.dart';
import 'package:flutter/material.dart';

class Home extends StatefulWidget {
  const Home({Key? key}) : super(key: key);

  @override
  State<Home> createState() => _HomeState();
}

class _HomeState extends State<Home> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Container(
        margin: const EdgeInsets.only(top: 20.0, left: 20.0, right: 10.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Row(
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: [
                Text(
                  "Hello Nilanchala,",
                  style: AppWidget.boldTextFieldStyle(),
                ),
                Padding(
                  padding: const EdgeInsets.fromLTRB(0, 0, 10.0, 0),
                  child: Container(
                    decoration: BoxDecoration(
                        color: Colors.black,
                        borderRadius: BorderRadius.circular(8)),
                    child: const Icon(Icons.shopping_cart, color: Colors.white), ),
                )
```

```
  ],
),

const SizedBox(height: 15.0),
// MAIN HEADING -
Text(
  "Order Your Food Now!",
  style: AppWidget.headerTextFieldStyle(),
),
// SUBHEADING -
Text(
  "Discover and Get Fresh Vegetables",
  style: AppWidget.lightTextFieldStyle(),
),

const SizedBox(height: 10.0),

Row(
  children: [
    ClipRRect(
      borderRadius: BorderRadius.circular(15.0),
      child: Material(
        elevation: 20.0,
        // ignore: avoid_unnecessary_containers
        child: Container(
          child: Image.asset(
            "images/mainBanner.jpg",
            height: 150,
            width: 340,
            fit: BoxFit.cover,
          ),
        ),
      ),
    ),
  ],
),

Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: [
    ClipRRect(
```

```
                  borderRadius: BorderRadius.circular(15.0),
                  child: Material(
                    elevation: 20.0,
                    // ignore: avoid_unnecessary_containers
                    child: Container(
                      child: Image.asset(
                        "images/fruitBasket.png",
                        height: 160,
                        width: 160,
                        fit: BoxFit.cover,
                      ),
                    ),
                  ),
                ),
                const SizedBox(width: 10), // Add some space between the images
                ClipRRect(
                  borderRadius: BorderRadius.circular(15.0),
                  child: Material(
                    elevation: 20.0,
                    // ignore: avoid_unnecessary_containers
                    child: Container(
                      child: Image.asset(
                        "images/fruitBasket.png", // Replace with your second image path
                        height: 160,
                        width: 160,
                        fit: BoxFit.cover,
                      ),
                    ),
                  ),
                ),
              ],
            )
          ],
        ),
      ),
    );
  }
}
```

OUTPUT :


12:40
Hello Nilanchala,
Order Your Food Now!
Discover and Get Fresh Vegetables
NYKAA
The Buzzword Across the Nation!
Here's Why

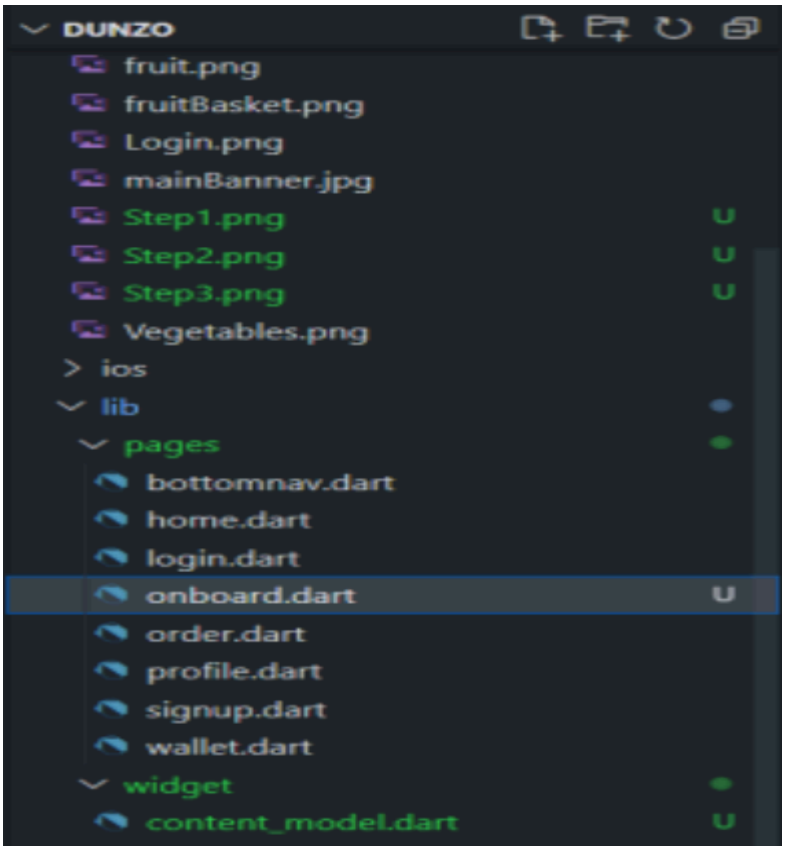
MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	41
Name	Nilanchala Panda
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 3

Aim: To include icons, images, fonts.

File Structure:



CODE:

```
import 'package:dunzo/pages/signup.dart'; import
'package:dunzo/widget/content_model.dart'; import
'package:dunzo/widget/widget_support.dart'; import
'package:flutter/material.dart';

class OnBoard extends StatefulWidget {
const OnBoard({super.key});

@override
State<OnBoard> createState()
_OnBoardState(); }

                class _OnBoardState extends
                State<OnBoard> { int currentIndex = 0;
```

```
late PageController _controller;

@override
void initState() {
  _controller = PageController(initialPage: 0);

  super.initState();
}

@override
void dispose() {
  _controller.dispose();
  super.dispose();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
body: Column(children: [
Expanded(
child: PageView.builder(
controller: _controller,
itemCount: contents.length,
onPageChanged: (int index) {
setState(() {
currentIndex = index;
});
},
itemBuilder: (_, i) {
return Padding(
padding: const EdgeInsets.all(20),
child: Column(
children: [
Image.asset(contents[i].image,
height: 450,
          width: MediaQuery.of(context).size.width,
fit: BoxFit.cover),
const SizedBox(
height: 40.0,
),
Text(contents[i].title,
```

```
                style: AppWidget.boldTextFieldStyle()),
const SizedBox(
height: 40.0,
),
Text(contents[i].description,
                style: AppWidget.boldTextFieldStyle()),
],
),
);
}),
),
Container(
child: Row(
mainAxisAlignment: MainAxisAlignment.center,
children: List.generate(
contents.length,
(index) => buildDot(index, context),
),
),
),
GestureDetector(
onTap: () {
if (currentIndex == contents.length - 1) {
Navigator.push(context,
                          MaterialPageRoute(builder: (context) =>
                          constSignUp())); }
                          _conroller.nextPage(
                              dration: const Duration(milliseconds: 100),
                              urve: Curves.bounceIn);
                        },
child: Container(
decoration: const BoxDecoration(color: Colors.red),
height: 60,
margin: const EdgeInsets.all(40),
width: double.infinity,
child: const Text(
"Next",
style: TextStyle(
color: Colors.white,
),
),
```

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

Container buildDot(int index, BuildContextcontext) { return
Container(
height: 100,
width: curentIndex == index ? 18 : 7,
margin: const EdgeInsets.only(right: 5),
decoration: BoxDecoration(
                    borderRadius: BorderRadius.circular(6),
                    color:Colors.black), );
}
```

MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	41
Name	Nilanchala Panda
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

EXPERIMENT 4

Name : Nilanchala Panda Div : D15A Batch : B Roll No : 41

Aim: To create an interactive form widget.

Code:

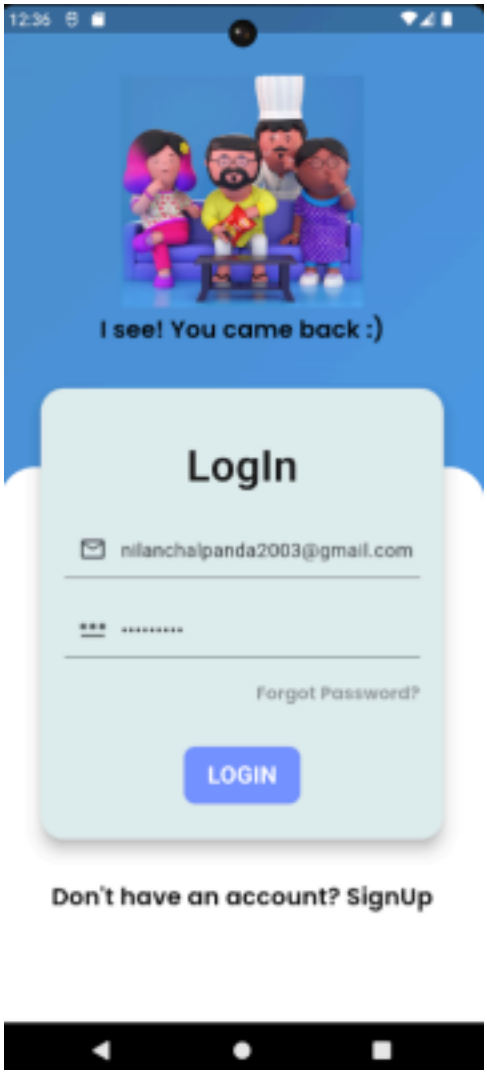
```
import "package:dunzo/pages/signup.dart";
import "package:flutter/material.dart";
import "package:dunzo/widget/widget_support.dart";

class LogIn extends StatefulWidget {
  const LogIn({super.key});

  @override
  State<LogIn> createState() => _LogInState();
}

class _LogInState extends State<LogIn> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      // ignore: avoid_unnecessary_containers
      body: Container(
        child: Stack(
          children: [
            Container(
              width: MediaQuery.of(context).size.width,
              height: MediaQuery.of(context).size.height / 2,
              decoration: const BoxDecoration(
                  gradient: LinearGradient(
                      begin: Alignment.topLeft,
                      end: Alignment.bottomRight,
                      colors: [
                    Color.fromRGBO(76, 144, 206, 1.0),
                    Color.fromRGBO(76, 152, 228, 1.0),
                  ])),
            ),
            Container(
                margin: EdgeInsets.only(
                    top: MediaQuery.of(context).size.height / 2.2),
                height: MediaQuery.of(context).size.height,
                width: MediaQuery.of(context).size.width,
                decoration: const BoxDecoration(
                    color: Colors.white,
                    borderRadius: BorderRadius.only(
                        topLeft: Radius.circular(30),
                        topRight: Radius.circular(30))),
```

```
    child: const Text("")),
Container(
  margin: const EdgeInsets.only(top: 60.0),
  child: SingleChildScrollView(
    child: Column(
      children: [
        Center(
          child: Image.asset(
            "images/dunzoLogin.png",
            width: MediaQuery.of(context).size.width / 2,
            fit: BoxFit.cover,
          ),
        ),
        // ignore: avoid_unnecessary_containers
        Container(
          child: Column(
            children: [
              const SizedBox(height: 5.0),
              Text(
                "I see! You came back :)",
                style: AppWidget.boldTextFieldStyle(),
              )
            ],
          ),
        ),

        // SIGNUP BOX
        // ignore: avoid_unnecessary_containers
        Card(
          color: Colors.white,
          margin: const EdgeInsets.all(35.0),
          shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(20.0)),
                        elevation: 10.0,
                        child: Padding(
                          padding:
                              const EdgeInsets.fromLTRB(20.0, 30.0,
                          20.0, 30.0), child: Column(
                            children: [
                              const SizedBox(height: 10.0),
                              const Text(
                                "LogIn",
                                style: TextStyle(
                                    fontSize: 37.0,
                                    fontFamily: "Poppins1",
                                    fontWeight: FontWeight.w600),
                              ),
                              const SizedBox(height: 20.0),
                              TextField(
```

```
            decoration: InputDecoration(
              hintText: "Email",
  hintStyle: AppWidget.lightTextFieldStyle(),
  prefixIcon: const Icon(Icons.email_outlined),
            ),
          ),
          const SizedBox(height: 20.0),
          TextField(
            obscureText: true,
            decoration: InputDecoration(
              hintText: "Password",
  hintStyle: AppWidget.lightTextFieldStyle(),
              prefixIcon: const
            Icon(Icons.password_outlined), ),
          ),
          const SizedBox(height: 20.0),
          Container(
              alignment: Alignment.topRight,
              child: Text(
                "Forgot Password?",
    style: AppWidget.lightTextFieldStyle(),
              )),

     // ignore: avoid_unnecessary_containers
          Container(
margin: const EdgeInsets.only(top: 35.0),
            padding: const EdgeInsets.only(
                top: 10.0,
                left: 20.0,
                right: 20.0,
                bottom: 10.0),
            decoration: BoxDecoration(
                color: const Color.fromRGBO(115, 148,
                255, 1.0), borderRadius:
                BorderRadius.circular(10)),
            child: const Text(
              "LOGIN",
              style: TextStyle(
                  color: Colors.white,
                  fontSize: 20.0,
                  fontFamily: "Poppins1",
           fontWeight: FontWeight.bold),
            ),
          ),
        ],
      ),
    ),
  ),
```

```
                GestureDetector(
                  onTap: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(
                    builder: (context) => const SignUp()),
                      );
                  },
                  child: Text(
                    "Don't have an account? SignUp",
                    style: AppWidget.boldTextFieldStyle(),
                  ),
                )
              ],
            ),
          ),
        ),
      ],
    ),
  ),
 );
 }
}
```

OUTPUT:


12:36
I see! You came back :)
LogIn
nilanchalpanda2003@gmail.com
Forgot Password?
LOGIN
Don't have an account? SignUp

MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	41
Name	Nilanchala Panda
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	14

EXPERIMENT NO 5

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

Theory:

Navigation routing in Flutter refers to the process of defining and managing the routes or screens within your app. Flutter's navigation routing system allows you to move between different screens or pages in a structured and organized way. Here's an overview:

Route Definition: In Flutter, each screen or page in your app is typically represented by a widget. To define routes, you create a mapping between route names and the corresponding widget classes. This mapping is often done in the MaterialApp widget using the routes parameter.

Named Routes: Named routes are a common approach in Flutter for defining routes using unique names. Each route is associated with a name, making it easier to navigate to specific screens by referencing these names. For example, you can define routes like '/home', '/profile', '/settings', etc.

Navigation Stack: Flutter's navigation system maintains a stack of route objects. When you navigate to a new screen, the new route is pushed onto the stack. When you navigate back, the top route is popped off the stack, returning you to the previous screen.

Navigator: The Navigator class in Flutter is responsible for managing the navigation stack and transitioning between routes. You can use methods like Navigator.push() to navigate to a new route, Navigator.pop() to return to the previous route, and Navigator.pushNamed() to navigate to a named route.

MaterialPageRoute and CupertinoPageRoute: Flutter provides two types of page route classes - MaterialPageRoute for Android-style navigation and CupertinoPageRoute for iOS-style navigation. These classes define transitions and behaviors specific to each platform.

Route Parameters: You can pass parameters between routes in Flutter using the arguments parameter of the navigation methods (Navigator.pushNamed() or Navigator.push()). This allows you to send data to the next screen or receive data from the previous screen.
Modal Routes: Modal routes are routes that appear on top of other routes and typically block interaction with the underlying routes until they are dismissed. You can create modal routes using methods like showDialog() or by using Navigator.push() with a PageRouteBuilder and setting the fullscreenDialog parameter to true.

Code:

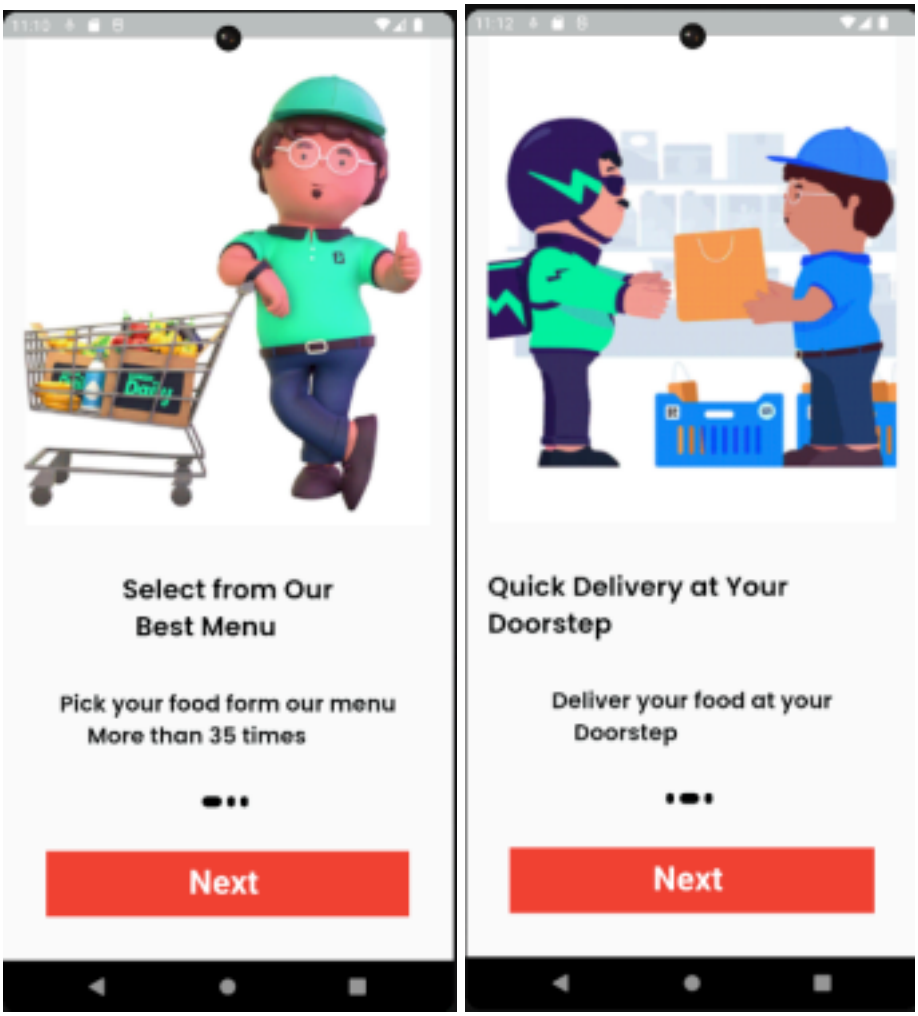
```
import 'package:dunzo/pages/signup.dart';
```

```
import 'package:dunzo/widget/content_model.dart';
import 'package:dunzo/widget/widget_support.dart';
import 'package:flutter/material.dart';

class OnBoard extends StatefulWidget {
  const OnBoard({super.key});

  @override
  State<OnBoard> createState() => _OnBoardState();
}

class _OnBoardState extends State<OnBoard> {
  int currentIndex = 0;
  late PageController _controller;

  @override
  void initState() {
    _controller = PageController(initialPage: 0);

    super.initState();
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Column(children: [
        Expanded(
          child: PageView.builder(
              controller: _controller,
              itemCount: contents.length,
              onPageChanged: (int index) {
                setState(() {
                  currentIndex = index;
                });
              },
              itemBuilder: (_, i) {
                return Padding(
                  padding: const EdgeInsets.all(20),
                  child: Column(
```

```
            children: [
              Image.asset(contents[i].image,
                  height: 450,
                  width: MediaQuery.of(context).size.width,
                  fit: BoxFit.cover),
              const SizedBox(
                height: 40.0,
              ),
              Text(contents[i].title,
                  style: AppWidget.headerTextFieldStyle()),
              const SizedBox(
                height: 40.0,
              ),
              Text(contents[i].description,
                  style: AppWidget.boldTextFieldStyle()),
            ],
          ),
        );
      }),
),
Container(
  child: Row(
    mainAxisAlignment: MainAxisAlignment.center,
    children: List.generate(
      contents.length,
      (index) => buildDot(index, context),
    ),
  ),
),
GestureDetector(
  onTap: () {
    if (currentIndex == contents.length - 1) {
      Navigator.push(context,
          MaterialPageRoute(builder: (context) => const
    SignUp())); }
    _controller.nextPage(
        duration: const Duration(milliseconds: 100),
        curve: Curves.bounceIn);
  },
  child: Container(
    decoration: const BoxDecoration(color:
    Colors.red), height: 60,
    padding: const EdgeInsets.only(left: 130, top: 7),
    margin: const EdgeInsets.all(40),
    width: double.infinity,
```

```
            child: const Text(
              "Next",
              style: TextStyle(
                color: Colors.white,
                fontSize: 30,
                fontWeight: FontWeight.bold,
                fontFamily: "Poppins1",
              ),
            ),
          ),
        )
      ]),
    );
  }

  Container buildDot(int index, BuildContext context)
    { return Container(
      height: 10.0,
      width: currentIndex == index ? 18 : 7,
      margin: const EdgeInsets.only(right: 5),
      decoration: BoxDecoration(
          borderRadius: BorderRadius.circular(6), color:
    Colors.black), );
  }
}
```

Output:


Select from Our
Best Menu
Pick your food form our menu
More than 35 times
Next
Quick Delivery at Your
Doorstep
Deliver your food at your
Doorstep
Next

MAD & PWA Lab
Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	41
Name	Nilanchala Panda
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	14

EXPERIMENT NO 6

Aim: To connect firebase database with flutter UI

Theory:

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

In this article, you will create a Firebase project for iOS and Android platforms using Flutter.

Prerequisites

To complete this tutorial, you will need:

- A Google account to use Firebase.
- Developing for iOS will require XCode.
- To download and install Flutter.
- To download and install Android Studio and Visual Studio Code.
- It is recommended to install plugins for your code editor:
    - `Flutter` and `Dart` plugins installed for Android Studio.
    - `Flutter` extension installed for Visual Studio Code.

This tutorial was verified with Flutter v2.0.6, Android SDK v31.0.2, and Android Studio v4.1.

Creating a New Flutter Project

This tutorial will require the creation of an example Flutter app.

Once you have your environment set up for Flutter, you can run the following to create a new application:

```
flutter create flutterfirebaseexample
```

1.
Copy

Navigate to the new project directory:

```
cd flutterfirebaseexample
```

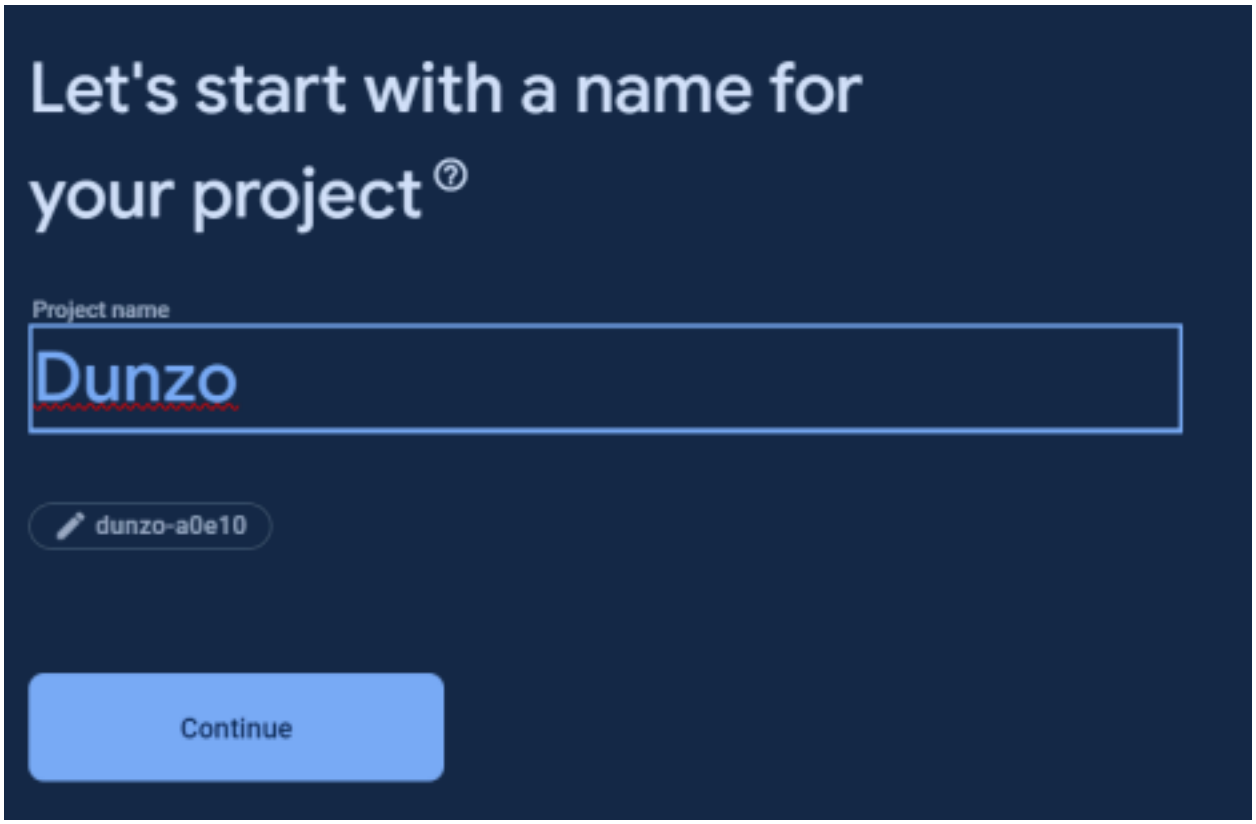
Copy

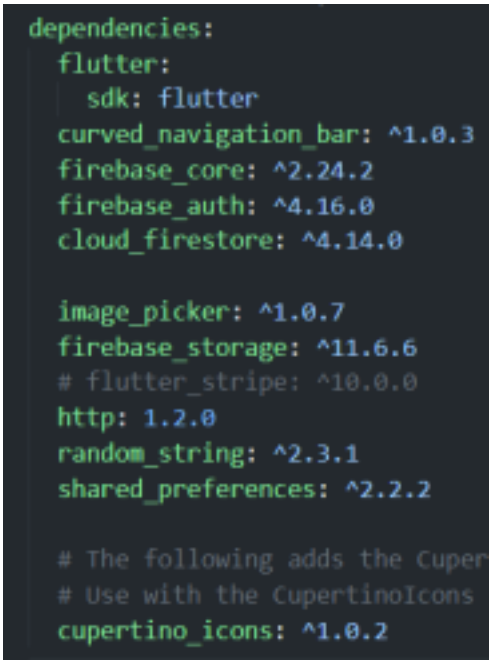
Using `flutter create` will produce a demo application that will display the number of times a button is clicked.

Now that we've got a Flutter project up and running, we can add Firebase.

Creating a New Firebase Project

First, log in with your Google account to manage your Firebase projects. From within the Firebase dashboard, select the Create new project button and give it a name:





```
dependencies:
  flutter:
    sdk: flutter
  curved_navigation_bar: ^1.0.3
  firebase_core: ^2.24.2
  firebase_auth: ^4.16.0
  cloud_firestore: ^4.14.0

  image_picker: ^1.0.7
  firebase_storage: ^11.6.6
  # flutter_stripe: ^10.0.0
  http: 1.2.0
  random_string: ^2.3.1
  shared_preferences: ^2.2.2

  # The following adds the Cuper
  # Use with the CupertinoIcons
  cupertino_icons: ^1.0.2
```

Be sure to move this file within Xcode to create the proper file references.

There are additional steps for installing the Firebase SDK and adding initialization code, but they are not necessary for this tutorial.

That's it!

Conclusion

In this article, you learned how to set up and ready our Flutter applications to be used with Firebase.

Flutter has official support for Firebase with the FlutterFire set of libraries.

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.	41
Name	Nilanchala Panda
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	15

EXPERIMENT NO 7

Aim:-

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

Theory:-

Regular Web App

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

Progressive Web App

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

Difference between PWAs vs. Regular Web Apps:

A Progressive Web is different and better than a Regular Web app with features like: 1. Native Experience

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

2. Ease of Access

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

easily accessible to the users and helps the brands remain in the users' minds,

and improving the chances of interaction.

3. Faster Services
PWAs can cache the data and serve the user with text stylesheets, images, and other web content even before the page loads completely. This lowers the waiting time for the end-users and helps the brands improve the user engagement and retention rate, which eventually adds value to their business.

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

5. Updated Real-Time Data Access
Another plus point of PWAs is that these apps get updated on their own. They do not demand the end-users to go to the App Store or other such platforms to download the update and wait until installed.
In this app type, the web app developers can push the live update from the server, which reaches the apps residing on the user's devices automatically. Therefore, it is easier for the mobile app developer to provide the best of the updated functionalities and services to the end-users without forcing them to update their app. 6. Discoverable
PWAs reside in web browsers. This implies higher chances of optimizing them as per the Search Engine Optimization (SEO) criteria and improving the Google rankings like that in websites and other web apps.

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

Pros and cons of the Progressive Web App
The main features are:

Progressive — They work for every user, regardless of the browser chosen because

they are built at the base with progressive improvement principles.
Responsive — They adapt to the various screen sizes: desktop, mobile, tablet, or dimensions that can later become available.

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

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

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

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

Reactivable — Make it easy to reactivate the application thanks to capabilities such as web notifications.
Installable — They allow the user to “save” the apps that he considers most useful with the corresponding icon on the screen of his mobile terminal (home screen) without having to face all the steps and problems related to the use of the app store.

Linkable — Easily shared via URL without complex installations.

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

CODE -

index.html -

```
<!DOCTYPE html>
<html>
  <head>
    <!-- Responsive -->
    <meta charset="utf-8" />
    <meta
      name="viewport"
      content="width=device-width,
```

```
                initial-scale=1"
    />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />

    <!-- Title -->
    <title>PWA Tutorial</title>

    <!-- Meta Tags required for
    Progressive Web App -->
    <meta name="apple-mobile-web-app-status-bar" content="#aa7700"
    /> <meta name="theme-color" content="black" />

    <!-- Manifest File link -->
    <link rel="manifest" href="manifest.json" />
  </head>

  <body>
    <h1 style="color: blue; font-family: 'Courier New',
Courier, monospace; font-size: 3rem;">NILANCHALA PANDA</h1>

    <p>This is a simple tutorial for creating a PWA application.</p>

    <script>
      // Add event listener to execute code when page loads
      window.addEventListener("load", () => {
        // Call registerSW function when page loads
        registerSW();
      });

      // Register the Service Worker
      async function registerSW() {
        // Check if browser supports Service Worker
        if ("serviceWorker" in navigator) {
          try {
            // Register the Service Worker named 'serviceworker.js'
          await navigator.serviceworker.register("serviceworker.js");
          } catch (e) {
            // Log error message if registration fails
            console.log("SW registration failed");
          }
        }
```

```
    }
  </script>
 </body>
</html>
```

manifest.json -

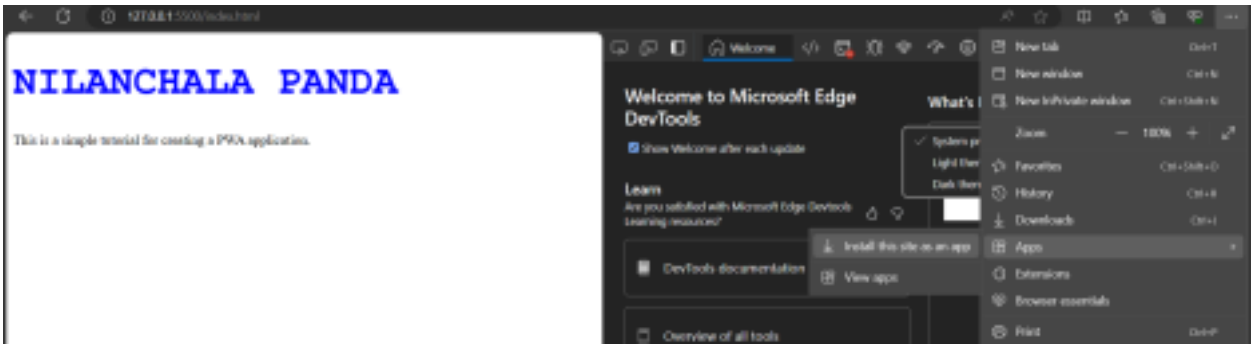
```
{
  "name": "PWA Tutorial",
  "short_name": "PWA",
  "start_url": "index.html",
  "display": "standalone",
  "background_color": "#5900b3",
  "theme_color": "black",
  "scope": ".",
  "description": "This is a PWA tutorial.",
  "icons": [
    {
      "src": "images/icon-192x192.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "images/icon-512x512.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ]
}
```

serviceworker.js -

```
var staticCacheName = "pwa";

self.addEventListener("install", function (e)
  { e.waitUntil(
    caches.open(staticCacheName).then(function (cache)
      { return cache.addAll(["/"]);
    })
  );
});

self.addEventListener("fetch", function (event) {
```

```
  console.log(event.request.url);

  event.respondWith(
    caches.match(event.request).then(function (response) {
      return response || fetch(event.request);
    })
  );
});
```

OUPUT :



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.	41
Name	Nilanchala Panda
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

Name : Nilanchala Panda **Batch : B**
Div . : 41 **Roll No. : 41**

EXPERIMENT NO 8

Aim:

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

Theory:

Service Worker

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

Things to note about Service Worker:

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

What can we do with Service Workers?

- You can dominate Network Traffic

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

- You can Cache

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

- You can manage Push Notifications

  You can manage push notifications with Service Worker and show any information message to the user.

- You can Continue

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

What can't we do with Service Workers?

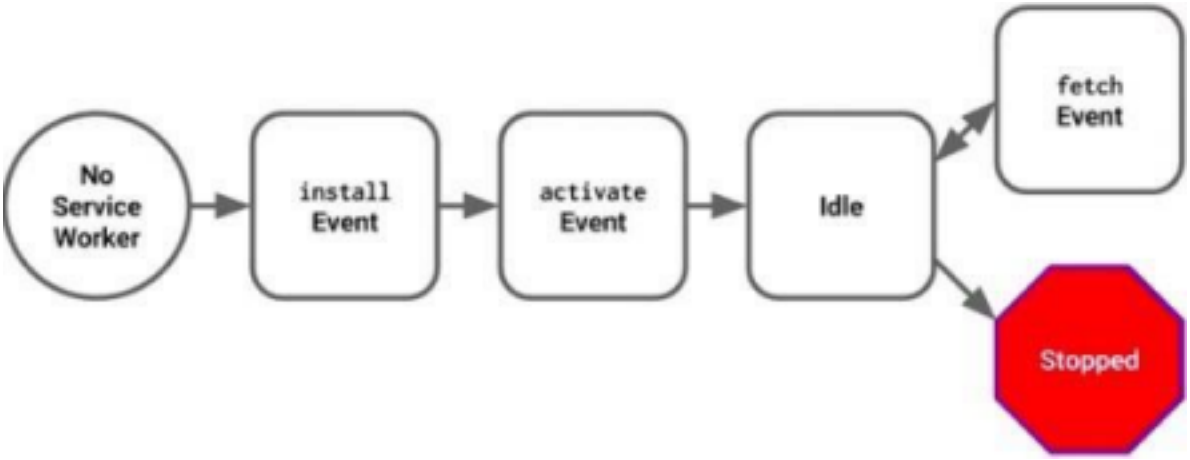
- You can't access the Window

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

- You can't work it on 80 Port

  Service Worker just can work on HTTPS protocol. But you can work on localhost during development.

Service Worker Cycle



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

- Registration
- Installation
- Activation

CODE -

index.html -

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
    <title>E-commerce PWA</title>
    <link rel="stylesheet" href="main.css" />
  </head>
  <body>
    <h1>Welcome to Our E-commerce Store</h1>
    <!-- E-commerce content goes here -->
    <script src="app.js"></script>
  </body>
</html>
```

service-worker.js -

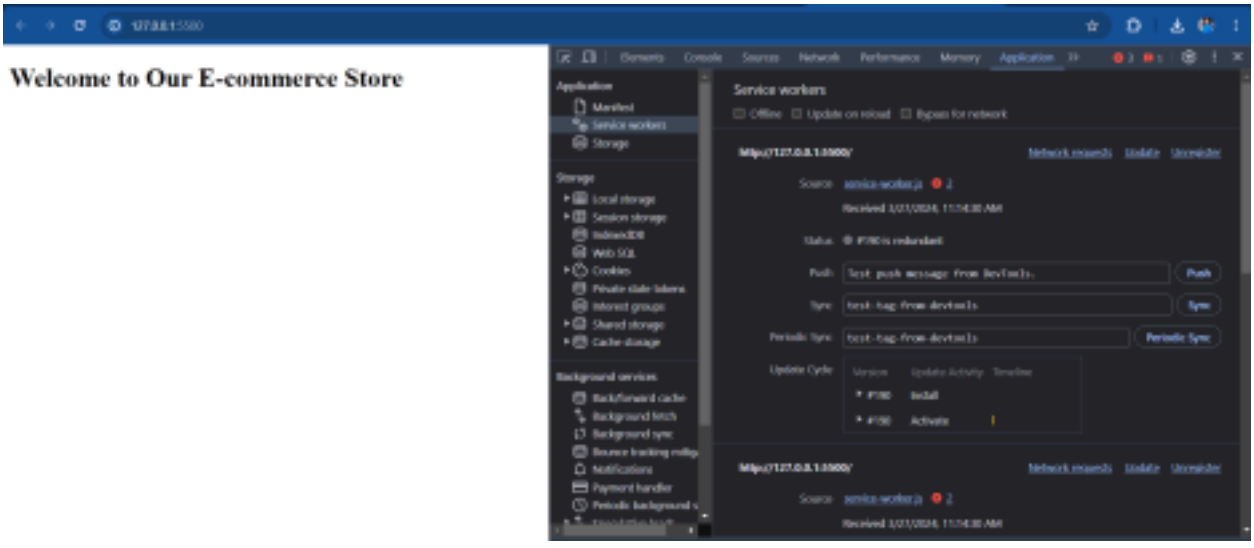
```
const cacheName = "ecommerce-pwa-v1";
const assetsToCache = [
  "/",
  "/index.html",
  "/app.js",
];

self.addEventListener("install", (event) => {
  event.waitUntil(
    caches.open(cacheName).then((cache) => {
      return cache.addAll(assetsToCache);
    })
  );
});

self.addEventListener("activate", (event) => {
  event.waitUntil(
    caches.keys().then((cacheNames) => {
      return Promise.all(
        cacheNames
          .filter((name) => {
            return name !== cacheName;
          })
```

```
          .map((name) => {
            return caches.delete(name);
          })
      );
    })
  );
});
```

app.js -

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

OUTPUT -



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.	41
Name	Nilanchala Panda
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

Name : Nilanchala Panda **Batch : B**
Div . : 41 **Roll No . : 41**

EXPERIMENT NO 9

Aim:

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

Theory:

Service Worker

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

Things to note about Service Worker:

● A service worker is a programmable network proxy that lets you control how network requests from your page are handled.

● Service workers only run over HTTPS. Because service workers can intercept network requests and modify responses, "man-in-the-middle" attacks could be very bad.

● The service worker becomes idle when not in use and restarts when it's next needed. You cannot rely on a global state persisting between events. If there is information that you need to persist and reuse across restarts, you can use IndexedDB databases.

● Service workers make extensive use of promises, so if you're new to promises, then you should stop reading this and check out Promises, an introduction.

Fetch Event

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

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

● CacheFirst - In this function, if the received request has cached before, the cached response is returned to the page. But if not, a new response requested from the network.

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

CODE -

index.html -

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>E-commerce PWA</title>
<link rel="stylesheet" href="main.css" />
</head>
<body>
<h1>Welcome to Our E-commerce Store</h1>
<!-- E-commerce content goes here -->
<script src="app.js"></script>
</body>
</html>
```

service-worker.js -

```
const CACHE_NAME = "my-ecommerce-app-cache-v1";
const urlsToCache = [
"/",
"index.html",
"service-worker.js",
"manifest.json",
];

var staticCacheName = "pwa";
self.addEventListener("install", function (e) {
e.waitUntil(
caches.open(staticCacheName).then(function (cache) { return
```

```
cache.addAll(["/"]);
})
);
});

self.addEventListener("fetch", function (event) {
console.log(event.request.url);

event.respondWith(
caches.match(event.request).then(function (response) { return response ||
fetch(event.request);
})
);
});

self.addEventListener("install", function (event) { event.waitUntil(
caches.open(CACHE_NAME).then(function (cache) {
console.log("Opened cache");
return cache.addAll(urlsToCache).catch(function (error) {
console.error("Cache.addAll error:", error);
});
})
);
});

self.addEventListener("activate", function (event) { // Perform activation
steps
event.waitUntil(
caches.keys().then(function (cacheNames) {
return Promise.all(
cacheNames.map(function (cacheName) {
if (cacheName !== CACHE_NAME) {
return caches.delete(cacheName);
}
})
);
})
);
});

// Fetch event listener
```

```
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));
});

// Sync event listener
self.addEventListener("sync", function (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); }
}
});
self.addEventListener("activate", async () => {
if (Notification.permission !== "granted") {
try {
const permission = await Notification.requestPermission(); if (permission
=== "granted") {
                            console.log("Notification permission granted.");
} else {
                            console.warn("Notification permission denied.");
}
```

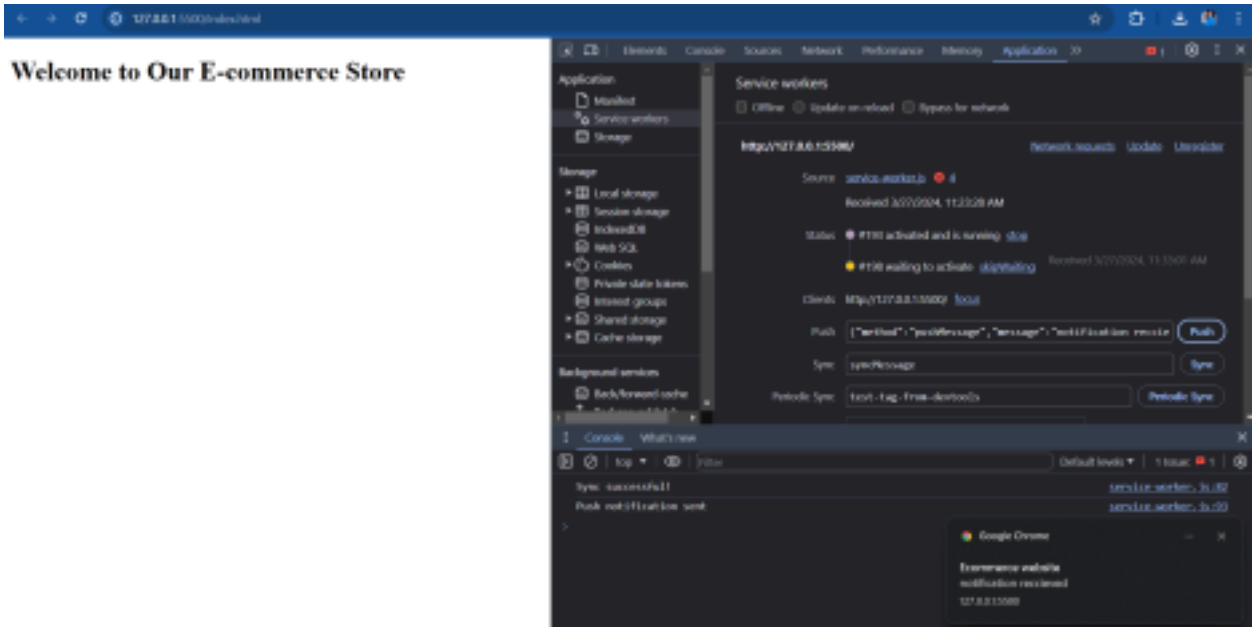
```
} catch (error) {
console.error("Failed to request notification permission:", error); }
}
});

var checkResponse = function (request) {
return new Promise(function (fulfill, reject) {
fetch(request)
.then(function (response) {
if (response.status !== 404) {
fulfill(response);
} else {
reject(new Error("Response not found"));
}
})
.catch(function (error) {
reject(error);
});
});
};

var returnFromCache = function (request) {
return caches.open("offline").then(function (cache) {
return cache.match(request).then(function (matching) {
if (!matching || matching.status == 404) {
return cache.match("offline.html");
} else {
return matching;
}
});
});
};

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

```
};
```

app.js -

```
if ("Notification" in window) {
Notification.requestPermission().then(function (permission) { if
(permission === "granted") {
console.log("Notification permission granted.");
} else {
console.warn("Notification permission denied.");
}
});
}
```

OUTPUT -



MAD & PWA Lab
Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	41
Name	Nilanchala Panda
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

Name : Nilanchala Panda **Batch : B**
Div . : 41 **Roll No . : 41**

EXPERIMENT NO 10

Aim:

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

Theory:

GitHub Pages

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

GitHub Pages provides the following key features:

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

Reasons for favoring this over Firebase:

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

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

Pros

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

Cons

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

3. Although Jekyll is supported, plug-in support is rather spotty.

Firebase

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

Some of the features offered by Firebase are:

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

Reasons for favoring over GitHub Pages:

1. Realtime backend made easy
2. Fast and responsive

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

Pros

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

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

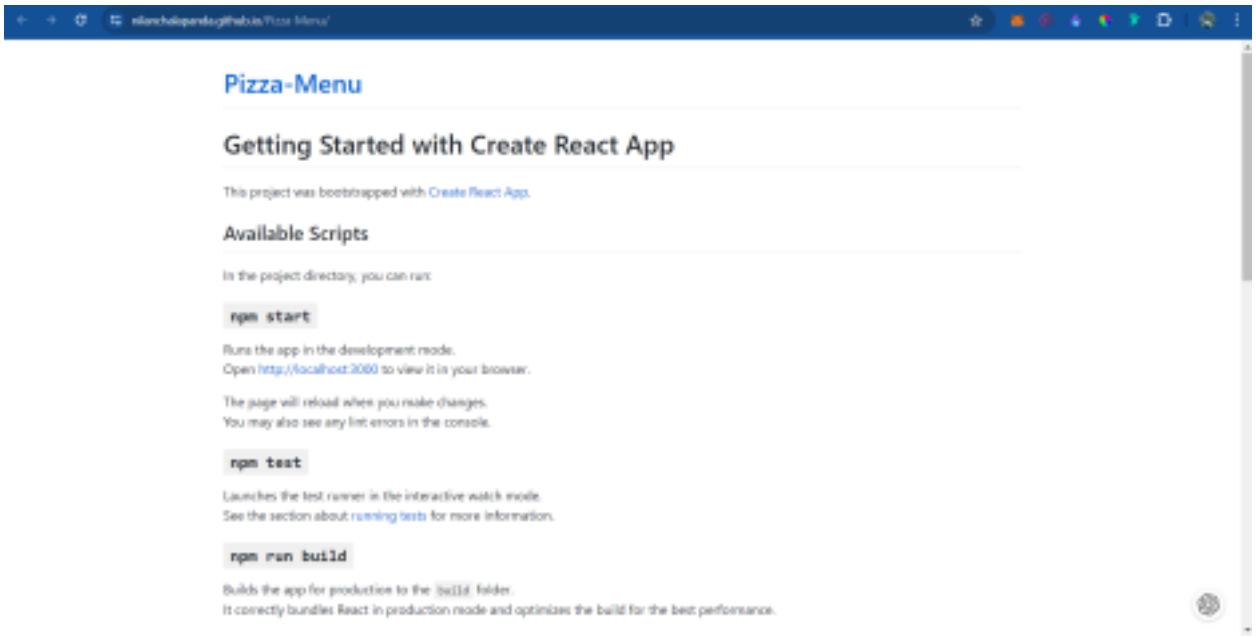
Cons

1. Only 10 GB of data transfer is allowed per month. But this is not really a

big problem, if you use a CDN or AMP.

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

OUTPUT -



MAD & PWA Lab
Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	41
Name	Nilanchala Panda
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	15

Name : Nilanchala Panda **Batch : B**
Div . : 41 **Roll No . : 41**

EXPERIMENT NO 12

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.

Key Features and Audit Metrics -

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

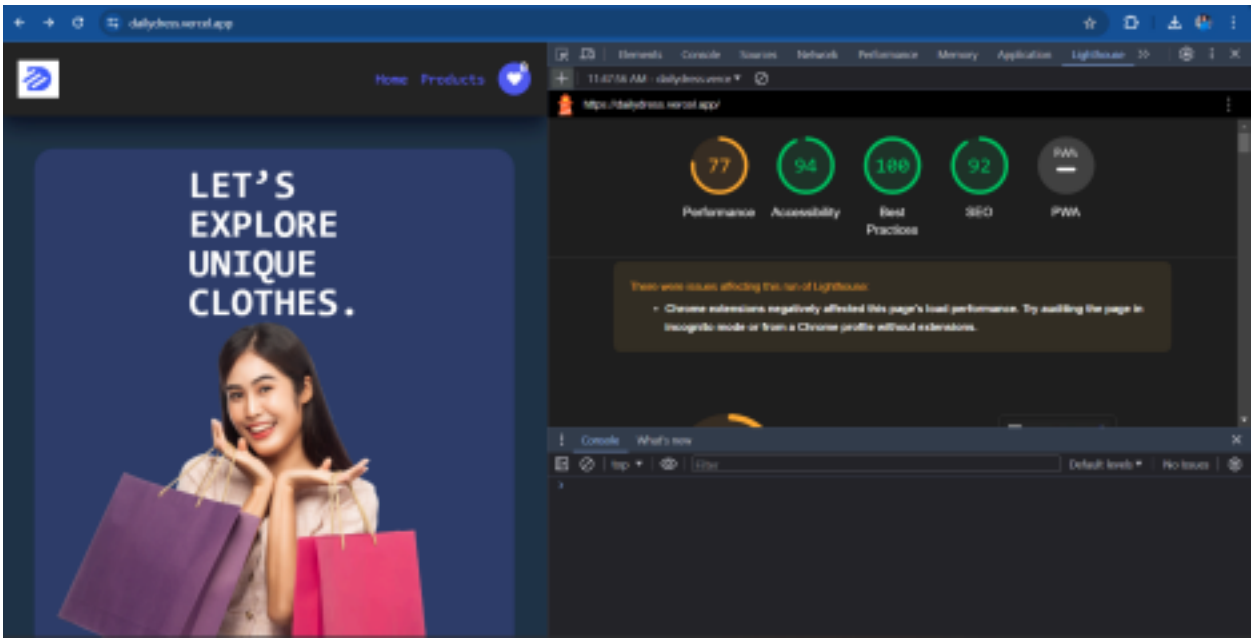
2) PWA Score (Mobile): Thanks to the rise of Service Workers, app manifests, etc., a lot of modern web applications are moving towards the PWA paradigm, where the objective is to make the application behave as close as possible to native mobile applications. Scoring points are based on the Baseline PWA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

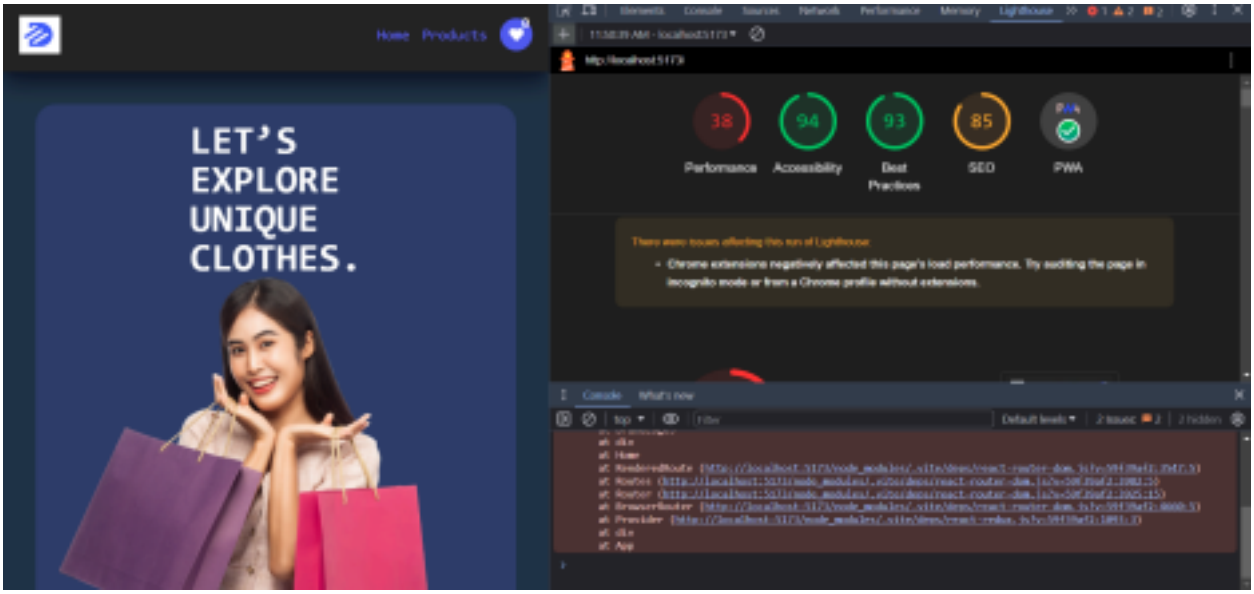
functionality, performance in script-disabled environments, etc.

3) Accessibility: As you might have guessed, this metric is a measure of how accessible your website is, across a plethora of accessibility features that can be implemented in your page (such as the 'aria-' attributes like aria-required, audio captions, button names, etc.).

4) Best Practices: As any developer would know, there are a number of practices that have been deemed 'best' based on empirical data. This metric is an aggregation of many such points, including but not limited to:Use of HTTPSAvoiding the use of deprecated code elements like tags, directives, libraries, etc

OUTPUT -




Home
Products
LET'S
EXPLORE
UNIQUE
CLOTHES.
38
94
93
85
Performance
Accessibility
Best
Practices
SEO
PWA
Console
What's new

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.	41
Name	Nilanchala Panda
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	03

Nilanchala Panda
DISA-41-B

FLUTTER ASSIGNMENT-01

Q1) FlutterOverview → Explain the key features and advantages of using flutter MAD.

Ans:

→ Flutter is a cross-platform development framwork developed by Google, allowing developer to create, mobile, desktop and web app; using a single codebase.

→ Key features:

ⓐ Cross-platform development:

Flutter enables developer to write code once and run it multiple platforms like iOS, Android and web.

ⓑ Hot Reload:

The Hot Reload feature helps develops to save and view the changes in realtime.

ⓒ Rich Set of Widgets:

Flutter provides a variety of widgets, which are the building blocks of flutter app.

ⓓ Support for multiple languages:

Flutter provides support for multiple languages like Dart, C++/C and Java, etc.

The flutter framework differs from traditional approach in several ways. Traditional android apps use Java code and graphics engine for rendering while cross-platform frameworks.

02) Widgets & Composition → Describe the concept of widget tree in Flutter.

Ans: → In flutter, widget tree is a a hierarchial structure where every widget is a node. The root of the tree is typically a material app or widget. Each widget can have zero or more child widgets forming a tree structure.

→ Widgets composition as a technique where you combine simple widgets to build more complex ones.

→ Commonly used widgets are -

(a) Stateless widgets : There are simpliest kind of widget. They describe part of the user interface which can depend on configuration.

(b) Container : This a convenience widget that combines painting, positioning, and sizing widgets.

Scenarios :

(1) Set State() → Basic calculator app where state changes are isolated to single screen.

(2) Provider → A weather app with multiple screens. where the chosen location / temperature.

(3) Riverpod → An E-Commerce app with complex user applications shopping cart management and real time updates, where riverpod advanced features that can be beneficial.

Q4) Firebase Integration in flutter. Explain the process of integrating firebase with flutter application. Discuss the benefits of using firebase as a backend solution. Highlight the firebase service commonly used in flutter development and provide a brief overview.

Ans:

Integrating firebase with flutter applic-ations :

a) Create firebase project.

b) Register your app.

web and native app experiences.

ⓒ Row and Column → These are flex widgets that allow you to arrange their children horiz--ontly and verdically.

Q3) State Management → Disuss the importance of state management in flutter applications.

Ans: State management is a critical aspect of flutter development, playing a pivotal role in everywhere a responsive and efficient user interface. Importance of state management -

ⓐ Performance optimization :
Effective management minimizes unexcessary widget rebuilds, improving performance by avoiding redundant UI updates.

ⓑ Scalabsility :
As an application grows in complexity, props state management becomes crucial for maintaining a manageble and organised database

ⓒ Code maintainability :
Well managed state promotes clean code architecture making it easier to under--stand, maintain and scale applications.

ⓓ Testing : Proper state management state. management facilities unit testing.

particular collection or documents.

3) Realtime Update - When the data changes in firestore the changes are instantly pushed to all connected clients.

4) Offline Support - Firebase firestore provides offline support allowing users to interact with the app even when offline. Data modifications made while offline are stored locally and synced with the server once the device is online.

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.	41
Name	Nilanchala Panda
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:	04

Nilanchala Panda
DISA_41

~~DMBI~~ PWA Assignment 2

Q1)

Soln: A progressive web app (PWA) is a type of web application that uses modern web capabilities to deliver an app-like experience to users PWA's are built using standard web technologies such as HTML, CSS, and Js but are designed to provide a more native-like experience for users, especially on mobile devices. Key characteristics of PWA are -

ⓐ Progressive Enhancement → They are built using progressive ehnacement principles meaning they provide basic experience for all users.

ⓑ Responsive Design → PWA's are designed to feel and behave like native mobile apps.

ⓒ Reliability → PWA's are designed to be reliable, even after user are offline or has slow or unreliable internet connection.

ⓓ Security → PWA's are served over HTTPs to ensure security.

The significance of PWA's in modern web development lies in their ability to bridge the gap between web and native app experiences.

Q2)

Soln: Responsive web design approach that ensure a website adapts its layout and content to seamlessly function and display well across various screen sizes. Importance of PWA is -

- PWA ensures proper layout and navigation across devices mimicking the adaptability of native apps.
- A responsive design can make PWA accessible to wider audience using diverse devices.

	Responsive	Fluid	Adaptive
Layout	Flexible, adapts to any screen size	Continuously adjusts based on rel units	Uses multiple fixed width.
Dev. Effort	Moderate	Less Effor for basic layout	More efforts to develop.
Control	Good Control over all layout.	Less controll on extreme sizes.	High control over device category.
Suitable PWA	Ideal Foundation due to its ver- -satility.	Can be used but may require adju- -sment for best experience	Less common for PWA's due to development overhead.

Sol":

Service workers are essential components of progressive web apps (PWA) that enable features such as offline support, push notifications and background synchronization. The lifecycle of service worker involves several key phases -

① REGISTRATION → Occurs in main Js file of web application.
→ Registration is done using "navigator.serviceWorker.register()" method, which takes the path to file.
→ This method returns a promise that resolves to a service worker.

② INSTALLATION → Once service worker registered, browser downloads and starts installation process of services worker script.
→ During installation, the service worker script is running and "install" event is fired.
→ This is critical for setting up the service worker's initial cache and preparing it to take control of the application.

③ ACTIVATION → After installation phase, Activation phase occurs when service worker is ready to take control of the web application.

Q4)

Soln:

→ Indexed DB is a low-level API for client side server of significant amount of structured data including files/blobs.

→ It is particularly useful in service workers. for several reasons:

(a) Asynchronous and non-blocking -

→ Service workers execute in a seperate thread from the main browser thread, and indexed DB's asynchronous nature allows service worker.

(b) Persistant storage -

→ It provides peristant storage, meaning data stored in indexed DB perists even when the browser is closed or device is restarted.

(c) Structured Data Storage -

→ Indexed DB stores data in a sturctured manner, allowing developers to organise and query data using indexes.