Vivekanand Education Society's Institute of Technology

An Autonomous Institute Affiliated to University of Mumbai

Hashu Advani Memorial Complex, Collector Colony, Chembur East, Mumbai - 400074.

Since 1962

Department of Information Technology

CERTIFICATE

This is to certify that **NIDHI GAWDE** 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.

Lab Objectives:

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

Lab Outcomes:

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

Index

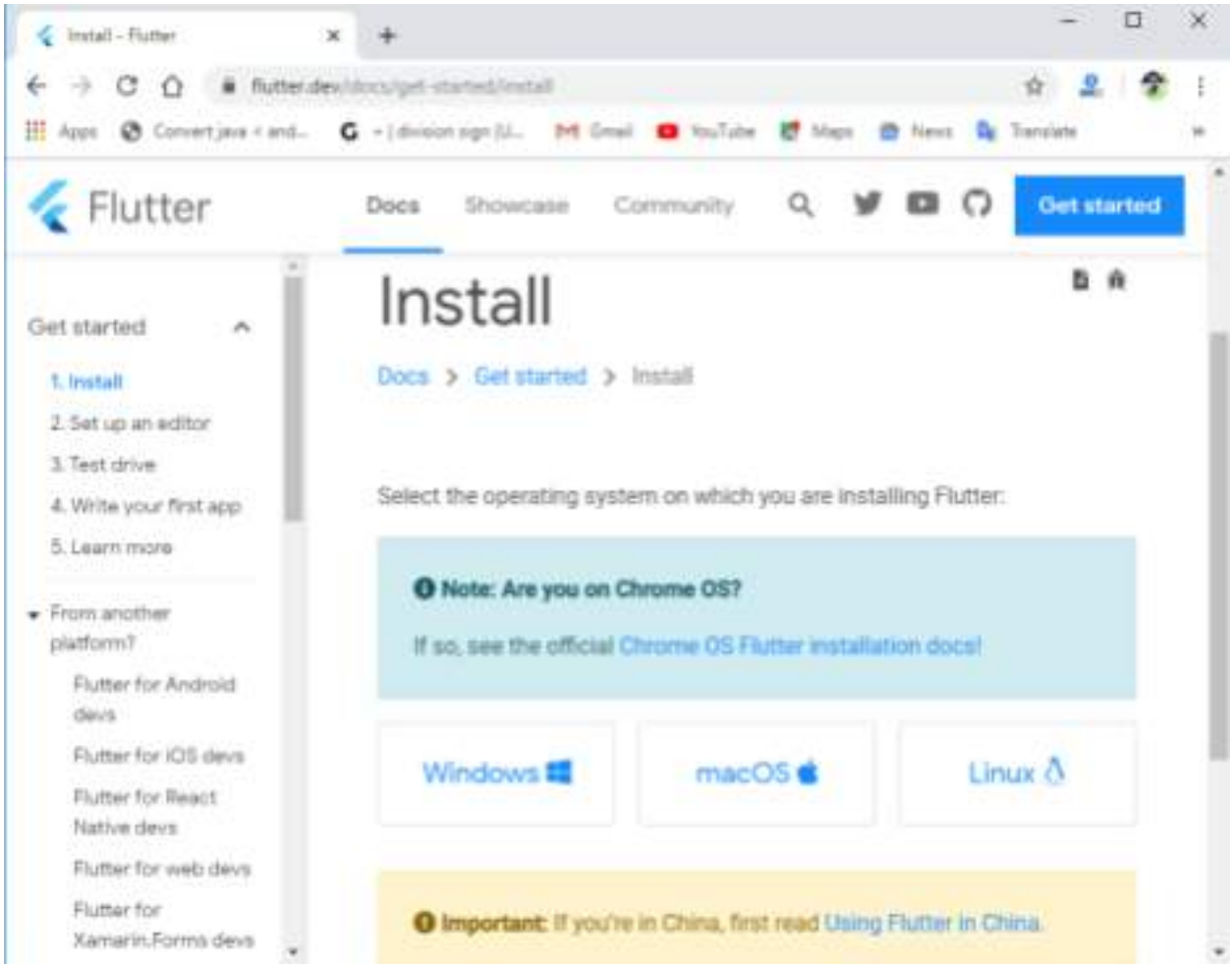
Sr. No	Experiment Title	LO	DOP	DOS	Grade
1.	To install and configure the Flutter Environment	LO1	16/1	23/1	15
2.	To design Flutter UI by including common widgets.	LO2	23/1	30/1	15
3.	To include icons, images, fonts in Flutter app	LO2	30/1	6/2	15
4.	To create an interactive Form using form widget	LO2	6/2	13/2	15
5.	To apply navigation, routing and gestures in Flutter App	LO2	13/2	20/2	15
6.	To Connect Flutter UI with fireBase database	LO3	20/2	5/3	15
7.	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.	LO4	5/3	12/3	15
8.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA	LO5	12/3	19/3	15
9.	To implement Service worker events like fetch, sync and push for E-commerce PWA	LO5	19/3	26/3	15
10.	To study and implement deployment of Ecommerce PWA to GitHub Pages.	LO5	26/3	2/4	15
11.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.	LO6	5/3	12/3	15
12.	Assignment-1	LO1,LO2,LO3	2/2	5/2	
13.	Assignment-2	LO4,LO5,LO6	19/3	21/3	

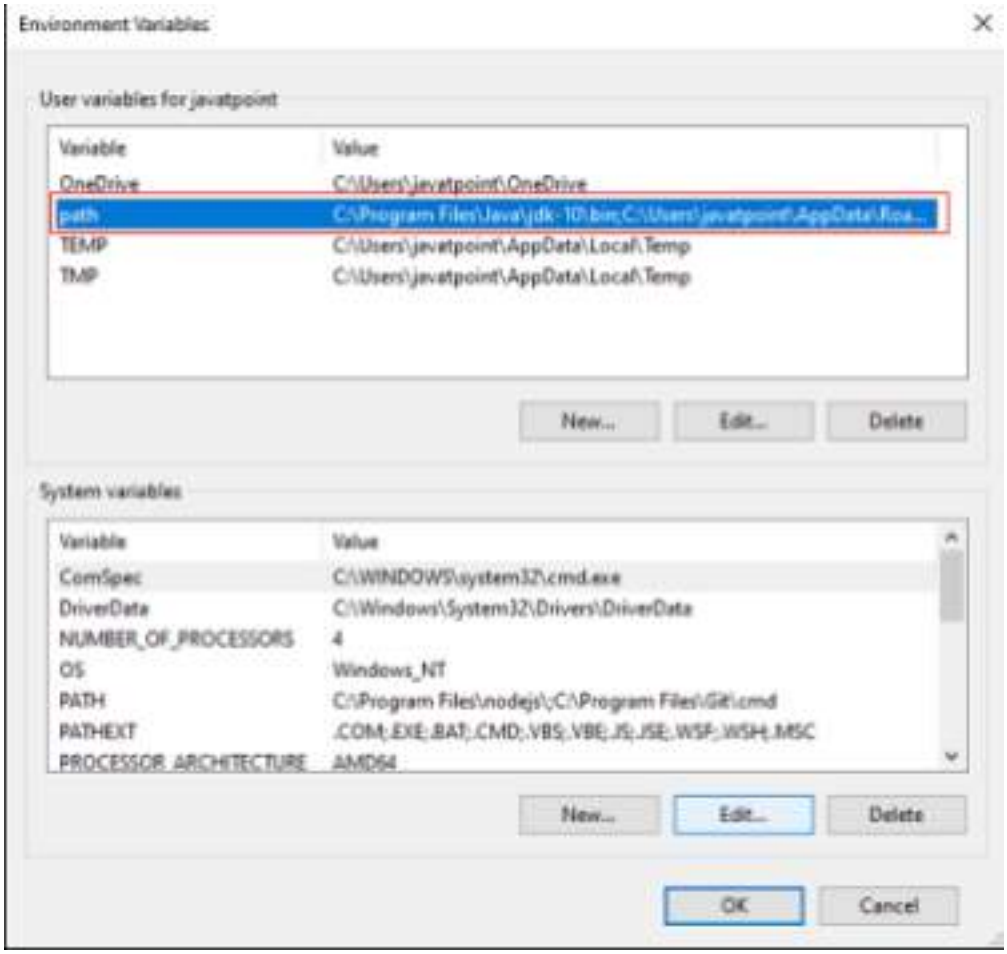
MAD & PWA Lab
Journal

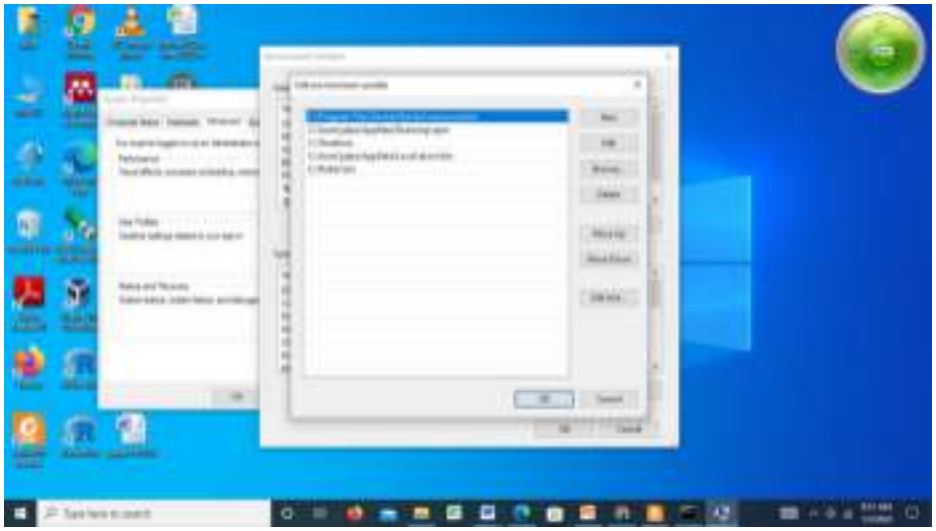
Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	20
Name	**NIDHI GAWDE**
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	15

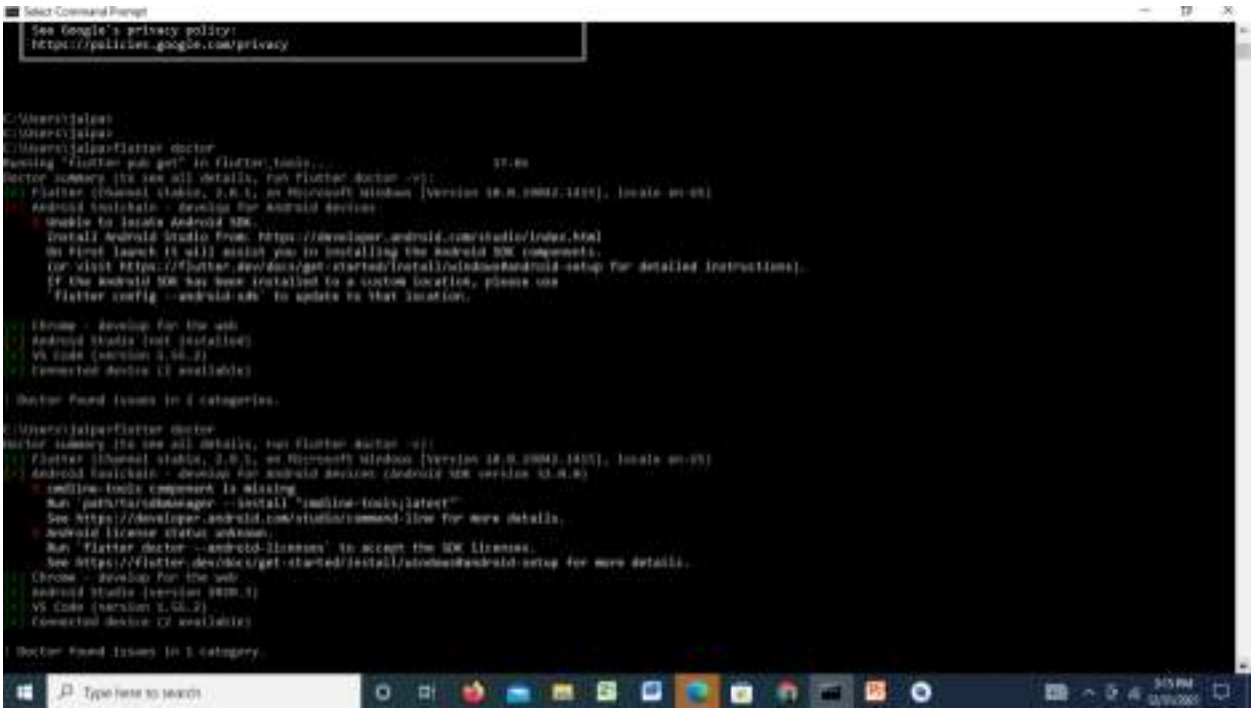
Experiment 01

AIM: Installation and Configuration of Flutter Environment.

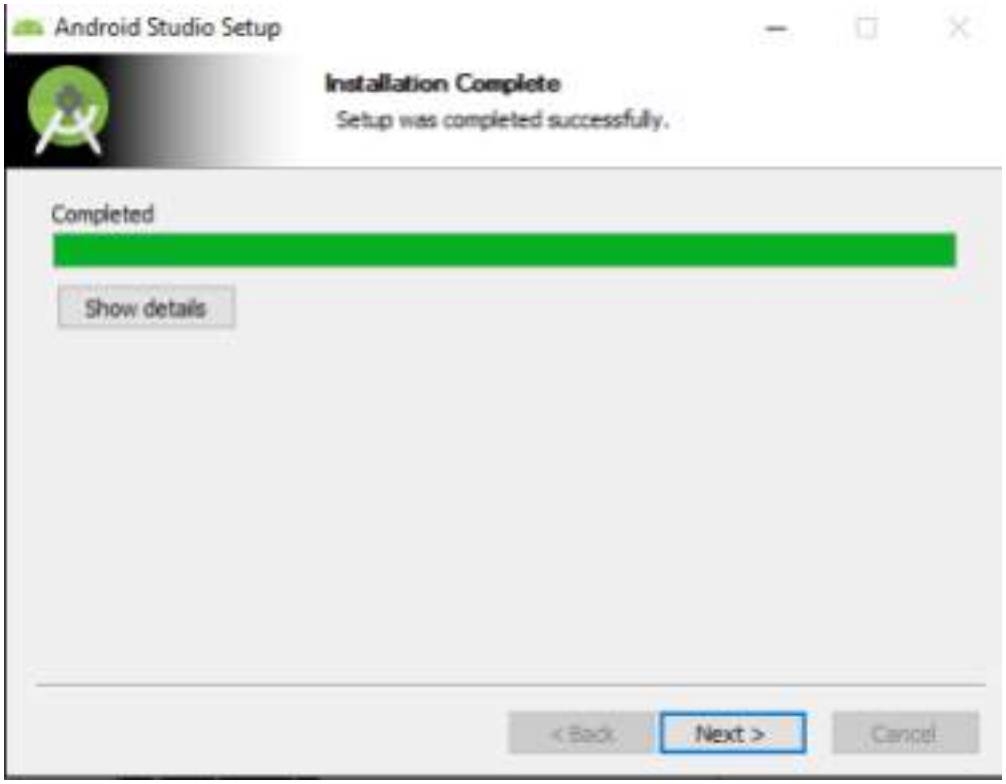


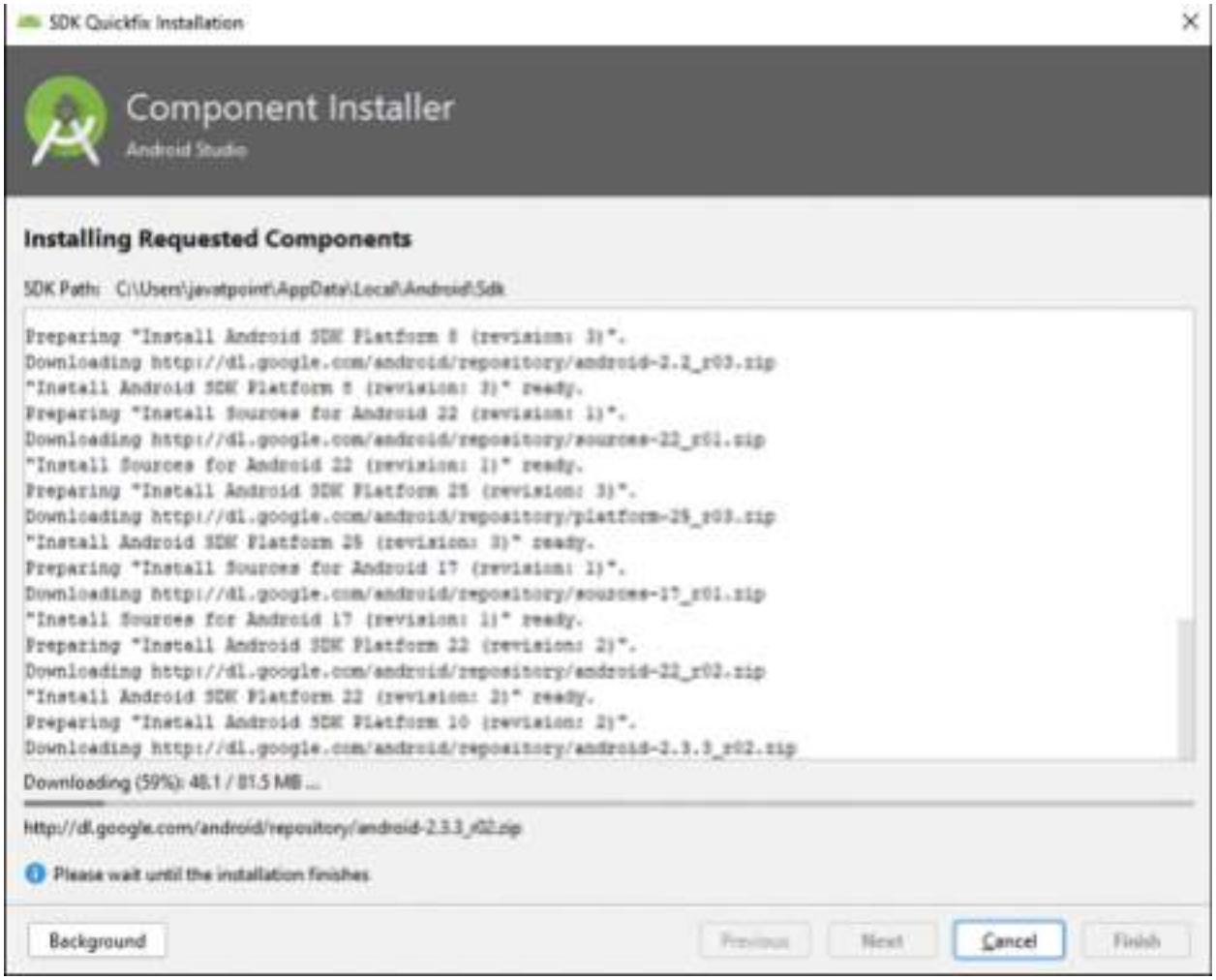

Environment Variables
User variables for javatpoint
Variable
Value
OneDrive
C:\Users\javatpoint\OneDrive
path
C:\Program Files\Java\jdk-10\bin;C:\Users\javatpoint\AppData\Roa...
TEMP
C:\Users\javatpoint\AppData\Local\Temp
TMP
C:\Users\javatpoint\AppData\Local\Temp
New...
Edit...
Delete
System variables
Variable
Value
ComSpec
C:\WINDOWS\system32\cmd.exe
DriverData
C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS
4
OS
Windows_NT
PATH
C:\Program Files\nodejs\;C:\Program Files\Git\cmd
PATHEXT
.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE
AMD64
New...
Edit...
Delete
OK
Cancel



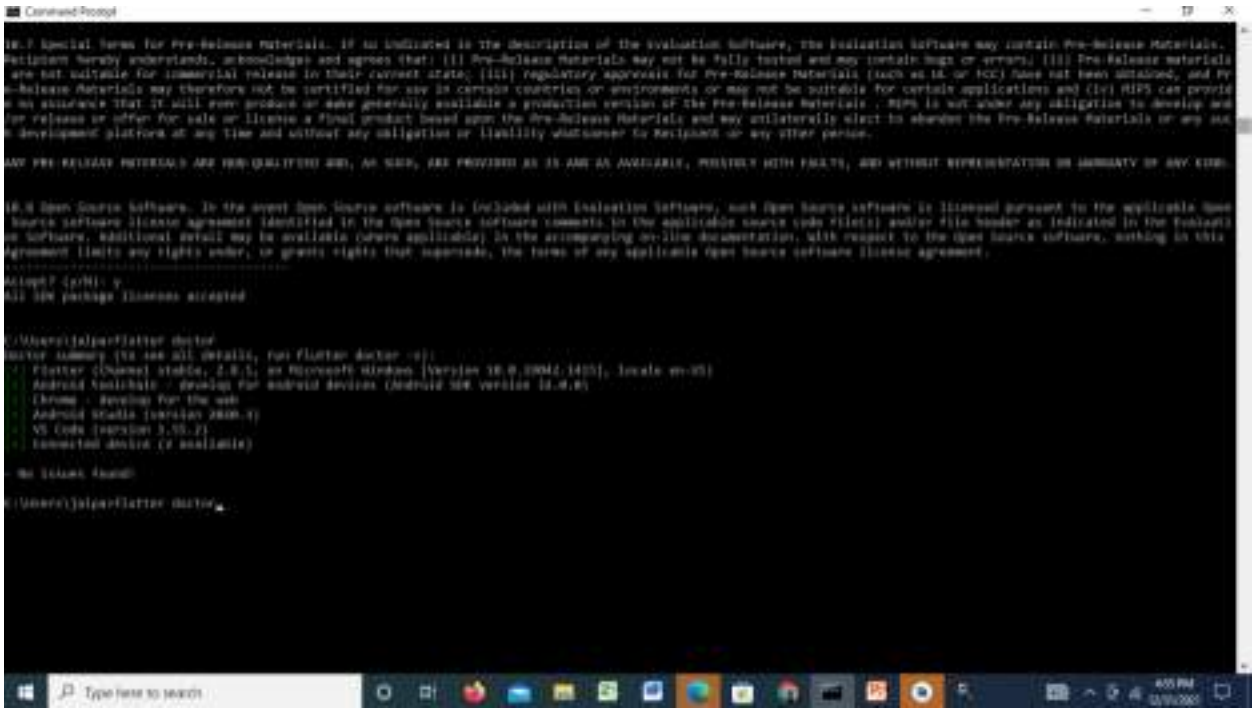

Select Command Prompt
See Google's privacy policy:
https://policies.google.com/privacy
Type here to search


Android Studio Setup
Welcome to Android Studio Setup
Setup will guide you through the installation of Android Studio.
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.
Click Next to continue.
Android Studio
< Back
Next >
Cancel

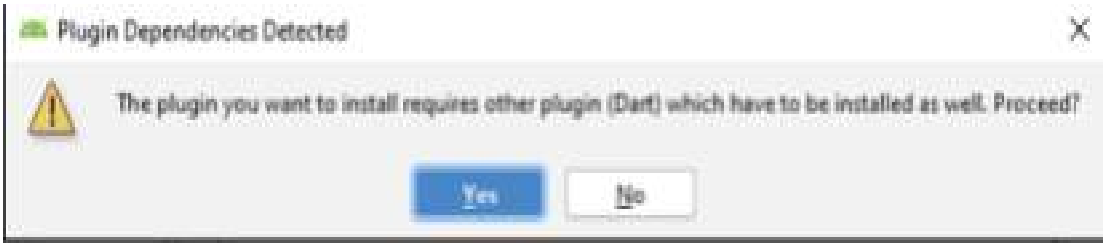

Android Studio Setup
Installation Complete
Setup was completed successfully.
Completed
Show details
< Back
Next >
Cancel

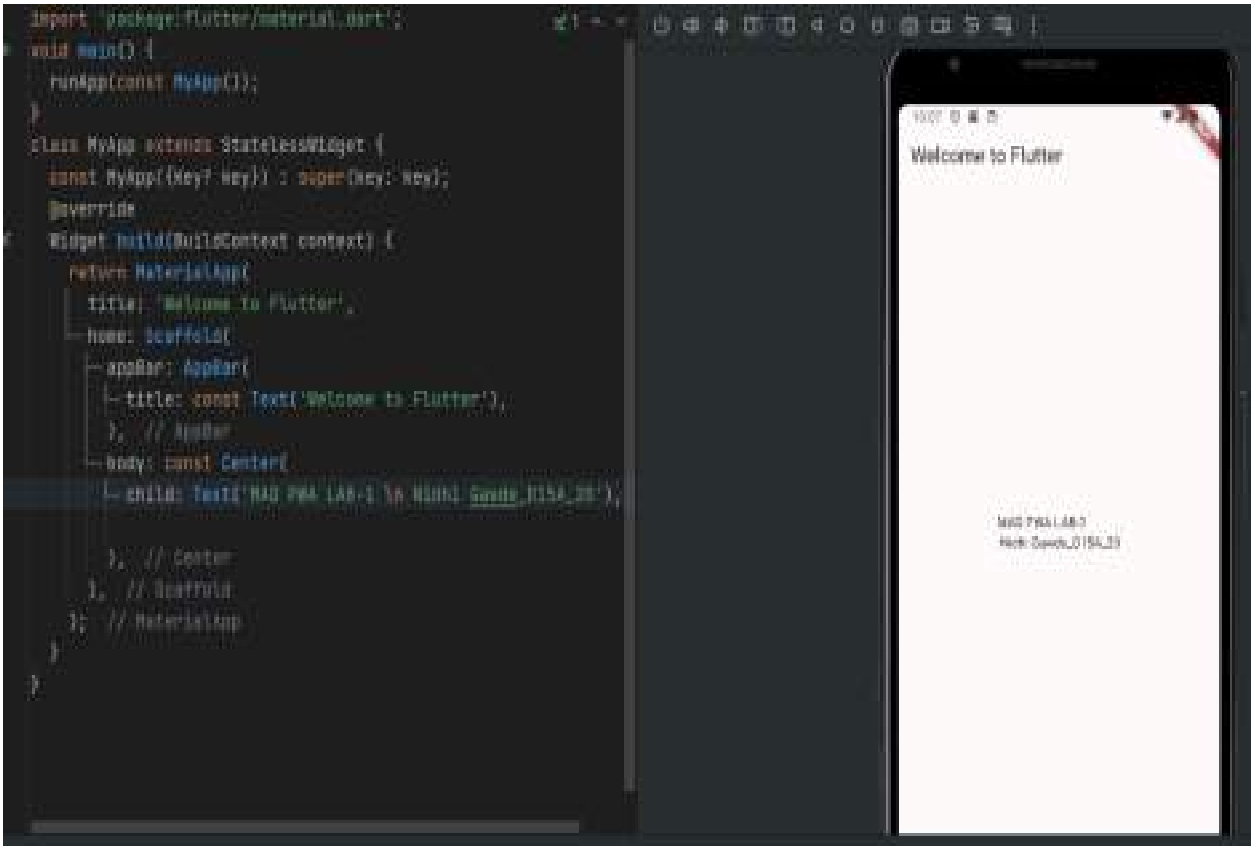

SDK Quickfix Installation
Component Installer
Android Studio
Installing Requested Components
SDK Path: C:\Users\javatpoint\AppData\Local\Android\Sdk
Preparing "Install Android SDK Platform 8 (revision: 3)".
Downloading http://dl.google.com/android/repository/android-2.2_r03.zip
"Install Android SDK Platform 8 (revision: 3)" ready.
Preparing "Install Sources for Android 22 (revision: 1)".
Downloading http://dl.google.com/android/repository/sources-22_r01.zip
"Install Sources for Android 22 (revision: 1)" ready.
Preparing "Install Android SDK Platform 25 (revision: 3)".
Downloading http://dl.google.com/android/repository/platform-25_r03.zip
"Install Android SDK Platform 25 (revision: 3)" ready.
Preparing "Install Sources for Android 17 (revision: 1)".
Downloading http://dl.google.com/android/repository/sources-17_r01.zip
"Install Sources for Android 17 (revision: 1)" ready.
Preparing "Install Android SDK Platform 22 (revision: 2)".
Downloading http://dl.google.com/android/repository/android-22_r02.zip
"Install Android SDK Platform 22 (revision: 2)" ready.
Preparing "Install Android SDK Platform 10 (revision: 2)".
Downloading http://dl.google.com/android/repository/android-2.3.3_r02.zip
Downloading (59%): 48.1 / 81.5 MB ...
http://dl.google.com/android/repository/android-2.3.3_r02.zip
Please wait until the installation finishes
Background
Previous
Next
Cancel
Finish

Project Title: Flutter - LinkedIn Clone / PWA-Netflix Website **Roll No.: 20**




Your Virtual Devices


Plugin Dependencies Detected
The plugin you want to install requires other plugin (Dart) which have to be installed as well. Proceed?
Yes
No



Conclusion :

The flutter environment was successfully installed and set up .

MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	20
Name	**NIDHI GAWDE**
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 02

Aim : To design Flutter UI by including common widgets.

Theory :

- In Flutter, widgets are the basic building blocks used to create user interfaces.
- Widgets in Flutter are immutable, meaning they cannot be changed once created. Widgets can be either **stateless** or **stateful.** Stateless widgets are static and do not change over time, while stateful widgets can change their state and appearance based on user interaction or other factors.
- Widgets are organized in a hierarchical structure, where each widget can have child widgets. This hierarchical structure defines the layout and appearance of the UI.
- Widgets are organized in a hierarchical structure, where each widget can have child widgets. This hierarchical structure defines the layout and appearance of the UI.

Some common widgets are listed below:

- **Container:**
  A versatile widget used to contain and arrange other widgets. It can be styled with various properties like padding, margin, color, border, etc.
- **Text:**
  Displays a string of text with customizable styles such as font size, color, alignment, etc. It's commonly used for displaying labels, titles, or paragraphs of text.
- **Image:**
  Displays an image from various sources such as assets, network, or memory. It supports different image formats and provides options for resizing and fitting.
- **Row:**
  Arranges its children widgets horizontally in a single row. It's useful for creating horizontally aligned layouts.
- **Column:**
  Arranges its children widgets vertically in a single column. It's commonly used for creating vertically stacked layouts.
- **ListView:**
  Displays a scrollable list of children widgets. It can be oriented vertically or horizontally and supports both fixed and dynamic lists.
- **Stack:**
  Overlays its children widgets on top of each other. It's useful for creating complex layouts where widgets can overlap or be positioned relative to each other.
- **AppBar:**
  A material design app bar that typically appears at the top of the screen. It's commonly used to display the title, actions, and navigation controls for the app.

- **TextField:**
  Allows users to input text. It supports various features like hint text, validation, input formatting, and keyboard customization.
- **Button:**
  Represents a clickable button widget that users can interact with. It can be styled and customized based on different states such as enabled, disabled, pressed, etc.
- **Icon:**
  Displays a graphical icon from the Material Icons library or custom icon sets. It's commonly used for adding visual cues and navigation elements.
- **AlertDialog:**
  Displays a dialog window with a title, content, and buttons. It's often used to present important information or prompt user actions.
- **Scaffold:**
  Implements the basic material design layout structure for a screen, including app bars, drawers, and bottom sheets. It serves as the root of the widget hierarchy for a screen.
- **Padding:**
  Adds padding space around its child widget. It's useful for controlling the spacing between widgets and the edges of the screen.
- **GestureDetector:**
  Detects gestures such as taps, drags, and swipes on its child widget. It's used to make any widget interactive and responsive to user input.

Code:

```
import 'package:flutter/material.dart';
import
'package:font_awesome_flutter/font_aweso
me_flutter.dart';
import
'package:linkedin_mobile_ui/theme/styles.d
art';

class CreatePage extends StatefulWidget {
  final VoidCallback? onCloneClickListener;
  const CreatePage({Key? key, required
this.onCloneClickListener})
      : super(key: key);

  @override
  State<CreatePage> createState() =>
_CreatePageState();
}

class _CreatePageState extends
State<CreatePage> {

  final TextEditingController
_postBodyController =
TextEditingController();

  bool _openTwoBottomModalSheetsOnce =
false;
```

```
  final FocusScopeNode
_subPostBottomModalSheetFocusNode =
FocusScopeNode();

 final FocusScopeNode
_superPostBottomModalSheetFocusNode =
FocusScopeNode();

  @override
  void dispose() {
    _postBodyController.dispose();
    super.dispose();
  }
  @override
  Widget build(BuildContext context) {
    if(_openTwoBottomModalSheetsOnce ==
false) {

WidgetsBinding.instance.addPostFrameCall
back((timeStamp) {
        _createSuperPostBottomModalSheet();
        _createSubPostBottomModalSheet();
        print("value before =
$_openTwoBottomModalSheetsOnce");
        setState(() {
          _openTwoBottomModalSheetsOnce =
true;
        });
        print("value after =
$_openTwoBottomModalSheetsOnce");
      });
    }

    return const Scaffold();
  }

  _createSuperPostBottomModalSheet() {
    showModalBottomSheet(
      isScrollControlled: true,
      enableDrag: false,
      isDismissible: false,
      context: context,
      builder: (context) {
        return StatefulBuilder(
          builder: (context, void Function(void
Function()) setState) {
            return FocusScope(
              node:
_superPostBottomModalSheetFocusNode,
              child: Container(
                child: Column(
                  crossAxisAlignment:
CrossAxisAlignment.start,
                  children: [
                    Container(
                      padding: const
EdgeInsets.symmetric(horizontal: 10),
                      width: double.infinity,
                      height: 110,
                      decoration:
                      const BoxDecoration(color:
linkedInWhiteFFFFFF, boxShadow: [
                        BoxShadow(
                          offset: Offset(0, 2),
                          color:
linkedInLightGreyCACCCE,
                          blurRadius: 5,
                          spreadRadius: 0.1),
                      ]),
                      child: Padding(
                        padding: const
EdgeInsets.only(bottom: 15.0),
                        child: Row(
                          crossAxisAlignment:
CrossAxisAlignment.end,
```

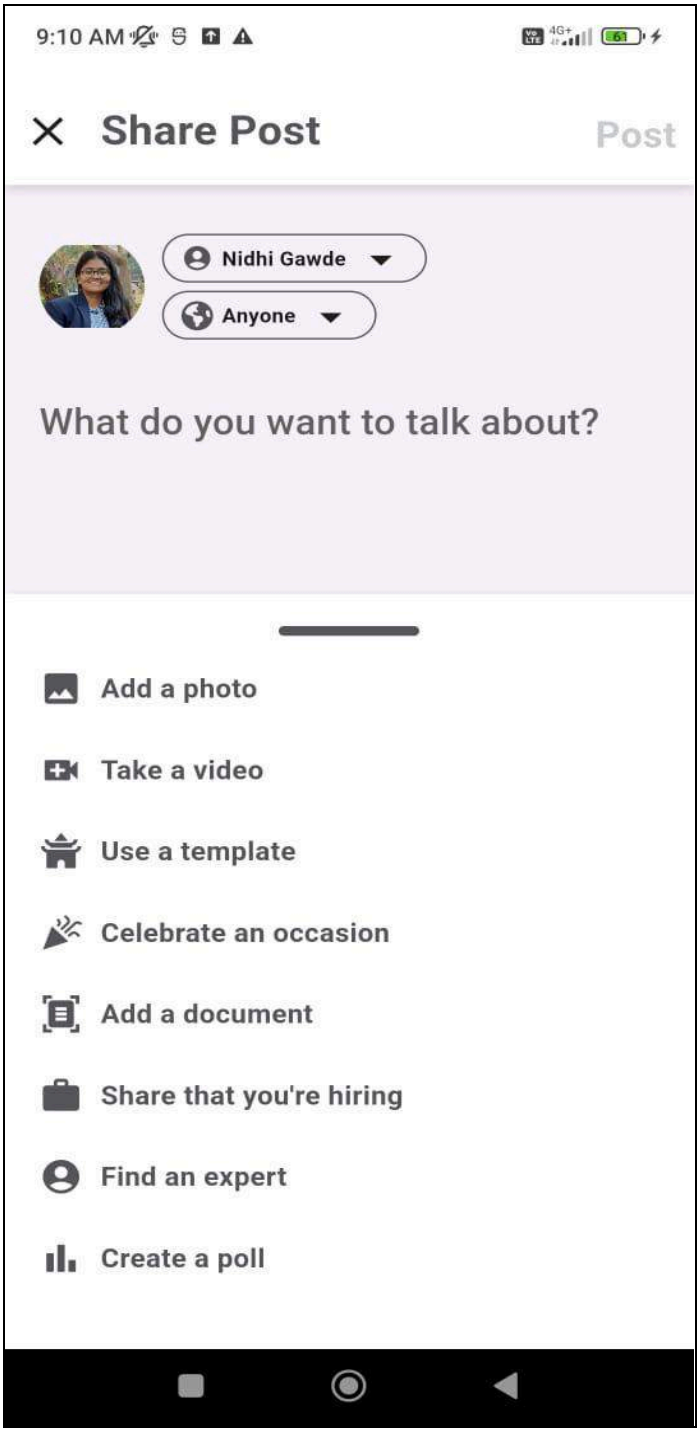
```
                    mainAxisAlignment:
MainAxisAlignment.spaceBetween,
                    children: [
                      Row(
                        children: [
                          GestureDetector(
                            onTap:
widget.onCloneClickListener,
                            child: const Icon(

Icons.close_outlined,
                              size: 30,
                            )),
                          const SizedBox(
                            width: 15,
                          ),
                          const Text(
                            "Share Post",
                            style: TextStyle(
                              fontSize: 25,
                              fontWeight:
FontWeight.bold,
                              color:
linkedInMediumGrey86888A),
                          )
                        ],
                      ),
                      Text(
                        "Post",
                        style: TextStyle(
                          fontSize: 22,
                          fontWeight:
FontWeight.bold,
                          color:
_postBodyController.text.isEmpty?
linkedInLightGreyCACCCE :
linkedInBlue0077B5),
                      )
                    ],
                  ),
                  ),
                ),
                const SizedBox(
                  height: 30,
                ),
                Expanded(
                  child: Container(
                    margin: const
EdgeInsets.symmetric(horizontal: 20),
                    child: Column(
                      children: [
                        Row(
                          children: [
                            Container(
                              width: 60,
                              height: 60,
                              child: ClipRRect(
                                borderRadius:
BorderRadius.circular(30),
                                child:
Image.asset("assets/profile_1.jpeg"),
                              ),
                            ),
                            const SizedBox(
                              width: 10,
                            ),
                            Column(
                              mainAxisAlignment:
MainAxisAlignment.end,
                              crossAxisAlignment:
CrossAxisAlignment.start,
                              children: [
                                _switchWidget(
                                  title: "Nidhi
Gawde",
                                  prefixIcon:
Icons.account_circle_rounded,
                                  suffixIcon:
Icons.arrow_drop_down_outlined),
                                const SizedBox(
```

```
                    height: 5,
                  ),
                  _switchWidget(
                    title: "Anyone",
                    prefixIcon:
FontAwesomeIcons.earth,
                    suffixIcon:
Icons.arrow_drop_down_outlined),
                  ],
                ),
              ],
            ),
            const SizedBox(
              height: 20,
            ),
            TextFormField(
              controller:
_postBodyController,
              onTap: () {

_postBodyController.addListener(() {

if(_postBodyController.text.length == 1) {
                  setState(() {});
                  print("call first IF
setState");
                } else if
(_postBodyController.text.length < 1) {
                  setState(() {});
                  print("2nd ELSE IF
setState");
                }

                print("onTap called");
              });
            },
            style: const
TextStyle(fontSize: 22),
            maxLines: 15,
            decoration: const
InputDecoration(
              hintText: "What do
you want to talk about?",
              border:
InputBorder.none),
            ),
          ],
        ),
      ),
    ),
    Container(
      margin: const
EdgeInsets.symmetric(horizontal: 30),
      child: Row(
        mainAxisAlignment:
MainAxisAlignment.spaceBetween,
        children: [
          Row(
            children: [
              const Icon(
                Icons.camera_alt,
                color:
linkedInMediumGrey86888A,
              ),
              const SizedBox(
                width: 15,
              ),
              const Icon(
                Icons.video_call,
                color:
linkedInMediumGrey86888A,
              ),
              const SizedBox(
                width: 15,
              ),
              const Icon(
                Icons.image,
                color:
linkedInMediumGrey86888A,
```

```
                    ),
                    const SizedBox(
                      width: 25,
                    ),
                    GestureDetector(
                      onTap: () {

_createSubPostBottomModalSheet();
                      },
                      child: const Icon(
                        Icons.more_horiz,
                        color:
linkedInMediumGrey86888A,
                      )),
                  ],
                ),
                const Row(
                  children: [
                    Icon(

Icons.message_outlined,
                        color:
linkedInMediumGrey86888A,
                    ),
                    SizedBox(
                      width: 10,
                    ),
                    Text(
                      "Anyone",
                      style: TextStyle(
                        fontWeight:
FontWeight.bold,
                        color:
linkedInMediumGrey86888A),
                    )
                  ],
                )
              ],
            ),
          ),
                  const SizedBox(
                    height: 30,
                  ),
                ],
              ),
            ),
          );
        }
      );
    },
  ).then((value) {

_superPostBottomModalSheetFocusNode.u
nfocus();
  });
}

_switchWidget({String? title, IconData?
prefixIcon, IconData? suffixIcon}) {
  return Container(
    height: 30,
    padding: const
EdgeInsets.symmetric(horizontal: 10),
    decoration: BoxDecoration(
      borderRadius:
BorderRadius.circular(20),
      border: Border.all(width: 1, color:
linkedInMediumGrey86888A)),
    child: Row(
      children: [
        Icon(
          prefixIcon,
          color: linkedInMediumGrey86888A,
          size: 18,
        ),
        const SizedBox(
          width: 5,
        ),
        Text(
          "$title",
```

```
          style: const TextStyle(fontWeight:
FontWeight.bold, fontSize: 12),
        ),
        const SizedBox(
          width: 5,
        ),
        Icon(
          suffixIcon,
          size: 30,
        ),
      ],
    ),
  );
}

_createSubPostBottomModalSheet() {
  showModalBottomSheet(
    shape: RoundedRectangleBorder(
      borderRadius:
BorderRadius.circular(20),
    ),
    barrierColor: Colors.transparent,
    context: context,
    builder: (context) {
      return FocusScope(
        node:
_subPostBottomModalSheetFocusNode,
        child: Container(
          decoration: BoxDecoration(color:
linkedInWhiteFFFFFF, boxShadow: [
            BoxShadow(
              offset: const Offset(5, 0),
              blurRadius: 1,
              color:
linkedInLightGreyCACCCE.withOpacity(.6
),
              spreadRadius: 0.5)
          ]),
          child: Padding(
            padding: const
EdgeInsets.symmetric(horizontal: 20.0,
vertical: 20),
            child: SingleChildScrollView(
              child: Column(
                crossAxisAlignment:
CrossAxisAlignment.start,
                children: [
                  Center(
                    child: Container(
                      width: 80,
                      height: 6,
                      decoration: BoxDecoration(
                        borderRadius:
BorderRadius.circular(10),
                        color:
linkedInMediumGrey86888A),
                    ),
                  ),
                  const SizedBox(height: 20,),

_createSubPostNavigationItem(title: "Add a
photo", iconData: Icons.image),
                  const SizedBox(height: 25,),

_createSubPostNavigationItem(title: "Take a
video", iconData: Icons.video_call),
                  const SizedBox(height: 25,),

_createSubPostNavigationItem(title: "Use a
template", iconData:
Icons.temple_buddhist),
                  const SizedBox(height: 25,),

_createSubPostNavigationItem(title:
"Celebrate an occasion", iconData:
Icons.celebration),
                  const SizedBox(height: 25,),

_createSubPostNavigationItem(title: "Add a
```

```
document", iconData:
Icons.document_scanner),
                const SizedBox(height: 25,),

_createSubPostNavigationItem(title: "Share
that you're hiring", iconData: Icons.work),
                const SizedBox(height: 25,),

_createSubPostNavigationItem(title: "Find
an expert", iconData:
Icons.account_circle_rounded),
                const SizedBox(height: 25,),

_createSubPostNavigationItem(title: "Create
a poll", iconData: Icons.bar_chart),
                const SizedBox(height: 25,),

_createSubPostNavigationItem(title: "Create
an event", iconData: Icons.event),
                const SizedBox(height: 25,),
              ],
            ),
          ),
        ),
          ),
        );
      },
    ).then((value) {

_subPostBottomModalSheetFocusNode.unf
ocus();
    });
  }

  _createSubPostNavigationItem({IconData?
iconData, String? title}) {
    return Row(
      children: [
        Icon(iconData, size: 25,color:
linkedInMediumGrey86888A,),
        const SizedBox(width: 10,),
        Text("$title", style: const
TextStyle(fontSize: 16, color:
linkedInMediumGrey86888A, fontWeight:
FontWeight.bold),)
      ],
    );
  }
}
```



Conclusion: Successfully design Flutter UI by including common widgets such as scaffold, container, stateful widgets , etc

MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	20
Name	**NIDHI GAWDE**
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 03

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

Theory :

In Flutter, icons, images, and fonts are essential elements for creating visually appealing and functional user interfaces. Here's a brief overview of each:

- Icons:

Icons are graphical symbols used to represent actions, features, or categories within an application.

Flutter provides a wide range of built-in icons through the Icons class, which includes commonly used icons like home, settings, search, etc.

Icons can be displayed using the Icon widget, where you specify the icon data (e.g., Icons.home) and customize its appearance with properties like color, size, and opacity.

Additionally, Flutter allows you to use custom icons by importing image files or vector graphics and converting them into Flutter-compatible assets.

- Images:

Images are visual content used to enhance the user interface or provide context within an application.

Flutter supports various image formats, including PNG, JPEG, GIF, and WebP.

Images can be displayed using the Image widget, which accepts different sources such as asset images, network images (from URLs), memory images, or file images.

To use images as assets in Flutter, you need to declare them in the pubspec.yaml file and specify their location within the project directory.

Flutter provides advanced features for working with images, such as caching, resizing, cropping, and applying filters, through libraries like cached_network_image and flutter_image_editor.

- Fonts:

Fonts are sets of typefaces or styles used to display text in various styles, weights, and sizes.

Flutter allows you to use custom fonts in your application by including font files (e.g., TTF or OTF files) and specifying them as assets in the pubspec.yaml file.

Custom fonts can be applied to text widgets using the TextStyle class, where you specify the font family, font weight, font style, and other properties.

Flutter also provides built-in support for Google Fonts, allowing you to use a wide variety of open-source fonts hosted by Google without downloading or importing them manually.

Additionally, Flutter supports text localization and internationalization, enabling developers to adapt the font and text layout based on the user's language and locale preferences.

Code:

a) home_page.dart

```
import 'package:flutter/material.dart';
import
'package:font_awesome_flutter/font_aweso
me_flutter.dart';
import
'package:linkedin_mobile_ui/data/post_entit
y.dart';
import
'package:linkedin_mobile_ui/pages/main/ho
me/widgets/single_post_card_widget.dart';
import
'package:linkedin_mobile_ui/theme/styles.d
art';

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

  @override
  State<HomePage> createState() =>
_HomePageState();
}

class _HomePageState extends
State<HomePage> {

  ScrollController _controller =
ScrollController(); //scroll behaviour

  bool _isShow = true;

  List<PostEntity> postData =
PostEntity.postListData;

  @override
  void initState() {
    _controller.addListener(() {
      if(_controller.position.pixels > 3) {
        setState(() {
          _isShow = false;
        });
      } else {
        setState(() {
          _isShow = true;
        });
      }
    });
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Column(
        children: [
          const SizedBox(height: 5,),
          _isShow ?Container(
            width: double.infinity,
            height: 8,
            color: linkedInLightGreyCACCCE,
          ) : Container(),

          Expanded(
            child: ListView.builder(
              controller: _controller,
              itemCount: postData.length,
              itemBuilder: (context, index) {
                final post = postData[index];
                return
SinglePostCardWidget(post: post);
              },
            ),
          ),
        ],
      )
    );
  }
```

```
}
```

a) Main_page.dart

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import
'package:font_awesome_flutter/font_awesome_flutter.dart';
import
'package:linkedin_mobile_ui/pages/main/create/create_page.dart';
import
'package:linkedin_mobile_ui/pages/main/home/home_page.dart';
import
'package:linkedin_mobile_ui/pages/main/jobs/jobs_page.dart';
import
'package:linkedin_mobile_ui/pages/main/main_page/widgets/drawer_widget.dart';
import
'package:linkedin_mobile_ui/pages/main/network/network_page.dart';
import
'package:linkedin_mobile_ui/pages/main/notifications/notifications_page.dart';
import
'package:linkedin_mobile_ui/theme/styles.dart';
import 'widgets/app_bar_widget.dart';

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

  @override
  State<MainPage> createState() =>
_MainPageState();
}

class _MainPageState extends
State<MainPage> {

  final GlobalKey<ScaffoldState>
_scaffoldState =
GlobalKey<ScaffoldState>();

  int _currentPageIndex = 0;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      drawer: const DrawerWidget(),
      key: _scaffoldState,
      appBar: _currentPageIndex == 4?
appBarWidget(
          context,
          title: "Search Jobs",
          isJobsTab: true,
          onLeadingTapClickListener: () {
            setState(() {

_scaffoldState.currentState!.openDrawer();
            });
          }
      ) :appBarWidget(
          context,
          title: "Search",
          isJobsTab: false,
          onLeadingTapClickListener: () {
            setState(() {

_scaffoldState.currentState!.openDrawer();
            });
          }
```

```
    ),
    bottomNavigationBar:
BottomNavigationBar(
      currentIndex: _currentPageIndex,
      onTap: (index) {
        setState(() {
          _currentPageIndex = index;
        });
      },
      selectedItemColor:
linkedInBlack000000,
      selectedLabelStyle: const
TextStyle(color: linkedInBlack000000),
      unselectedItemColor:
linkedInMediumGrey86888A,
      unselectedLabelStyle: const
TextStyle(color:
linkedInMediumGrey86888A),
      showUnselectedLabels: true,
      items: const [
        BottomNavigationBarItem(
          icon:
Icon(CupertinoIcons.house_fill),
          label: "Home",
        ),
        BottomNavigationBarItem(
          icon:
Icon(FontAwesomeIcons.userGroup),
          label: "Network",
        ),
        BottomNavigationBarItem(
          icon: Icon(
            Icons.add_box,
            size: 30,
          ),
          label: "Post",
        ),
        BottomNavigationBarItem(
          icon: Icon(
            Icons.notifications,
            size: 30,
          ),
          label: "Notifications",
        ),
        BottomNavigationBarItem(
          icon:
Icon(FontAwesomeIcons.briefcase),
          label: "Jobs",
        ),
      ],
    ),
    body: _switchPages(_currentPageIndex)
  );
}

_switchPages(int index) {
  switch (index) {
    case 0:
      {
        return const HomePage();
      }
    case 1:
      {
        return const NetworkPage();
      }
    case 2:
      {
        return
CreatePage(onCloneClickListener: () {
          Navigator.pop(context);
          setState(() {
            _currentPageIndex = 0;
          });
        },);
      }
    case 3:
      {
        return const NotificationsPage();
      }
    case 4:
```

```
{
  return const JobsPage();
}
```

```
    }
  }
}
```

Conclusion :
Successfully implemented the icons, images and fonts in Flutter App.

MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	20
Name	**NIDHI GAWDE**
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 04

Aim : To create an interactive Form using form widget

Theory :

In Flutter, the Form widget is used to create interactive forms, allowing users to input data and submit it for processing. The Form widget is part of the Flutter framework's flutter/widgets.dart library and is commonly used in combination with other form-related widgets like TextFormField, DropdownButton, and ElevatedButton. Here's an overview of how the Form widget works and its key features:

- Form Widget:

The Form widget is a container that holds form fields and manages their state.
It provides methods for validation, submission, and resetting of form fields.
The Form widget maintains the state of each form field within it and can be used to retrieve and manipulate field values.

- Form Fields:

Form fields are widgets used to collect input from users, such as text, numbers, dates, and selections.
Flutter provides various form field widgets like TextFormField for text input, DropdownButton for selection input, Checkbox for boolean input, etc.
Each form field widget typically requires a key, an optional initialValue, and a validator function for input validation.
Form fields can be customized with properties like decoration, style, onChanged, onSaved, and more.

- Validation:

Validation is the process of ensuring that the data entered by the user meets certain criteria or constraints.
Flutter's Form widget supports both built-in and custom validation using the validator parameter of form field widgets.
Built-in validators are provided by Flutter for common validation tasks like required fields, email format, numeric range, etc.
Custom validation logic can be implemented by defining validator functions that return error messages when validation fails.

- Submission:

Form submission involves processing the data entered by the user and performing actions like saving to a database, sending to a server, or updating the UI.
Flutter provides the onFormSubmitted callback for handling form submission, which is triggered when the user submits the form.
Inside the onFormSubmitted callback, you can access the current values of form fields using their corresponding keys and perform any necessary processing.
State Management:

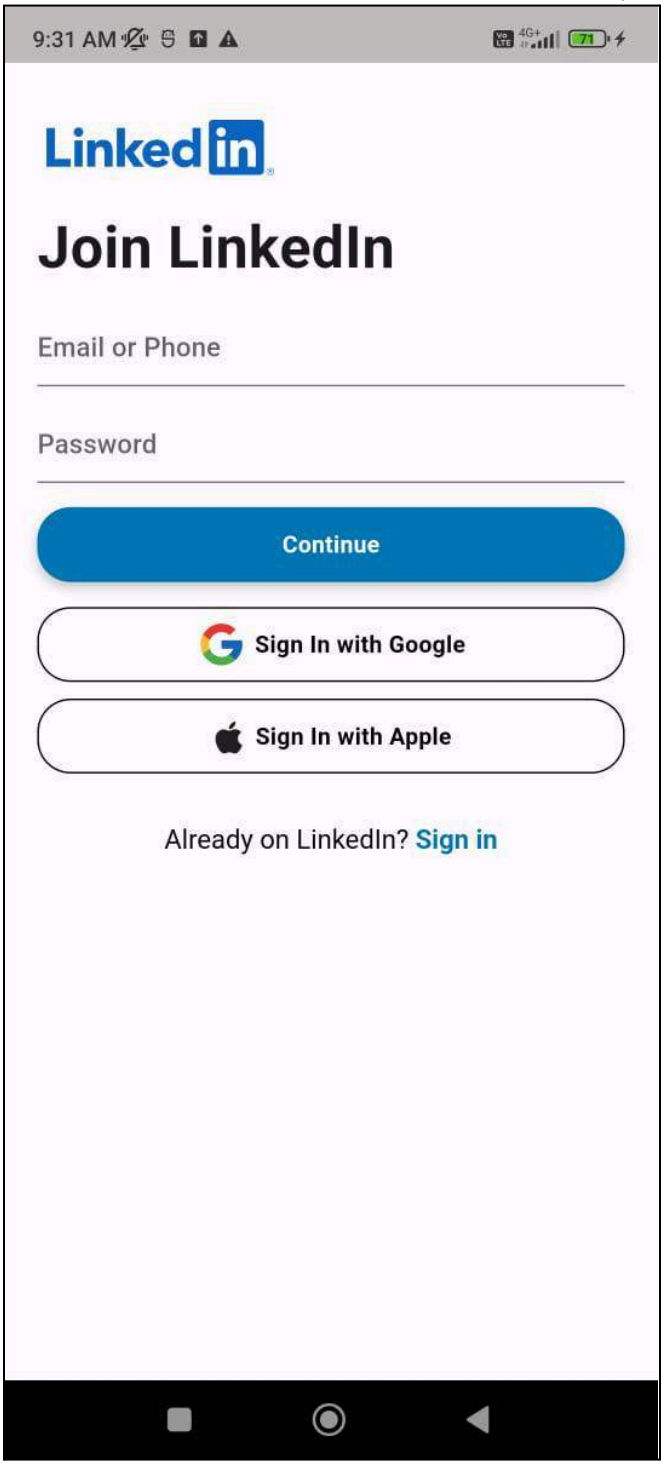
The Form widget manages the state of form fields internally using the FormState class.
The FormState class provides methods for accessing field values, validating fields, saving field values, resetting fields, and more.
You can obtain a reference to the FormState object using a GlobalKey<FormState> assigned to the Form widget and call its methods to interact with the form fields programmatically.

Code :

```
import 'package:flutter/gestures.dart';
import 'package:flutter/material.dart';
import 'package:flutter_svg/flutter_svg.dart';
import
'package:font_awesome_flutter/font_awesome_flutter.dart';
import
'package:linkedin_mobile_ui/pages/auth/sign_in_page.dart';
import
'package:linkedin_mobile_ui/pages/main/main_page/main_page.dart';
import
'package:linkedin_mobile_ui/theme/styles.dart';
import
'package:linkedin_mobile_ui/widgets/button_container_widget.dart';

import
'../../widgets/google_button_container_widget.dart';

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

  @override
  State<SignUpPage> createState() =>
_SignUpPageState();
}

class _SignUpPageState extends
State<SignUpPage> {
  bool _isContinued = false;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Container(
        margin: const EdgeInsets.only(top: 60),
        child: SingleChildScrollView(
          child: Column(
            crossAxisAlignment:
CrossAxisAlignment.start,
            children: [
              Padding(
                padding: const
EdgeInsets.only(left: 5.0),
                child: SvgPicture.asset(
                  "assets/app_logo_svg.svg",
                  width: 50,
                  height: 50,
                ),
              ),
              const SizedBox(
                height: 10,
              ),
              Container(
                margin: const
EdgeInsets.only(left: 20, right: 20),
                child: Column(
                  crossAxisAlignment:
CrossAxisAlignment.start,
                  children: [
                    const Text("Join LinkedIn",
style: TextStyle(fontSize: 35, fontWeight:
FontWeight.bold),),
                    const SizedBox(height: 10,),
                    TextFormField(
```

```
                decoration: const
InputDecoration(
                  hintText: "Email or Phone",
                ),
              ),
              const SizedBox(height: 10,),
              _isContinued ==
true?TextFormField(
                decoration: const
InputDecoration(
                  hintText: "Password",
                ),
              ) : Container(),
              _isContinued == true? const
SizedBox(height: 15,) : const
SizedBox(height: 0),
              ButtonContainerWidget(
                title: "Continue",
                onTap: () {
                  // You must also check if the
email
                  // is correctly formatted is not
empty
                  if(_isContinued == false) {
                    setState(() {
                      _isContinued = true;
                    });
                    return;
                  }

                  // Next operation

Navigator.pushAndRemoveUntil(context,
MaterialPageRoute(builder: (_) => const
MainPage()), (route) => false);
                },
              ),
              const SizedBox(height: 15,),
              GoogleButtonContainerWidget(
                hasIcon: true,
                icon:
SvgPicture.asset("assets/google_logo_svg.
svg", width: 30, height: 30,),
                title: "Sign In with Google",
              ),
              const SizedBox(height: 10,),
              const
GoogleButtonContainerWidget(
                hasIcon: true,
                icon:
Icon(FontAwesomeIcons.apple, size: 22,),
                title: "Sign In with Apple",
              ),
              const SizedBox(height: 30,),
              Center(
                child: RichText(
                  text: TextSpan(
                    text: "Already on
LinkedIn? ",
                    style: TextStyle(color:
linkedInBlack000000, fontSize: 16),
                    children: [
                      TextSpan(
                        recognizer:
TapGestureRecognizer()..onTap = () {

Navigator.pushAndRemoveUntil(context,
MaterialPageRoute(builder: (_) => const
SignInPage()), (route) => false,);
                        },
                        text: "Sign in",
                        style: TextStyle(color:
linkedInBlue0077B5, fontWeight:
FontWeight.bold, fontSize: 16)
                      )
                    ]
                  ),
                ),
              )
            ],
          ),
        )
      ],
    ),
  ),
);
}
```

```
}
```



Conclusion: Successfully created an interactive form in the flutter application.

MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	20
Name	**NIDHI GAWDE**
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 05

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

Theory :

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

- Navigation:

Navigation refers to the movement between different screens or pages within an app.

In Flutter, navigation is managed by the Navigator class, which maintains a stack of routes representing the app's navigation history.

Navigation can be triggered by user interactions, such as tapping on a button or selecting an item from a list, or programmatically in response to events or user input.

Flutter provides various navigation methods, such as push, pushReplacement, pop, popUntil, etc., to navigate between routes and manipulate the navigation stack.

- Routing:

Routing is the process of defining and configuring routes within a Flutter app.

A route represents a distinct screen or page in the app's UI hierarchy.

In Flutter, routes are typically defined using the MaterialApp widget's routes parameter or by manually creating instances of MaterialPageRoute or CupertinoPageRoute.

Routes can have parameters or arguments that are passed during navigation, allowing data to be shared between screens.

- Gestures:

Gestures are user actions, such as tapping, dragging, pinching, etc., that are detected and handled by the app to trigger specific actions or interactions.

In Flutter, gestures are implemented using gesture recognizer classes like GestureDetector, InkWell, DragGestureRecognizer, etc.

GestureDetector: Detects various gestures, such as taps, drags, long presses, etc., and invokes corresponding callbacks.

InkWell: A material widget that responds to taps with a splash effect. It wraps its child widget and triggers the onTap callback when tapped.

GestureDetector and InkWell can be used to make UI elements interactive and responsive to user input, enhancing the user experience.

- Gesture Recognition:

Gesture recognition is the process of identifying and interpreting user gestures to perform specific actions or trigger events.

Flutter provides built-in gesture recognizers for common gestures like taps, drags, scrolls, etc.

Developers can also implement custom gesture recognizers by subclassing the GestureRecognizer class and overriding its methods to detect and handle custom gestures.

Code:

a) main_page.dart

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import
'package:font_awesome_flutter/font_aweso
me_flutter.dart';
import
'package:linkedin_mobile_ui/pages/main/cre
ate/create_page.dart';
import
'package:linkedin_mobile_ui/pages/main/ho
me/home_page.dart';
import
'package:linkedin_mobile_ui/pages/main/job
s/jobs_page.dart';
import
'package:linkedin_mobile_ui/pages/main/ma
in_page/widgets/drawer_widget.dart';
import
'package:linkedin_mobile_ui/pages/main/net
work/network_page.dart';
import
'package:linkedin_mobile_ui/pages/main/not
ifications/notifications_page.dart';
import
'package:linkedin_mobile_ui/theme/styles.d
art';
import 'widgets/app_bar_widget.dart';

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

  @override
  State<MainPage> createState() =>
_MainPageState();
}

class _MainPageState extends
State<MainPage> {

  final GlobalKey<ScaffoldState>
_scaffoldState =
GlobalKey<ScaffoldState>();

  int _currentPageIndex = 0;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      drawer: const DrawerWidget(),
      key: _scaffoldState,
      appBar: _currentPageIndex == 4?
appBarWidget(
          context,
          title: "Search Jobs",
          isJobsTab: true,
          onLeadingTapClickListener: () {
            setState(() {

_scaffoldState.currentState!.openDrawer();
            });
          }
      ) :appBarWidget(
          context,
          title: "Search",
          isJobsTab: false,
          onLeadingTapClickListener: () {
            setState(() {

_scaffoldState.currentState!.openDrawer();
            });
          }
      ),
```

```
      bottomNavigationBar:
BottomNavigationBar(
        currentIndex: _currentPageIndex,
        onTap: (index) {
          setState(() {
            _currentPageIndex = index;
          });
        },
        selectedItemColor:
linkedInBlack000000,
        selectedLabelStyle: const
TextStyle(color: linkedInBlack000000),
        unselectedItemColor:
linkedInMediumGrey86888A,
        unselectedLabelStyle: const
TextStyle(color:
linkedInMediumGrey86888A),
        showUnselectedLabels: true,
        items: const [
          BottomNavigationBarItem(
            icon:
Icon(CupertinoIcons.house_fill),
            label: "Home",
          ),
          BottomNavigationBarItem(
            icon:
Icon(FontAwesomeIcons.userGroup),
            label: "Network",
          ),
          BottomNavigationBarItem(
            icon: Icon(
              Icons.add_box,
              size: 30,
            ),
            label: "Post",
          ),
          BottomNavigationBarItem(
            icon: Icon(
              Icons.notifications,
              size: 30,
            ),
            label: "Notifications",
          ),
          BottomNavigationBarItem(
            icon:
Icon(FontAwesomeIcons.briefcase),
            label: "Jobs",
          ),
        ],
      ),
      body: _switchPages(_currentPageIndex)
    );
  }

  _switchPages(int index) {
    switch (index) {
      case 0:
        {
          return const HomePage();
        }
      case 1:
        {
          return const NetworkPage();
        }
      case 2:
        {
          return
CreatePage(onCloneClickListener: () {
            Navigator.pop(context);
            setState(() {
              _currentPageIndex = 0;
            });
          },);
        }
      case 3:
        {
          return const NotificationsPage();
        }
      case 4:
        {
```

```
    return const JobsPage();
  }
```

```
    }
  }
}
```

b) Network_page.dart

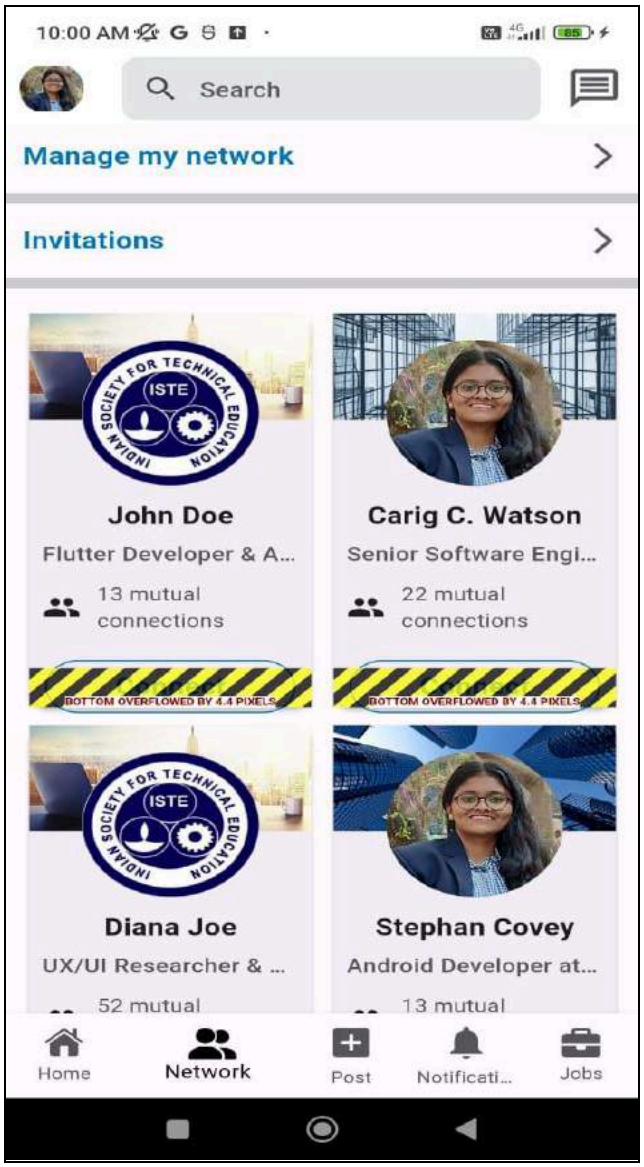
```
import 'package:flutter/material.dart';
import
'package:linkedin_mobile_ui/data/ne
twork_entity.dart';
import
'package:linkedin_mobile_ui/pages/
main/network/widgets/single_networ
k_user_widget.dart';
import
'package:linkedin_mobile_ui/theme/s
tyles.dart';

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

  @override
  State<NetworkPage> createState()
=> _NetworkPageState();
}

class _NetworkPageState extends
State<NetworkPage> {

  List<NetworkEntity> networkData
= NetworkEntity.networkData;

  @override
  Widget build(BuildContext context)
{
    return Scaffold(
      body: SingleChildScrollView(
        child: Column(
          children: [
            const SizedBox(
              height: 10,
            ),
            const Padding(
              padding:
EdgeInsets.symmetric(horizontal:
10.0),
              child: Row(
                mainAxisAlignment:
MainAxisAlignment.spaceBetween,
                children: [
                  Text(
                    "Manage my network",
                    style: TextStyle(
                      fontSize: 18,
                      fontWeight:
FontWeight.bold,
                      color:
linkedInBlue0077B5),
                  ),
                  Icon(

Icons.arrow_forward_ios,
                    color:
linkedInMediumGrey86888A,
                  ),
                ],
              ),
            ),
            const SizedBox(
              height: 15,
            ),
            Container(
              width: double.infinity,
              height: 8,
              color:
linkedInLightGreyCACCCE,
            ),
```

```
          const SizedBox(
            height: 15,
          ),
          const Padding(
            padding:
EdgeInsets.symmetric(horizontal:
10.0),
            child: Row(
              mainAxisAlignment:
MainAxisAlignment.spaceBetween,
              children: [
                Text(
                  "Invitations",
                  style: TextStyle(
                      fontSize: 18,
                      fontWeight:
FontWeight.bold,
                      color:
linkedInBlue0077B5),
                ),
                Icon(

Icons.arrow_forward_ios,
                  color:
linkedInMediumGrey86888A,
                ),
              ],
            ),
          ),
          const SizedBox(
            height: 15,
          ),
          Container(
            width: double.infinity,
            height: 8,
            color:
linkedInLightGreyCACCCE,
          ),
          const SizedBox(
            height: 15,
          ),
          GridView.builder(
            padding: const
EdgeInsets.symmetric(horizontal:
10),
            shrinkWrap: true,
            physics: const
ScrollPhysics(),
            itemCount:
networkData.length,
            gridDelegate: const
SliverGridDelegateWithFixedCross
AxisCount(
                crossAxisCount: 2,
                mainAxisSpacing: 6,
                crossAxisSpacing: 6,
                childAspectRatio: 0.6),
            itemBuilder: (context,
index) {
              final network =
networkData[index];
              return
SingleNetworkUserWidget(network:
network);
            },
          )
        ],
      ),
    ),
  );
}
}
```

c) network_entity.dart

```
class NetworkEntity {

  final String? userBgImage;
  final String? userProfileImage;
  final String? username;
  final String? userBio;
  final num? mutualConnections;

  NetworkEntity(
      {this.userBgImage,
      this.userProfileImage,
      this.username,
      this.userBio,
      this.mutualConnections});

  static List<NetworkEntity>
networkData = [

    NetworkEntity(
      userBgImage:
"bg_image_1.jpeg",
      userProfileImage:
"profile_2.jpeg",
      mutualConnections: 13,
      userBio: "Flutter Developer &
Advocate",
      username: "ISTE",
    ),

    NetworkEntity(
      userBgImage:
"bg_image_2.png",
      userProfileImage:
"profile_1.jpeg",
      mutualConnections: 22,
      userBio: "Senior Software
Engineer",
      username: "",
    ),

    NetworkEntity(
      userBgImage:
"bg_image_1.jpeg",
      userProfileImage:
"profile_2.jpeg",
      mutualConnections: 52,
      userBio: "UX/UI Researcher &
Designer",
      username: "Diana Joe",
    ),

    NetworkEntity(
      userBgImage:
"bg_image_3.jpeg",
      userProfileImage:
"profile_1.jpeg",
      mutualConnections: 13,
      userBio: "Android Developer at
Google",
      username: "Stephan Covey",
    ),

    NetworkEntity(
      userBgImage:
"bg_image_1.jpeg",
      userProfileImage:
"profile_2.jpeg",
      mutualConnections: 88,
      userBio: "Flutter Developer &
Advocate",
      username: "Elon Musk",
    ),

    NetworkEntity(
```

```
      userBgImage:
"bg_image_2.png",
      userProfileImage:
"profile_1.jpeg",
      mutualConnections: 11,
      userBio: "Flutter Developer &
Advocate",
      username: "Robert Frost",
    ),

    NetworkEntity(
      userBgImage:
"bg_image_3.jpeg",
      userProfileImage:
"profile_2.jpeg",
      mutualConnections: 13,
      userBio: "Flutter Developer &
Advocate",
      username: "Steve Wozniak",
    ),

    NetworkEntity(
      userBgImage:
"bg_image_3.jpeg",
      userProfileImage:
"profile_1.jpeg",
      mutualConnections: 76,
      userBio: "Flutter Developer &
Advocate",
      username: "Doug Stevenson",
    ),
  ];

}
```



Conclusion : Successfully applied navigation, routing and gestures in Flutter App

MAD & PWA Lab
Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	20
Name	**NIDHI GAWDE**
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:	15

Experiment 06

Aim : To Connect Flutter UI with fireBase database

Theory :

Firebase is a comprehensive mobile and web application development platform provided by Google. It offers a wide range of features and services that enable developers to build high-quality apps quickly and efficiently.

At its core, Firebase provides tools for various aspects of app development, including authentication, real-time database, cloud storage, hosting, and more. Here's a breakdown of some key Firebase components:

Authentication: Firebase Authentication allows developers to easily integrate secure authentication methods into their apps. It supports various authentication methods, including email/password, phone number, Google, Facebook, Twitter, and more. This feature handles user management, authentication, and security, making it simple to add user sign-up, sign-in, and access control to apps.

Realtime Database: Firebase Realtime Database is a NoSQL cloud database that enables developers to store and sync data in real time. It's particularly useful for applications that require real-time updates, such as chat apps, collaboration tools, and gaming apps. The database is JSON-based, which makes it flexible and easy to use. It automatically synchronizes data across all connected clients and devices, ensuring that changes made by one user are immediately reflected on others' screens.

Cloud Firestore: Cloud Firestore is Firebase's newer, more scalable NoSQL database solution. It offers more powerful querying capabilities, better scalability, and improved performance compared to the Realtime Database. Firestore organizes data into collections and documents, allowing for more structured and efficient data storage. It also supports real-time updates, offline data access, and powerful querying capabilities.

Cloud Storage: Firebase Cloud Storage provides secure and reliable cloud storage for user-generated content, such as images, videos, and documents. It allows developers to easily upload and download files from their apps using simple APIs. Cloud Storage integrates seamlessly with other Firebase services, making it easy to store and serve user-generated content in Firebase-powered apps.

Hosting: Firebase Hosting allows developers to quickly and securely deploy web apps and static content to a global content delivery network (CDN). It provides fast and reliable hosting with SSL encryption, custom domain support, and automatic scaling. With Firebase Hosting, developers can deploy web apps with just a few simple commands, eliminating the need for complex server configurations and maintenance.

Cloud Functions: Firebase Cloud Functions allow developers to run backend code in response to events triggered by Firebase features and HTTPS requests. It enables developers to extend the

functionality of their apps without managing servers or infrastructure. Cloud Functions are written in JavaScript or TypeScript and can be deployed with a single command using the Firebase CLI (Command Line Interface).

Analytics: Firebase Analytics provides insights into app usage and user behavior, helping developers understand how users interact with their apps. It tracks key metrics such as active users, retention, engagement, and conversion, allowing developers to make informed decisions about app improvements and optimizations.

Performance Monitoring: Firebase Performance Monitoring helps developers identify and fix performance issues in their apps. It provides detailed insights into app performance, including app startup time, network latency, and UI responsiveness. Performance Monitoring helps developers optimize their apps for better user experiences and higher ratings.

In addition to these core features, Firebase offers many other services, including Cloud Messaging for push notifications, Remote Config for dynamic app configuration, Machine Learning for integrating machine learning models into apps, and more. Overall, Firebase provides a powerful and comprehensive platform for building, managing, and scaling mobile and web applications.

Code:

```
import 'package:flutter/material.dart';
import
'package:font_awesome_flutter/font_aweso
me_flutter.dart';
import
'package:cloud_firestore/cloud_firestore.dart
'; // Import Firestore
import
'package:linkedin_mobile_ui/theme/styles.d
art';

class CreatePage extends StatefulWidget {
  final VoidCallback? onCloneClickListener;
  const CreatePage({Key? key, required
this.onCloneClickListener})
      : super(key: key);
  @override
  State<CreatePage> createState() =>
_CreatePageState();
}

class _CreatePageState extends
State<CreatePage> {

  final TextEditingController
_postBodyController =
TextEditingController();

  bool _openTwoBottomModalSheetsOnce =
false;
```

```
  final FocusScopeNode
_subPostBottomModalSheetFocusNode =
FocusScopeNode();
  final FocusScopeNode
_superPostBottomModalSheetFocusNode =
FocusScopeNode();

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

  // Function to add post to Firestore
  Future<void> _addPostToFirestore(String
postBody) async {
    try {
      await
FirebaseFirestore.instance.collection('posts').
add({
        'body': postBody,
        'timestamp': DateTime.now(),
      });
      print('Post added to Firestore');
    } catch (e) {
      print('Error adding post to Firestore: $e');
    }
  }

  @override
  Widget build(BuildContext context) {
    if(_openTwoBottomModalSheetsOnce ==
false) {

WidgetsBinding.instance!.addPostFrameCal
lback((timeStamp) {
        _createSuperPostBottomModalSheet();
        _createSubPostBottomModalSheet();
        print("value before =
$_openTwoBottomModalSheetsOnce");
        setState(() {
          _openTwoBottomModalSheetsOnce =
true;
        });
        print("value after =
$_openTwoBottomModalSheetsOnce");
      });
    }

    return const Scaffold();
  }

  _createSuperPostBottomModalSheet() {
    showModalBottomSheet(
      isScrollControlled: true,
      enableDrag: false,
      isDismissible: false,
      context: context,
      builder: (context) {
        return StatefulBuilder(
          builder: (context, void Function(void
Function()) setState) {
            return FocusScope(
              node:
_superPostBottomModalSheetFocusNode,
              child: Container(
                child: Column(
                  crossAxisAlignment:
CrossAxisAlignment.start,
                  children: [
                    Container(
                      padding: const
EdgeInsets.symmetric(horizontal: 10),
                      width: double.infinity,
                      height: 110,
                      decoration:
                      const BoxDecoration(color:
linkedInWhiteFFFFFF, boxShadow: [
                        BoxShadow(
                          offset: Offset(0, 2),
```

```
                    color:
linkedInLightGreyCACCCE,
                    blurRadius: 5,
                    spreadRadius: 0.1),
                ]),
                child: Padding(
                  padding: const
EdgeInsets.only(bottom: 15.0),
                  child: Row(
                    crossAxisAlignment:
CrossAxisAlignment.end,
                    mainAxisAlignment:
MainAxisAlignment.spaceBetween,
                    children: [
                      Row(
                        children: [
                          GestureDetector(
                              onTap:
widget.onCloneClickListener,
                              child: const Icon(

Icons.close_outlined,
                                size: 30,
                              )),
                          const SizedBox(
                            width: 15,
                          ),
                          const Text(
                            "Share Post",
                            style: TextStyle(
                                fontSize: 25,
                                fontWeight:
FontWeight.bold,
                                color:
linkedInMediumGrey86888A),
                          )
                        ],
                      ),
                      Text(
                        "Post",
                        style: TextStyle(
                          fontSize: 22,
                          fontWeight:
FontWeight.bold,
                          color:
_postBodyController.text.isEmpty?
linkedInLightGreyCACCCE :
linkedInBlue0077B5),
                        )
                      ],
                    ),
                  ),
                ),
                const SizedBox(
                  height: 30,
                ),
                Expanded(
                  child: Container(
                    margin: const
EdgeInsets.symmetric(horizontal: 20),
                    child: Column(
                      children: [
                        Row(
                          children: [
                            Container(
                              width: 60,
                              height: 60,
                              child: ClipRRect(
                                borderRadius:
BorderRadius.circular(30),
                                child:
Image.asset("assets/profile_1.jpeg"),
                              ),
                            ),
                            const SizedBox(
                              width: 10,
                            ),
                            Column(
                              mainAxisAlignment:
MainAxisAlignment.end,
```

```
                    crossAxisAlignment:
CrossAxisAlignment.start,
                    children: [
                      _switchWidget(
                        title: "Nidhi
Gawde",
                        prefixIcon:
Icons.account_circle_rounded,
                        suffixIcon:
Icons.arrow_drop_down_outlined),
                      const SizedBox(
                        height: 5,
                      ),
                      _switchWidget(
                        title: "Anyone",
                        prefixIcon:
FontAwesomeIcons.earth,
                        suffixIcon:
Icons.arrow_drop_down_outlined),
                    ],
                  ),
                ],
              ),
              const SizedBox(
                height: 20,
              ),
              TextFormField(
                controller:
_postBodyController,
                onTap: () {

_postBodyController.addListener(() {

if(_postBodyController.text.length == 1) {
                      setState(() {});
                      print("call first IF
setState");
                    } else if
(_postBodyController.text.length < 1) {
                      setState(() {});
                      print("2nd ELSE IF
setState");
                    }

                    print("onTap called");
                  });
                },
                style: const
TextStyle(fontSize: 22),
                maxLines: 15,
                decoration: const
InputDecoration(
                  hintText: "What do
you want to talk about?",
                  border:
InputBorder.none),
                ),
              ],
            ),
          ),
        ),
        Container(
          margin: const
EdgeInsets.symmetric(horizontal: 30),
          child: Row(
            mainAxisAlignment:
MainAxisAlignment.spaceBetween,
            children: [
              Row(
                children: [
                  const Icon(
                    Icons.camera_alt,
                    color:
linkedInMediumGrey86888A,
                  ),
                  const SizedBox(
                    width: 15,
                  ),
                  const Icon(
                    Icons.video_call,
```

```
                    color:
linkedInMediumGrey86888A,
                  ),
                  const SizedBox(
                    width: 15,
                  ),
                  const Icon(
                    Icons.image,
                    color:
linkedInMediumGrey86888A,
                  ),
                  const SizedBox(
                    width: 25,
                  ),
                  GestureDetector(
                      onTap: () {

_createSubPostBottomModalSheet();
                      },
                      child: const Icon(
                        Icons.more_horiz,
                        color:
linkedInMediumGrey86888A,
                      )),
                ],
              ),
              const Row(
                children: [
                  Icon(

Icons.message_outlined,
                    color:
linkedInMediumGrey86888A,
                  ),
                  SizedBox(
                    width: 10,
                  ),
                  Text(
                    "Anyone",
                    style: TextStyle(
                      fontWeight:
FontWeight.bold,
                      color:
linkedInMediumGrey86888A),
                  )
                ],
              )
            ],
          ),
        ),


      ],
    ),
  ),
);
}
);
},
).then((value) {

_superPostBottomModalSheetFocusNode.u
nfocus();
});
}

_switchWidget({String? title, IconData?
prefixIcon, IconData? suffixIcon}) {
  return Container(
    height: 30,
    padding: const
EdgeInsets.symmetric(horizontal: 10),
    decoration: BoxDecoration(
      borderRadius:
BorderRadius.circular(20),
      border: Border.all(width: 1, color:
linkedInMediumGrey86888A)),
    child: Row(
      children: [
        Icon(
```

```
          prefixIcon,
          color: linkedInMediumGrey86888A,
          size: 18,
        ),
        const SizedBox(
          width: 5,
        ),
        Text(
          "$title",
          style: const TextStyle(fontWeight:
FontWeight.bold, fontSize: 12),
        ),
        const SizedBox(
          width: 5,
        ),
        Icon(
          suffixIcon,
          size: 30,
        ),
      ],
    ),
  );
}

 _createSubPostBottomModalSheet() {
  showModalBottomSheet(
    shape: RoundedRectangleBorder(
      borderRadius:
BorderRadius.circular(20),
    ),
    barrierColor: Colors.transparent,
    context: context,
    builder: (context) {
      return FocusScope(
        node:
_subPostBottomModalSheetFocusNode,
        child: Container(
          decoration: BoxDecoration(color:
linkedInWhiteFFFFFF, boxShadow: [
            BoxShadow(
              offset: const Offset(5, 0),
              blurRadius: 1,
              color:
linkedInLightGreyCACCCE.withOpacity(.6
),
              spreadRadius: 0.5)
          ]),
          child: Padding(
            padding: const
EdgeInsets.symmetric(horizontal: 20.0,
vertical: 20),
            child: SingleChildScrollView(
              child: Column(
              crossAxisAlignment:
CrossAxisAlignment.start,
              children: [
                Center(
                  child: Container(
                    width: 80,
                    height: 6,
                    decoration: BoxDecoration(
                      borderRadius:
BorderRadius.circular(10),
                      color:
linkedInMediumGrey86888A),
                  ),
                ),
                const SizedBox(height: 20,),

_createSubPostNavigationItem(title: "Add a
photo", iconData: Icons.image),
                const SizedBox(height: 25,),

_createSubPostNavigationItem(title: "Take a
video", iconData: Icons.video_call),
                const SizedBox(height: 25,),

_createSubPostNavigationItem(title: "Use a
template", iconData:
Icons.temple_buddhist),
```

```
                const SizedBox(height: 25,),

_createSubPostNavigationItem(title:
"Celebrate an occasion", iconData:
Icons.celebration),
                const SizedBox(height: 25,),

_createSubPostNavigationItem(title: "Add a
document", iconData:
Icons.document_scanner),
                const SizedBox(height: 25,),

_createSubPostNavigationItem(title: "Share
that you're hiring", iconData: Icons.work),
                const SizedBox(height: 25,),

_createSubPostNavigationItem(title: "Find
an expert", iconData:
Icons.account_circle_rounded),
                const SizedBox(height: 25,),

_createSubPostNavigationItem(title: "Create
a poll", iconData: Icons.bar_chart),
                const SizedBox(height: 25,),

_createSubPostNavigationItem(title: "Create
an event", iconData: Icons.event),
                const SizedBox(height: 25,),
              ],
              ),
            ),
          ),
        ),
      );
    },
  ).then((value) {

_subPostBottomModalSheetFocusNode.unf
ocus();
  });
 }

 _createSubPostNavigationItem({IconData?
iconData, String? title}) {
  return Row(
    children: [
      Icon(iconData, size: 25,color:
linkedInMediumGrey86888A,),
      const SizedBox(width: 10,),
      Text("$title", style: const
TextStyle(fontSize: 16, color:
linkedInMediumGrey86888A, fontWeight:
FontWeight.bold),)
    ],
  );
 }
}
```

Add Firebase to your Android app

1 Register app

Android package name

com.android.application

App nickname (optional)

linked-in-clone

Debug signing certificate SHA-1 (optional)

00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:0

Required for Dynamic Links, and Google Sign-In or phone number support in Auth. Edit SHA-1s in Settings.

Register app

3 Add Firebase SDK

Instructions for Gradle | Unity C++

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

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

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

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

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

```
plugins {
  // ...

  // Add the dependency for the Google services Gradle plugin
  id 'com.google.gms.google-services' version '4.4.1' apply false
}
```

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

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

```
plugins {
  id 'com.android.application'
  // Add the Google services Gradle plugin
  id 'com.google.gms.google-services'
  ...
}

dependencies {
  // Import the Firebase BoM
  implementation platform('com.google.firebase:firebase-bom:32.7.2')

  // TODO: Add the dependencies for Firebase products you want to use
  // When using the BoM, don't specify versions in Firebase dependencies
  implementation 'com.google.firebase:firebase-analytics'

  // Add the dependencies for any other desired Firebase products
  // https://firebase.google.com/docs/android/setup#available-libraries
}
```

By using the Firebase Android BoM, your app will always use compatible Firebase library versions. Learn more

3. After adding the plug-in and the desired SDKs, sync your Android project with the Gradle files.

Your project

Project name	linked-in-clone1
Project ID	linked-in-clone1-6b851
Project number	557237544811
Default GCP resource location	Not yet selected
Web API key	AIzaSyC6rIcYaABsa3NAB4cXjwATd2gvwA4zz_Q

Environment

This setting customises your project for different stages of the app lifecycle

Environment type	Unspecified

Firestore:

posts > 9IsKSSQJEGUt... More in Google Cloud

(default)	posts	9IsKSSQJEGUtLzoTv5wX
+ Start collection	+ Add document	+ Start collection
posts	9IsKSSQJEGUtLzoTv5wX	+ Add field
	EwTh7gzzDLXFSDJFHOXu	author: "Nidhi Gawde"
	FIMfjN1yNL1CAaIMfC06	content: "This is my first post"
	N1va2jWfaE9iaFtp1f9R	timestamp: 23 February 2024 at 00:40:39 UTC+5:30
	WS6GWcY7JfI3m6EEmsBX	

Query results

Document ID	author	content	timestamp
9IsKSSQJEGUtLzoTv5wX	"Nidhi Gawde"	"This is my first post"	23 February 2024 at 00:40:39 UTC+5:30
EwTh7gzzDLXFSDJFHOXu	"Nidhi Gawde"	"Common mistakes in flutter"	23 February 2024 at 01:18:37 UTC+5:30
FIMfjNlyNLICAaIMfC06	"Nidhi Gawde"	"Flutter"	23 February 2024 at 08:57:56 UTC+5:30
N1va2jWfaE9iaFtpIf9R	"Nidhi Gawde"	"Development"	23 February 2024 at 08:57:09 UTC+5:30
WS6GWcY7Jfl3m6EEmsBX	"Nidhi Gawde"	"Updates in flutter"	23 February 2024 at 01:18:50 UTC+5:30
iueFrrcr1DM5b4bWDmNT	"Nidhi Gawde"	"Flutter mistakes"	23 February 2024 at 08:57:44 UTC+5:30
vxQ0Yyv5YUpvm5ImTubt	"Nidhi Gawde"	"hello"	23 February 2024 at 08:55:52 UTC+5:30

Authentication:

Sign-in providers

Add new provider

Provider Status

Select a sign-in provider (step 1 of 2)

Native providers: Email/Password

Additional providers: Google, Facebook, Play Games

Custom providers: OpenID Connect

Authentication

Users | Sign-in method | Templates | Usage | Settings | Extensions

Search by email address, phone number or user UID

Add user

Identifier	Providers	Created	Signed in	User UID
nidhigawde1@gmail.com		22 Feb 2024	22 Feb 2024	PC84UY2Dn9QIHfAITN939sTu...

Rows per page 50 1 – 1 of 1

Conclusion: Successfully connected Firebase with the Flutter Application.

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.	20
Name	**NIDHI GAWDE**
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	15

Experiment 07

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

Theory:-

- Regular Web App:

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

- Progressive Web App:

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

- Difference between PWAs vs. Regular Web Apps:

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

1. Native Experience

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

2. Ease of Access

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

3. Faster Services

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

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

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

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

- <u>Pros and cons of the Progressive Web App</u>

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.

- <u>Weaknesses refer to:</u>

IOS support from version 11.3 onwards;
Greater use of the device battery;
Not all devices support the full range of PWA features (same speech for iOS and Android operating systems);
It is not possible to establish a strong re-engagement for iOS users (URL scheme, standard web notifications);
Support for offline execution is however limited;
Lack of presence on the stores (there is no possibility to acquire traffic from that channel);
There is no “body” of control (like the stores) and an approval process;
Limited access to some hardware components of the devices;
Little flexibility regarding “special” content for users (eg loyalty programs, loyalty, etc.).

<u>Code :</u>

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

```
<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>



//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":"app ui ecom/assets/img/aclogoshop.png",
```

```
        "sizes":"192x192",
        "type":"image/png",
        "purpose":"any"
        }
]
}


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

Output:

Application :

Identity

Name	PWA Tutorial
Short name	PWA
Description	This is a PWA tutorial.
Computed App ID	http://127.0.0.1:5500/app%20ui%20ecom/index.html Learn more

Note: `id` is not specified in the manifest, `start_url` is used instead. To specify an App ID that matches the current identity, set the `id` field to `/app%20ui%20ecom/index.html`.

Presentation

Start URL	index.html
Theme color	black
Background color	#5900b3
Orientation	
Display	standalone

Icons

Show only the minimum safe area for maskable icons

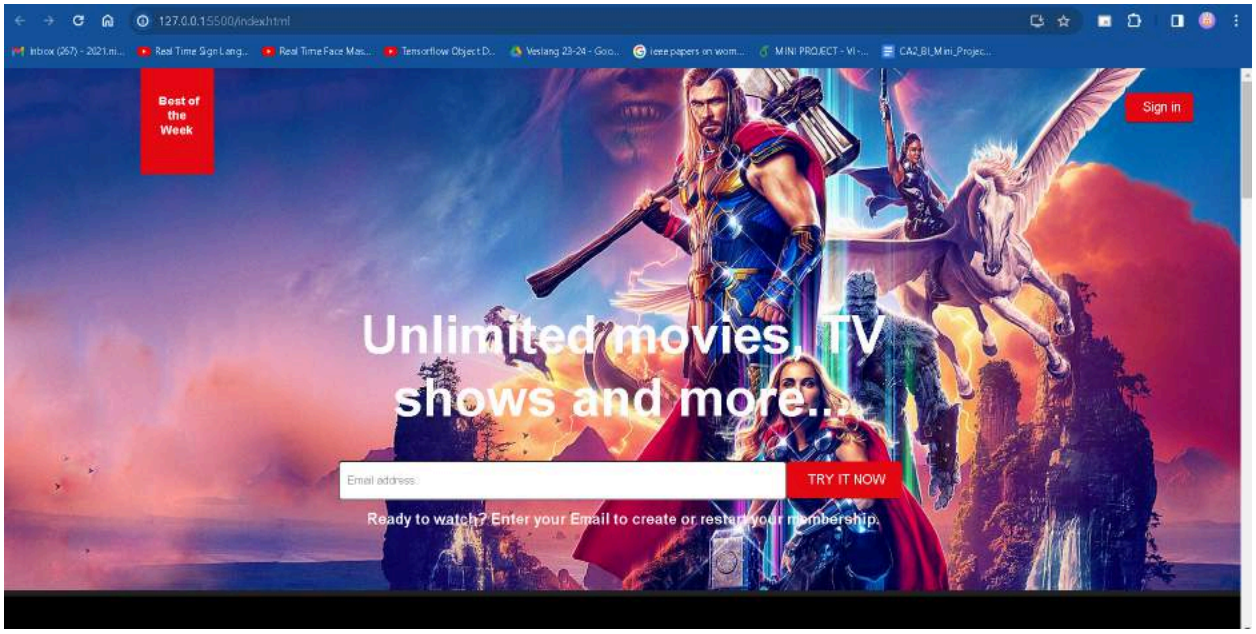
Need help? Read the documentation on maskable icons.

192×192px

image/png

Add to home screen (here on desktop):

Home Screen :



Conclusion:

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

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.	20
Name	**NIDHI GAWDE**
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

Experiment 08

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

Theory:

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

Things to note about Service Worker:

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

What can we do with Service Workers?

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

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

- You can manage **Push Notifications**
  You can manage push notifications with Service Worker and show any information message to the user.
- You can **Continue**
  Although Internet connection is broken, you can start any process with Background

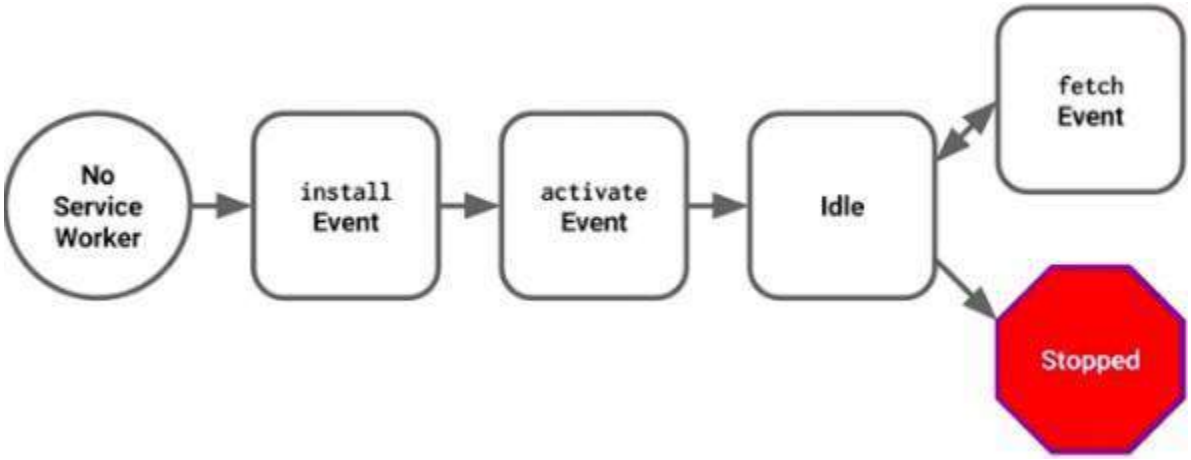
Sync of Service Worker.

What can't we do with Service Workers?

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

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

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. Let's look at an example:

main.js

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

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

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

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

main.js

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

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

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

main.js

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

Installation

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

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

service-worker.js

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

Activation

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

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

service-worker.js

```
self.addEventListener('activate', function(event) {
  // Perform some task
});
```

Once activated, the service worker controls all pages that load within its scope, and starts listening for events from those pages. However, pages in your app that were loaded before the service worker activation will not be under service worker control. The new service worker will only take over when you close and reopen your app, or if the service worker calls **clients.claim()**. Until then, requests from this page will not be intercepted by the new service worker. This is intentional as a way to ensure consistency in your site.

CODE:

index.html

```
<!DOCTYPE html>

<html>
<head>
	<title>Netflix - Watch TV Shows Online,
Watch Movies Online</title>
	<meta charset="utf-8">
	<meta name="viewport"
content="width=device-width, initial-scale=1">

	<!--for making the website more
responsive-->
	<meta http-equiv="X-UA-compatible"
content="ie=">
	<!--for Favicon-->
	<link rel="shortcut icon" type="image/png"
href="https://img.techpowerup.org/200517/prodipto.
png">
	<!--for CSS-->
	<link rel="stylesheet" type="text/css"
href="netflixstyles.css">
	<!--link rel="stylesheet"
href="https://www.w3schools.com/lib/w3.css"/ -->

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

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

<body>
	<header class="showcase">
		<div class="showcase-top">
			<p>Best of the Week</p>
		</div>

		<nav>
			<a href="#" class="btn
btn-m"> Sign in <i class="fas fa-chevron-right
btn-icon"></i>
			</a>
		</nav>
		<!--inside the following div we could
write : style="background-image:
url('https://img.techpowerup.org/200613/netflix-back
groung-image.jpg');"-->
		<div class="showcase-content" >

<br><br><br><br><br><br><br><br><br><br><br
><br><br><br><br>
			<h1>Unlimited movies,
TV<br>shows and more...</h1>
			<br><br>
			<div class="search-box">
			<input class="search-txt"
type="text" name="box" placeholder="Email
address."><a href="#" class="btn  btn-x1">TRY IT
NOW<i class="fas fa-chevron-right btn-icon"></i>
			</a>
			</div>
		</div>

		<div>
```

```
                <h3>Ready to watch?  Enter
your Email to create or restart your membership.
</h3>
                <br><br>

            </div>
        </header>


        <div class="enjoy-tv">
            <div>
                <h2> Enjoy on your TV. </h2>
                <p>Watch on your smart-TV,
PlayStation,<br>XBox, ChromeCast, AppleTV,
blu-<br>ray players and more.</p>
            </div>
            <img
src="https://t.ctcdn.com.br/hxMIOa4911i56IHic6sET
QtHCBg=/1270x714/smart/i541471.jpeg">
        </div>

        <div class="Watch-offline">
            <div>
                <h2>Download your
shows<br>to watch offline.</h2>
                <p>Save your favourites easily
and<br>always have something to watch.</p>
            </div>
            <img
src="https://play-lh.googleusercontent.com/4DOuvV
tWKp_dcvPvueS-g3tUQw08AtSgumH1b7iNlX4Ykc
wUyBcAsKWIljaDjxT3hKc=w526-h296-rw">
        </div>

        <div class="watch-everywhere">
            <h2>Watch Everywhere.</h2>
            <p>Stream unlimited movies and
TV<br>shows on your phon, tablet, laptop,<br>and
TV.</p>
        </div>

        <div class="ask-question">
            <h1>Frequently Asked
Questions...</h1>
            <button>What is
Netfiix?</button><br><br>
            <button>How much does Netflix
costs?</button><br><br>
            <button>Where can I
watch?</button><br><br>
            <button>How do I
cancel?</button><br><br>
            <button>What can I watch on
netflix?</button>
        </div>

        <div>
            <input class="search-txt" type="text"
name="box" placeholder="Email address"><a
href="#" class="btn btn-x1">TRY IT NOW><i
class="fas fa-chevron-right btn-icon"></i></a>
        </div>
        <br>

        <div>
            <h3>Ready to watch? Enter your
email to create or restart your
membership.</h3><br><br>
        </div>

        <footer class="page-down">
                <p style="font-weight:
600;"><a href="#">Quetions? Call
000-800-040-1842</a></p>
                <p style="font-weight:
600;"><a href="#">FAQ</a></p>
                <p style="font-weight:
600;"><a href="#">Investor Relations.</a></p>
                <p style="font-weight:
600;"><a href="#">Privacy.</a></p>
                <p style="font-weight:
600;"><a href="#">Speed Test.</a></p>
                <p><a href="#">Netflix
India</a></p>
                <p><a href="#">Help
Center.</a></p>
                <p><a
href="#">Jobs.</a></p>
```

```
                    <p><a href="#">Cookie
Preference.</a></p>
                    <p><a href="#">Legal
Notices.</a></p>
                    <p><a
href="#">Account.</a></p>
                    <p><a href="#">Way to
Watch.</a></p>
                    <p><a href="#">Corporate
Information.</a></p>
                    <p><a href="#">Netflix
Originals.</a></p>
                    <p><a href="#">Media
Centre.</a></p>
                    <p><a href="#">Terms of
Use.</a></p>
                    <p><a href="#">Contact
us</a></p>
        </footer>

        <script src="/app.js"></script>

</body>
</html>
```

<u>serviceworker.js</u>

```
var cacheName = "pwa-1";
const assetsToCache = [
   "/",
   "/index.html",
   "/netflixstyles.css",
   "/icon-192x192.jpg", // Add more product images as needed
   "/icon-512x512.jpg",
   "manifest.json", // Ensure manifest is cached
   "serviceworker.js",// Ensure serviceworker is cached
   "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 => {
```

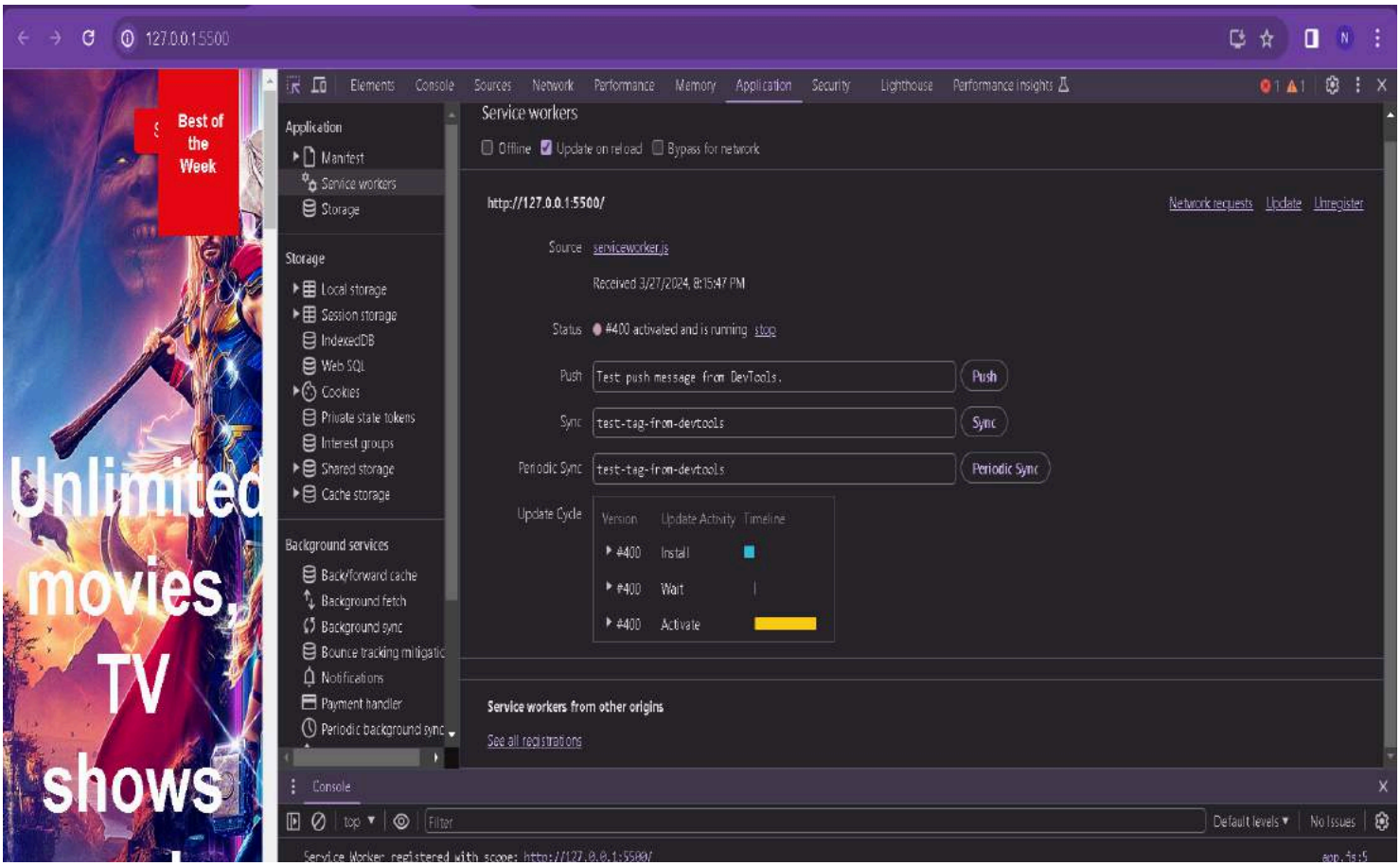
```
                return name !== cacheName;
            }).map(name => {
        return caches.delete(name);
    })
  );
})
);
});
```

app.js

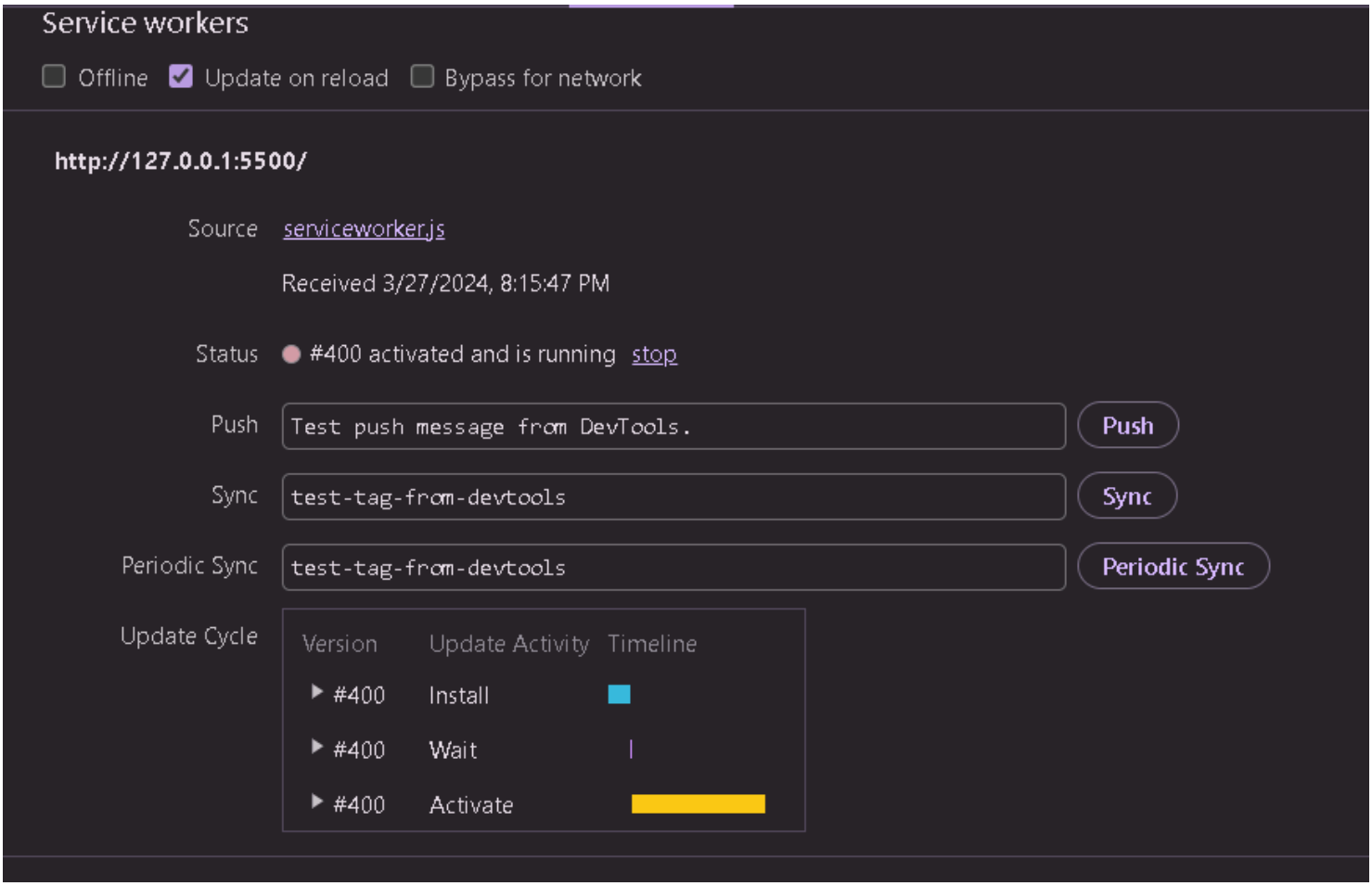
```
if ('serviceWorker' in navigator) {
  window.addEventListener('load', () => {
    navigator.serviceWorker.register('/serviceworker.js')
      .then(registration => {
        console.log('Service Worker registered with scope:', registration.scope);
  })
  .catch(error => {
    console.error('Service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

OUTPUT:

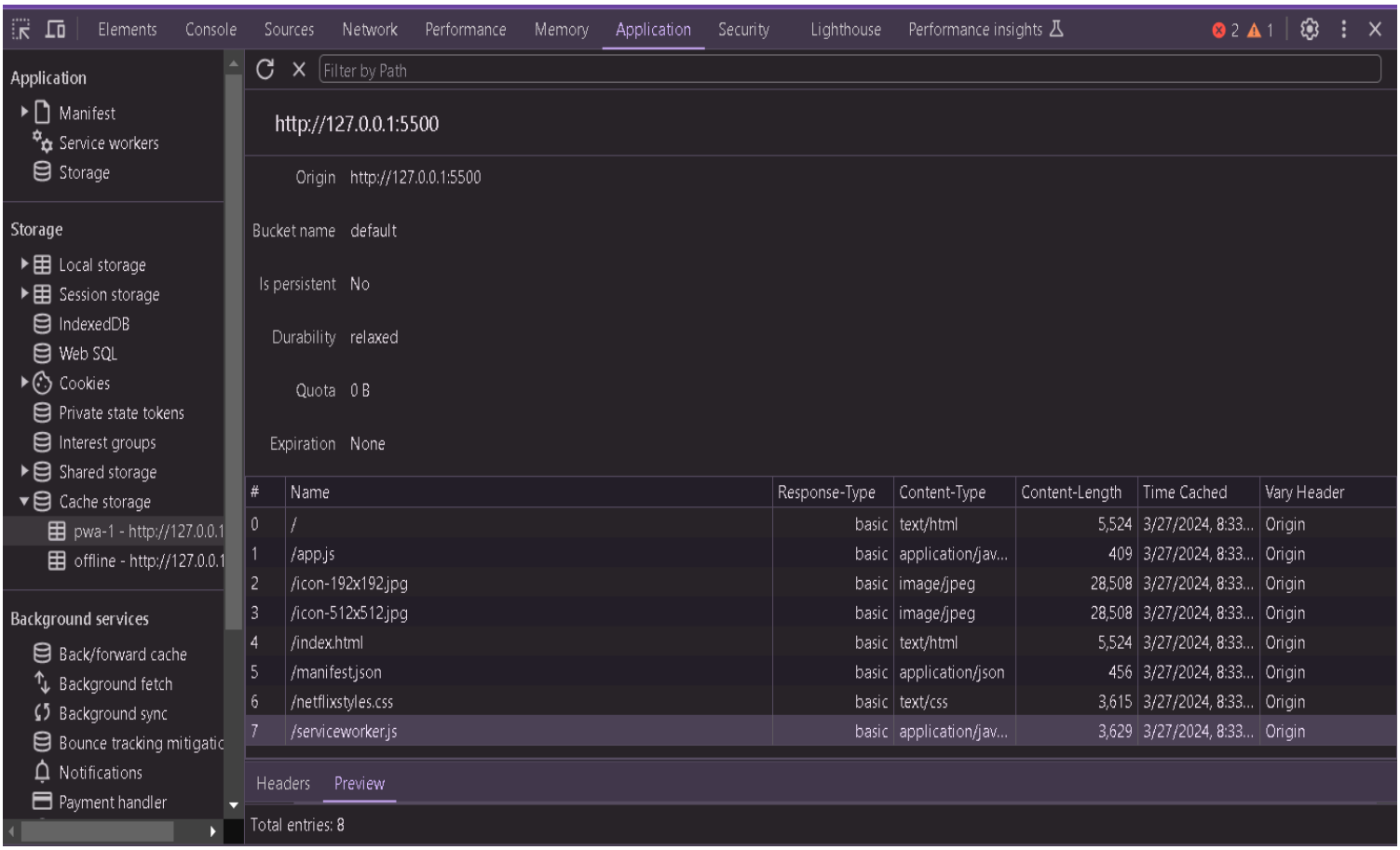
Project Title: Flutter - LinkedIn Clone / PWA-Netflix Website **Roll No.: 20**



Project Title: Flutter - LinkedIn Clone / PWA-Netflix Website **Roll No.: 20**



Project Title: Flutter - LinkedIn Clone / PWA-Netflix Website **Roll No.: 20**



Conclusion: Successfully registered a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA.

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.	20
Name	**NIDHI GAWDE**
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

Experiment 09

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

Theory:

Service Worker

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

Things to note about Service Worker:

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

Fetch Event

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

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

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

But if this process fails, we check whether the request has been cached before or not. If a cache exists, it is returned to the page, but if not, this is up to you. You can return dummy content or information messages to the page.

```
self.addEventListener("fetch", function (event) {
    const req = event.request;
    const url = new URL(req.url);

    if (url.origin === location.origin) {
        event.respondWith(cacheFirst(req));
    }
    else {
        event.respondWith(networkFirst(req));
    }
});

async function cacheFirst(req) {
    return await caches.match(req) || fetch(req);
}

async function networkFirst(req) {
    const cache = await caches.open("pwa-dynamic");
    try {
        const res = await fetch(req);
        cache.put(req, res.clone());
        return res;
    } catch (error) {
        const cachedResponse = await cache.match(req);
        return cachedResponse || await caches.match("./noconnection.json");
    }
}
```

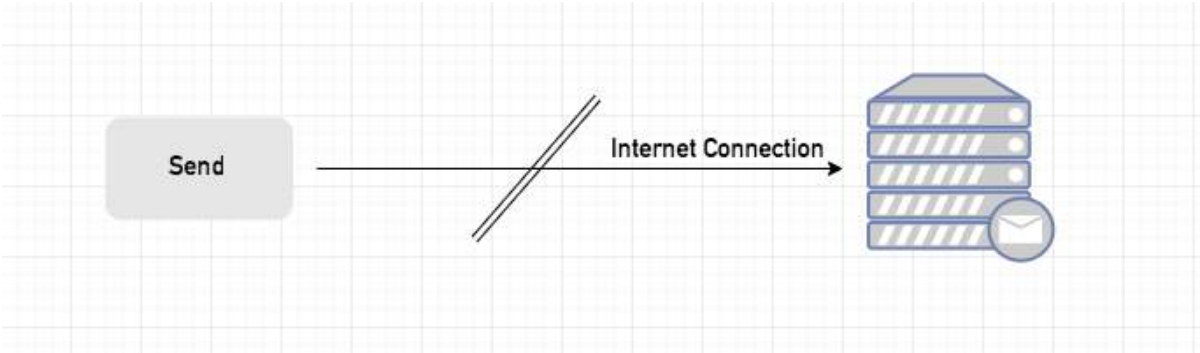
Sync Event

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

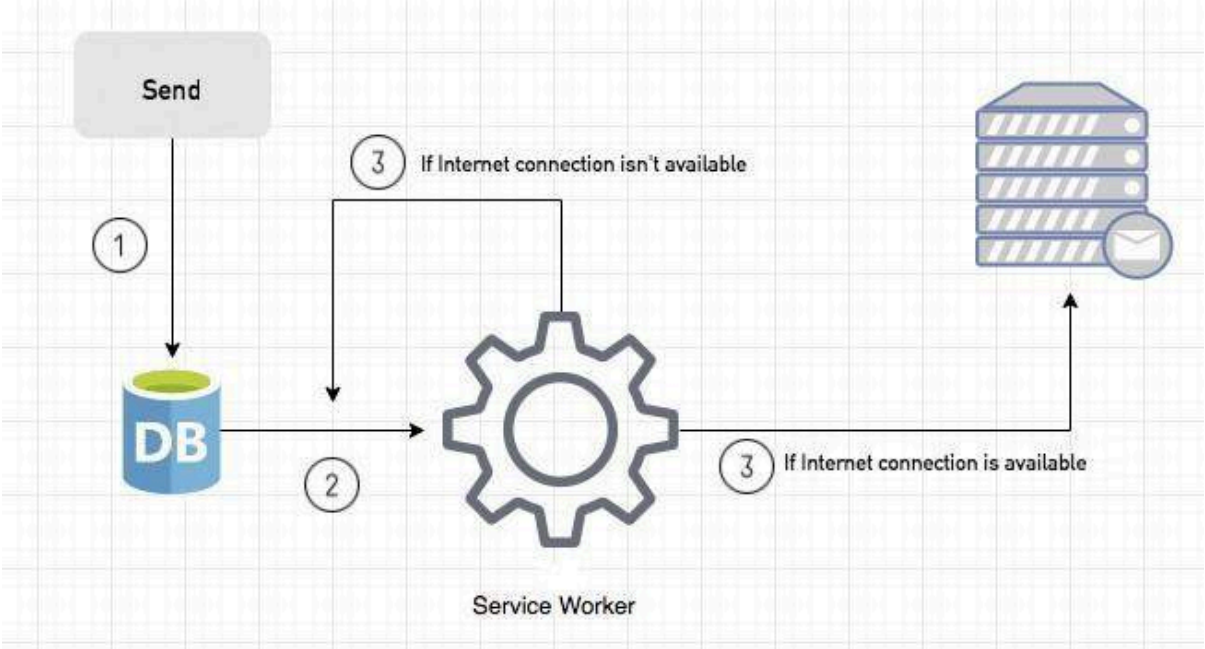
click the send button.

Here is a job for the Background Sync.

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



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



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

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

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

Event Listener for Background Sync Registration

```
document.querySelector("button").addEventListener("click", async () => {
    var swRegistration = await navigator.serviceWorker.register("sw.js");
    swRegistration.sync.register("helloSync").then(function () {
        console.log("helloSync success [main.js]");
    });
});
```

Event Listener for sw.js

```
self.addEventListener('sync', event => {
    if (event.tag == 'helloSync') {
        console.log("helloSync [sw.js]");
    }
});
```

Push Event

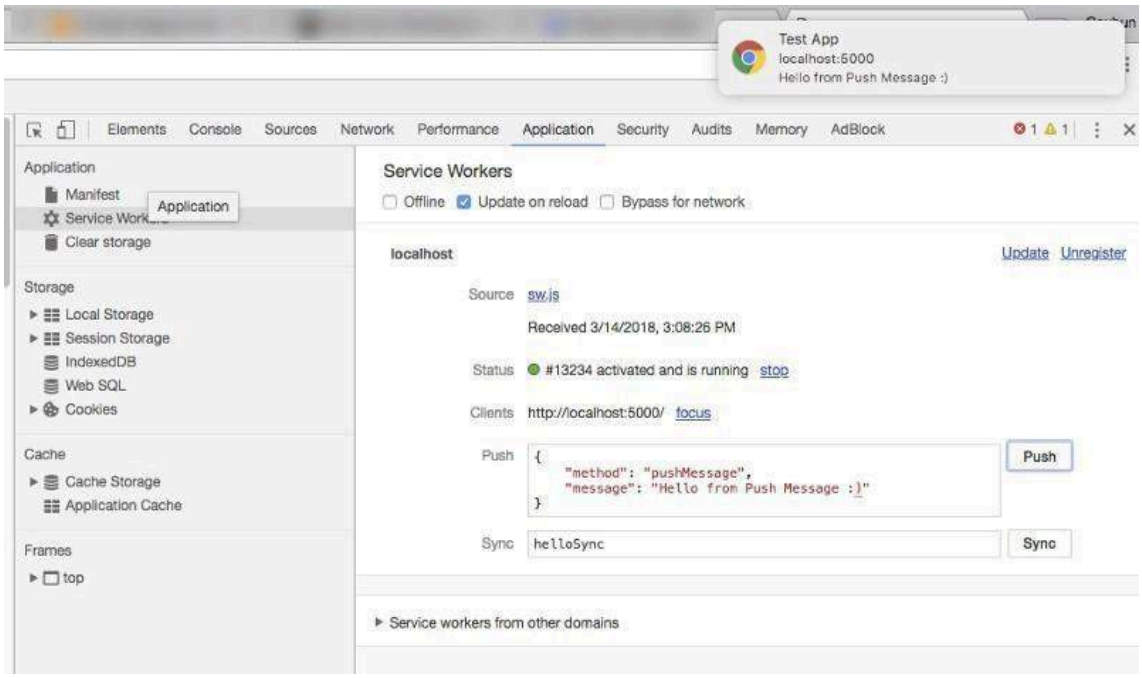
This is the event that handles push notifications that are received from the server. You can apply any method with received data.

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.

```
self.addEventListener('push', event => {
    if (event && event.data) {
        var data = event.data.json();
        if (data.method === "pushMessage") {
            event.waitUntil(self.registration.showNotification("Test App", {
                body: data.message
            }));
        }
    }
});
```

You can use Application Tab from Chrome Developer Tools for testing push notification.



code:

<u>serviceworker.js:</u>

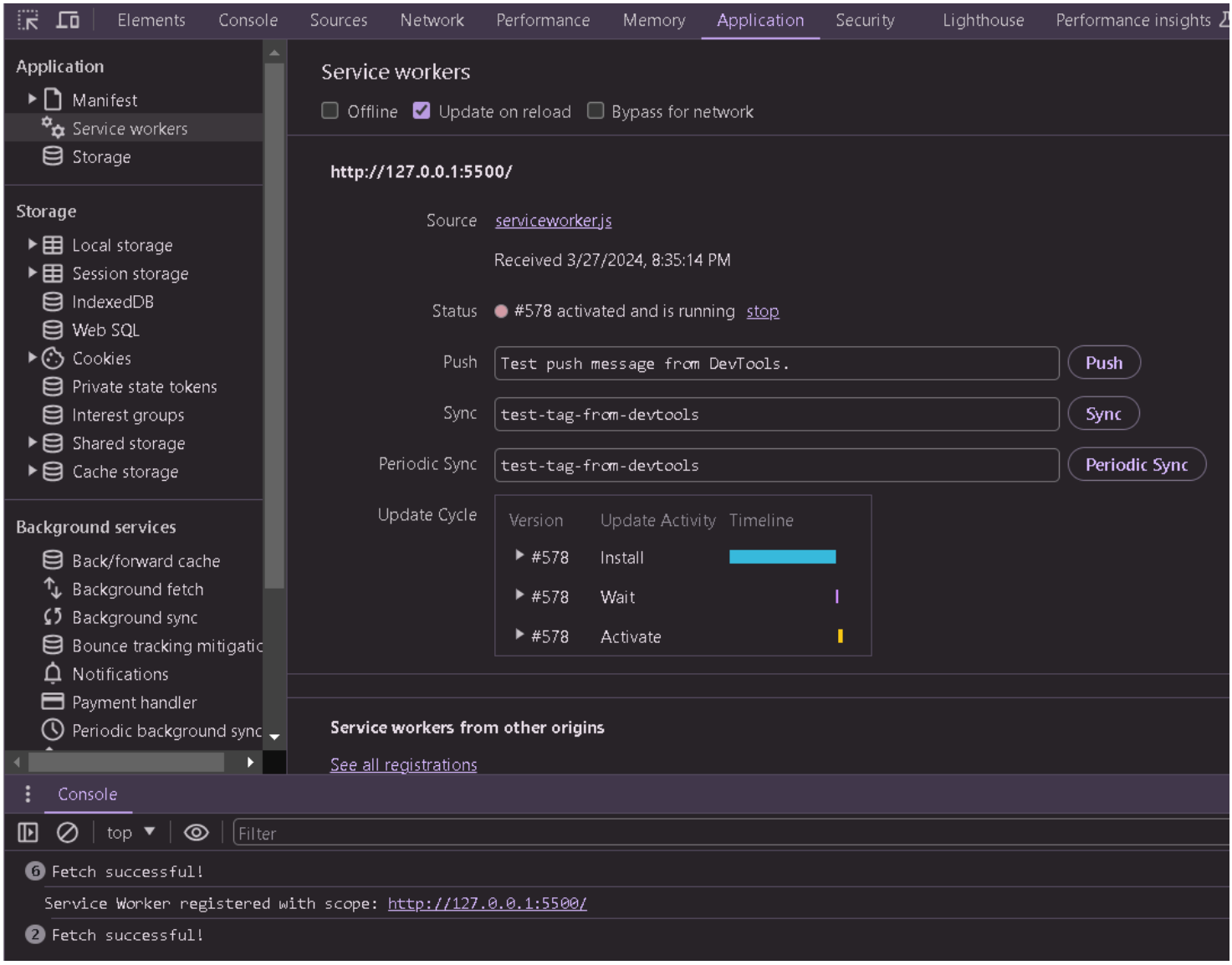
```
self.addEventListener("install", function (event) {
   event.waitUntil(preLoad());
   });

//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.requesst));
   });

//Sync event listener
   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("Avengers!!!Assemble", {
    body: data.message,
    });
    }
    } catch (error) {
    console.error("Error parsing push data:", error);
    }
    }
    });
    var preLoad = function () {
    return caches.open("offline").then(function (cache) {
    // caching index and important routes
    return cache.addAll([
        '/',
        '/index.html',
        '/netflixstyles.css',
        '/app.js',
        '/icon-192x192.jpg',
        '/icon-512x512.jpg',
        '/manifest.json',
        '/serviceworker.js'
    ]);
    });
    };
    var checkResponse = function (request) {
    return new Promise(function (fulfill, reject) {
    fetch(request)
    .then(function (response) {
    if (response.status !== 404) {
    fulfill(response);
    } else {
    reject(new Error("Response not found"));
    }
    })
    .catch(function (error) {
    reject(error);
    });
    });
```

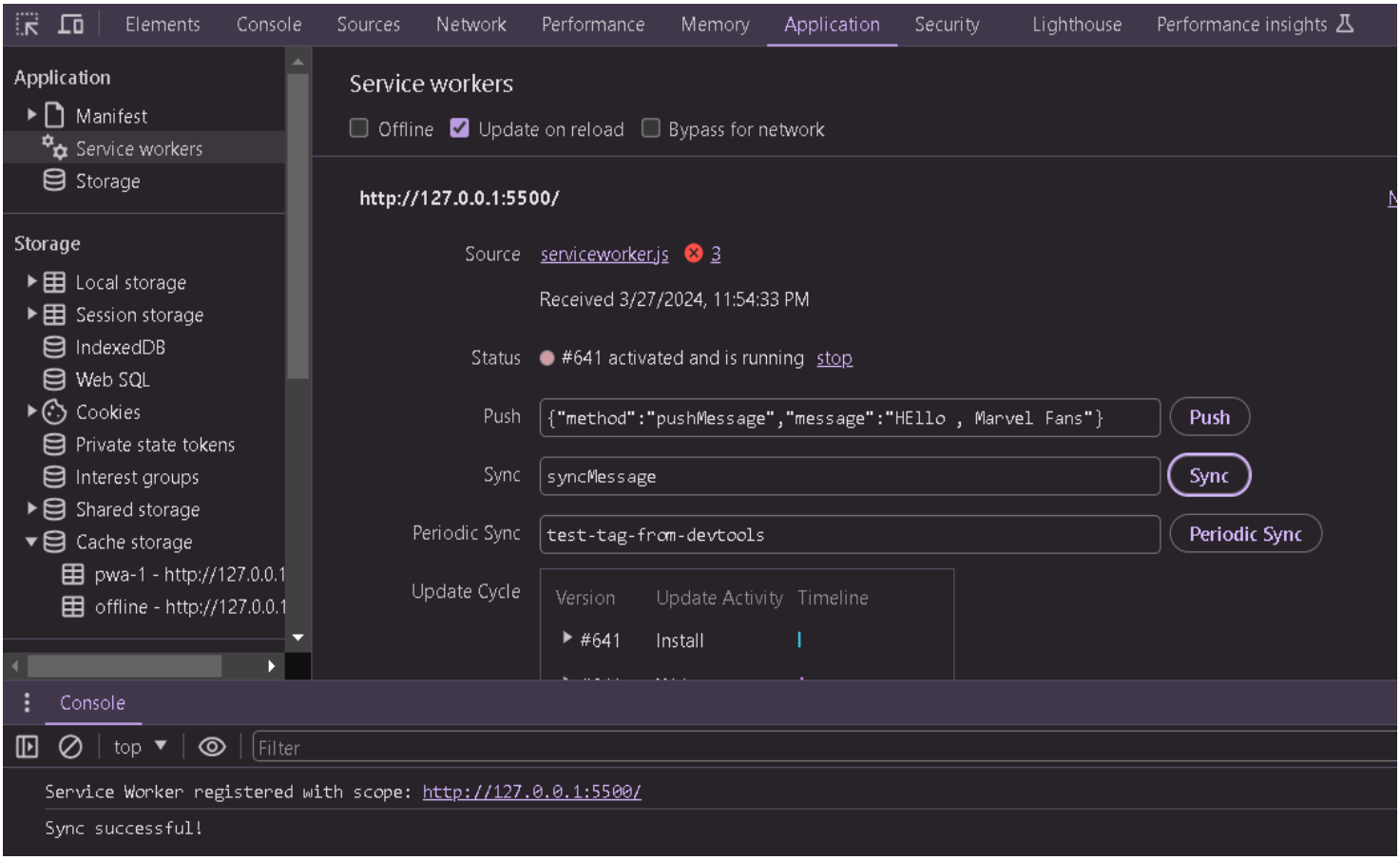
```
};
var returnFromCache = function (request) {
return caches.open("offline").then(function (cache) {
return cache.match(request).then(function (matching) {
if (!matching || matching.status == 404) {
return cache.match("offline.html");
} else {
return matching;
}
});
});
};
var addToCache = function (request) {
return caches.open("offline").then(function (cache) {
return fetch(request).then(function (response) {
return cache.put(request, response.clone()).then(function () {
return response;
});
});
});
};
```

Output:

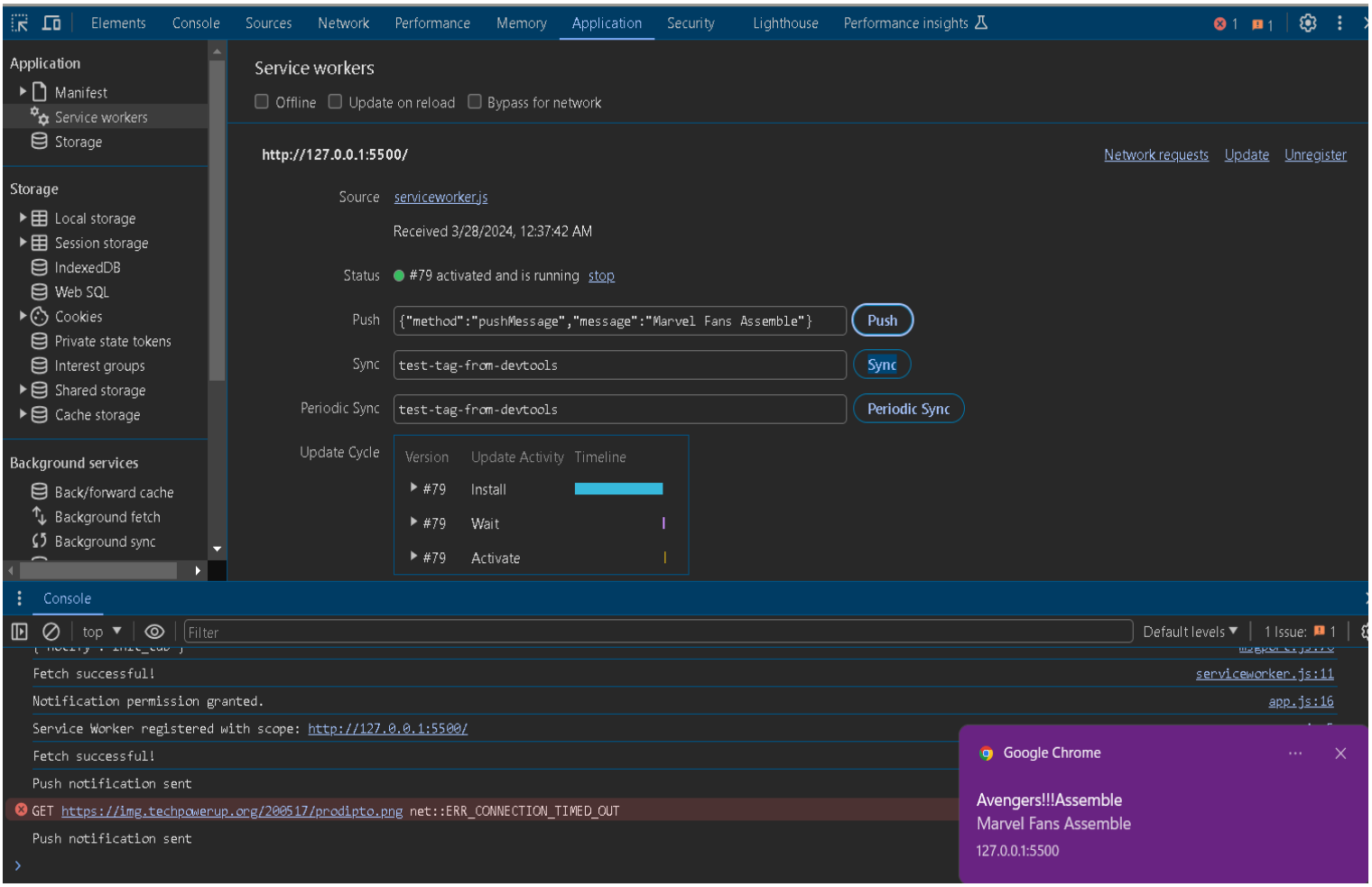
1. **Fetch**

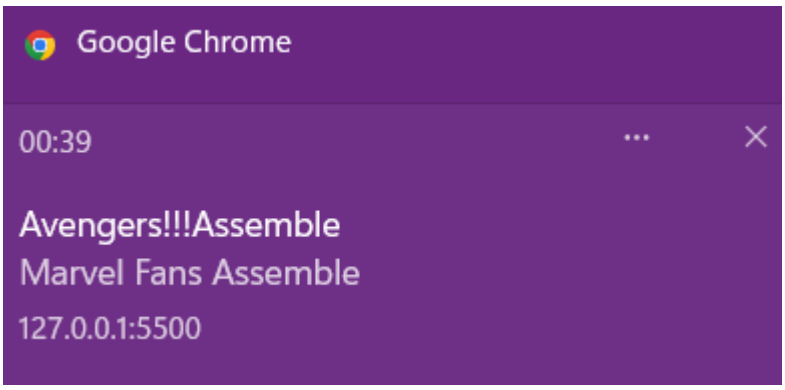

Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Performance insights
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
IndexedDB
Web SQL
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking mitigatic
Notifications
Payment handler
Periodic background sync
Service workers
Offline
Update on reload
Bypass for network
http://127.0.0.1:5500/
Source
serviceworker.js
Received 3/27/2024, 8:35:14 PM
Status
#578 activated and is running
stop
Push
Test push message from DevTools.
Push
Sync
test-tag-from-devtools
Sync
Periodic Sync
test-tag-from-devtools
Periodic Sync
Update Cycle
Version
Update Activity
Timeline
#578
Install
#578
Wait
#578
Activate
Service workers from other origins
See all registrations
Console
top
Filter
Fetch successful!
Service Worker registered with scope: http://127.0.0.1:5500/
Fetch successful!

2. **Sync**



3. Push Notification:





Conclusion:

Successfully implemented Service worker events like fetch, sync and push for E-commerce PWA.

MAD & PWA Lab
Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	20
Name	**NIDHI GAWDE**
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

Experiment 11

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

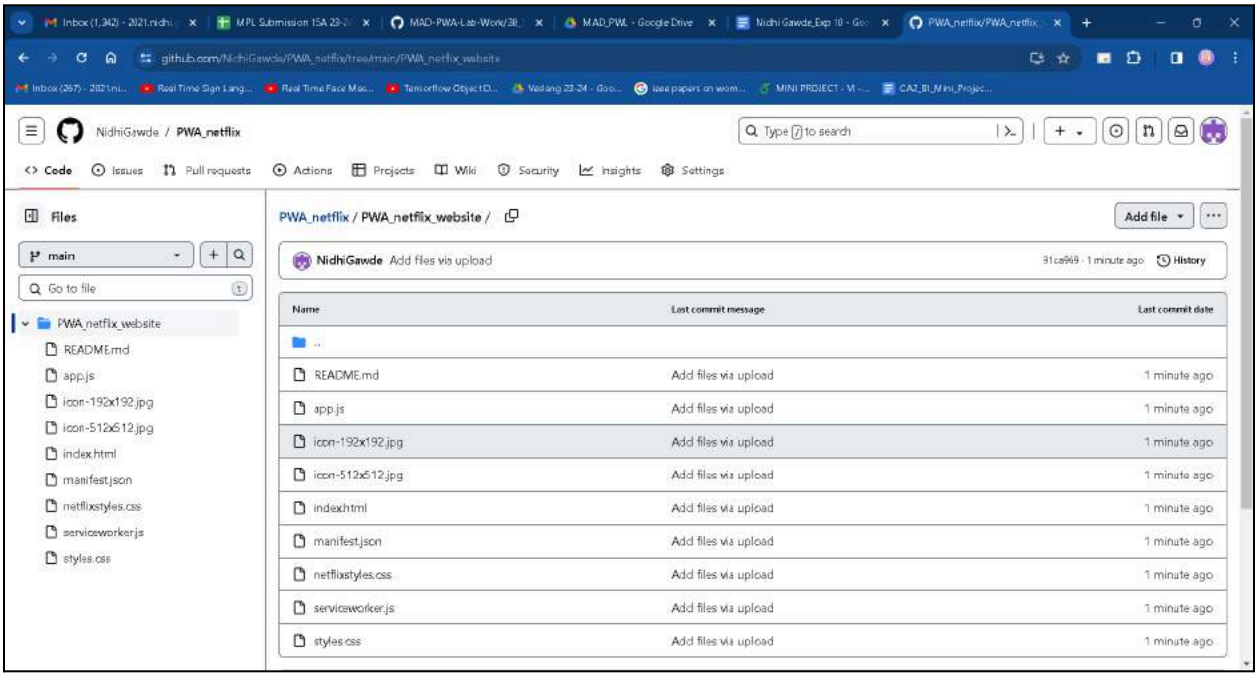
1. Hosted by Google. Enough said.
2. Authentication, Cloud Messaging, and a whole lot of other handy services will be available to you.
3. A real-time database will be available to you, which can store 1 GB of data.
4. You'll also have access to a blob store, which can store another 1 GB of data.
5.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

Cons

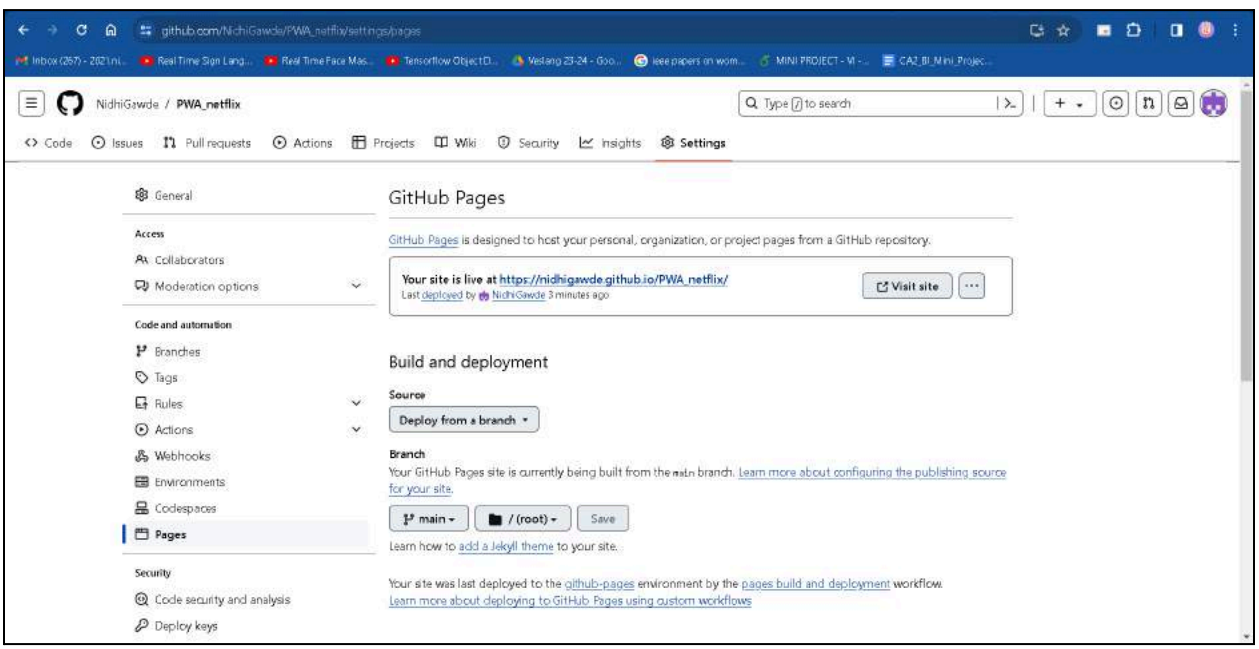
1. Only 10 GB of data transfer is allowed per month. But this is not really a big problem, if you use a CDN or AMP.
2. Command-line interface only.
3. No in-built support for any static site generator.

GITHUB REPOSITORY : PWA_website

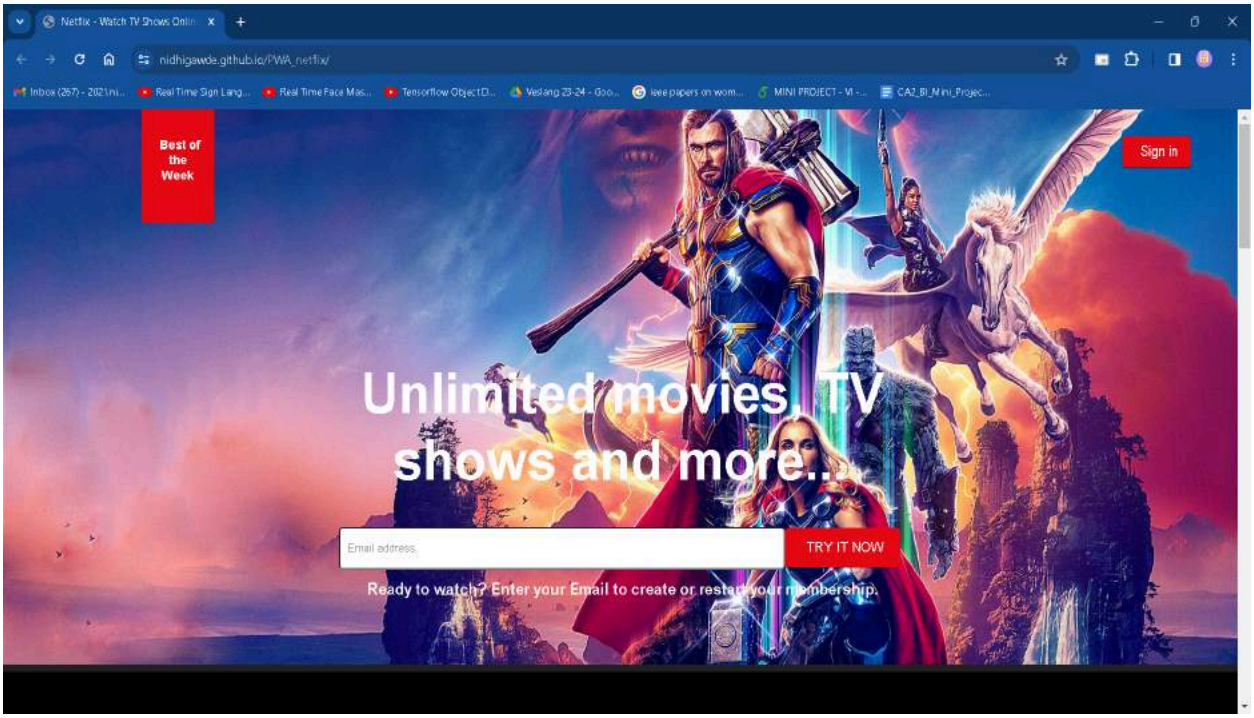
Project Title: Flutter - LinkedIn Clone / PWA-Netflix Website **Roll No.: 20**



Hosted Website: https://nidhigawde.github.io/PWA_netflix/



Project Title: Flutter - LinkedIn Clone / PWA-Netflix Website **Roll No.: 20**



Conclusion : Successfully implement deployment of Ecommerce PWA to GitHub Pages.

MAD & PWA Lab
Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	20
Name	**NIDHI GAWDE**
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	15

Experiment 11

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

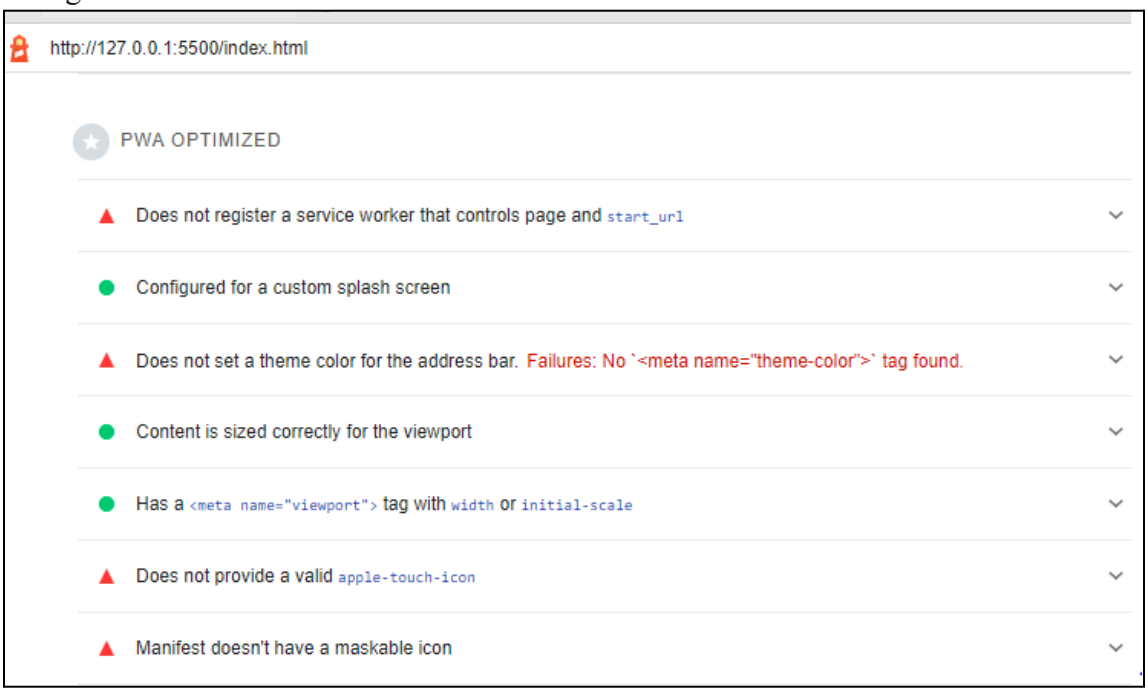
Google Lighthouse has the option of running the Audit for Desktop as well as mobile version of your page(s). The top metrics that will be measured in the Audit are:

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

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

4. **Best Practices:** As any developer would know, there are a number of practices that have been deemed 'best' based on empirical data. This metric is an aggregation of many such points, including but not limited to:Use of HTTPS Avoiding the use of deprecated code elements like tags, directives, libraries, etc. Password input with paste-into disabled Geo-Location and cookie usage alerts on load, etc.

Changes made to the code :



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

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

For a maskable icon add "purpose": "any maskable" to the icons in

manifest.json file For apple touch icon add the following meta tag in

index.html-

Changes in 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":"app ui ecom/assets/img/aclogoshop.png",

        "sizes":"192x192",
        "type":"image/png",
        "purpose":"maskable"
        },
        {
                "src":"app ui ecom\\assets\\img\\company6.png",

                "sizes":"512x512",
                "type":"image/png",
                "purpose":"any"
                }

]
```

Project Title: Flutter - LinkedIn Clone / PWA-Netflix Website **Roll No.: 20**

Generate a Lighthouse report

Analyze page load

Mode Learn more

- (●) Navigation (Default)
- ( ) Timespan
- ( ) Snapshot

Device

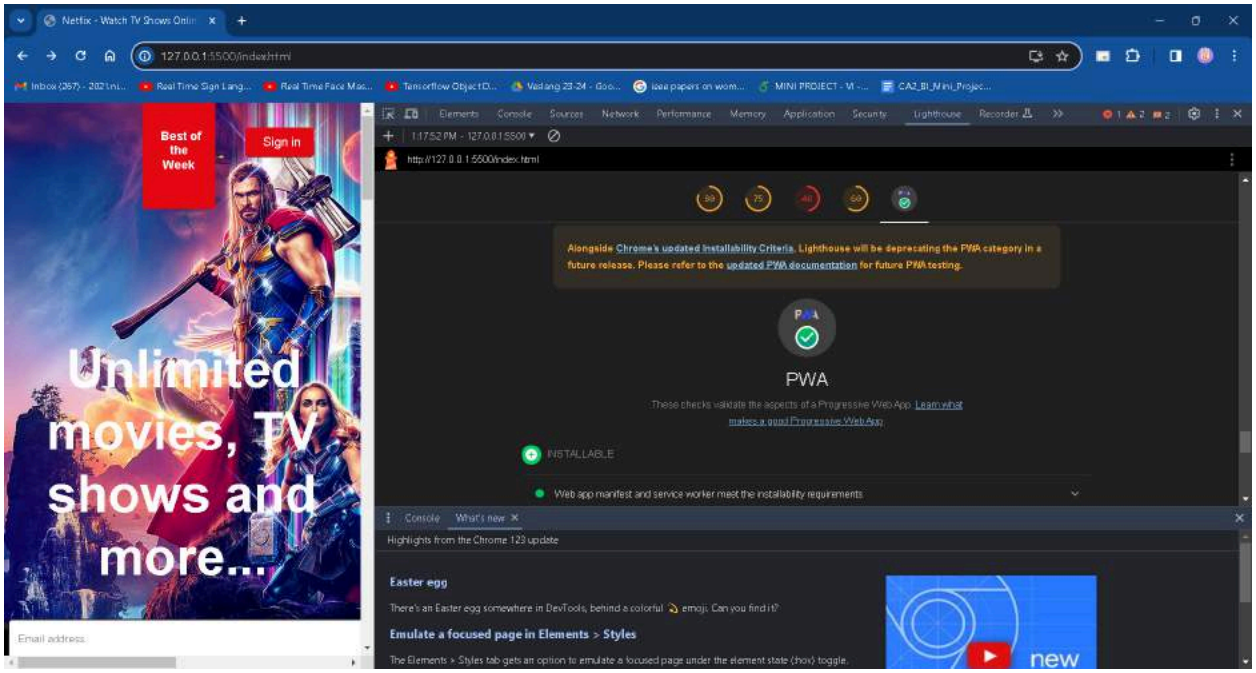
- (●) Mobile
- ( ) Desktop

Categories

- [x] Performance
- [x] Accessibility
- [x] Best practices
- [x] SEO
- [x] Progressive Web App

Plugins

- [ ] Publisher Ads



Generate a Lighthouse report

Analyze page load

Mode Learn more

- (●) Navigation (Default)
- ( ) Timespan
- ( ) Snapshot

Device

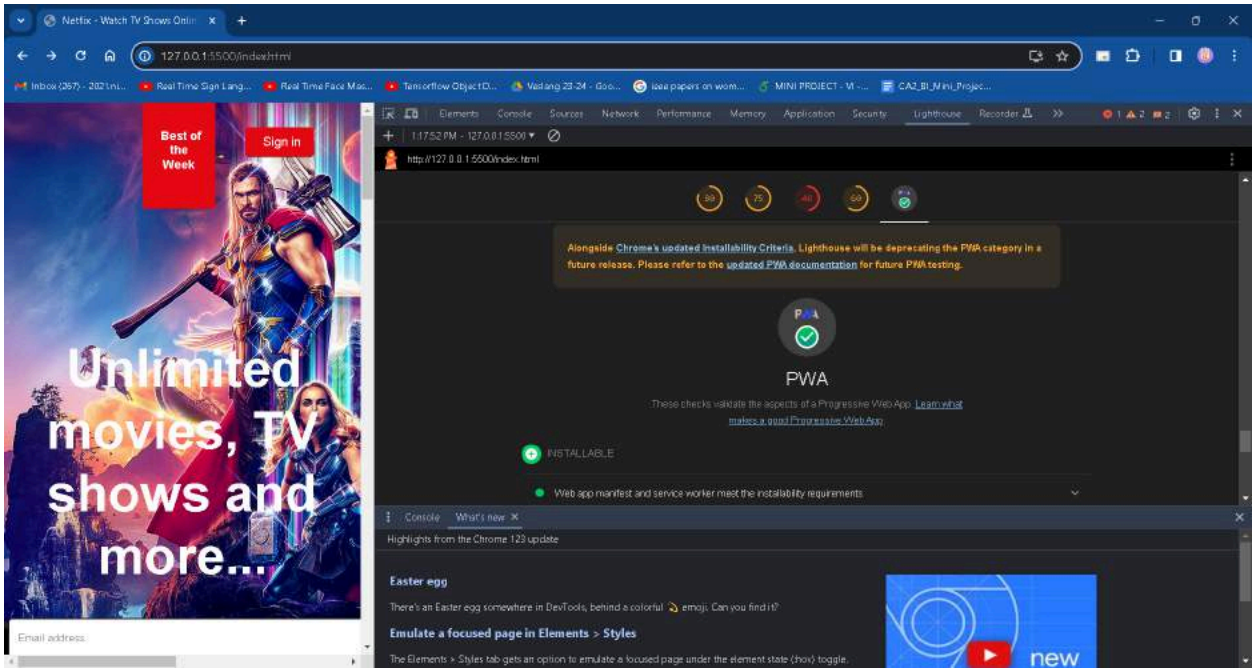
- ( ) Mobile
- (●) Desktop

Categories

- [x] Performance
- [x] Accessibility
- [x] Best practices
- [x] SEO
- [x] Progressive Web App

Plugins

- [ ] Publisher Ads



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

MAD & PWA Lab
Journal

Experiment No.	Assignment-1
Assignment 1 Questions	1. Flutter Overview:Explain the key features and advantages of using Flutter for mobile app development. Discuss how the Flutter framework differs from traditional approaches and why it has gained popularity in the developer community. 2. Widget Tree and Composition: Describe the concept of the widget tree in Flutter. Explain how widget composition is used to build complex user interfaces. Provide examples of commonly used widgets and their roles in creating a widget tree. 3. State Management in Flutter: Discuss the importance of state management in Flutter applications. Compare and contrast the different state management approaches available in Flutter, such as setState, Provider, and Riverpod. Provide scenarios where each approach is suitable. 4. Firebase Integration in Flutter: Explain the process of integrating Firebase with a Flutter application. Discuss the benefits of using Firebase as a backend solution. Highlight the Firebase services commonly used in Flutter development and provide a brief overview of how data synchronization is achieved.
Roll No.	20
Name	**NIDHI GAWDE**
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:	05

NIDHI GAWDE
D15A
20

DATE: 5/2/24.

Flutter

ASSIGNMENT NO: 01

Q.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.

→ Key features of flutter:

- Single codebase for multiple platforms -
  Flutter allows developers to write code and deploy it on both iOS and Android platforms.
- Hot Reload -
  This enables developers to instantly see the results of the code changes they make.
- Expressive UI -
  Developers have the flexibility to create expressive and flexible UIs.
- Integration with other tools -
  Flutter can easily integrate with other popular development tools and frameworks.

→ Advantages of Flutter

- Faster development
  Uses single codebase for multiple platforms.
- Consistent UI Across Platforms
  Widgets provide a consistent look and feel across different platforms.
- Cost - Efficiency
  Developing and maintaining single codebase for both iOS and Android reduces development cost and resources.

DATE:

→ Differ from Traditional Approach -

- Traditional approach uses a hierarchical structure for UI components, whereas Flutter uses a widget-based approach.
- Flutter compiles to native ARM code, providing performance comparable to native applications.
- Hot Reloads allows to see changes made instantly

Flutter's popularity is driven by increased productivity, a growing community, flexibility in UI design, cross-platform development capabilities and adoption by major companies.

Q.2] Widget Tree and Compositions. Describe the concept of 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.

→ Widget Tree

- The widget tree is a hierarchical structure of widgets that defines the user interface of an application.
- Every visual element, from simple components to complex layouts, is represented by a widget.
- Widget can be categorized in 2 types -
  - Stateless widget
    It is immutable and cannot change over time.
    Eg: images, text

DATE:

- Stateful widget
  Widget that can change its state over time.
  eg: buttons, forms

→ Widget Composition
- Widget composition in Flutter involves combining multiple simple widgets to create more complex and compound widgets.
- This composability is a powerful concept that allows developers to build sophisticated user interfaces by nesting widgets within each other.

→ Commonly used Widgets
- Container
  A box model for padding, margin and decoration.
- Column and Row
  Layout widgets for arranging children vertically or horizontally.
- Stack
  Overlapping widgets, allowing them to be layered on top of each other.
- List View
  A scrollable list of widgets.
- Grid View
  A scrollable grid of widgets
- AppBar
  A material design app bar typically at top of screen.
- Text field
  An input field for users to enter text.

DATE:

- Button widgets
  Interactive buttons for user actions.

Q.3] State Management in Flutter
Discuss the importance of state management in flutter applications. Compare and contrast the difference in state management approaches available in Flutter, such as setState, Provider and Riverpod. Provide scenario where each approach is suitable

→ • State management is crucial in Flutter applications because it involves managing the data that can change over time.
- Flutter, is reactive, meaning the UI rebuilds when the underlying data changes.

Set State	Provider	Riverpod
• Built-in flutter method	External package named 'provider'	External package 'riverpod'
• Local state within a widget	Global state within widget tree	Global state with additional features
• Limited scalability for large apps	Suitable for medium sized apps	Designed for large and complex apps
• May lead to code redundancy	Balances simplicity and readability	Emphasizes readability and clean syntax.

DATE:

• Testing can be more challenging	Good testability support	Enhances testing experience.
• Widely used, well established	Widely adopted well supported	Gaining popularity growing community

Scenario where each is applicable -

1. Set State
   - For small to moderately complex applications
   - when managing local state within a widget.

   Eg: Simple forms, UI components with local UI - specific state.

2. Provider
   - For medium to large-sized applications.
   - When a centralized state is needed, accessible by multiple widgets.

   Eg: Managing user authentication, theme changes or app-wide configuration.

3. Riverpod
   - For large and complex applications.
   - when testability and maintainability are top priorities.

   Eg: Complex applications with multiple feature dynamic UIs.

DATE:

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.

Integration

go to Firebase console and create new project

Add firebase SDK by including dependencies in pubspec.yaml

dependencies:

firebase_core : ^version

firebase_auth : ^version

cloud_firestore : ^version

Run flutter pub get.

Initialise firebase by calling

firebase.initialiseApp() in main

```
import package.firebase_core/ firebase_core.dart

void main() async {
    widgetsFlutterBinding.ensureInitialized();
    await Firebase.initializeApp();
    runApp (MyApp());
}
```

DATE:

Benefits of using Firebase as Backend

- Real-time Database
  Firebase offers a real-time NoSQL database.
- Authentication
  Provides a secure and easy-to-implement solution for user authentication.
- Cloud firestore
  Firebase's Cloud firestore provides a secure and easy-to-implement scalable NoSQL database that allows you to store and sync data in real time.
- Hosting
  Firebase Hosting provides a simple and efficient way to deploy and host web applications.

Data Synchronization

- Real-time Database
  When data changes on one client, it triggers events that automatically update data on other clients.
- Cloud Firestore
  It notifies clients when data changes, allowing for seamless real-time updates.
- Authentication
  If user sign in or out on one device, the authentication state is automatically reflected on other devices.

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.	20
Name	**NIDHI GAWDE**
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

NIDHI GAWDE
D15A
20.

DATE: 22/3/24

PWA MAD LAB

PWA ASSIGNMENT

Q.1] Define Progressive web App (PWA) & explain its significance in modern web development. Discuss the key characteristics that differentiate PWAs from traditional mobile apps.

→ A Progressive web App (PWA) is a type of web application that utilizes modern web capabilities to deliver an app-like experience to users.

→ PWA are designed to work on any device that uses a standard compliant browser, including desktops, laptops tablets & smartphones.

→ They leverage technologies such as HTML, CSS & JS to create fast, reliable and engaging user experiences.

• Significance:

→ Cross-platform compatibility: PWAs are built using web technologies, making them compatible with various platforms & devices

→ Responsive Design: PWAs - responsive by nature, seamless adaption to different screen sizes & orientations ensuring consistent user experience across devices improving accessibility & usability.

→ Offline Functionality: Allows offline work; uses service workers to cache content and assets.

→ App-like Experience: it has features such as push-notifications, home-screen installation, etc.

→ Improved Performance: optimized performance with faster loading times and smoother navigation.

→ Security: served over HTTPS ensuring communication is encrypted.

DATE:

- Key characteristics of differentiation:

→ Installation: PWAs can directly be ~~used~~ installed directly from browser without need for app stores.

→ Updates - PWAs are updated automatically, ensuring access to latest version of app without manual download

→ Resource Consumption: Typically consume less device storage compared to native apps since they don't require large downloads or installations.

→ Platform Independence: PWAs are platform independence and run in any modern ~~all~~ web browser, eliminating the need ~~of~~ to develop seperate versions of different platforms like IOS, Android.

Q.2] Define responsive web design & explain its significance in context of PWA. Compare & contrast responsive fluid, and adaptive web design approaches.

→ • Responsive web design:

→ It is an approach in web development that aims to create applications that provide an optimal viewing experience across a wide range of devices and screen sizes.

→ Primary goal - ensure that ~~layat~~ layout and content of a website adapt dynamically to the size and orientation of the user's devices.

→ IMPORTANCE:

→ Responsive web design in context of PWA is significant due to multi~~path~~device nature of PWAs.

→ Since, PWAs are designed to work seamlessly across various platforms and screen sizes, responsive design ensure that user experience remains consistent and user-friendly regardless of the devices used.

DATE:

→ This is crucial for maintaining engagement and accessibility especially considering that PWAs are often accessed on mobile devices with different screen resolution & aspect ratios.
→ Ensures consistency across different devices, maintaining user experience standards.
→ Improving accessibility allowing users to access PWAs on any device without encountering usability issues.
→ Optimizing Performance by reducing the need ~~of~~ for unnecessary downloads and minimizing rendering times on different devices.

	Responsive Web Design	Fluid Web Design	Adaptive Web Design.
Definition:	Uses flexible grids, layout and media queries to adapt layout & content of a website dynamically based on size ~~of~~ and orientation of user's device.	→ It focuses on creating websites with flexible layouts that adjust to width of browser window rather than specific device sizes.	→ It involves creating multiple layouts or versions of a website tailored to specific device category or screen sizes.
Features:	Fluid grids & flexible images allow content to resize to fit to screen sizes.	→ Elements resize smoothly as the browser window is resized.	→ Uses server-side detection to deliver most appropriate layout or version of website based on user device characteristics.
Advantages:	Provides a consistent user experience across devices, ~~imp~~	offers great flexibility in accomodating various screen sizes	Allows for precise optimization of user experience for different

DATE:

	improves accessibility	and resolutions, this maintaining proportions and alignment.	devices improving performance & usability.
Disadvantages	Requires more development effort and may result in complex CSS code.	→ May require additional testing to ensure consistent performance across different browsers & devices.	→ Requires more upfront planning & development effort to create & maintain multiple versions of website.

Q.3] Describe the lifecycle of service workers, including registration installation and activation phases.

i) Registration:

→ This phase begins when the service worker script is registered with browser using 'navigator.serviceWorker.register()'

→ During registration, the browser parses the serviceworker script and checks for syntax errors. If registration is successful, browser starts the ~~browser~~ process of installing service worker in the background.

→ Registration defines scope of the service worker which determines the URLs for which the serviceworker will control network requests and handle events.

(ii) Installation:

→ After successful registration, browser intiates installation phase by downloading & caching service worker script.

→ It allows service worker to perform tasks such as caching ~~state~~ assets, setting up event listeners and preparing for activation.

DATE:

ii) Activation :

→ After installation, the service worker moves to activation phase.

→ This occurs after currently active instances of the service worker have been terminated.

→ During activation, the browser fires an 'activate' event in service worker script which allows the service worker to clean up any outdated caches, update existing resources and perform necessary housekeeping tasks.

→ After activation, service worker becomes fully operational and can intercept network requests, handle push notifications and execute background synchronization tasks as defined in its scripts.

Q.4] Explain the use of Indexed DB in the service worker for data storage.

→ Indexed DB : powerful client-side storage mechanism available to web browsers.

→ It provides a way for web applications to store large amounts of structured data locally, allowing them to work offline, improve performance and provide more responsive user experience.

• Indexed DB utilization w.r.t service worker.

~~→ Offline Data storage~~ -

i) Offline Data storage -

→ It stores data locally for offline access allowing interception of network requests and cache responses.

→ It allows web application to serve content from local database when the user is offline ensuring that application remains functional in absence of network connection.

DATE:

(ii) Caching Dynamic Content :

→ service workers can dynamically cache data retrieve from network using Indexed DB

→ This enables web application to provide a faster and more responsive user experience. by serving cached content from local database instead of making repeated network requests.

(iii) Background Data synchronization :

→ Indexed DB can be used in conjuction with background data synchronization in service worker.

→ This allows web application to periodically sync data synchronization in service worker.

(iv) Structured Data storage :

→ Indexed DB provides a structured storage mechanism that allows web application to store data in key: value pairs.

(v) Asynchronous Data Operations :

→ Indexed DB operations are asynchronous ie ~~do~~ they do not block the main thread of the web application

→ service workers can perform database operations in background, allowing web application.