Vivekanand Education Society's Institute of Technology

An Autonomous Institute Affiliated to University of Mumbai
Hashu Advani Memorial Complex, Collector Colony, Chembur East, Mumbai - 400074.

Department of Information Technology

CERTIFICATE

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

Lab Assistant

Subject Teacher

Mrs. Kajal Joseph

Principal

Head of Department

Dr. Mrs. Shalu Chopra

Name of the Course : MAD & PWA Lab

Course Code : ITL604

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

Faculty Incharge : Mrs. Kajal Joseph.

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

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

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

Program specific Outcomes

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

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

Project Title: Flutter:Netflix Movie App/ PWA:Glowing **Roll No. 49**

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	19/01	26/01	15
2.	To design Flutter UI by including common widgets.	LO2	26/01	02/02	15
3.	To include icons, images, fonts in Flutter app	LO2	02/02	09/02	15
4.	To create an interactive Form using form widget	LO2	09/02	16/02	15
5.	To apply navigation, routing and gestures in Flutter App	LO2	16/02	23/02	15
6.	To Connect Flutter UI with fireBase database	LO3	23/02	08/03	15
7.	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.	LO4	08/03	15/03	15
8.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA	LO5	15/03	22/03	15
9.	To implement Service worker events like fetch, sync and push for E-commerce PWA	LO5	22/03	31/03	15
10.	To study and implement deployment of Ecommerce PWA to GitHub Pages.	LO5	29/03	31/03	15
11.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.	LO6	29/03	31/03	15
12.	Assignment-1	LO1,LO2 ,LO3	28/01	05/02	5
13.	Assignment-2	LO4,LO5 ,LO6	14/03	21/03	4

MAD & PWA Lab
Journal

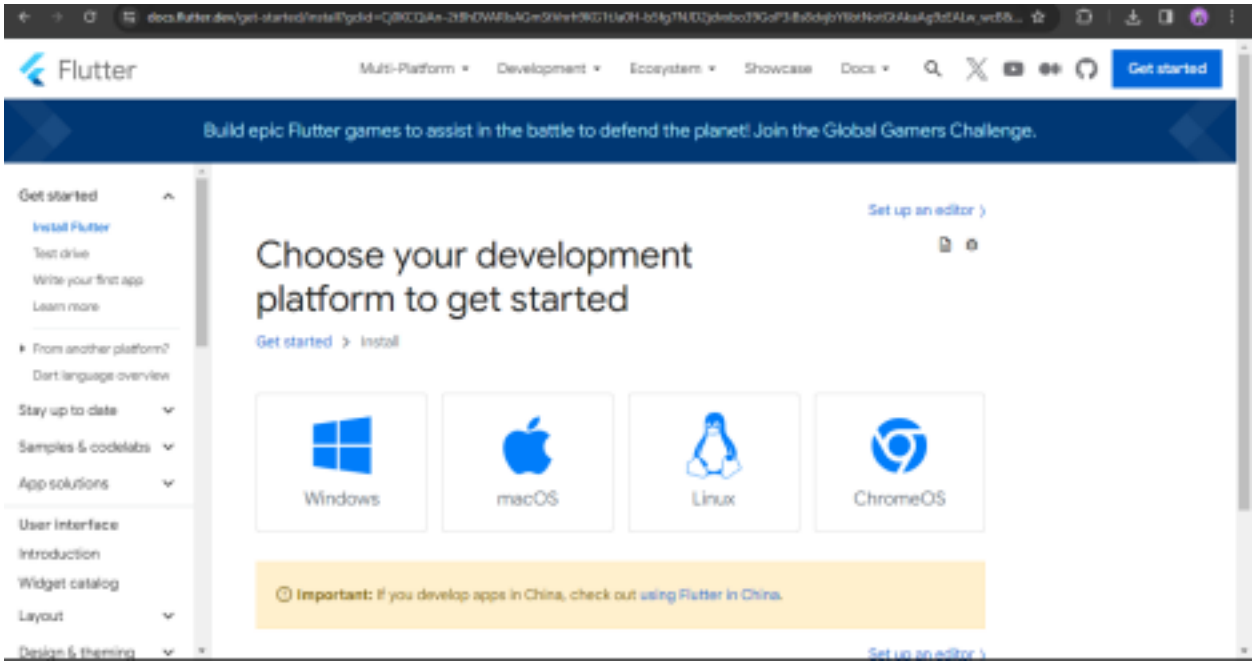
Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	49
Name	Gaurang A Roarane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	15

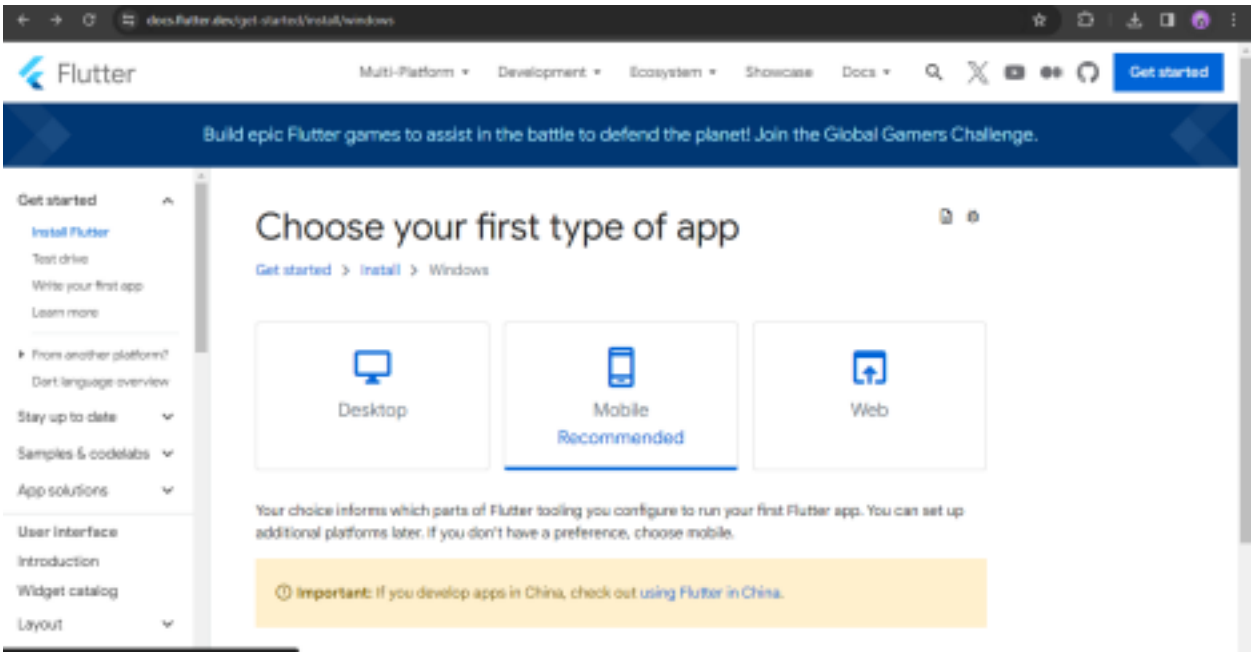
Experiment - 1

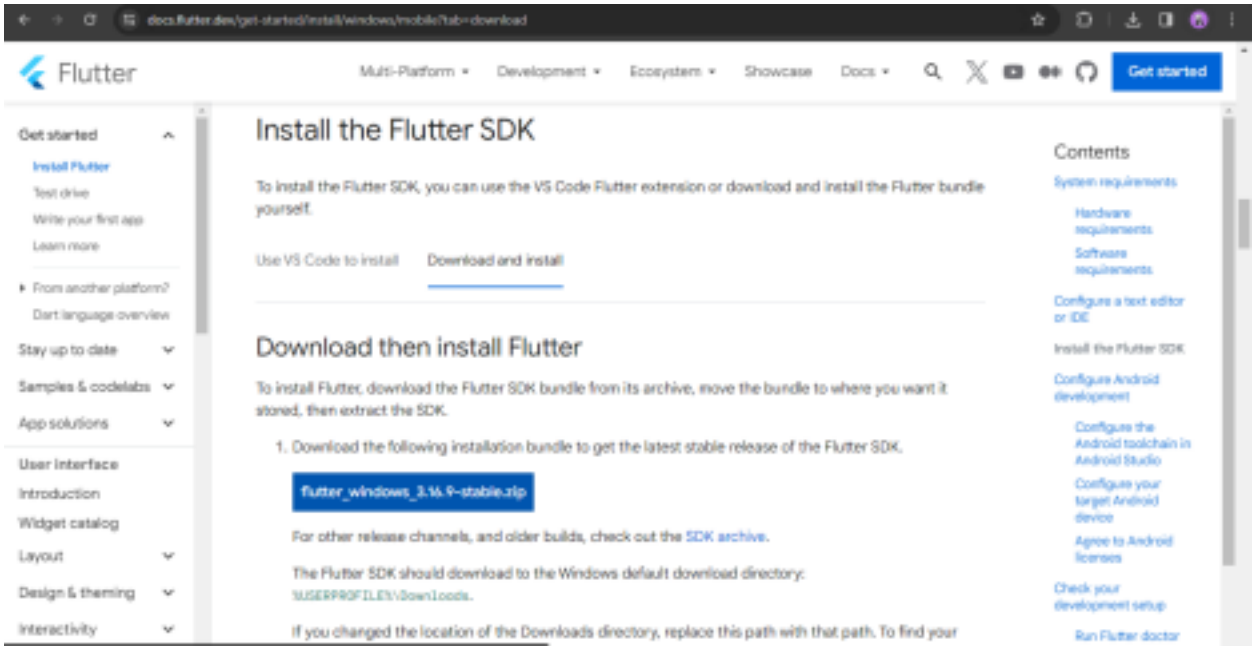
Aim - Installation and Configuration of Flutter Environment.

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

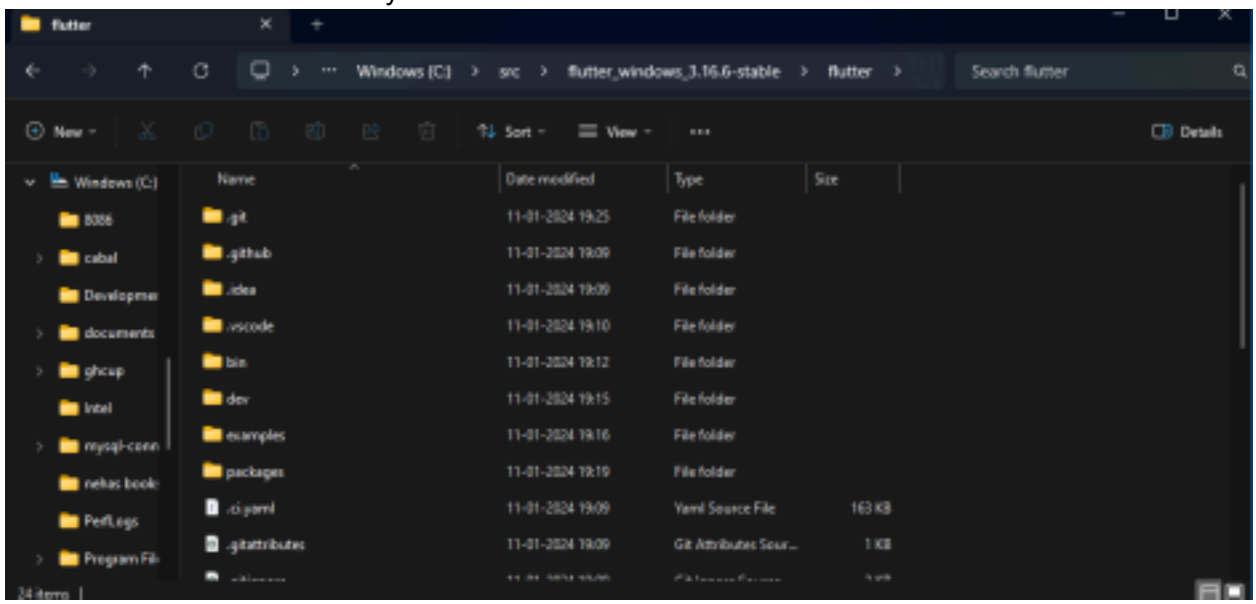
Installation -



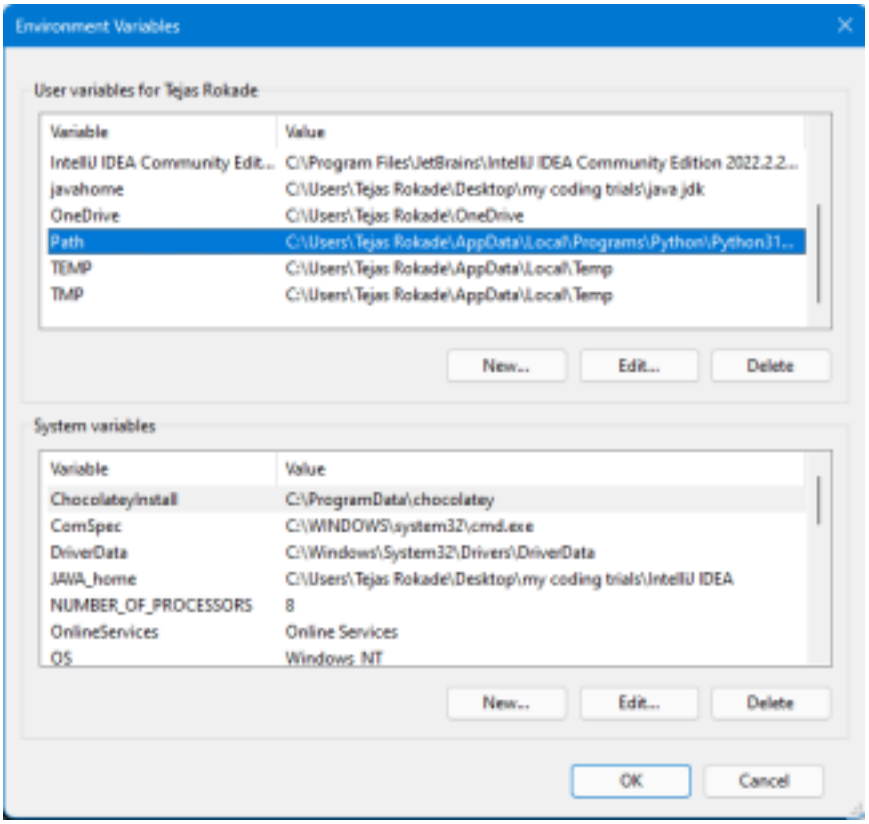




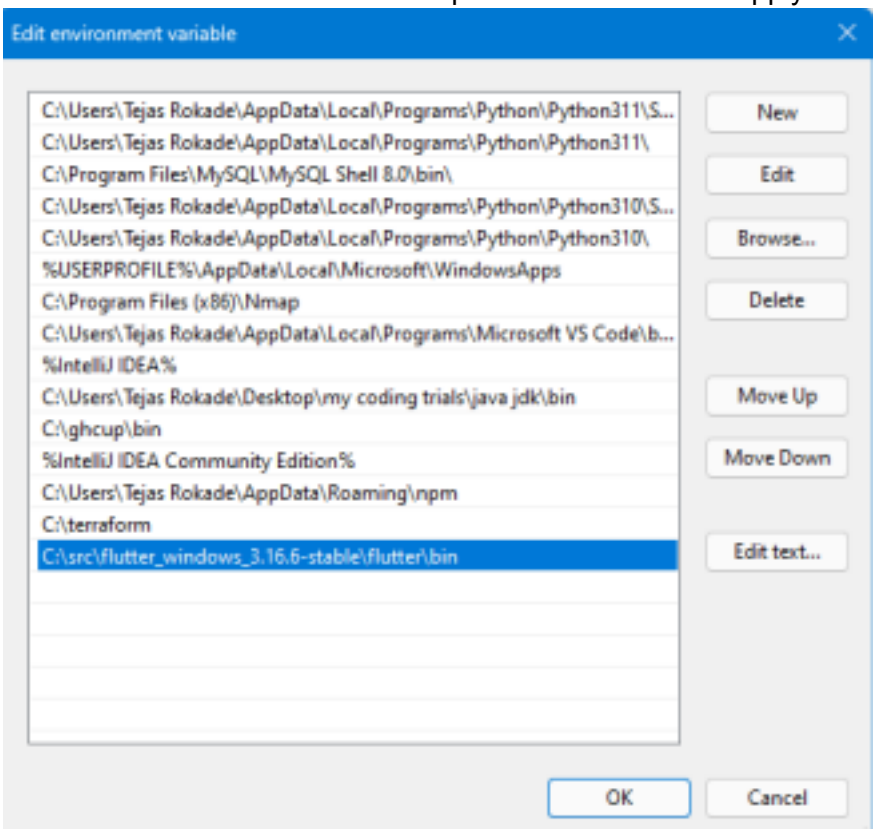
Extract the files in the directory -



Setup path in Environment variables -



Create new variable and insert the path to bin folder and apply -



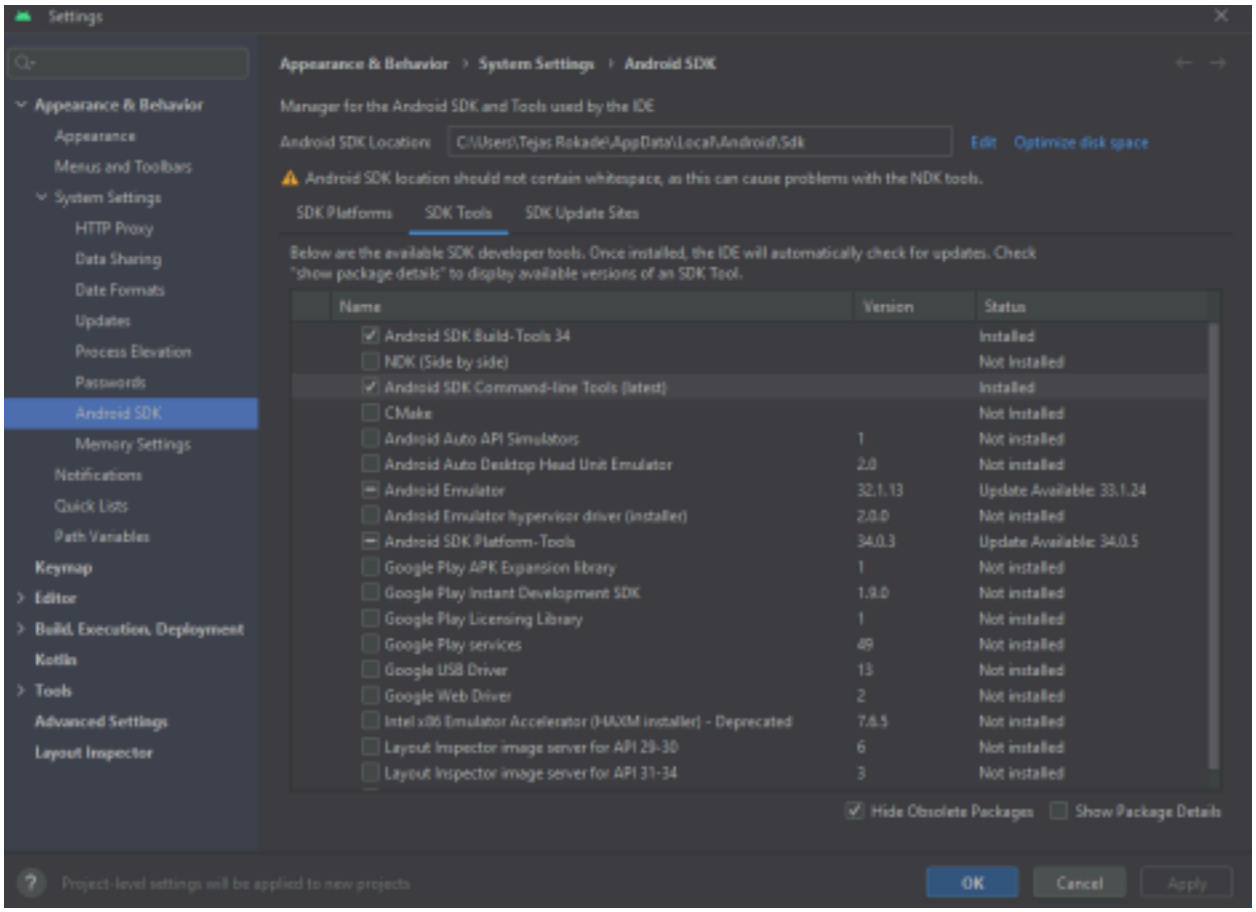
Then on CMD -

```
C:\Users\Tejas Rokade>flutter
Manage your Flutter app development.

Common commands:

  flutter create <output directory>
    Create a new Flutter project in the specified directory.

  flutter run [options]
    Run your Flutter application on an attached device or in an emulator.

Usage: flutter <command> [arguments]

Global options:
-h, --help                  Print this usage information.
-v, --verbose               Noisy logging, including all shell commands executed.
                            If used with "--help", shows hidden options. If used with "flutter d
                            diagnostic information. (Use "-vv" to force verbose logging in those
-d, --device-id             Target device id or name (prefixes allowed).
    --version               Reports the version of this tool.
```

```
C:\Users\Tejas Rokade>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.16.6, on Microsoft Windows [Version 10.0.22621.3807], locale en-IN)
[√] Windows Version (Installed version of Windows is version 10 or higher)
[√] Android toolchain - develop for Android devices (Android SDK version 34.0.0)
[√] Chrome - develop for the web
[√] Visual Studio - develop Windows apps (Visual Studio Build Tools 2019 16.11.24)
[√] Android Studio (version 2022.2)
[√] VS Code (version 1.85.2)
[√] Connected device (3 available)
[√] Network resources

• No issues found!

C:\Users\Tejas Rokade>
```

The error for Android toolchain will occur if havent installed android SDK Command Line Tools in Android Studios SDK Tools.

And IF issue of android licenses occurs run Flutter –android-licenses command in the prompt and type y+enter till process is complete

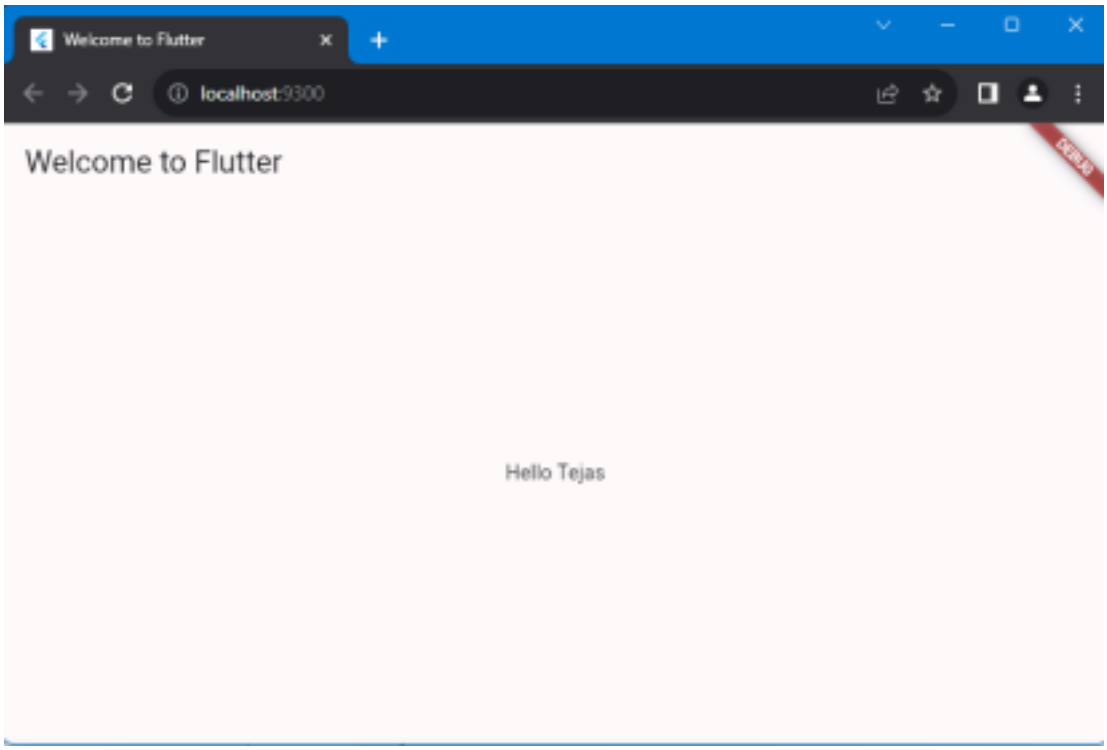


Code -

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

```
}
```

Output -


Welcome to Flutter
localhost:9300
Welcome to Flutter
DEBUG
Hello Tejas

MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	49
Name	Gaurang A Roarane
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 - 2

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

Theory :-

In Flutter, widgets are the basic building blocks for creating user interfaces. They represent the visual elements in the UI, such as buttons, text fields, and containers. Here's a theoretical overview of some common Flutter widgets that you might encounter in a laboratory (lab) setting:

Container Widget:

- Theory: The Container widget is a versatile and powerful widget that can contain other widgets. It allows customization of dimensions, padding, margin, decoration, and more.
- Lab Use: Use Container to structure and style the layout of your lab interface. Adjust its properties to control spacing and alignment.

Text Widget:

- Theory: The Text widget displays a string of text with various styling options. It is used to present information to the user.
- Lab Use: Utilize Text to display important information, labels, or instructions within your lab app.

Row and Column Widgets:

- Theory: Row and Column are layout widgets that allow you to arrange child widgets in a horizontal (row) or vertical (column) sequence.
- Lab Use: Organize and structure your UI elements in a row or column format for better readability and organization.

ListView Widget:

- Theory: ListView is a scrollable list of widgets. It is used to display a scrolling, linear list of widgets.
- Lab Use: Implement ListView to display a list of items, such as experiment steps, results, or any dynamic data.

AppBar Widget:

- Theory: The AppBar widget provides a top app bar with options for navigation, title, and actions.
- Lab Use: Include an AppBar to give your lab app a consistent and recognizable navigation structure.

Form and TextFormField Widgets:

- Theory: Form and TextFormField are used for creating forms and handling user input.
- Lab Use: Implement a form structure using Form and capture user input with TextFormField for data entry and interaction.

Button Widgets (ElevatedButton, TextButton, and OutlinedButton):

- Theory: Flutter provides various button widgets, each with its visual style and behavior.

- Lab Use: Use buttons like ElevatedButton, TextButton, and OutlinedButton for actions such as submitting data, navigating, or triggering experiments.

Image Widget:

- Theory: The Image widget displays an image from various sources, such as assets or the internet.
- Lab Use: Use the Image widget to show diagrams, graphs, or other visual content in your lab app.

AlertDialog and SnackBar Widgets:

- Theory: AlertDialog and SnackBar are used for displaying pop-up messages to the user.
- Lab Use: Provide important notifications, alerts, or feedback to users through these pop-up widgets.

Scaffold Widget:

- Theory: Scaffold is a basic skeletal structure that contains the visual elements of a material design app.
- Lab Use: Use Scaffold as the overall structure of your lab app, providing a consistent layout with an app bar, body, and other elements.

Input :-

```
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:osiris/Services/auth.dart';
import 'package:osiris/routes.dart';
import 'package:provider/provider.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(const App());
}

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

  @override
  State<App> createState() => _AppState();
}

class _AppState extends State<App> {
  @override
  Widget build(BuildContext context) {
```

```
    return ChangeNotifierProvider(
      create: (context) => GoogleSignInProvider(),
      child: MaterialApp.router(
        debugShowCheckedModeBanner: false,
        routerConfig: router,
      ),
    );
  }
}
```

MainScreen.dart

```
import 'package:flutter/material.dart';
import 'package:flutter/rendering.dart';
import 'package:osiris/Models/PopularMovies.dart';
import 'package:osiris/Models/TvShow.dart';
import 'package:osiris/Services/API.dart';
import 'package:osiris/Services/consts.dart';
import 'package:osiris/Widgets/BottomNavBar.dart';
import 'package:osiris/Widgets/CarouselCard.dart';
import 'package:osiris/Widgets/CustomLists.dart';
import 'package:osiris/Widgets/LoadingScreen.dart';
import 'package:osiris/Widgets/SectionText.dart';

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

  @override
  State<MainScreen> createState() => _MainScreenState();
}

class _MainScreenState extends State<MainScreen> {
  ScrollController _scrollController = ScrollController();
  bool isVisible = true;

  late List<Results> popularMovie;
  late List<Results> topRatedMovie;
  late List<Results> nowPLayingMovie;
  late List<TvShow> popularShows;
  late List<TvShow> topRatedShows;
  bool isLoading = true;

  @override
  void initState() {
    super.initState();
```

```
  fetchData();
  _scrollController = ScrollController();
  _scrollController.addListener(listen);
 }

 @override
 void dispose() {
  _scrollController.removeListener(listen);
  _scrollController.dispose();
  super.dispose();
 }

 void listen() {
  final direction = _scrollController.position.userScrollDirection;
  if (direction == ScrollDirection.forward) {
    show();
  } else if (direction == ScrollDirection.reverse) {
    hide();
  }
 }

 void show() {
  if (!isVisible) {
    (setState(
      () => isVisible = true,
    ));
  }
 }

 void hide() {
  if (isVisible) {
    (setState(
      () => isVisible = false,
    ));
  }
 }

 Future<void> fetchData() async {
  topRatedShows = await APIService().getTopRatedShow();
  popularMovie = await APIService().getPopularMovie();
  topRatedMovie = await APIService().getTopRatedMovie();
  popularShows = await APIService().getRecommendedTvShows("1396");
  nowPLayingMovie = await APIService().getNowPLayingMovie();
  setState(() {
```

```
    isLoading = false;
  });
 }

 @override
 Widget build(BuildContext context) {
  var size = MediaQuery.of(context).size;
  return Scaffold(
    bottomNavigationBar: AnimatedBuilder(
        animation: _scrollController,
        builder: ((context, child) {
          return AnimatedContainer(
            duration: const Duration(milliseconds: 800),
            curve: Curves.fastLinearToSlowEaseIn,
            height: isVisible ? 75 : 0,
            child: BottomNavBar(
              currentIndex: 0,
            ),
          );
        })),
    extendBody: true,
    body: isLoading
        ? const LoadingScreen()
        : Container(
            height: size.height,
            width: size.width,
            color: background_primary,
            child: ListView(
              padding: EdgeInsets.zero,
              scrollDirection: Axis.vertical,
              physics: const BouncingScrollPhysics(),
              controller: _scrollController,
              shrinkWrap: true,
              children: [
                CustomCarouselSlider(topRatedShows),
                SectionText("Popular", "Movies"),
                CustomListMovie(popularMovie),
                SectionText("TOP Rated", "Movies"),
                CustomListMovie(topRatedMovie),
                SectionText("Popular", "Shows"),
                CustomListTV(popularShows),
                SectionText("NoW PLAying", "Movies"),
                CustomListMovie(nowPLayingMovie),
              ],
```

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

CarouselCard.dart

```
import 'package:cached_network_image/cached_network_image.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:go_router/go_router.dart';
import 'package:osiris/Models/TvShow.dart';
import 'package:osiris/Widgets/LandingCard.dart';

class CustomCarouselSlider extends StatelessWidget {
  CustomCarouselSlider(this.data, {super.key});
  List<TvShow> data;

  @override
  Widget build(BuildContext context) {
    var size = MediaQuery.of(context).size;
    return SizedBox(
        width: size.width,
        height: (size.height * 0.33 < 300) ? 300 : size.height * 0.33,
        child: PageView.builder(
          scrollDirection: Axis.horizontal,
          physics: const BouncingScrollPhysics(),
          pageSnapping: true,
          itemCount: 20,
          itemBuilder: ((context, index) {
            var url = data[index].backdropPath.toString();
            return GestureDetector(
              onTap: () {
                HapticFeedback.mediumImpact();
                GoRouter.of(context).push('/tv/${data[index].id}');
              },
              child: LandingCard(
                  CachedNetworkImageProvider(
                      "https://image.tmdb.org/t/p/original$url"),
                  data[index].name.toString()),
            );
          }),
        ));
  }
```

```
}
```

MovieCard.dart

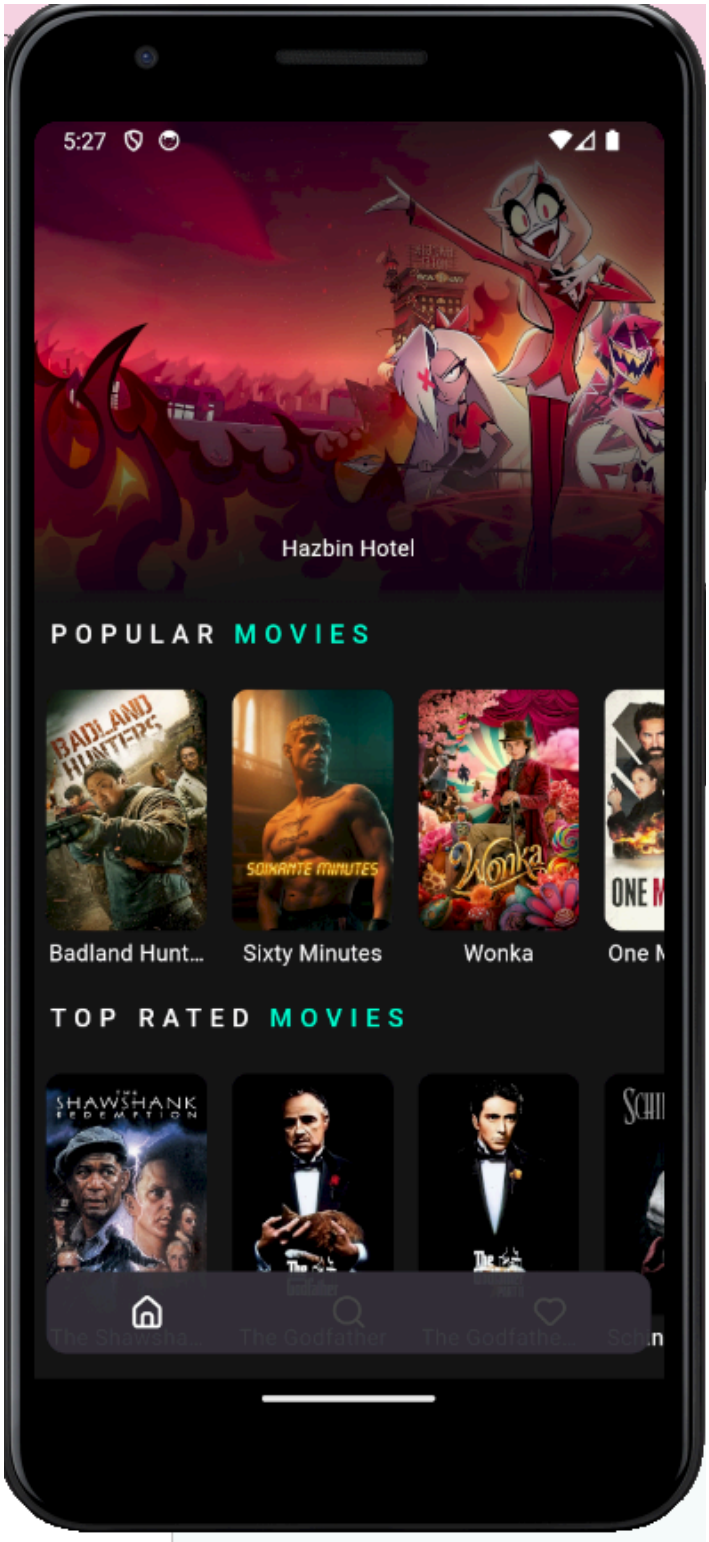
```
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:go_router/go_router.dart';

class MovieCard extends StatelessWidget {
  MovieCard(this.title, this.image, this.Id, this.mediaType, {super.key});
  String title;
  ImageProvider image;
  String Id;
  String mediaType;

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: () {
        HapticFeedback.mediumImpact();
        GoRouter.of(context).push('/$mediaType/$Id');
      },
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        crossAxisAlignment: CrossAxisAlignment.center,
        children: [
          Container(
            height: 150,
            width: 100,
            margin: const EdgeInsets.fromLTRB(8, 4, 8, 4),
            decoration: BoxDecoration(
              image: DecorationImage(fit: BoxFit.cover, image: image),
              borderRadius: BorderRadius.circular(10),
            ),
          ),
          SizedBox(
            width: 100,
            child: Text(
              title,
              style: const TextStyle(color: Colors.white),
              maxLines: 1,
              overflow: TextOverflow.ellipsis,
              textAlign: TextAlign.center,
            ),
          )
        ],
```

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

Pubspec.yaml

```
flutter:
  uses-material-design: true
  assets:
    - assets/

flutter_icons:
    android: "launcher_icon"
    image_path_android: "assets/icon.png"
    adaptive_icon_background: "assets/launcher_icon/background.png"
    adaptive_icon_foreground: "assets/launcher_icon/foreground.png"
```

Output:-


5:27
Hazbin Hotel
POPULAR MOVIES
Badland Hunt...
Sixty Minutes
Wonka
TOP RATED MOVIES

MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	49
Name	Gaurang A Roarane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

Experiment - 3

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

Theory:-

1) Icons:
   Icons are small, visual representations of actions, objects, or concepts. In Flutter, icons can be added using the Icon widget, which allows developers to incorporate a wide range of pre-built icons from the Material Icons library or other icon packs. To use icons, developers simply specify the icon data or icon name within the Icon widget.
2) Images:
   Images are graphical assets that can be used to convey information, illustrate content, or enhance the visual appeal of an app. Flutter provides support for displaying both local and network images using the Image widget. Local images, stored in the project's assets folder, can be imported and displayed using the AssetImage class. Network images can be loaded asynchronously using the NetworkImage class.
3) Fonts:
   Fonts define the visual appearance of text in an app and are instrumental in maintaining brand consistency and UI design. In Flutter, developers can incorporate custom fonts by adding font files (e.g., TrueType or OpenType) to the project and defining font families in the pubspec.yaml file. Once imported, custom fonts can be applied to text using the TextStyle widget by specifying the desired font family.

Integration Process:

Icons:

- Import the Material Icons package in the pubspec.yaml file.
- Use the Icon widget to add icons to Flutter widgets by specifying the icon data or icon name.

Images:

- Add image assets to the project's assets folder.
- Declare the image assets in the pubspec.yaml file.
- Display images using the Image widget and specifying the image asset path.

Fonts:

- Add custom font files (e.g., .ttf, .otf) to the project directory.
- Define font families in the pubspec.yaml file by specifying the font file paths.
- Apply custom fonts to text using the TextStyle widget and specifying the font family.

Best Practices:

- Optimize image assets to reduce file size and improve app performance.
- Use vector-based icons whenever possible to ensure scalability and resolution independence.
- Maintain consistency in font usage across the app to reinforce brand identity and enhance readability.

- Test the app on different devices and screen sizes to ensure proper rendering of icons, images, and fonts.
-

TVShowScreen.dart

```
import 'package:flutter/services.dart';
import 'package:intl/intl.dart';
import 'package:flutter/material.dart';
import 'package:osiris/Models/TvShow.dart';
import 'package:osiris/Services/API.dart';
import 'package:osiris/Services/consts.dart';
import 'package:osiris/Widgets/CustomLists.dart';
import 'package:osiris/Widgets/LoadingScreen.dart';
import 'package:osiris/Widgets/SeasonsList.dart';
import 'package:unicons/unicons.dart';
import 'package:osiris/Services/extraServices.dart';
import 'package:osiris/Widgets/DetailScreenComponents.dart';

class TVShowScreen extends StatefulWidget {
  TVShowScreen(this.movieId, {super.key});
  String movieId;

  @override
  State<TVShowScreen> createState() => _TVShowScreenState();
}

class _TVShowScreenState extends State<TVShowScreen> {
  bool isLoading = true;
  late List<TvShow> recommendedTvShows;

  Future<void> fetchData() async {
    recommendedTvShows =
        await APIService().getRecommendedTvShows(widget.movieId);
    setState(() {
      isLoading = false;
    });
  }

  @override
  void initState() {
    super.initState();
    fetchData();
  }

  var selectedSeason = 0;
```

```
@override
Widget build(BuildContext context) {
  var size = MediaQuery.of(context).size;
  return Scaffold(
    backgroundColor: background_primary,
    body: isLoading
        ? const LoadingScreen()
        : FutureBuilder(
            future: APIService().getTvShowDetail(widget.movieId),
            builder: (context, AsyncSnapshot snapshot) {
              if (snapshot.hasData) {
                var status = snapshot.data!.status.toString();
                var releaseDate = snapshot.data!.firstAirDate.toString();
                var seasonCount = snapshot.data!.numberOfSeasons;
                var seasonList = [];
                for (var i = 1; i <= seasonCount; i++) {
                  seasonList.add("Season $i");
                }
                return ListView(
                    scrollDirection: Axis.vertical,
                    physics: const AlwaysScrollableScrollPhysics(
                        parent: BouncingScrollPhysics()),
                    padding: EdgeInsets.zero,
                    shrinkWrap: true,
                    children: [
                      Stack(
                        children: [
                          Container(
                            width: size.width,
                            height: size.height * 0.40 > 300
                                ? size.height * 0.40
                                : 300,
                            decoration: BoxDecoration(
                              image: DecorationImage(
                                image: snapshot.data!.backdropPath == null
                                    ? const AssetImage(
                                          "assets/LoadingImage.png")
                                      as ImageProvider
                                    : NetworkImage(
                                        "https://image.tmdb.org/t/p/original${snapshot.data!.backdropPath}",
                                      ),
                                fit: BoxFit.cover,
                              ),
                            ),
```

```
),
Container(
  width: size.width,
  height: size.height * 0.40 > 300
      ? size.height * 0.40
      : 300,
  decoration: BoxDecoration(
    gradient: LinearGradient(
        begin: Alignment.topCenter,
        end: Alignment.bottomCenter,
        colors: [
          Colors.transparent,
          Colors.transparent,
          background_primary.withOpacity(0.50),
          background_primary.withOpacity(0.75),
          background_primary.withOpacity(0.85),
          background_primary.withOpacity(0.90),
          background_primary.withOpacity(0.95),
          background_primary.withOpacity(1.00)
        ]),
  ),
),
Container(
  width: size.width,
  height: size.height * 0.35 > 300
      ? size.height * 0.35
      : 300,
  margin: const EdgeInsets.all(8),
  child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    mainAxisAlignment: MainAxisAlignment.end,
    children: [
      Text(
        snapshot.data!.voteAverage
            .toString()
            .substring(0, 3),
        style: const TextStyle(
            fontSize: 40,
            fontWeight: FontWeight.w500,
            color: Colors.white),
      ),
      Text(
        snapshot.data!.name.toString(),
        overflow: TextOverflow.ellipsis,
```

```
                maxLines: 2,
                style: const TextStyle(
                    fontSize: 24,
                    fontWeight: FontWeight.w500,
                    color: Colors.white),
              ),
              Row(
                children: [
                  CircularButtons(
                    UniconsLine.play,
                    onTap: () {
                      HapticFeedback.lightImpact();
                      APIService()
                          .getTrailerLink(
                              snapshot.data!.id.toString(),
                              "tv")
                          .then(
                              (value) => LaunchUrl(value));
                    },
                  ),
                  CircularButtons(
                    UniconsLine.plus,
                    onTap: () {
                      HapticFeedback.lightImpact();
                      pshowDialog(
                          context, widget.movieId, "tv");
                    },
                  ),
                  Visibility(
                    visible: snapshot.data!.adult,
                    child: CircularButtons(
                      UniconsLine.eighteen_plus,
                      onTap: () {},
                    ),
                  ),
                ],
              )
            ],
          ),
        )
      ],
    ),
    FutureBuilder(
      future:
```

```
                    APIService().getMovieGenres(widget.movieId, "tv"),
                builder: (context, AsyncSnapshot snapshot) {
                  if (snapshot.hasData) {
                    return Container(
                      height: 36,
                      width: size.width,
                      margin: const EdgeInsets.only(left: 8),
                      child: ListView.builder(
                        scrollDirection: Axis.horizontal,
                        shrinkWrap: true,
                        physics: const BouncingScrollPhysics(),
                        itemCount: snapshot.data.length,
                        itemBuilder: (context, index) {
                          return TextContainer(
                              snapshot.data![index].name.toString(),
                              const EdgeInsets.only(right: 8),
                              const Color(0xFF14303B));
                        },
                      ),
                    );
                  } else {
                    return TextContainer(
                        "Loading",
                        const EdgeInsets.all(8),
                        const Color(0xFF14303B));
                  }
                },
              ),
              Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  TitleText("Status"),
                  Row(
                    children: [
                      TextContainer(
                          status,
                          const EdgeInsets.only(
                              left: 8, right: 8, bottom: 8),
                          const Color(0xFF382E39)),
                      TextContainer(
                          "Release:
${DateFormat.yMMMMd().format(DateTime.parse(releaseDate))}",
                          const EdgeInsets.only(
                              left: 8, right: 8, bottom: 8),
```

```
            const Color(0xFF545551)),
      ],
    ),
    TitleText("Overview"),
    TextContainer(
      snapshot.data!.overview.toString().isEmpty ||
          snapshot.data!.overview.toString() ==
              "null"
          ? "No overview available"
          : snapshot.data!.overview.toString(),
      const EdgeInsets.all(8),
      const Color(0xFF0F1D39)),
    TitleText("Seasons"),
    Container(
      height: 36,
      width: size.width,
      margin: const EdgeInsets.only(left: 8),
      child: ListView.builder(
        scrollDirection: Axis.horizontal,
        shrinkWrap: true,
        physics: const BouncingScrollPhysics(),
        itemCount: seasonCount,
        itemBuilder: (context, index) {
          return GestureDetector(
            onTap: () {
              HapticFeedback.lightImpact();
              setState(() {
                selectedSeason = index;
              });
            },
            child: TextContainer(
                seasonList[index].toString(),
                const EdgeInsets.only(right: 8),
                index == selectedSeason
                    ? const Color(0xFF545551)
                    : const Color(0xFF14303B)),
          );
        },
      ),
    ),
    SeasonsList(selectedSeason + 1, widget.movieId),
    TitleText("Recommendations"),
    CustomListTV(recommendedTvShows),
  ],
```

```
                    )
                ]);
            } else {
              return const LoadingScreen();
            }
          },
        ),
    );
  }
}
```

SectionText.dart

```
import 'package:flutter/material.dart';
import 'package:osiris/Services/consts.dart';

Widget SectionText(String ktitle, String ntitle) {
  return Container(
      width: double.infinity,
      margin: const EdgeInsets.all(8),
      child: Row(
        mainAxisAlignment: MainAxisAlignment.start,
        crossAxisAlignment: CrossAxisAlignment.center,
        children: [
          Padding(
            padding: const EdgeInsets.only(right: 8.0),
            child: Text(
              ktitle.toUpperCase(),
              style: const TextStyle(
                  color: Colors.white,
                  fontSize: 16,
                  fontWeight: FontWeight.w500,
                  letterSpacing: 5),
            ),
          ),
          Text(
            ntitle.toUpperCase(),
            style: TextStyle(
                color: accent_secondary,
                fontSize: 16,
                fontWeight: FontWeight.w500,
                letterSpacing: 5),
          ),
        ],
      ));
```

```
}
```

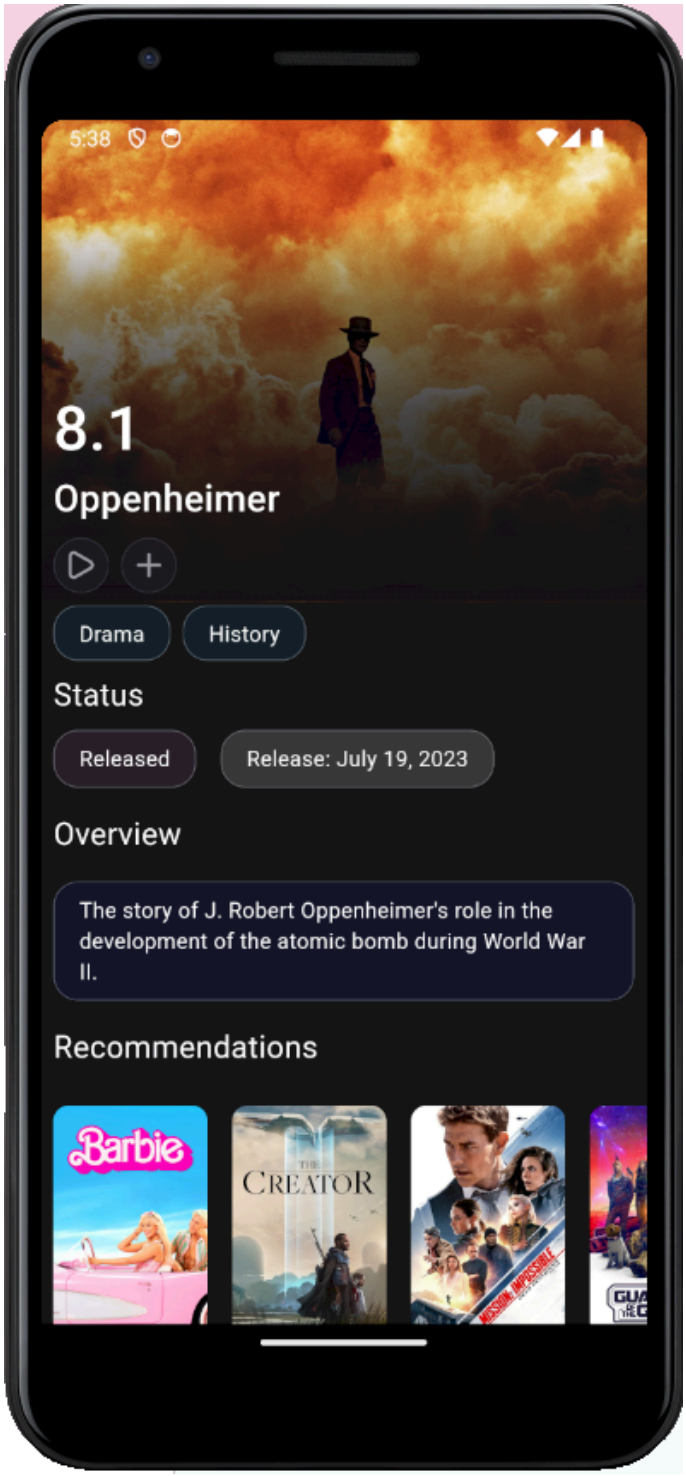
SeasonsList.dart

```
import 'package:flutter/material.dart';
import 'package:osiris/Services/API.dart';

class SeasonsList extends StatefulWidget {
  SeasonsList(this.seasonNumber, this.movieId, {super.key});
  int seasonNumber;
  String movieId;

  @override
  State<SeasonsList> createState() => _SeasonsListState();
}

class _SeasonsListState extends State<SeasonsList> {
  @override
  Widget build(BuildContext context) {
    return SizedBox(
      height: 150,
      child: FutureBuilder(
          future: APIService()
              .getEpisodes(widget.movieId, widget.seasonNumber.toString()),
          builder: (context, AsyncSnapshot snapshot) {
            if (snapshot.hasData) {
              return ListView.builder(
                scrollDirection: Axis.horizontal,
                shrinkWrap: true,
                physics: const AlwaysScrollableScrollPhysics(
                    parent: BouncingScrollPhysics()),
                itemCount: snapshot.data.length,
                itemBuilder: (context, index) {
                  var url = snapshot.data[index].stillPath.toString();
                  return Container(
                    margin: const EdgeInsets.all(10),
                    decoration: BoxDecoration(
                      color: const Color(0xFF14303B).withOpacity(0.25),
                      borderRadius: BorderRadius.circular(10),
                      border: Border.all(
                        color: const Color(0xFFA3A3B0).withOpacity(0.5),
                        width: 0.75,
                      ),
                    ),
                    child: Column(children: [
```

```
Container(
  height: 100,
  width: 220,
  alignment: Alignment.bottomRight,
  decoration: BoxDecoration(
    borderRadius: const BorderRadius.only(
        topLeft: Radius.circular(10),
        topRight: Radius.circular(10)),
    image: DecorationImage(
        image: url == "null"
            ? const AssetImage("assets/LoadingImage.png")
                as ImageProvider
            : NetworkImage(
                "https://image.tmdb.org/t/p/w500$url"),
        fit: BoxFit.cover),
  ),
  child: Container(
    margin: const EdgeInsets.all(10),
    child: Text(
      snapshot.data[index].episodeNumber.toString(),
      style: const TextStyle(
          color: Colors.white,
          fontSize: 24,
          shadows: <Shadow>[
            Shadow(
              offset: Offset(0, 0),
              blurRadius: 10,
              color: Colors.black,
            ),
          ]),
    ),
  ),
),
Container(
  margin: const EdgeInsets.only(left: 8, top: 4),
  width: 200,
  child: Text(snapshot.data[index].name.toString(),
      textAlign: TextAlign.left,
      maxLines: 1,
      overflow: TextOverflow.ellipsis,
      style: const TextStyle(
        color: Colors.white,
      )),
),
```

```
                ]),
              );
            },
          );
        }
        return const Center(child: CircularProgressIndicator());
      }),
    );
  }
}
```

Output:-


5:38
8.1
Oppenheimer
Drama
History
Status
Released
Release: July 19, 2023
Overview
The story of J. Robert Oppenheimer's role in the development of the atomic bomb during World War II.
Recommendations
Barbie
THE CREATOR

MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	49
Name	Gaurang A Roarane
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 - 4

Aim :- To create an interactive Form using form widget

Theory:-

Forms are essential components of user interaction in mobile and web applications, allowing users to input data, make selections, and submit information. In Flutter, the Form widget serves as the foundation for creating interactive and validated forms. Let's explore the theory behind creating an interactive form using the Form widget:

Form Widget:

- The Form widget in Flutter provides a container for form fields and facilitates form validation and submission.
- It maintains the state of form fields and manages their validation status.
- The Form widget should be placed as the parent widget of form fields within the widget tree.

Form Fields:

- Form fields represent input elements within the form, such as text fields, checkboxes, radio buttons, dropdowns, etc.
- Flutter provides various built-in form field widgets like TextFormField, Checkbox, Radio, DropdownButton, etc.
- Each form field widget must be wrapped inside a FormField widget to register with the Form and participate in form validation.

Validation:

- Form validation ensures that user input meets specified criteria or constraints before submitting the form.
- Flutter offers built-in validation mechanisms for form fields using validators.
- Validators are callback functions applied to form fields to validate input data. They return an error message if validation fails.
- Validators can be synchronous or asynchronous, depending on the complexity of validation logic.

Submission:

- Once the form is filled out and validated, it can be submitted to process the user input.
- Form submission involves handling the data collected from form fields and performing any necessary actions, such as saving data to a database, sending it to a server, etc.
- Submission can be triggered by a button press or any other user action.

LoginScreen.dart

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

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:osiris/Services/auth.dart';
import 'package:osiris/Services/consts.dart';
import 'package:provider/provider.dart';

import 'package:flutter_svg/flutter_svg.dart';

class LoginScreen extends StatelessWidget {
  const LoginScreen({super.key, Key? key1});

  @override
  Widget build(BuildContext context) {
    var size = MediaQuery.of(context).size;
    TextEditingController emailController = TextEditingController();
    TextEditingController passwordController = TextEditingController();

    return Scaffold(
      body: Container(
        color: background_primary,
        child: SafeArea(
          child: SingleChildScrollView(
            child: Container(
              color: background_primary,
              height: size.height,
              width: size.width,
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  Container(
                    height: size.height * 0.3,
                  ),
                  SvgPicture.asset(
                    "assets/netflix.svg",
                    width: size.width * 0.60,
),
                  Padding(
                    padding: const EdgeInsets.all(24.0),
                    child: Column(
                      children: [
                        TextFormField(
                          controller: emailController,
                          decoration: const InputDecoration(
                            labelText: 'Email',
                            filled: true,
```

```
    fillColor: Colors.white,
   ),
  ),
  const SizedBox(height: 20),
  TextFormField(
   controller: passwordController,
   decoration: const InputDecoration(
    labelText: 'Password',
    filled: true,
    fillColor: Colors.white,
   ),
   obscureText: true,
  ),
  const SizedBox(height: 20),
  ElevatedButton(
   onPressed: () async {
    try {
     await FirebaseAuth.instance
       .signInWithEmailAndPassword(
      email: emailController.text.trim(),
      password: passwordController.text.trim(),
     );
     // Authentication successful, navigate to next screen
    } catch (e) {
     // Handle authentication failure
     print('Error: $e');
    }
   },
   child: const Text('Sign In with Email'),
  ),
  const SizedBox(height: 20),
  ElevatedButton.icon(
   onPressed: () {
    final provider = Provider.of<GoogleSignInProvider>(
      context,
      listen: false);
    provider.googleLogin();
   },
   icon: const FaIcon(
    FontAwesomeIcons.google,
    color: Colors.white,
   ),
   label: const Text(
    'Continue with Google',
```

```
                    style: TextStyle(color: Colors.white),
                  ),
                  style: ElevatedButton.styleFrom(
                    shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(14)),
                    backgroundColor: const Color(0xFF2A292F)),
                ),
              ],
            ),
          ),
          Container(
            margin: const EdgeInsets.fromLTRB(28, 0, 28, 24),
            child: Text(
              footer_text,
              textAlign: TextAlign.center,
              style: const TextStyle(color: Color(0xFF423E50)),
            ),
          )
        ],
      ),
    ),
  ),
 ),
 );
 }
}
```

SignUpScreen.dart

```
import 'package:flutter/material.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:osiris/Screens/LoginScreen.dart';
import 'package:osiris/Services/auth.dart';
import 'package:osiris/Services/consts.dart';
import 'package:provider/provider.dart';

import 'package:flutter_svg/flutter_svg.dart';

class SignUpScreen extends StatelessWidget {
  const SignUpScreen({super.key, Key? key1});

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

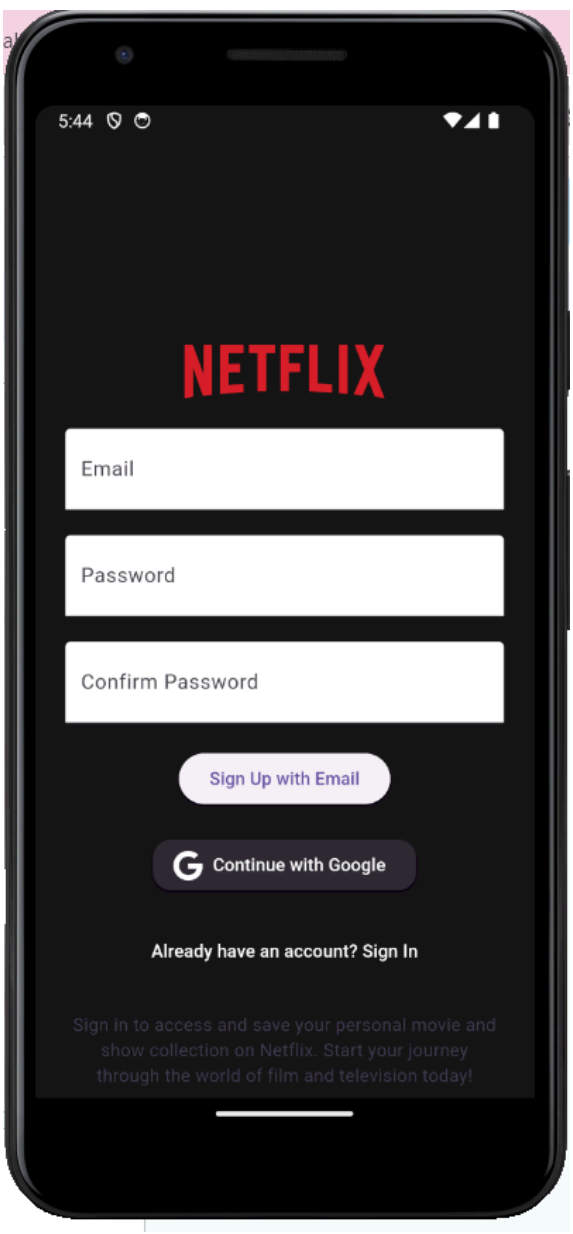
```
var size = MediaQuery.of(context).size;
TextEditingController emailController = TextEditingController();
TextEditingController passwordController = TextEditingController();
TextEditingController confirmPasswordController = TextEditingController();

return Scaffold(
  body: Container(
    color: background_primary,
    child: SafeArea(
      child: SingleChildScrollView(
        child: Container(
          color: background_primary,
          height: size.height,
          width: size.width,
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Container(
                height: size.height * 0.2,
              ),
              SvgPicture.asset(
                "assets/netflix.svg",
                width: size.width * 0.40,
              ),
              Padding(
                padding: const EdgeInsets.all(24.0),
                child: Column(
                  children: [
                    TextFormField(
                      controller: emailController,
                      decoration: const InputDecoration(
                        labelText: 'Email',
                        filled: true,
                        fillColor: Colors.white,
                      ),
                    ),
                    const SizedBox(height: 20),
                    TextFormField(
                      controller: passwordController,
                      decoration: const InputDecoration(
                        labelText: 'Password',
                        filled: true,
                        fillColor: Colors.white,
                      ),
```

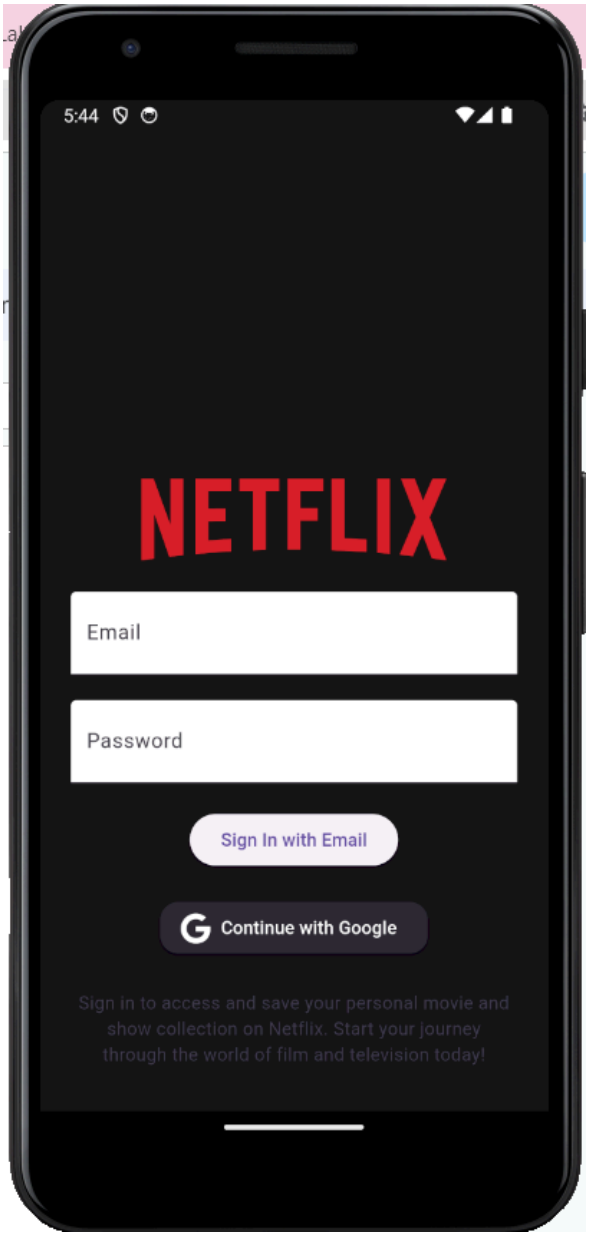
```
  obscureText: true,
),
const SizedBox(height: 20),
TextFormField(
  controller: confirmPasswordController,
  decoration: const InputDecoration(
    labelText: 'Confirm Password',
    filled: true,
    fillColor: Colors.white,
  ),
  obscureText: true,
),
const SizedBox(height: 20),
ElevatedButton(
  onPressed: () async {
    if (passwordController.text.trim() == confirmPasswordController.text.trim()) {
      try {
        await FirebaseAuth.instance
            .createUserWithEmailAndPassword(
          email: emailController.text.trim(),
          password: passwordController.text.trim(),
        );
        // Sign up successful, navigate to login screen
        Navigator.pushReplacement(
          context,
          MaterialPageRoute(builder: (context) => const LoginScreen()),
        );
      } catch (e) {
        // Handle sign up failure
        print('Error: $e');
      }
    } else {
      // Handle password mismatch
      print('Error: Passwords do not match');
    }
  },
  child: const Text('Sign Up with Email'),
),
const SizedBox(height: 20),
ElevatedButton.icon(
  onPressed: () {
    final provider = Provider.of<GoogleSignInProvider>(
        context,
        listen: false);
```

```
                provider.googleLogin();
              },
              icon: const FaIcon(
                FontAwesomeIcons.google,
                color: Colors.white,
              ),
              label: const Text(
                'Continue with Google',
                style: TextStyle(color: Colors.white),
              ),
              style: ElevatedButton.styleFrom(
                  shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.circular(14)),
                  backgroundColor: const Color(0xFF2A292F)),
            ),
            const SizedBox(height: 20),
            TextButton(
              onPressed: () {
                Navigator.push(
                  context,
                  MaterialPageRoute(builder: (context) => const LoginScreen()),
                );
              },
              child: const Text(
                'Already have an account? Sign In',
                style: TextStyle(color: Colors.white),
              ),
            ),
          ],
        ),
      ),
      Container(
        margin: const EdgeInsets.fromLTRB(28, 0, 28, 24),
        child: Text(
          footer_text,
          textAlign: TextAlign.center,
          style: const TextStyle(color: Color(0xFF423E50)),
        ),
      )
    ],
  ),
 ),
),
```

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

Output:-


5:44
NETFLIX
Email
Password
Confirm Password
Sign Up with Email
Continue with Google
Already have an account? Sign In
Sign in to access and save your personal movie and show collection on Netflix. Start your journey through the world of film and television today!


5:44
NETFLIX
Email
Password
Sign In with Email
Continue with Google
Sign in to access and save your personal movie and show collection on Netflix. Start your journey through the world of film and television today!

MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	49
Name	Gaurang A Roarane
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 - 5

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

Theory:-

Navigation, routing, and gestures are essential aspects of mobile app development that enable users to navigate between screens, interact with app elements, and perform various actions. In Flutter, these features are implemented using widgets and event handlers to create intuitive and engaging user experiences. Let's delve into the theory behind navigation, routing, and gestures in a Flutter app:

Navigation and Routing:

- Navigation refers to the process of moving between different screens or pages within an app.
- Routing involves defining the routes or paths that users can take to navigate to specific screens.
- Flutter provides a built-in routing mechanism using the Navigator widget and MaterialApp widget's routes property to manage app navigation.
- Routes are mapped to unique route names or paths, making it easy to navigate to a specific screen using its route name.

Navigator Widget:

- The Navigator widget manages a stack of Route objects representing the app's navigation history.
- It allows pushing new routes onto the navigation stack, popping routes off the stack, and replacing routes with new ones.
- Routes can be pushed onto the stack using Navigator.push() and popped using Navigator.pop() methods.

Routes:

- A route represents a screen or page in the app.
- Each route is associated with a unique route name and a builder function that constructs the UI for the corresponding screen.
- Routes can be declared statically using the routes property of MaterialApp or dynamically using the Navigator widget.

Gesture Detection:

- Gestures enable users to interact with app elements through touch-based actions like tapping, swiping, dragging, pinching, etc.
- Flutter provides gesture detection widgets such as GestureDetector, InkWell, InkResponse, etc., to handle user input gestures.
- These widgets detect user gestures and trigger corresponding event handlers to perform specific actions.

Gesture Recognition:

- Gesture recognition involves identifying and interpreting user gestures to determine the intended action.
- Flutter's gesture detection widgets come with built-in gesture recognizers that recognize common gestures like taps, drags, long presses, etc.
- Developers can customize gesture recognition behavior and implement complex gestures using gesture recognizer callbacks.

BottomNavBar.dart

```
import 'package:flutter/material.dart';
import 'package:go_router/go_router.dart';
import 'package:unicons/unicons.dart';
import 'package:flutter/services.dart';
import 'package:osiris/Services/consts.dart';

class BottomNavBar extends StatefulWidget {
  BottomNavBar({Key? key, required this.currentIndex}) : super(key: key);
  int currentIndex = 0;

  @override
  _BottomNavBarState createState() => _BottomNavBarState();
}

class _BottomNavBarState extends State<BottomNavBar> {
  @override
  Widget build(BuildContext context) {
    return Container(
      height: 50,
      margin: const EdgeInsets.fromLTRB(8, 8, 8, 16),
      decoration: BoxDecoration(
          color: accent_t.withOpacity(0.95),
          borderRadius: BorderRadius.circular(12)),
      child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceAround,
        crossAxisAlignment: CrossAxisAlignment.center,
        children: [
          IconButton(
            icon: Icon(
              UniconsLine.home_alt,
              color: widget.currentIndex == 0 ? Colors.white : inactive_accent,
            ),
            onPressed: () {
              HapticFeedback.mediumImpact();
              setState(() {
                widget.currentIndex = 0;
```

```
            });
            GoRouter.of(context).go('/main');
          },
        ),
        IconButton(
          icon: Icon(
            UniconsLine.search,
            color: widget.currentIndex == 1 ? Colors.white : inactive_accent,
          ),
          onPressed: () {
            HapticFeedback.mediumImpact();
            setState(() {
              widget.currentIndex = 1;
            });
            GoRouter.of(context).go('/search');
          },
        ),
        IconButton(
          icon: Icon(
            UniconsLine.heart,
            color: widget.currentIndex == 2 ? Colors.white : inactive_accent,
          ),
          onPressed: () {
            HapticFeedback.mediumImpact();
            setState(() {
              widget.currentIndex = 2;
            });
            GoRouter.of(context).go('/profile');
          },
        ),
      ],
    ),
  );
 }
}
```

Routes.dart

```
import 'package:flutter/material.dart';
import 'package:go_router/go_router.dart';
import 'package:osiris/Screens/LoginScreen.dart';
import 'package:osiris/Screens/MainScreen.dart';
import 'package:osiris/Screens/MovieScreen.dart';
import 'package:osiris/Screens/NavScreen.dart';
import 'package:osiris/Screens/ProfileScreen.dart';
```

```
import 'package:osiris/Screens/SearchScreen.dart';
import 'package:osiris/Screens/TvShowScreen.dart';
import 'package:osiris/Screens/SignUpScreen.dart';



GoRouter router = GoRouter(initialLocation: '/', routes: [
  GoRoute(
    path: '/',
    builder: (context, state) => const NavScreen(),
  ),

  GoRoute(
    path: '/',
    builder: (context, state) => const SignUpScreen(),
    pageBuilder: defaultPageBuilder<SignUpScreen>(const SignUpScreen()),
  ),
  GoRoute(
    path: '/login',
    builder: (context, state) => const LoginScreen(),
    pageBuilder: defaultPageBuilder<LoginScreen>(const LoginScreen()),
  ),
  GoRoute(
    path: '/main',
    builder: (context, state) => const MainScreen(),
    pageBuilder: defaultPageBuilder<MainScreen>(const MainScreen()),
  ),
  GoRoute(
    path: '/search',
    builder: (context, state) => const SearchScreen(),
    pageBuilder: defaultPageBuilder<SearchScreen>(const SearchScreen()),
  ),
  GoRoute(
    path: '/profile',
    builder: (context, state) => const ProfileScreen(),
    pageBuilder: defaultPageBuilder<ProfileScreen>(const ProfileScreen()),
  ),
  GoRoute(
    path: '/movie/:id',
    builder: (context, state) => MovieScreen(state.params['id']!),
  ),
  GoRoute(
    path: '/tv/:id',
    builder: (context, state) => TVShowScreen(state.params['id']!),
```

```
  )
]);

CustomTransitionPage buildPageWithDefaultTransition<T>({
  required BuildContext context,
  required GoRouterState state,
  required Widget child,
}) {
  return CustomTransitionPage<T>(
    key: state.pageKey,
    child: child,
    transitionsBuilder: (context, animation, secondaryAnimation, child) =>
        FadeTransition(opacity: animation, child: child),
  );
}

Page<dynamic> Function(BuildContext, GoRouterState) defaultPageBuilder<T>(
        Widget child) =>
    (BuildContext context, GoRouterState state) {
      return buildPageWithDefaultTransition<T>(
        context: context,
        state: state,
        child: child,
      );
    };
```

Output:-


5:57

MAD & PWA Lab
Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	49
Name	Gaurang A Roarane
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 - 6

Aim:- To Connect Flutter UI with fireBase database

Theory:-

Firebase is a comprehensive platform provided by Google for building mobile and web applications. It offers various services, including a real-time database, authentication, cloud storage, and more. Connecting a Flutter UI with a Firebase database enables developers to store, retrieve, and synchronize data in real-time, providing a seamless experience for users. Let's explore the theory behind connecting Flutter UI with Firebase Database:

Firebase Realtime Database:

- Firebase Realtime Database is a cloud-hosted NoSQL database that allows developers to store and sync data between users in real-time.
- It provides a JSON-based data model, making it easy to organize and structure data.
- The database automatically synchronizes data across all connected clients, ensuring that changes made by one client are immediately reflected on other clients.

Firebase SDK for Flutter:

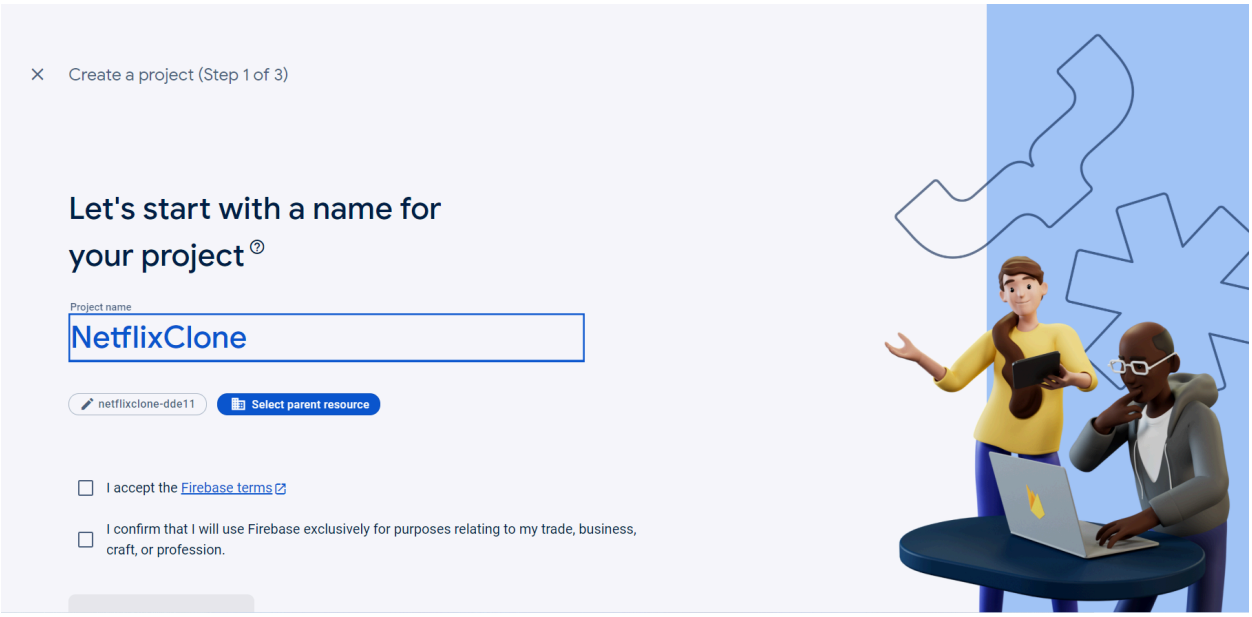
- Google provides the Firebase Flutter plugin, which allows Flutter apps to integrate seamlessly with Firebase services.
- The Firebase plugin for Flutter provides APIs to interact with Firebase services, including authentication, database, cloud storage, etc.
- It simplifies the process of connecting Flutter apps to Firebase services, enabling developers to focus on building the app's functionality.

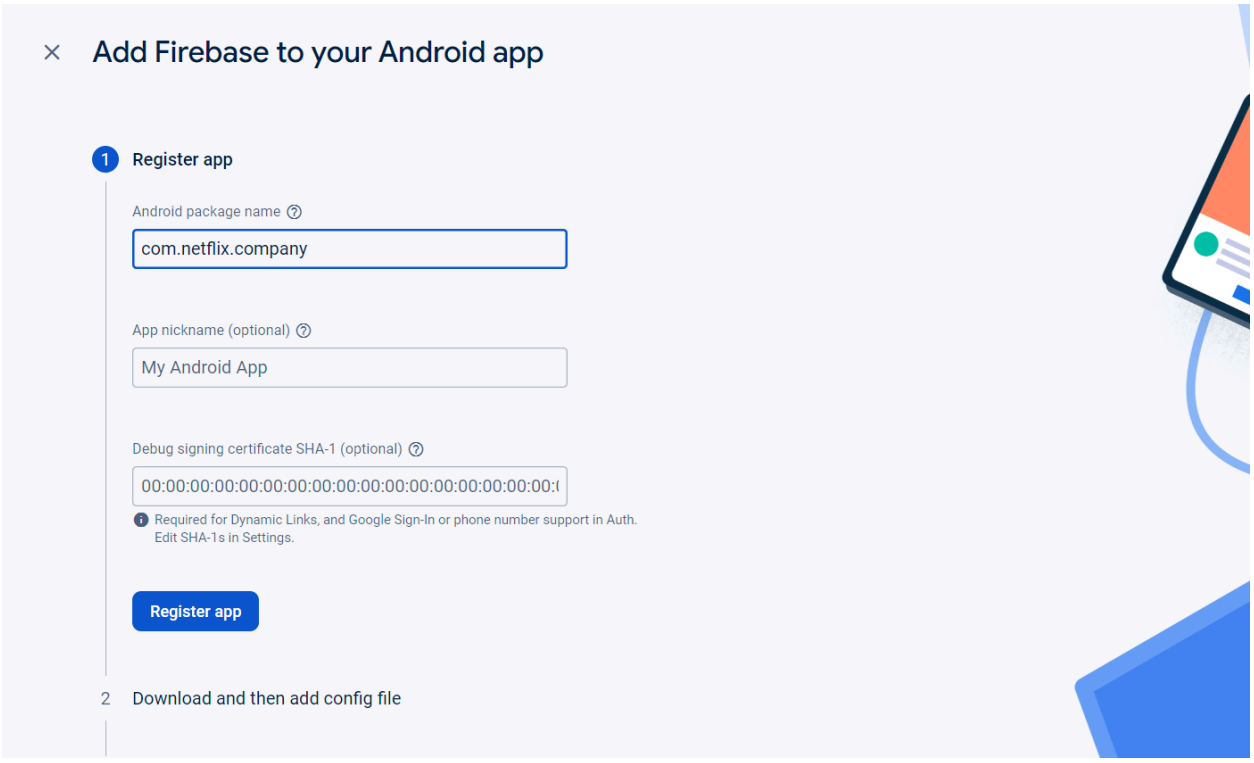
Integration Process:

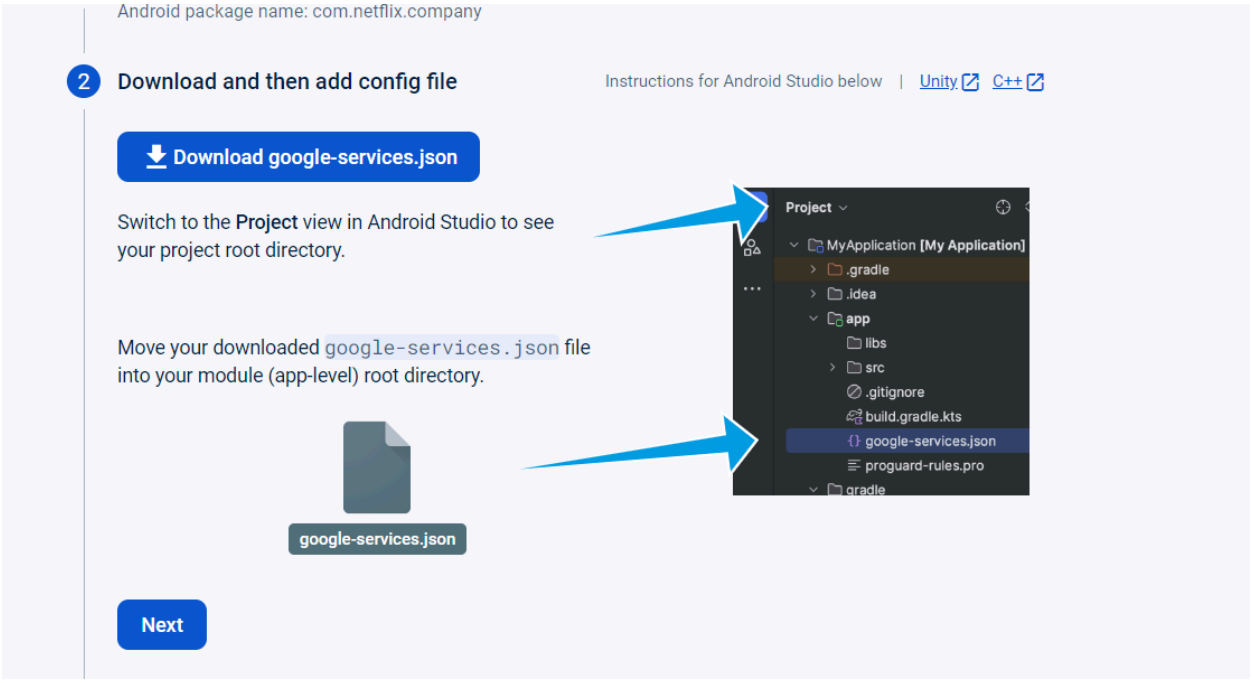
- Step 1: Set up Firebase Project:
    - Create a Firebase project from the Firebase Console (https://console.firebase.google.com).
    - Add your Flutter app to the Firebase project by registering its package name.
    - Download the google-services.json configuration file and place it in the android/app directory of your Flutter project.
- Step 2: Configure Flutter Project:
    - Add the Firebase Flutter plugin to your Flutter project by adding the necessary dependencies in the pubspec.yaml file.
    - Initialize Firebase in your Flutter app by calling Firebase.initializeApp() in the main() function.
- Step 3: Access Firebase Database:
    - Use the Firebase Database SDK for Flutter to interact with the Firebase Realtime Database.

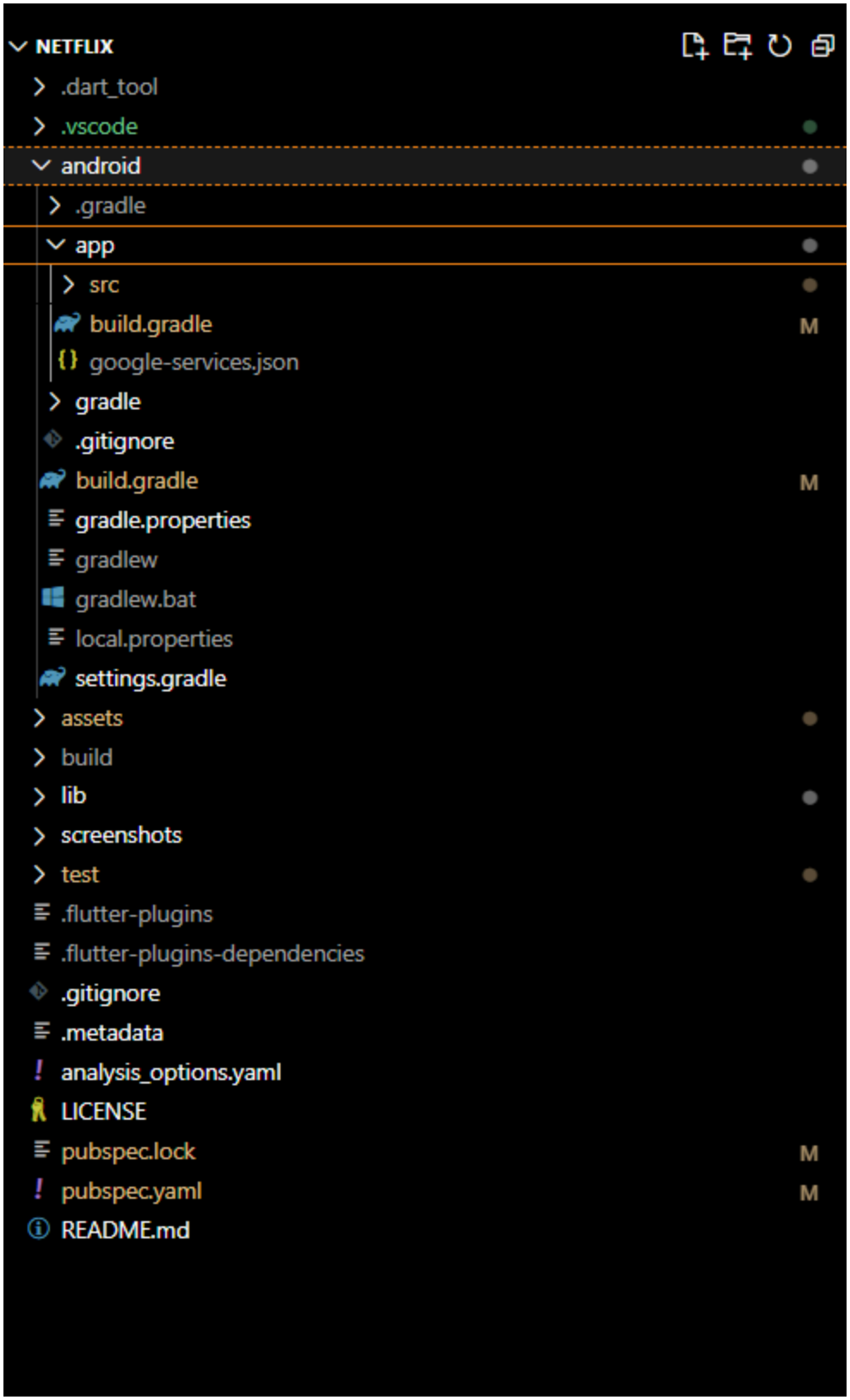
- Create references to database locations using the FirebaseDatabase class.
- Perform CRUD (Create, Read, Update, Delete) operations on the database using methods like set(), push(), update(), remove(), etc.

- Step 4: Display Data in Flutter UI:
  - Retrieve data from the Firebase database and display it in the Flutter UI.
  - Use Flutter widgets like ListView, GridView, Text, Image, etc., to render database content in the app's UI.
  - Implement real-time data synchronization to automatically update the UI when changes occur in the database.

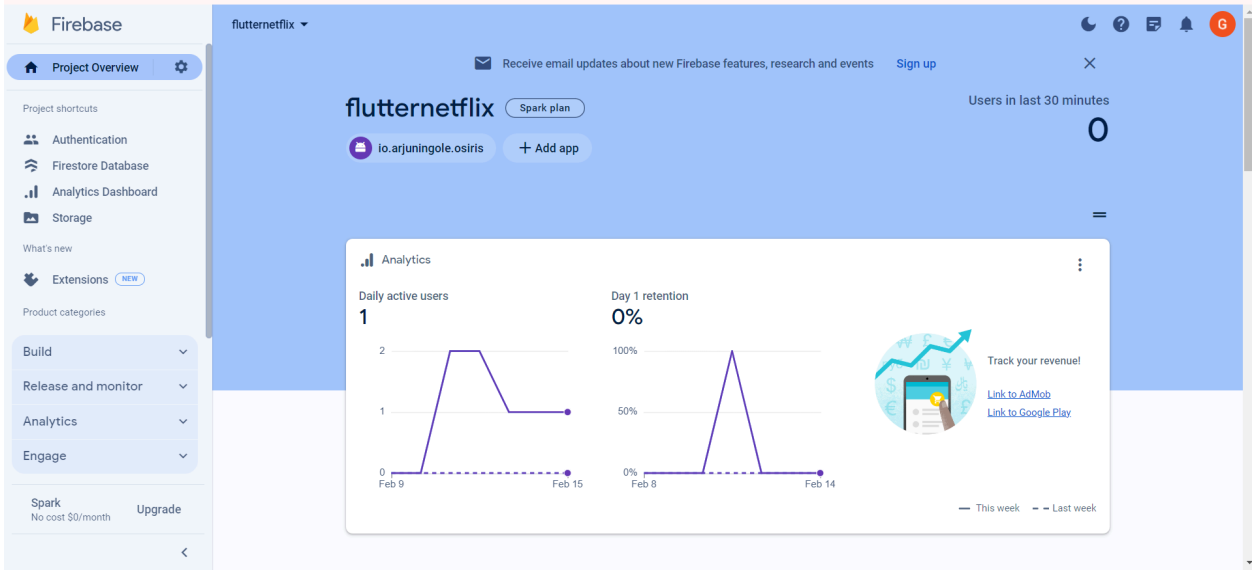


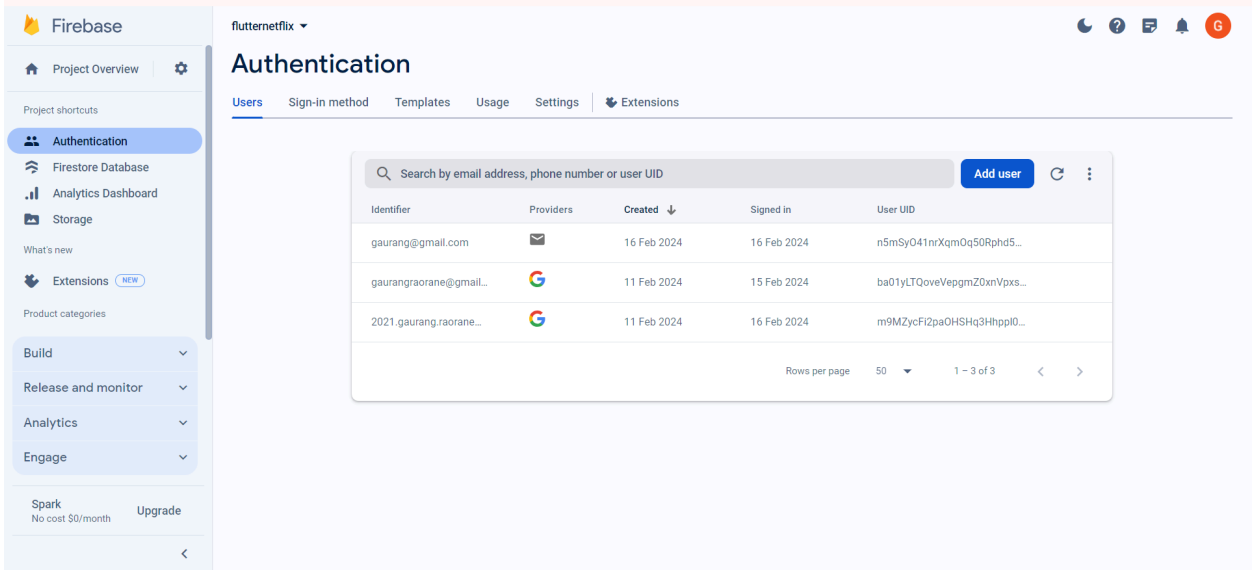

Add Firebase to your Android app
1 Register app
Android package name
com.netflix.company
App nickname (optional)
My Android App
Debug signing certificate SHA-1 (optional)
00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:0
Required for Dynamic Links, and Google Sign-In or phone number support in Auth. Edit SHA-1s in Settings.
Register app
2 Download and then add config file

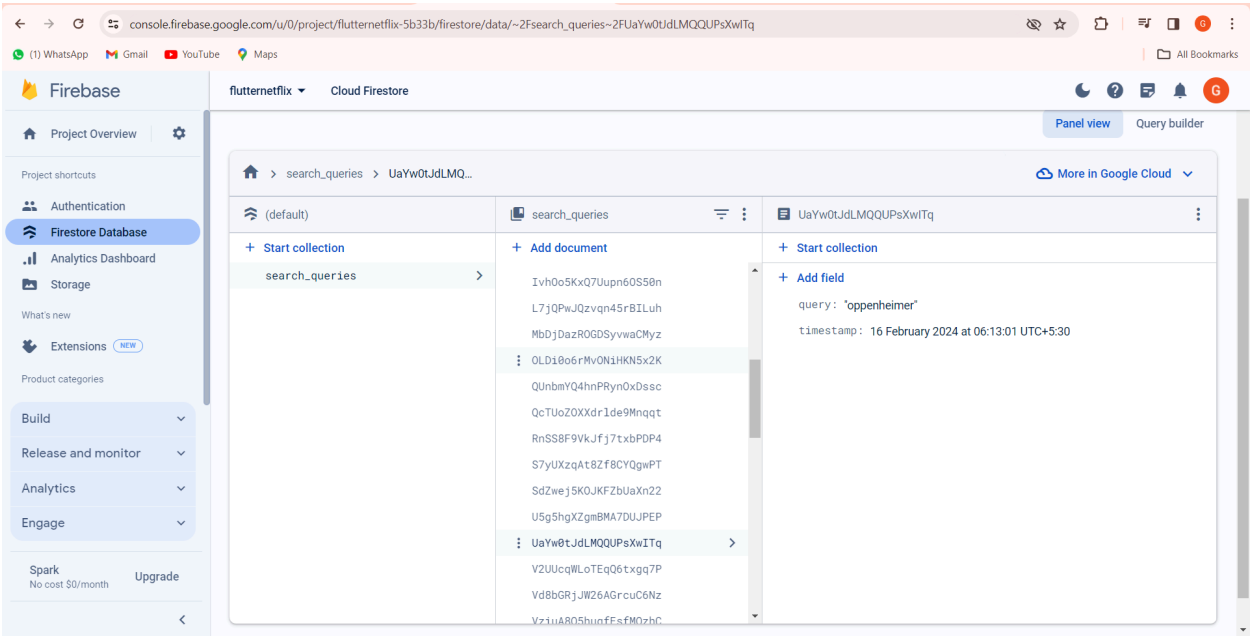

Android package name: com.netflix.company
2 Download and then add config file
Instructions for Android Studio below | Unity C++
Download google-services.json
Switch to the Project view in Android Studio to see your project root directory.
Move your downloaded google-services.json file into your module (app-level) root directory.
google-services.json
Project
MyApplication [My Application]
.gradle
.idea
app
libs
src
.gitignore
build.gradle.kts
google-services.json
proguard-rules.pro
gradle
Next


NETFLIX
.dart_tool
.vscode
android
.gradle
app
src
build.gradle
M
google-services.json
gradle
.gitignore
build.gradle
M
gradle.properties
gradlew
gradlew.bat
local.properties
settings.gradle
assets
build
lib
screenshots
test
.flutter-plugins
.flutter-plugins-dependencies
.gitignore
.metadata
analysis_options.yaml
LICENSE
pubspec.lock
M
pubspec.yaml
M
README.md

Project Title: Flutter:Netflix Movie App/ PWA:Glowing **Roll No. 49**







Hence Firebase is Successfully Connected.

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.	49
Name	Gaurang A Roarane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	15

Experiment - 7

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

Theory -

It's clear that PWAs offer several advantages over traditional web apps, particularly in terms of providing a more native-like experience, ease of access, faster services, engaging features, and real-time data access.
PWAs leverage modern web technologies and principles to deliver a more app-like experience to users while maintaining the flexibility and accessibility of web apps. By incorporating features such as service workers, PWAs can offer offline capabilities, faster loading times, and seamless updates without requiring users to download or install new versions manually.
The key features of PWAs that distinguish them from regular web apps include:
1. Progressive Enhancement: PWAs are built with progressive improvement principles, ensuring that they work for every user regardless of the browser or device they are using.

2. Responsive Design: PWAs adapt to various screen sizes and orientations, providing a consistent user experience across desktops, mobile devices, and tablets.

3. App-like Experience: PWAs behave like native apps in terms of interaction and navigation, providing a smoother and more immersive user experience.

4. Automatic Updates: PWAs can receive updates automatically from the server, ensuring that users always have access to the latest features and content without needing to manually update the app.

5. Security: PWAs are served over HTTPS to ensure secure connections and prevent unauthorized access or tampering of data.

6. Search Engine Indexing: PWAs are identified as applications and can be indexed by search engines, improving discoverability and visibility.

7. Reactivatable: PWAs make it easy for users to reactivate the application and receive notifications, enhancing user engagement and retention.

Overall, PWAs represent a significant advancement in web app development, offering a compelling alternative to traditional native apps with their enhanced performance, user experience, and accessibility across various platforms and devices.

Code -

Index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="theme-color" content="#4285f4">
  <meta name="viewport"
      content="width=device-width,
            initial-scale=1">
  <meta http-equiv="X-UA-Compatible"
      content="ie=edge">
  <title>Glowing - Reveal The Beauty of Skin</title>

  <link rel="shortcut icon" href="./favicon.svg" type="image/svg+xml">

  <link rel="stylesheet" href="./assets/css/style.css">

  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link
href="https://fonts.googleapis.com/css2?family=Urbanist:wght@400;500;600;700;800&display=
swap" rel="stylesheet">


  <link rel="preload" as="image" href="./assets/images/logo.png">
  <link rel="preload" as="image" href="./assets/images/hero-banner-1.jpg">
  <link rel="preload" as="image" href="./assets/images/hero-banner-2.jpg">
  <link rel="preload" as="image" href="./assets/images/hero-banner-3.jpg">
  <link rel="manifest" href="manifest.json">
  <script src="serviceworker.js"></script>
</head>

<body id="top">
  <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.error('ServiceWorker registration failed: ', e);
                    }
                }
            }
            if ('Notification' in window) {
                Notification.requestPermission().then(function (permission) {
                    if (permission === 'granted') {
                        console.log('Notification permission granted.');
                    } else {
                        console.warn('Notification permission denied.');
                    }
                });
            }
        </script>
<header class="header">

  <div class="alert">
    <div class="container">
      <p class="alert-text">Free Shipping On All U.S. Orders $50+</p>
    </div>
  </div>

  <div class="header-top" data-header>
    <div class="container">

      <button class="nav-open-btn" aria-label="open menu" data-nav-toggler>
        <span class="line line-1"></span>
        <span class="line line-2"></span>
        <span class="line line-3"></span>
      </button>

      <div class="input-wrapper">
        <input type="search" name="search" placeholder="Search product" class="search-field">

        <button class="search-submit" aria-label="search">
```

```
    <ion-icon name="search-outline" aria-hidden="true"></ion-icon>
  </button>
</div>

<a href="#" class="logo">
  <img src="./assets/images/logo.png" width="179" height="26" alt="Glowing">
</a>

<div class="header-actions">

  <button class="header-action-btn" aria-label="user">
    <ion-icon name="person-outline" aria-hidden="true" aria-hidden="true"></ion-icon>
  </button>

  <button class="header-action-btn" aria-label="favourite item">
    <ion-icon name="star-outline" aria-hidden="true" aria-hidden="true"></ion-icon>

    <span class="btn-badge">0</span>
  </button>

  <button class="header-action-btn" aria-label="cart item">
    <data class="btn-text" value="0">$0.00</data>

    <ion-icon name="bag-handle-outline" aria-hidden="true" aria-hidden="true"></ion-icon>

    <span class="btn-badge">0</span>
  </button>

</div>

<nav class="navbar">
  <ul class="navbar-list">

    <li>
      <a href="#home" class="navbar-link has-after">Home</a>
    </li>

    <li>
      <a href="#collection" class="navbar-link has-after">Collection</a>
    </li>

    <li>
      <a href="#shop" class="navbar-link has-after">Shop</a>
    </li>
```

```
        <li>
          <a href="#offer" class="navbar-link has-after">Offer</a>
        </li>

        <li>
          <a href="#blog" class="navbar-link has-after">Blog</a>
        </li>

      </ul>
    </nav>

  </div>
 </div>

</header>
<div class="sidebar">
 <div class="mobile-navbar" data-navbar>
  <div class="wrapper">
   <a href="#" class="logo">
    <img src="./assets/images/logo.png" width="179" height="26" alt="Glowing">
   </a>
   <button class="nav-close-btn" aria-label="close menu" data-nav-toggler>
    <ion-icon name="close-outline" aria-hidden="true"></ion-icon>
   </button>
  </div>
  <ul class="navbar-list">
   <li>
    <a href="#home" class="navbar-link" data-nav-link>Home</a>
   </li>
   <li>
    <a href="#collection" class="navbar-link" data-nav-link>Collection</a>
   </li>
   <li>
    <a href="#shop" class="navbar-link" data-nav-link>Shop</a>
   </li>
   <li>
    <a href="#offer" class="navbar-link" data-nav-link>Offer</a>
   </li>

   <li>
    <a href="#blog" class="navbar-link" data-nav-link>Blog</a>
   </li>
  </ul>
```

```
    </div>
    <div class="overlay" data-nav-toggler data-overlay></div>
  </div>
  <main>
    <article>
      <section class="section hero" id="home" aria-label="hero" data-section>
        <div class="container">

          <ul class="has-scrollbar">

            <li class="scrollbar-item">
              <div class="hero-card has-bg-image" style="background-image:
url('./assets/images/hero-banner-1.jpg')">

                <div class="card-content">

                  <h1 class="h1 hero-title">
                    Reveal The <br>
                    Beauty of Skin
                  </h1>

                  <p class="hero-text">
                    Made using clean, non-toxic ingredients, our products are designed for everyone.
                  </p>

                  <p class="price">Starting at $7.99</p>

                  <a href="#" class="btn btn-primary">Shop Now</a>

                </div>

              </div>
            </li>

            <li class="scrollbar-item">
              <div class="hero-card has-bg-image" style="background-image:
url('./assets/images/hero-banner-2.jpg')">

                <div class="card-content">

                  <h1 class="h1 hero-title">
                    Reveal The <br>
                    Beauty of Skin
                  </h1>
```

```
            <p class="hero-text">
              Made using clean, non-toxic ingredients, our products are designed for everyone.
            </p>

            <p class="price">Starting at $7.99</p>

            <a href="#" class="btn btn-primary">Shop Now</a>

          </div>

        </div>
      </li>

      <li class="scrollbar-item">
        <div class="hero-card has-bg-image" style="background-image:
url('./assets/images/hero-banner-3.jpg')">

          <div class="card-content">
            <h1 class="h1 hero-title">
              Reveal The <br>
              Beauty of Skin
            </h1>
            <p class="hero-text">
              Made using clean, non-toxic ingredients, our products are designed for everyone.
            </p>
            <p class="price">Starting at $7.99</p>
            <a href="#" class="btn btn-primary">Shop Now</a>
          </div>
        </div>
      </li>
    </ul>

  </div>
</section>




<!--
  - #COLLECTION
-->
```

```
    <section class="section collection" id="collection" aria-label="collection" data-section>
      <div class="container">

        <ul class="collection-list">

          <li>
            <div class="collection-card has-before hover:shine">

              <h2 class="h2 card-title">Summer Collection</h2>

              <p class="card-text">Starting at $17.99</p>

              <a href="#" class="btn-link">
                <span class="span">Shop Now</span>

                <ion-icon name="arrow-forward" aria-hidden="true"></ion-icon>
              </a>

              <div class="has-bg-image" style="background-image:
url('./assets/images/collection-1.jpg')"></div>

            </div>
          </li>

          <li>
            <div class="collection-card has-before hover:shine">

              <h2 class="h2 card-title">What's New?</h2>

              <p class="card-text">Get the glow</p>

              <a href="#" class="btn-link">
                <span class="span">Discover Now</span>

                <ion-icon name="arrow-forward" aria-hidden="true"></ion-icon>
              </a>

              <div class="has-bg-image" style="background-image:
url('./assets/images/collection-2.jpg')"></div>

            </div>
          </li>

          <li>
```

```
            <div class="collection-card has-before hover:shine">

              <h2 class="h2 card-title">Buy 1 Get 1</h2>

              <p class="card-text">Starting at $7.99</p>

              <a href="#" class="btn-link">
                <span class="span">Discover Now</span>

                <ion-icon name="arrow-forward" aria-hidden="true"></ion-icon>
              </a>

              <div class="has-bg-image" style="background-image:
url('./assets/images/collection-3.jpg')"></div>

            </div>
          </li>

        </ul>

      </div>
    </section>




    <!--
      - #SHOP
    -->

    <section class="section shop" id="shop" aria-label="shop" data-section>
      <div class="container">

        <div class="title-wrapper">
          <h2 class="h2 section-title">Our Bestsellers</h2>

          <a href="#" class="btn-link">
            <span class="span">Shop All Products</span>

            <ion-icon name="arrow-forward" aria-hidden="true"></ion-icon>
          </a>
        </div>
```

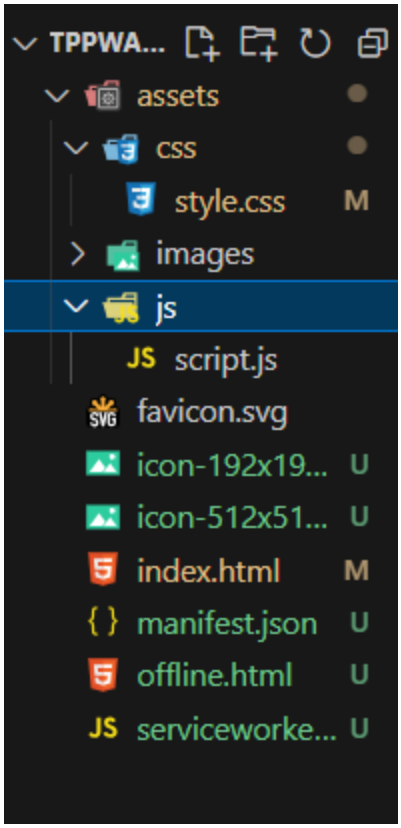
```
<ul class="has-scrollbar">

  <li class="scrollbar-item">
    <div class="shop-card">

      <div class="card-banner img-holder" style="--width: 540; --height: 720;">
        <img src="./assets/images/product-01.jpg" width="540" height="720" loading="lazy"
          alt="Facial cleanser" class="img-cover">

        <span class="badge" aria-label="20% off">-20%</span>

        <div class="card-actions">

          <button class="action-btn" aria-label="add to cart">
            <ion-icon name="bag-handle-outline" aria-hidden="true"></ion-icon>
          </button>

          <button class="action-btn" aria-label="add to whishlist">
            <ion-icon name="star-outline" aria-hidden="true"></ion-icon>
          </button>

          <button class="action-btn" aria-label="compare">
            <ion-icon name="repeat-outline" aria-hidden="true"></ion-icon>
          </button>

        </div>
      </div>

      <div class="card-content">

        <div class="price">
          <del class="del">$39.00</del>

          <span class="span">$29.00</span>
        </div>

        <h3>
          <a href="#" class="card-title">Facial cleanser</a>
        </h3>

        <div class="card-rating">

          <div class="rating-wrapper" aria-label="5 start rating">
            <ion-icon name="star" aria-hidden="true"></ion-icon>
```

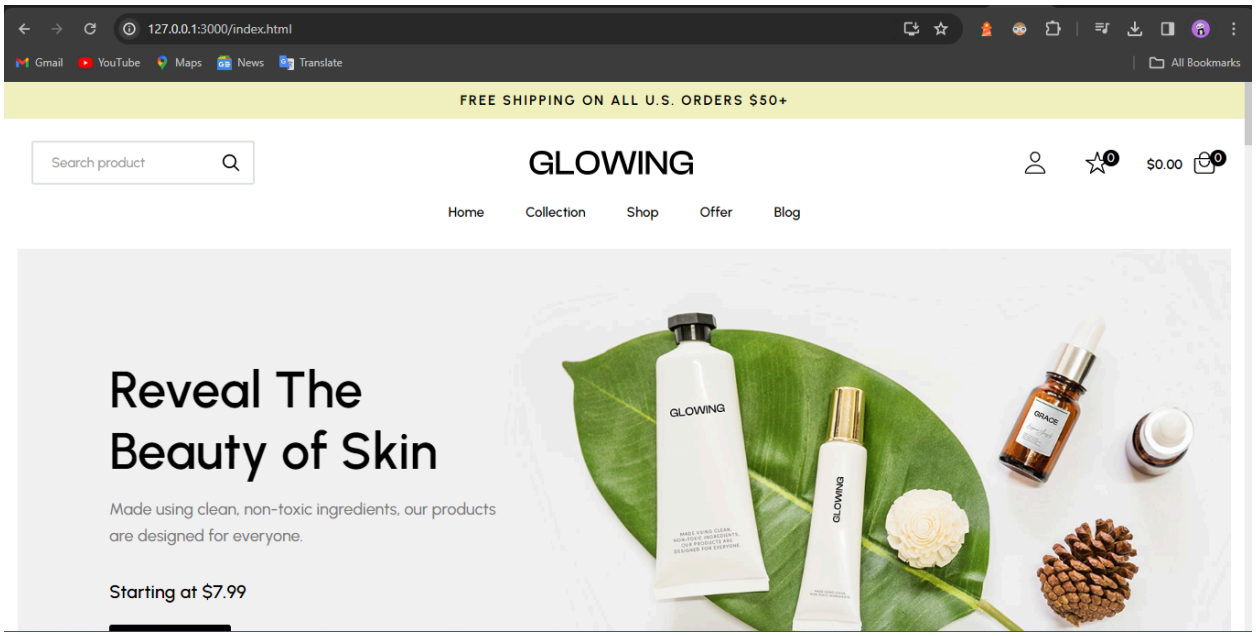
```
            <ion-icon name="star" aria-hidden="true"></ion-icon>
            <ion-icon name="star" aria-hidden="true"></ion-icon>
            <ion-icon name="star" aria-hidden="true"></ion-icon>
            <ion-icon name="star" aria-hidden="true"></ion-icon>
          </div>

          <p class="rating-text">5170 reviews</p>

        </div>

      </div>

    </div>
  </li>

  <li class="scrollbar-item">
    <div class="shop-card">

      <div class="card-banner img-holder" style="--width: 540; --height: 720;">
        <img src="./assets/images/product-02.jpg" width="540" height="720" loading="lazy"
          alt="Bio-shroom Rejuvenating Serum" class="img-cover">

        <div class="card-actions">

          <button class="action-btn" aria-label="add to cart">
            <ion-icon name="bag-handle-outline" aria-hidden="true"></ion-icon>
          </button>

          <button class="action-btn" aria-label="add to whishlist">
            <ion-icon name="star-outline" aria-hidden="true"></ion-icon>
          </button>

          <button class="action-btn" aria-label="compare">
            <ion-icon name="repeat-outline" aria-hidden="true"></ion-icon>
          </button>

        </div>
      </div>

      <div class="card-content">

        <div class="price">
          <span class="span">$29.00</span>
        </div>
```

```
            <h3>
              <a href="#" class="card-title">Bio-shroom Rejuvenating Serum</a>
            </h3>

            <div class="card-rating">

              <div class="rating-wrapper" aria-label="5 start rating">
                <ion-icon name="star" aria-hidden="true"></ion-icon>
                <ion-icon name="star" aria-hidden="true"></ion-icon>
                <ion-icon name="star" aria-hidden="true"></ion-icon>
                <ion-icon name="star" aria-hidden="true"></ion-icon>
                <ion-icon name="star" aria-hidden="true"></ion-icon>
              </div>

              <p class="rating-text">5170 reviews</p>

            </div>

          </div>

        </div>
      </li>
……
<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.error('ServiceWorker registration failed: ', e);
      }
    }
```

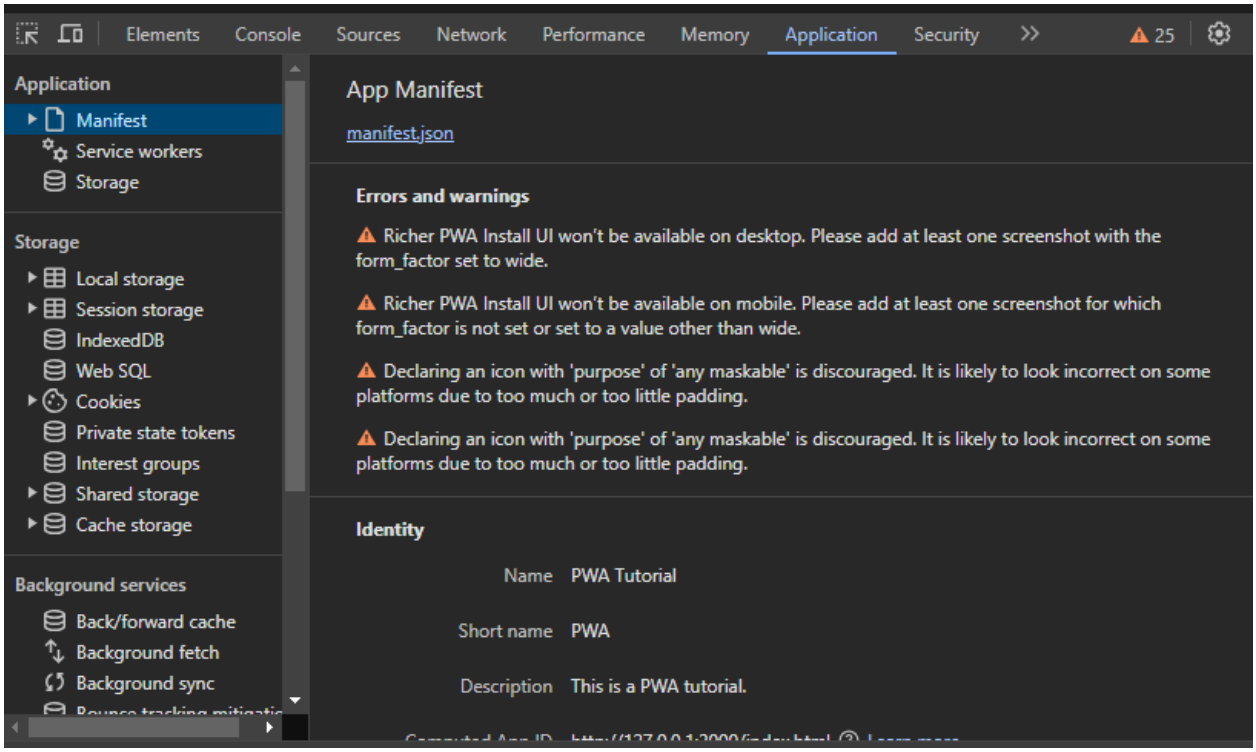
```
    }
    if ('Notification' in window) {
      Notification.requestPermission().then(function (permission) {
        if (permission === 'granted') {
          console.log('Notification permission granted.');
        } else {
          console.warn('Notification permission denied.');
        }
      });
    }
  </script>
```

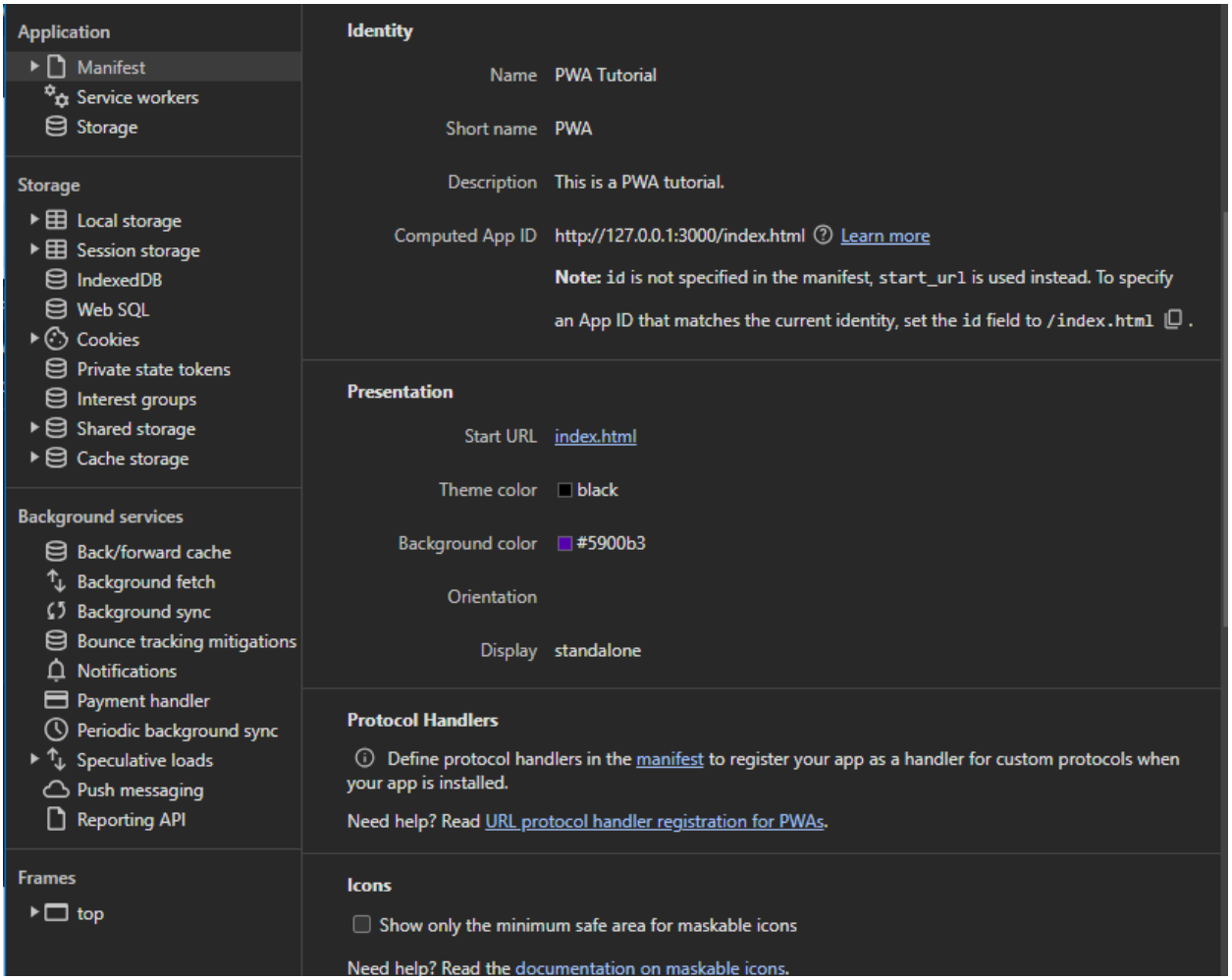
```
{
    "name":"PWA Tutorial",
    "short_name":"PWA",
    "start_url":"index.html",
    "display":"standalone",
    "background_color":"#5900b3",
    "theme_color":"black",
    "scope": ".",
    "description":"This is a PWA tutorial.",
    "icons":[
    {
    "src":"icon-192x192.png",
    "sizes":"192x192",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


TPPWA...
assets
css
style.css M
images
js
script.js
favicon.svg
icon-192x19... U
icon-512x51... U
index.html M
manifest.json U
offline.html U
serviceworke... U

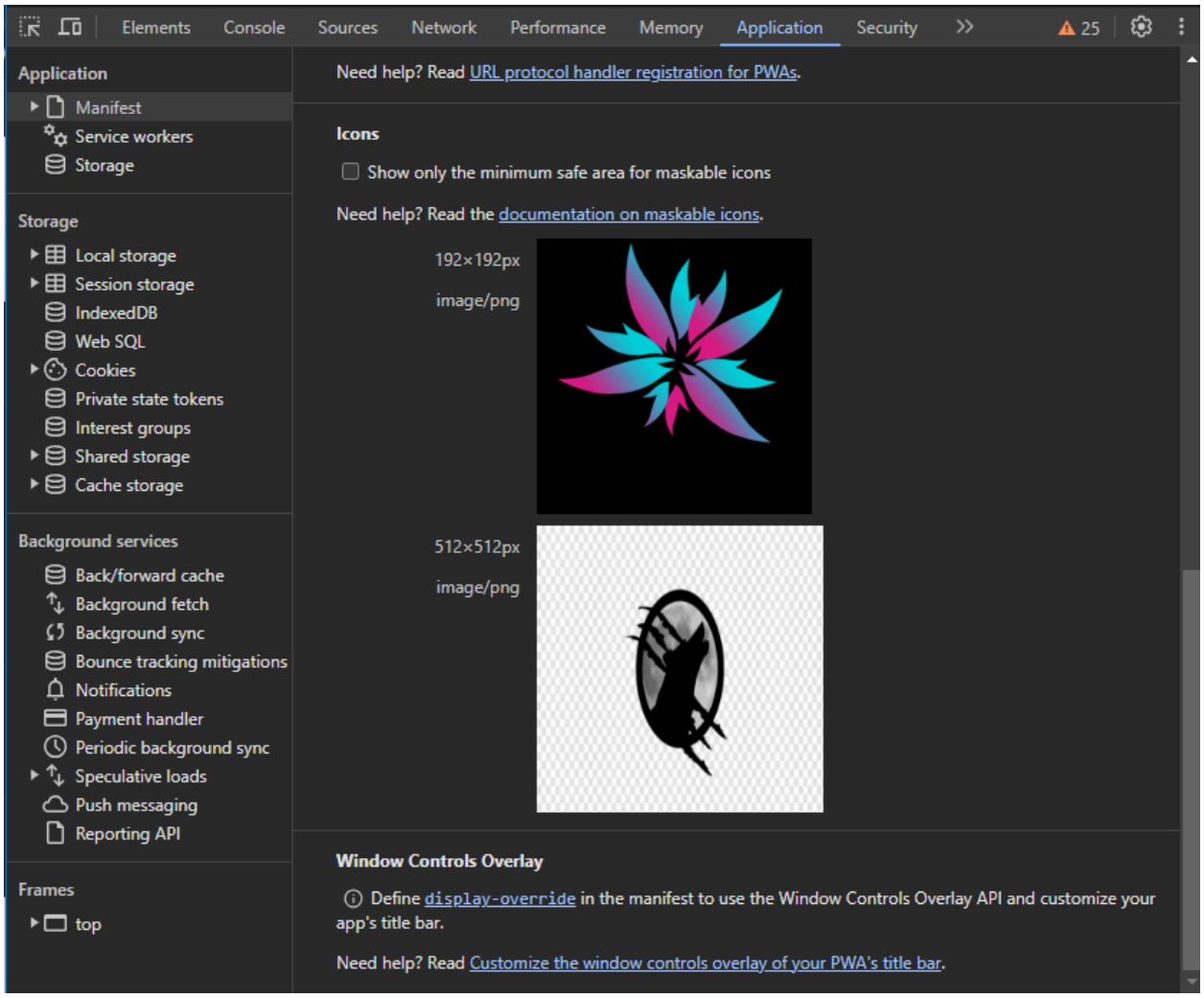

127.0.0.1:3000/index.html
Gmail
YouTube
Maps
News
Translate
All Bookmarks
FREE SHIPPING ON ALL U.S. ORDERS $50+
Search product
GLOWING
$0.00
Home
Collection
Shop
Offer
Blog
Reveal The Beauty of Skin
Made using clean, non-toxic ingredients, our products are designed for everyone.
Starting at $7.99
GLOWING
GRACE

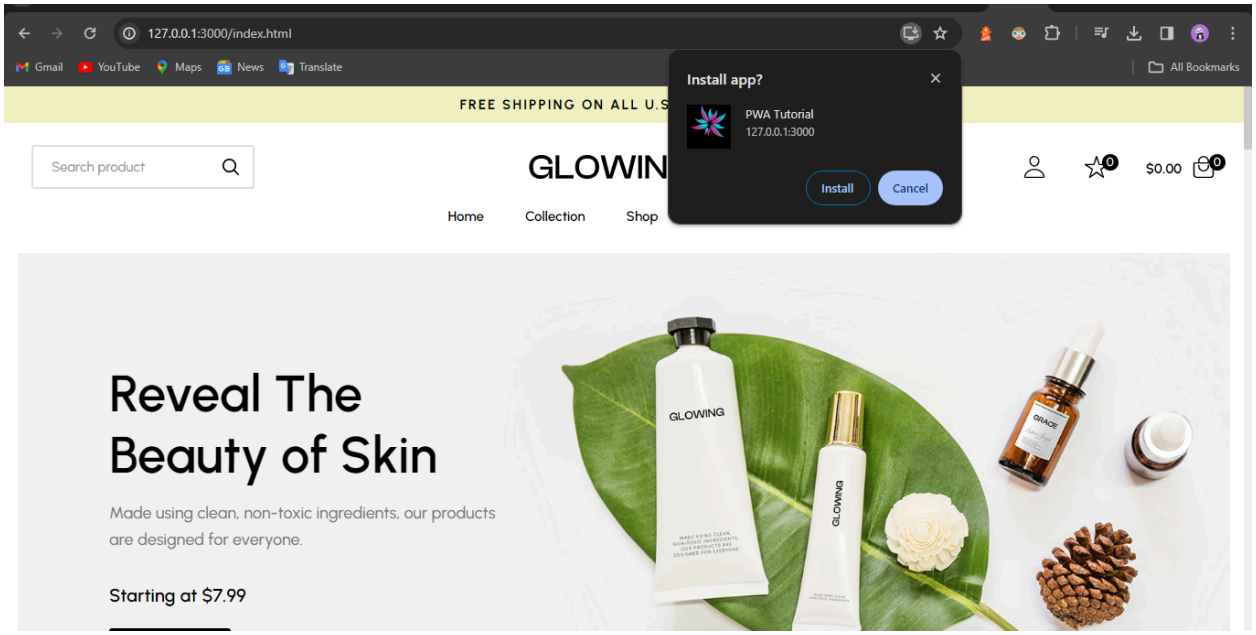
Project Title: Flutter:Netflix Movie App/ PWA:Glowing **Roll No. 49**



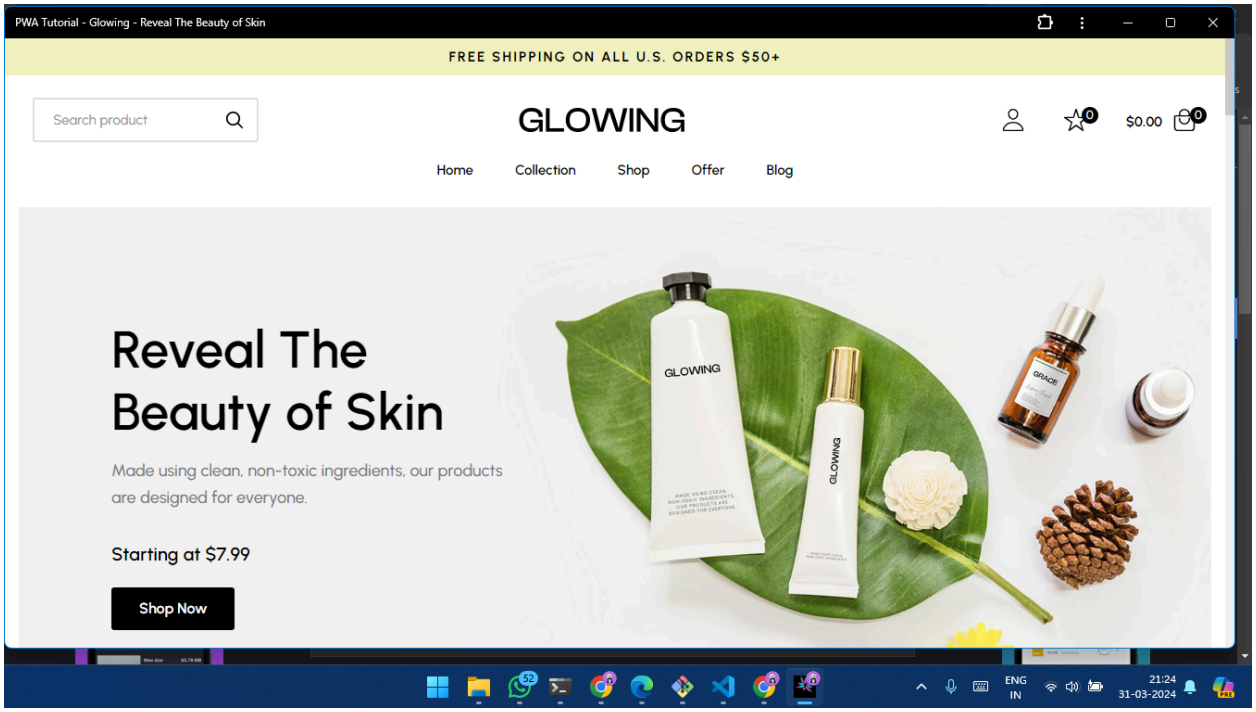

Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
IndexedDB
Web SQL
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking mitigations
Notifications
Payment handler
Periodic background sync
Speculative loads
Push messaging
Reporting API
Frames
top
Identity
Name PWA Tutorial
Short name PWA
Description This is a PWA tutorial.
Computed App ID http://127.0.0.1:3000/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 /index.html .
Presentation
Start URL index.html
Theme color black
Background color #5900b3
Orientation
Display standalone
Protocol Handlers
Define protocol handlers in the manifest to register your app as a handler for custom protocols when your app is installed.
Need help? Read URL protocol handler registration for PWAs.
Icons
Show only the minimum safe area for maskable icons
Need help? Read the documentation on maskable icons.

Project Title: Flutter:Netflix Movie App/ PWA:Glowing **Roll No. 49**





Project Title: Flutter:Netflix Movie App/ PWA:Glowing **Roll No. 49**



Conclusion -
In this experiment, we have successfully created a basic progressive web app of our web page and installed it in our desktop successfully.

MAD & PWA Lab
Journal

Experiment No.	08
Experiment Title.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA
Roll No.	49
Name	Gaurang A Roarane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

Experiment - 8

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

Theory -
Service Worker is a script that operates independently in the background of a browser, acting as a programmable network proxy. It enables developers to control how network requests from a web page are handled. It's crucial to note that Service Workers only function over HTTPS due to security concerns, as they have the capability to intercept network requests and modify responses.

Here's a summary of what you can do and can't do with Service Workers:

What You Can Do:

1. Dominate Network Traffic: You can manage all network traffic of the page and manipulate requests and responses as needed. This includes altering responses for different types of requests.

2. Cache Content: Service Workers allow caching of request/response pairs using the Cache API. This enables offline access to cached content, improving performance and user experience.

3. Manage Push Notifications: You can handle push notifications with Service Workers, allowing you to display messages to users even when the internet connection is broken.

4. Background Sync: Service Workers facilitate background processes, enabling tasks to continue despite interruptions in internet connectivity.

What You Can't Do:

1. Access the Window: Service Workers cannot access the window or manipulate DOM elements directly. Communication with the window is possible through postMessage.

2. Work on Port 80: Service Workers only function on HTTPS, but during development, they can operate on localhost.

Registration:
To install a Service Worker, it must be registered in the main JavaScript code. This registration informs the browser where the Service Worker is located and initiates the installation process in the background.

Installation:
Once registered, the browser attempts to install the Service Worker if it's considered new or updated. This triggers an install event, during which tasks such as precaching parts of the web app can be performed to enhance subsequent loading times.

Overall, Service Workers offer powerful capabilities for enhancing web applications, particularly in terms of offline functionality, network traffic management, and push notifications. However, developers must be mindful of security considerations and the limitations imposed by the Service Worker environment.

Code -
Serviceworker.js

```
const CACHE_NAME = 'my-ecommerce-app-cache-v1';
const urlsToCache = [
  '/',
  'index.html',
  'assets/js/script.js',
  'assets/css/style.css',
  'serviceworker.js',
  'manifest.json',
  'offline.html'
  // Add more files to cache as needed
];

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

self.addEventListener('activate', function(event) {
  // Perform activation steps
  event.waitUntil(
    caches.keys().then(function(cacheNames) {
      return Promise.all(
```

```
        cacheNames.map(function(cacheName) {
          if (cacheName !== CACHE_NAME) {
            return caches.delete(cacheName);
          }
        })
      );
    })
  );
});
self.addEventListener('activate', async () => {
  if (Notification.permission !== 'granted') {
    try {
      const permission = await Notification.requestPermission();
      if (permission === 'granted') {
        console.log('Notification permission granted.');
      } else {
        console.warn('Notification permission denied.');
      }
    } catch (error) {
      console.error('Failed to request notification permission:', error);
    }
  }
});
```

manifest.js

```
{
    "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":"icon-192x192.png",
    "sizes":"192x192",
    "type":"image/png",
    "purpose":"any maskable"
    },
    {
    "src":"icon-512x512.png",
    "sizes":"512x512",
```

```
    "type":"image/png",
    "purpose":"any maskable"
    }
]
}
```

Offline.html

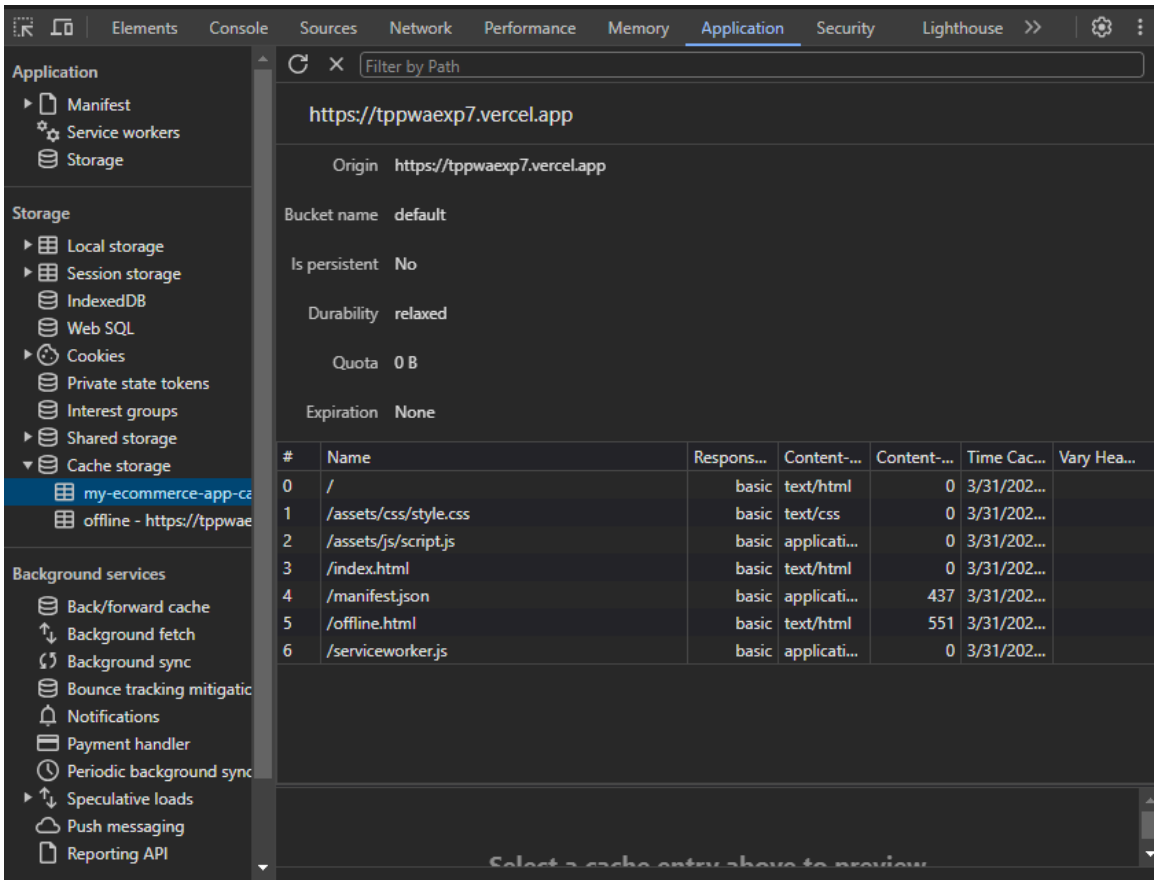
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Offline</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f0f0f0;
      color: #333;
      text-align: center;
      padding: 50px;
    }
    h1 {
      margin-bottom: 20px;
    }
  </style>
</head>
<body>
  <h1>Oops! You're offline.</h1>
  <p>It seems like you're currently offline. Please check your internet connection and try again
later.</p>
</body>
</html>
```

script.js-

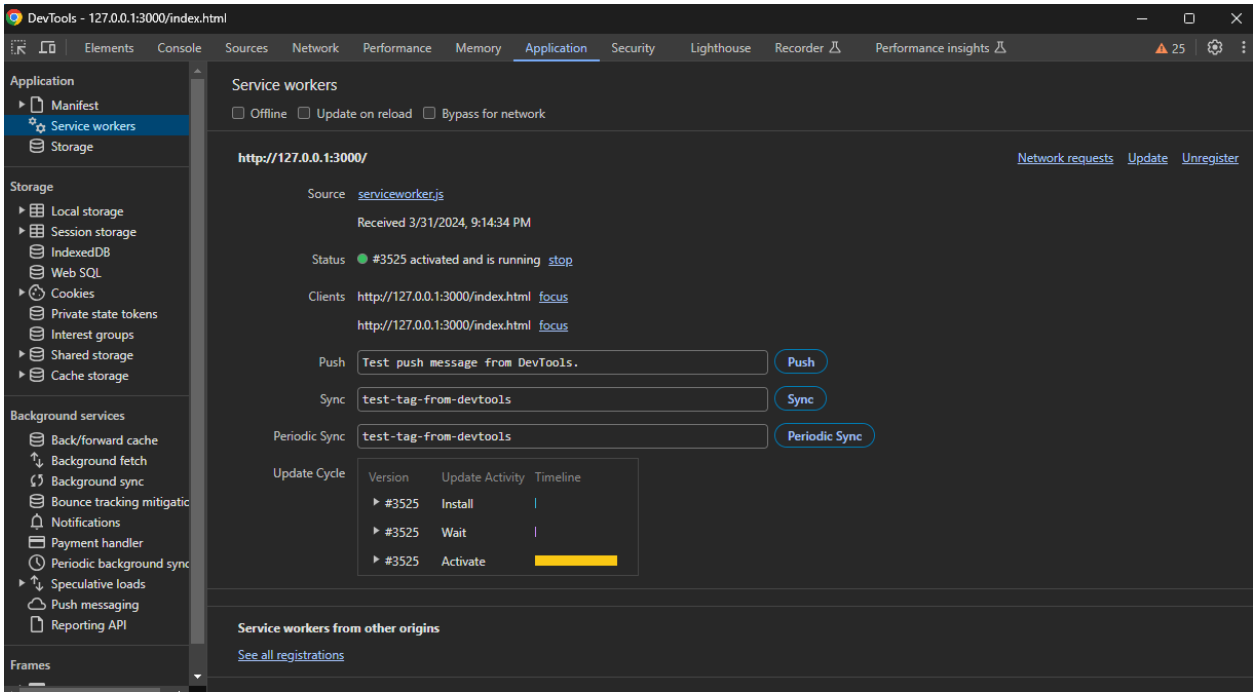
```
'use strict';

const addEventOnElem = function (elem, type, callback) {
  if (elem.length > 1) {
    for (let i = 0; i < elem.length; i++) {
      elem[i].addEventListener(type, callback);
    }
  } else {
    elem.addEventListener(type, callback);
  }
```

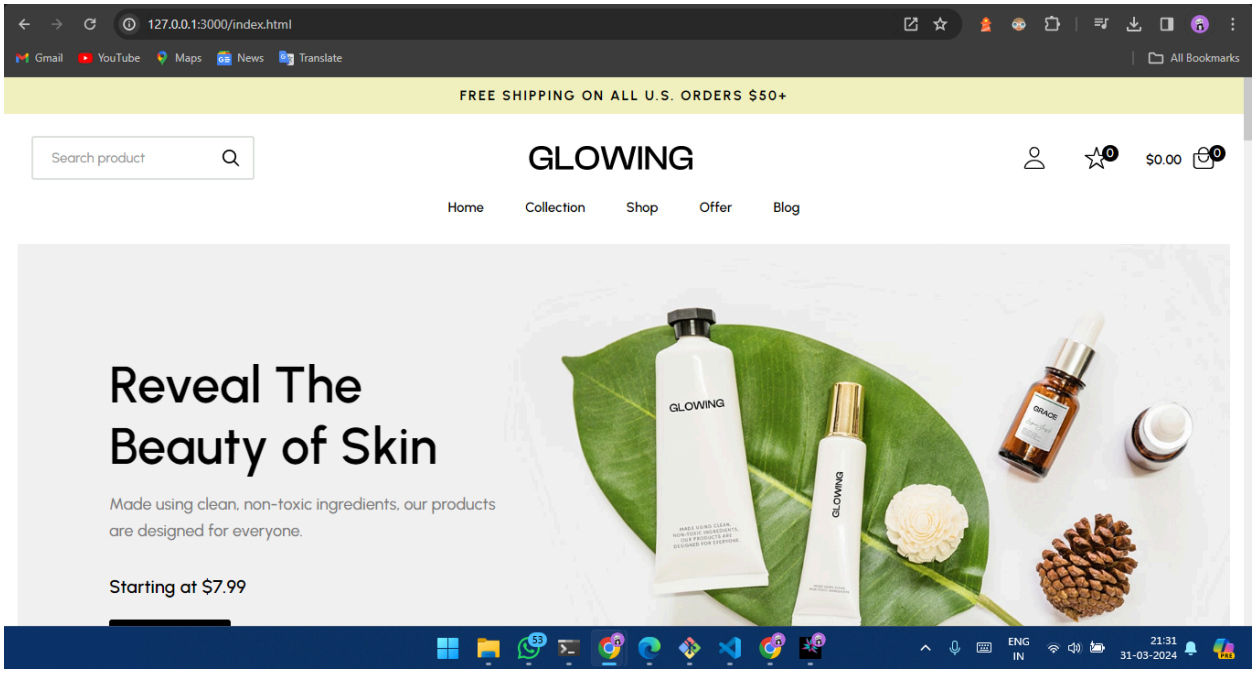
```
}
const navTogglers = document.querySelectorAll("[data-nav-toggler]");
const navbar = document.querySelector("[data-navbar]");
const navbarLinks = document.querySelectorAll("[data-nav-link]");
const overlay = document.querySelector("[data-overlay]");

const toggleNavbar = function () {
  navbar.classList.toggle("active");
  overlay.classList.toggle("active");
}

addEventOnElem(navTogglers, "click", toggleNavbar);

const closeNavbar = function () {
  navbar.classList.remove("active");
  overlay.classList.remove("active");
}

addEventOnElem(navbarLinks, "click", closeNavbar);
const header = document.querySelector("[data-header]");
const backTopBtn = document.querySelector("[data-back-top-btn]");

const headerActive = function () {
  if (window.scrollY > 150) {
    header.classList.add("active");
    backTopBtn.classList.add("active");
  } else {
    header.classList.remove("active");
    backTopBtn.classList.remove("active");
  }
}

addEventOnElem(window, "scroll", headerActive);

let lastScrolledPos = 0;

const headerSticky = function () {
  if (lastScrolledPos >= window.scrollY) {
    header.classList.remove("header-hide");
  } else {
    header.classList.add("header-hide");
  }

  lastScrolledPos = window.scrollY;
```

```
}
addEventOnElem(window, "scroll", headerSticky);
const sections = document.querySelectorAll("[data-section]");
const scrollReveal = function () {
  for (let i = 0; i < sections.length; i++) {
    if (sections[i].getBoundingClientRect().top < window.innerHeight / 2) {
      sections[i].classList.add("active");
    }
  }
}
scrollReveal();
addEventOnElem(window, "scroll", scrollReveal);
```



Project Title: Flutter:Netflix Movie App/ PWA:Glowing **Roll No. 49**





Conclusion -

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

MAD & PWA Lab
Journal

Experiment No.	09
Experiment Title.	To implement Service worker events like fetch, sync and push for E-commerce PWA
Roll No.	49
Name	Gaurang A Roarane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

Experiment - 9

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

Theory -
Service Worker is a powerful script that operates in the background of a browser independently, similar to a proxy working on the user's side. It allows developers to track network traffic, manage push notifications, and develop "offline-first" web applications using the Cache API. Here are some important aspects to note about Service Workers:

- Programmable Network Proxy: Service Worker acts as a programmable network proxy, giving developers control over how network requests from their web page are handled.

- HTTPS Requirement: Service Workers only function over HTTPS due to security reasons. They can intercept network requests and modify responses, so running them over insecure connections could pose significant security risks.

- Idle State and Restart: When not in use, a Service Worker becomes idle and restarts when needed next. Developers can't rely on a global state persisting between events. IndexedDB databases can be used to persist and reuse information across restarts.

- Fetch Event: This event allows tracking and managing of page network traffic. Developers can implement strategies such as "cache first" and "network first" approaches. For example:
  - CacheFirst: If the requested resource is cached, return the cached response; otherwise, request a new response from the network.
  - NetworkFirst: Attempt to get an updated response from the network. If successful, cache and return the new response. If unsuccessful, check if the request is cached and return it if available.

- Sync Event: Background Sync API delays a process until the internet connection is stable. For example, in an email client application, if the internet connection is lost while composing an email, Background Sync can ensure the email is sent once the connection is restored.

- Push Event: Handles push notifications received from the server. Developers can apply various actions based on received data. For instance, displaying a notification to the user. The `Notification.requestPermission();` line is necessary to show notifications to the user.

Service Workers offer developers extensive capabilities for enhancing web applications, including offline functionality, network traffic management, and push notifications. However, they require careful implementation due to security considerations and their asynchronous nature.

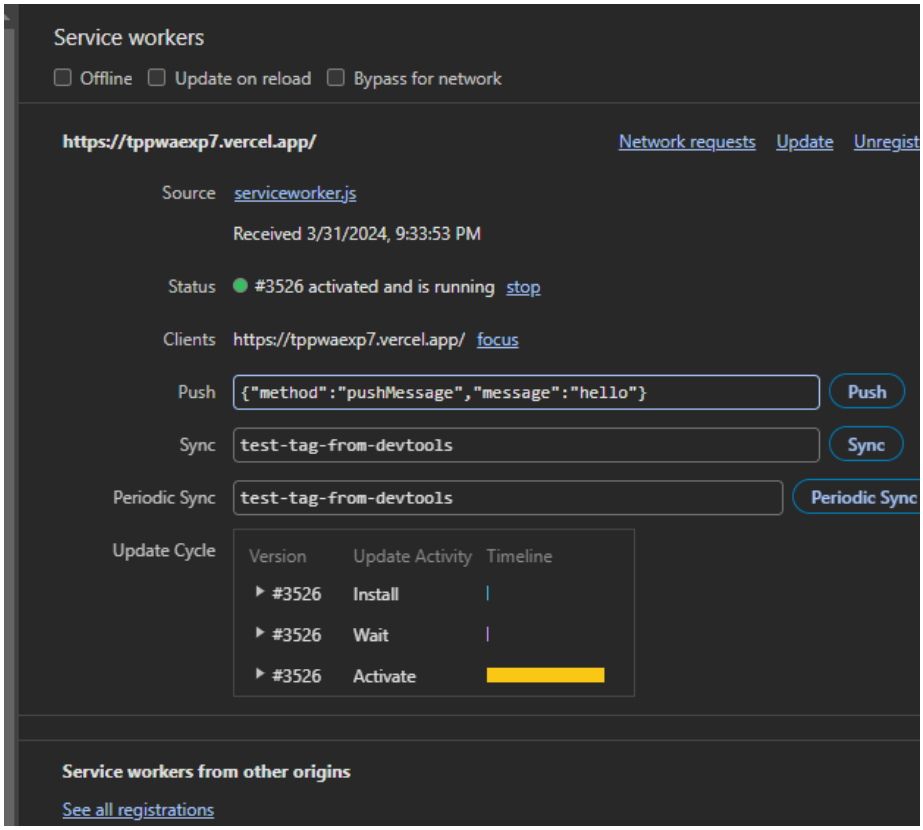
Code -
Service worker.js

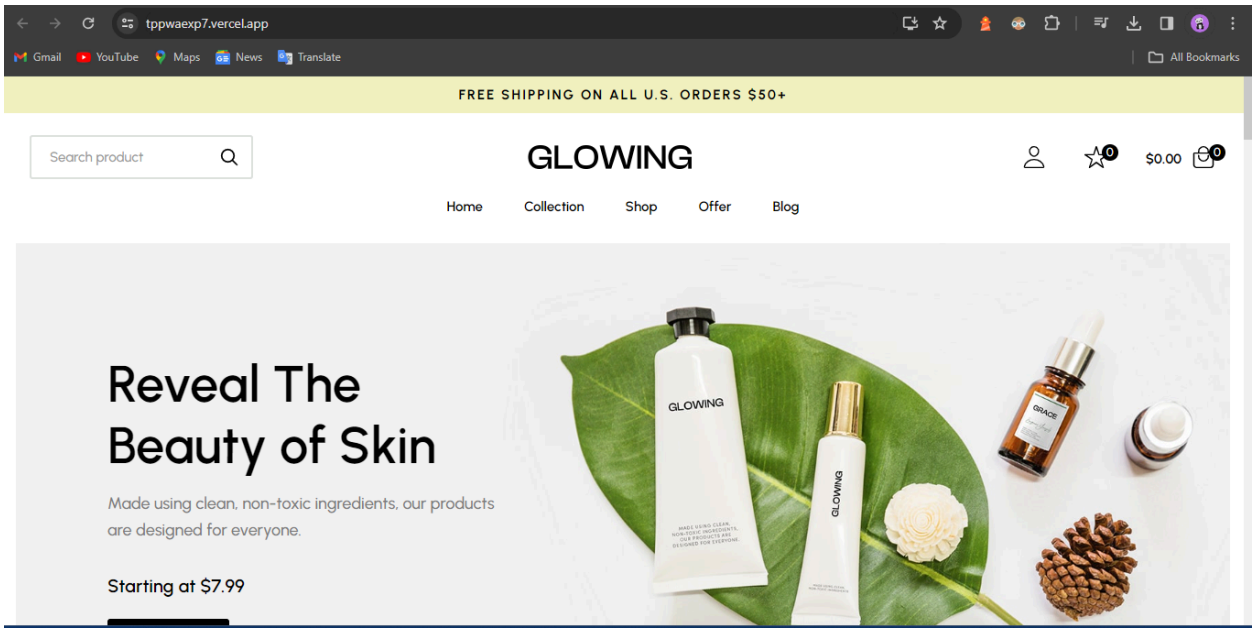
```
const CACHE_NAME = 'my-ecommerce-app-cache-v1';
const urlsToCache = [
  '/',
  'index.html',
  'assets/js/script.js',
  'assets/css/style.css',
  'serviceworker.js',
  'manifest.json',
  'offline.html'
  // Add more files to cache as needed
];

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

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

// Fetch event listener
self.addEventListener("fetch", function (event) {
  event.respondWith(checkResponse(event.request).catch(function () {
```

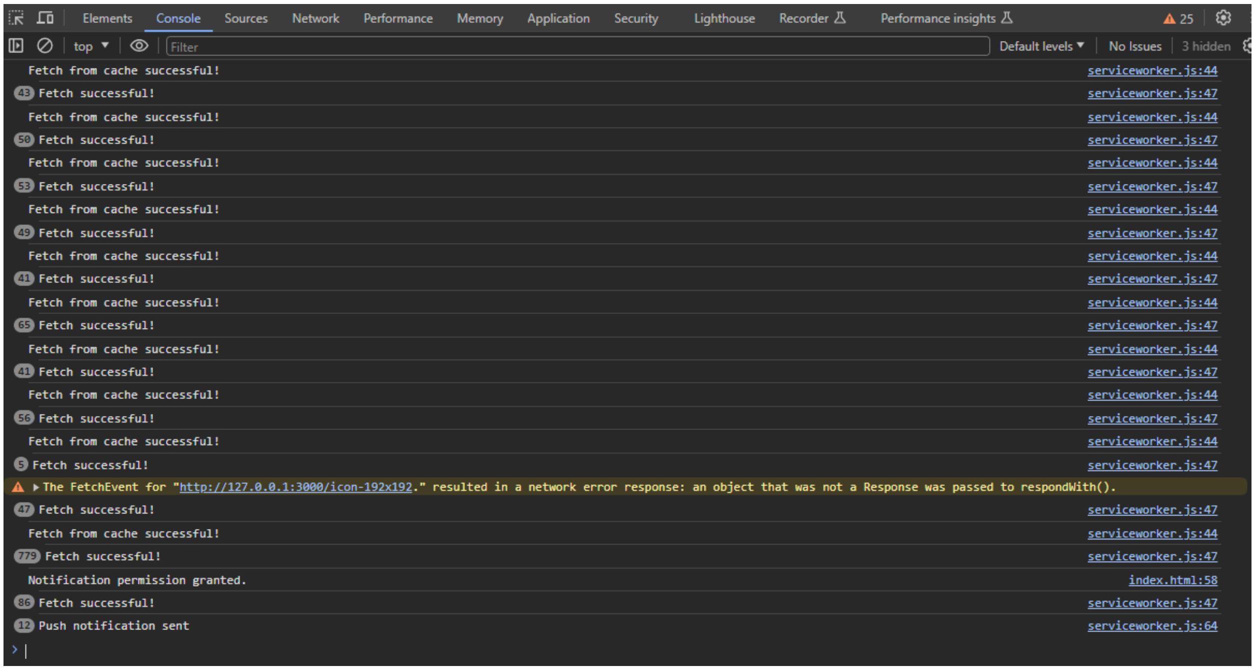
```
    console.log("Fetch from cache successful!");
    return returnFromCache(event.request);
  }));
  console.log("Fetch successful!");
  event.waitUntil(addToCache(event.request));
});

// Sync event listener
self.addEventListener('sync', function(event) {
  if (event.tag === 'syncMessage') {
    console.log("Sync successful!");
  }
});

// Push event listener
self.addEventListener("push", function (event) {
  if (event && event.data) {
    try {
      var data = event.data.json();
      if (data && data.method === "pushMessage") {
        console.log("Push notification sent");
        self.registration.showNotification("Ecommerce website", { body: data.message });
      }
    } catch (error) {
      console.error("Error parsing push data:", error);
    }
  }
});

self.addEventListener('activate', async () => {
  if (Notification.permission !== 'granted') {
    try {
      const permission = await Notification.requestPermission();
      if (permission === 'granted') {
        console.log('Notification permission granted.');
      } else {
        console.warn('Notification permission denied.');
      }
    } catch (error) {
      console.error('Failed to request notification permission:', error);
    }
  }
});
```

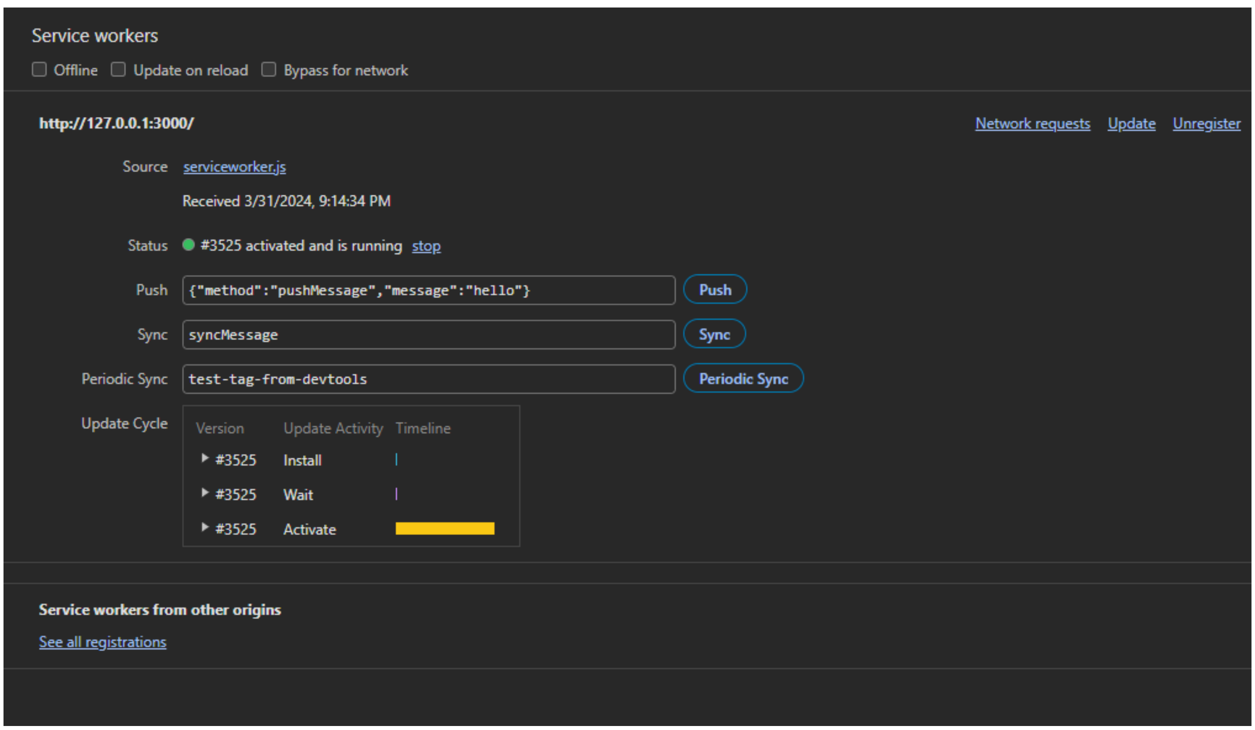
```
var checkResponse = function (request) {
  return new Promise(function (fulfill, reject) {
    fetch(request)
      .then(function (response) {
        if (response.status !== 404) {
          fulfill(response);
        } else {
          reject(new Error("Response not found"));
        }
      })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

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

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

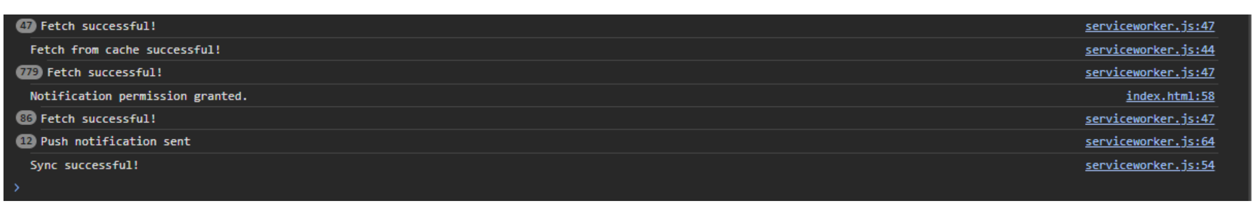
Project Title: Flutter:Netflix Movie App/ PWA:Glowing **Roll No. 49**











Conclusion -

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

MAD & PWA Lab
Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	49
Name	Gaurang A Roarane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

Experiment - 10

Aim - To study and implement deployment of Ecommerce PWA to GitHub Pages

Theory -
GitHub Pages and Firebase are both excellent platforms for hosting websites and web applications, each with its own set of features, advantages, and limitations. Here's a comparison of the two:

GitHub Pages:
Pros:
1. Familiar interface for GitHub users.
2. Easy setup process.
3. Supports Jekyll for blogging out of the box.
4. Allows custom domains.

Cons:
1. Website code is public unless using a private repository.
2. No HTTPS support for custom domains (currently).
3. Limited plugin support for Jekyll.

Firebase:
Pros:
1. Hosted by Google, ensuring reliability and scalability.
2. Offers authentication, cloud messaging, and other useful services.
3. Provides a real-time database and blob store.
4. HTTPS support with free SSL certificate provisioned for custom domains.

Cons:
1. Limited to 10 GB of data transfer per month.
2. Command-line interface only.

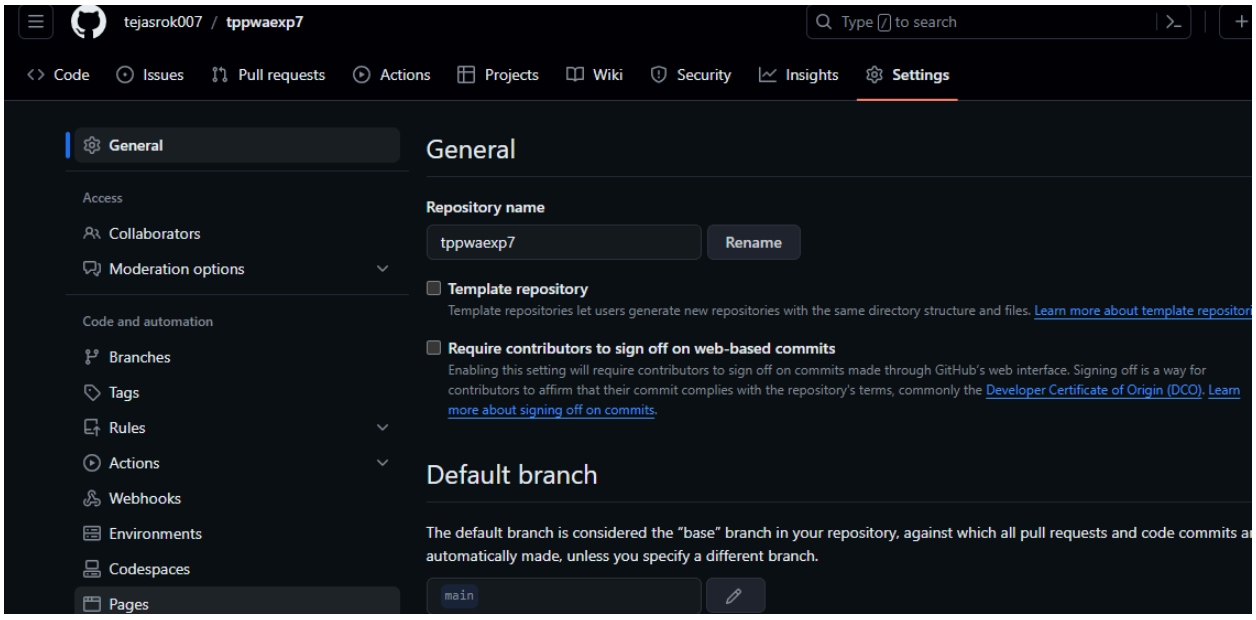
Reasons to Choose GitHub Pages:
1. Free to use and integrates seamlessly with GitHub repositories.
2. Easy setup process, particularly for static websites.
3. Great for hosting blogs with Jekyll.
4. Supports custom domains.

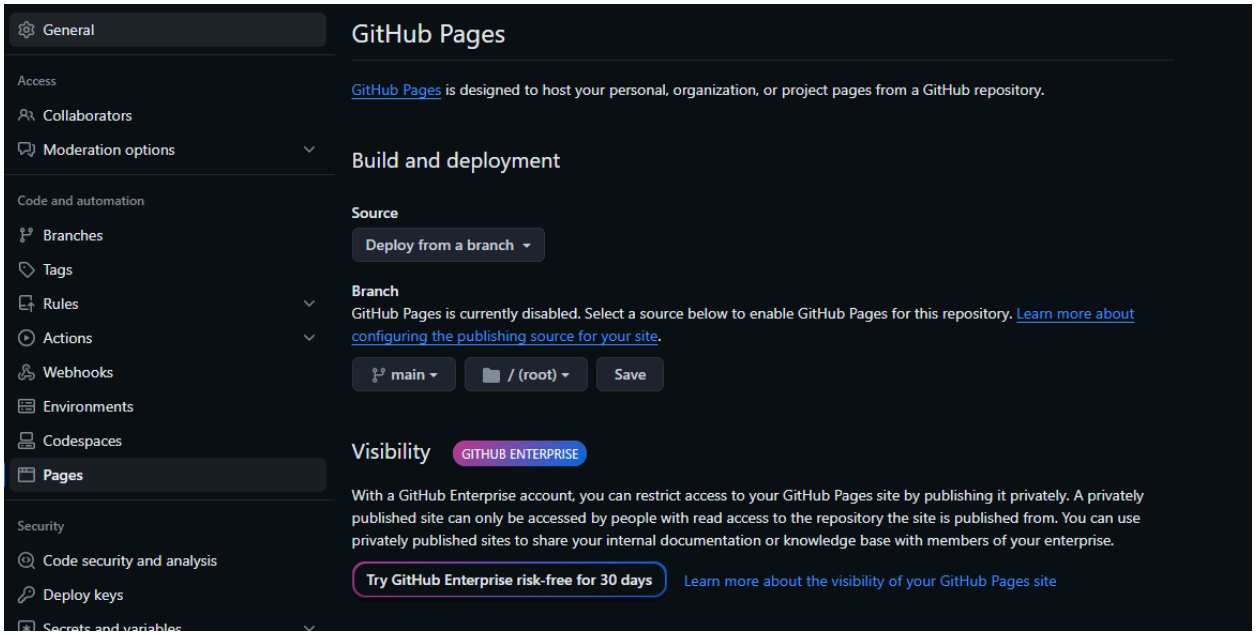
Reasons to Choose Firebase:
1. Real-time backend capabilities make it suitable for dynamic and collaborative applications.
2. Provides a broader range of services beyond hosting, such as authentication and cloud messaging.

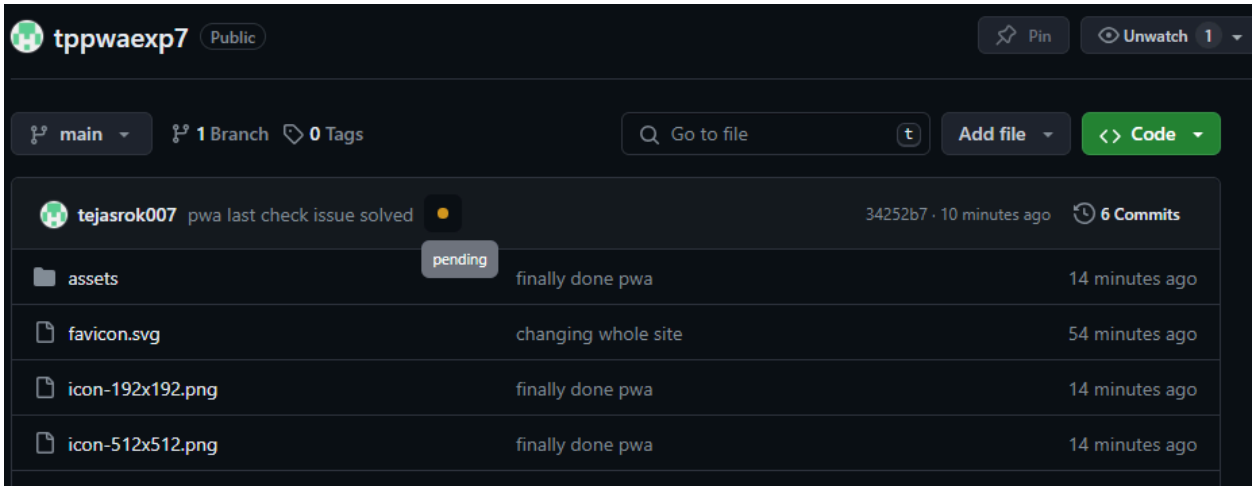
3. HTTPS support and free SSL certificates for custom domains.
4. Hosted by Google, ensuring reliability and scalability.

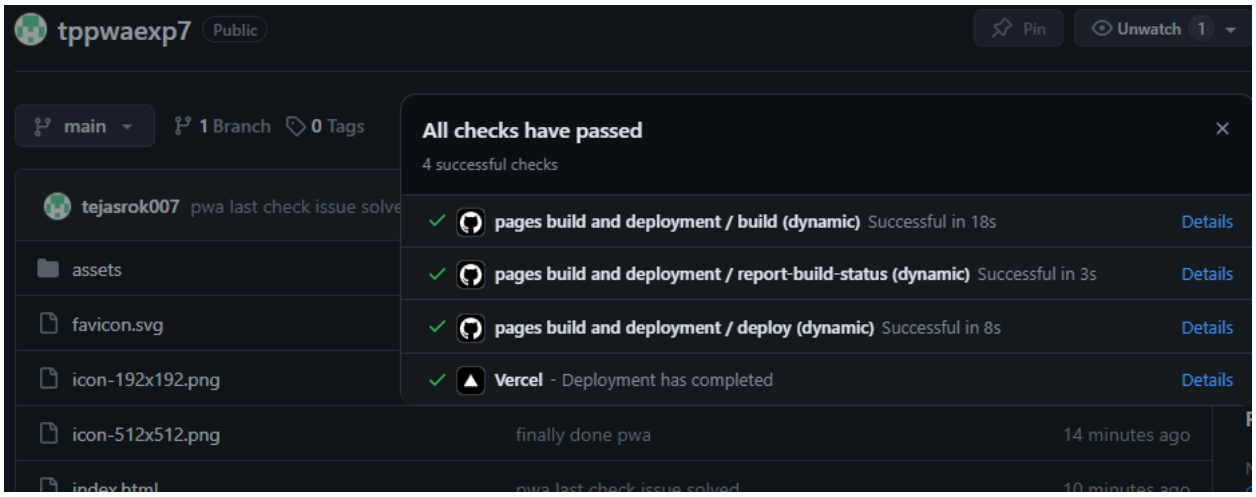
In summary, GitHub Pages is a great choice for static websites, particularly for users already familiar with GitHub, while Firebase offers a more comprehensive platform for building dynamic and real-time applications with additional backend services. Your choice should depend on the specific needs and requirements of your project.

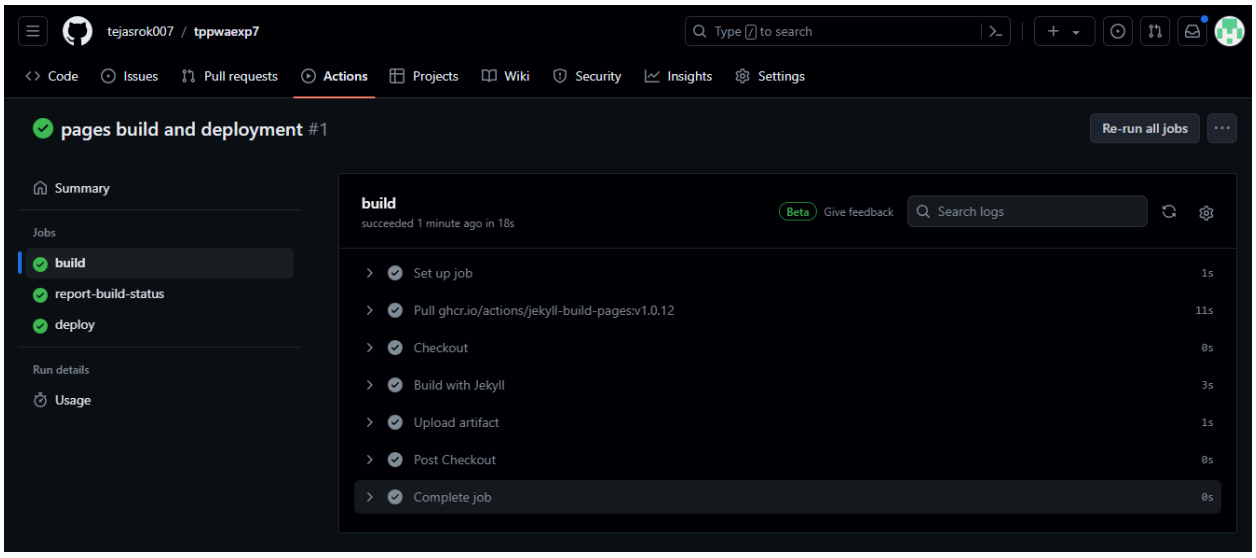
Screenshot -



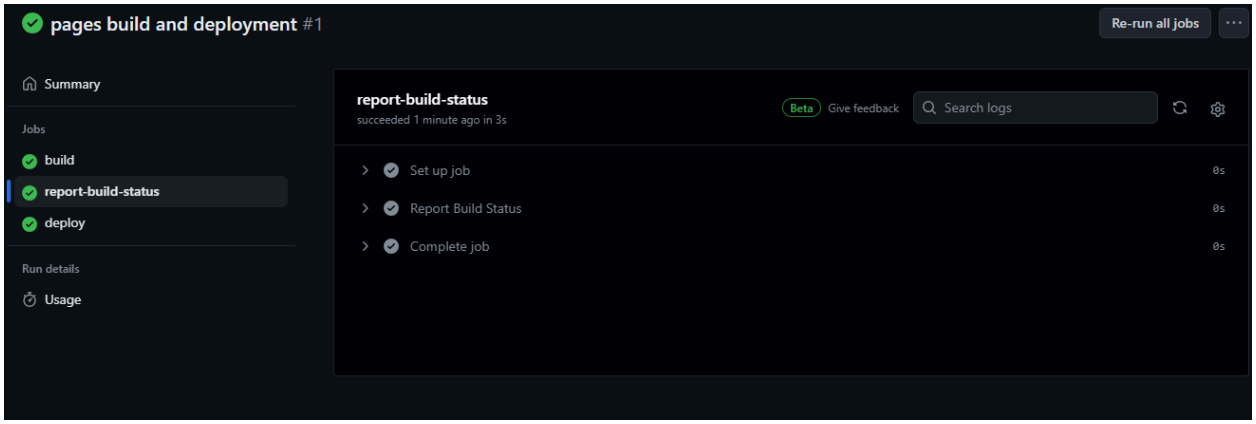


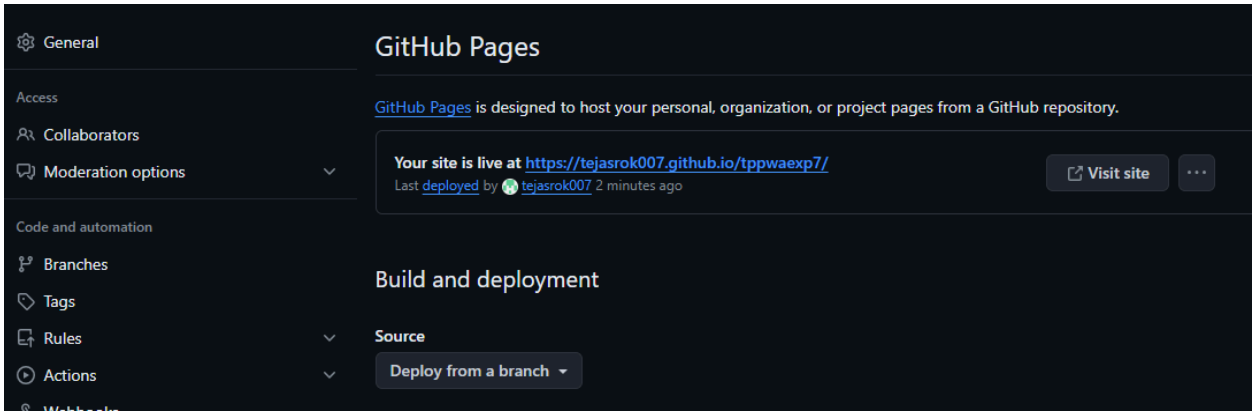

tppwaexp7
Public
Pin
Unwatch 1
main
1 Branch
0 Tags
Go to file
t
Add file
Code
tejasrok007 pwa last check issue solved
pending
34252b7 · 10 minutes ago
6 Commits
assets
finally done pwa
14 minutes ago
favicon.svg
changing whole site
54 minutes ago
icon-192x192.png
finally done pwa
14 minutes ago
icon-512x512.png
finally done pwa
14 minutes ago

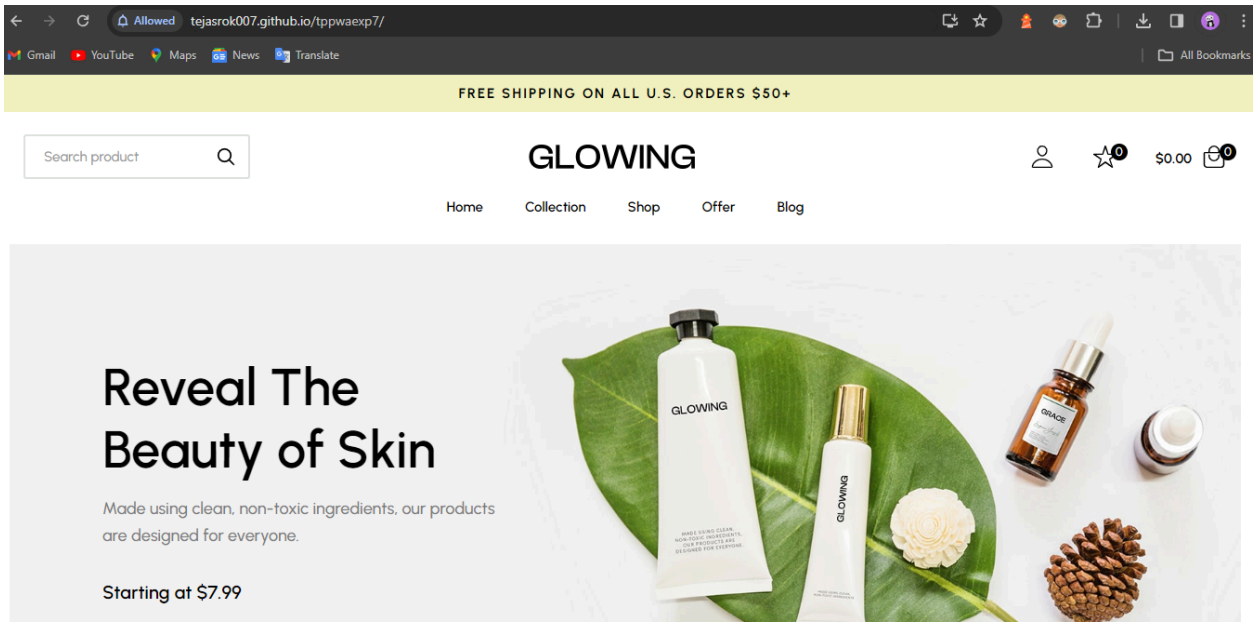

tppwaexp7
Public
Pin
Unwatch 1
main
1 Branch
0 Tags
All checks have passed
4 successful checks
tejasrok007 pwa last check issue solve
pages build and deployment / build (dynamic) Successful in 18s
Details
pages build and deployment / report-build-status (dynamic) Successful in 3s
Details
pages build and deployment / deploy (dynamic) Successful in 8s
Details
Vercel - Deployment has completed
Details
assets
favicon.svg
icon-192x192.png
icon-512x512.png
finally done pwa
14 minutes ago


tejasrok007 / tppwaexp7
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
pages build and deployment #1
Re-run all jobs
Summary
Jobs
build
report-build-status
deploy
Run details
Usage
build
succeeded 1 minute ago in 18s
Beta
Give feedback
Search logs
Set up job
1s
Pull ghcr.io/actions/jekyll-build-pages:v1.0.12
11s
Checkout
0s
Build with Jekyll
3s
Upload artifact
1s
Post Checkout
0s
Complete job
0s

Project Title: Flutter:Netflix Movie App/ PWA:Glowing **Roll No. 49**







Github - https://tejasrok007.github.io/tppwaexp7/

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

MAD & PWA Lab
Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	49
Name	Gaurang A Roarane
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 is a powerful open-source tool designed to improve the quality of web pages by providing comprehensive audits across various metrics. Here's a breakdown of its features:

1. Performance:
- Provides valuable metrics to optimize site performance, including opportunities for improvement.
- Offers insights into render-blocking resources such as stylesheets, scripts, and HTML imports.
- Combines data from both real-world usage (Field Data) and simulated conditions (Lab Data) for a comprehensive analysis.

2. Accessibility:
- Highlights potential improvements to enhance accessibility and user experience.
- Ensures that websites are easily navigable and usable for all users, including those with disabilities.
- Improving accessibility can also positively impact search engine rankings.

3. Best Practices:
- Offers recommendations to improve overall site performance and user experience, particularly for mobile sites.
- Provides suggestions for adhering to best practices in web development and design.

4. SEO:
- Evaluates a website's optimization for search engine rankings.
- Provides essential tools to analyze and improve a page's visibility in search engine results.
- Covers fundamental aspects of SEO, though not all factors may be considered.

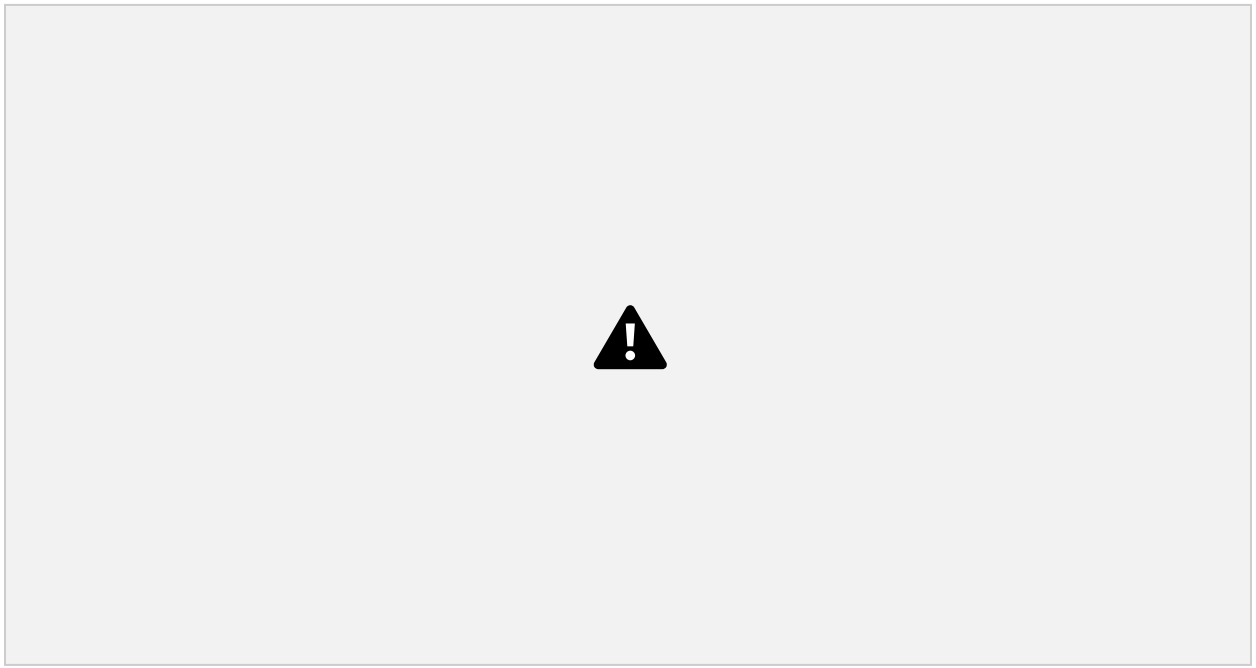
5. Progressive Web Applications (PWAs):
- Assess whether a website meets the criteria for being classified as a Progressive Web App.
- Focuses on key factors such as service worker registration, enabling push notifications, and offline capabilities.
- Helps developers enhance the user experience and engagement with modern web app features.

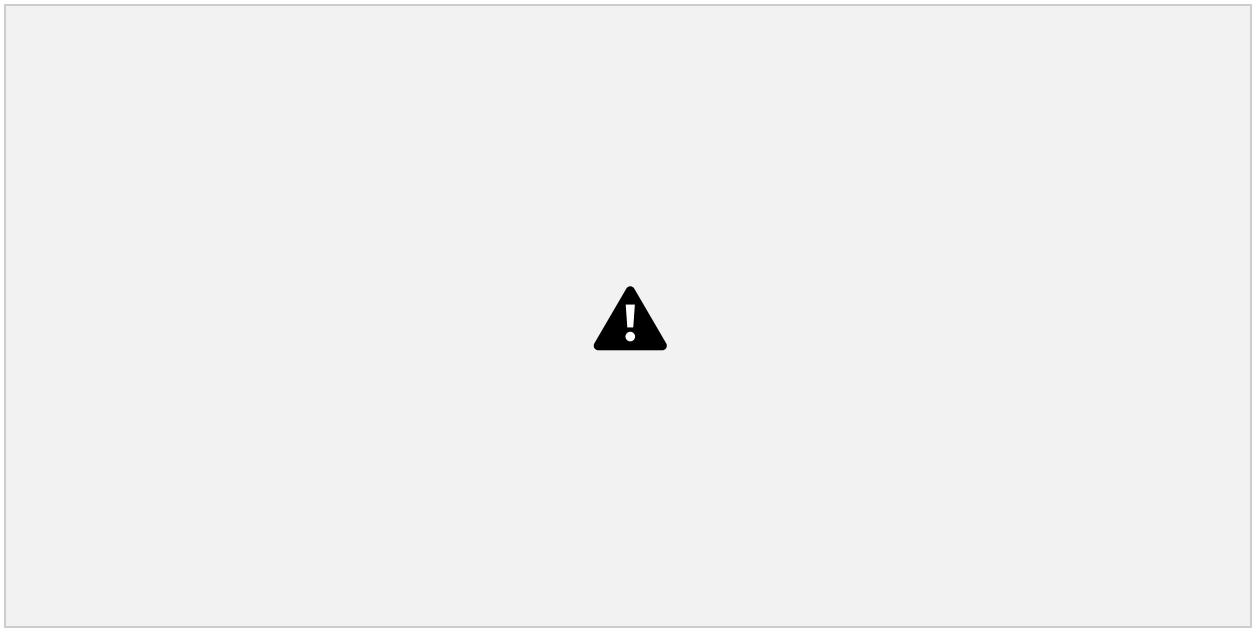
Overall, Google Lighthouse offers a comprehensive suite of metrics and recommendations to help developers optimize their websites for performance, accessibility, SEO, and PWA features. By addressing the issues identified in the Lighthouse report, developers can improve their site's user experience, search engine visibility, and overall quality.

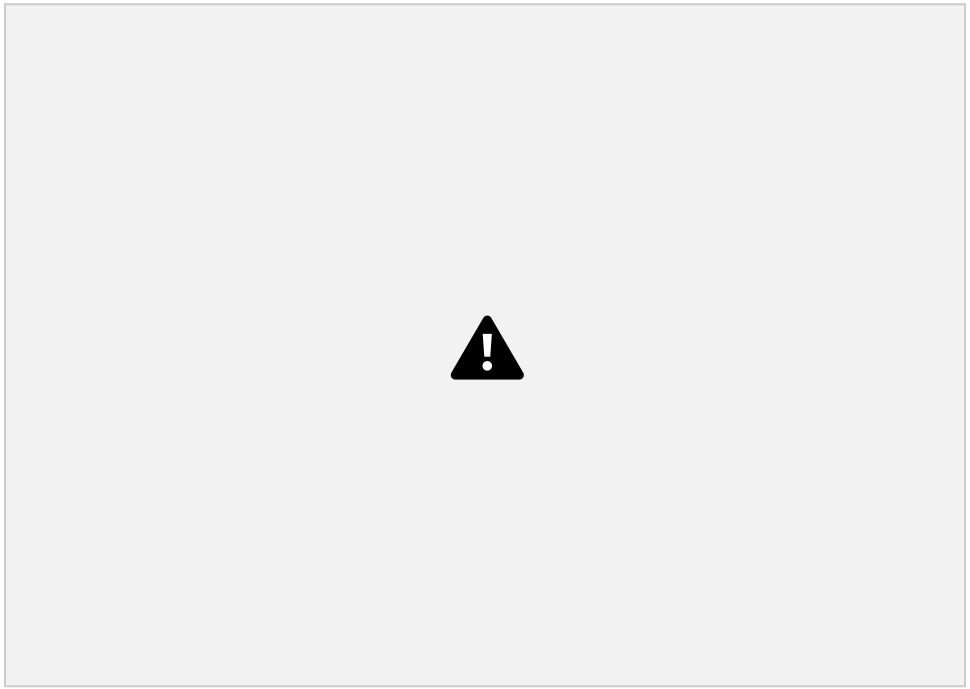
```
{
   "name":"PWA Tutorial",
   "short_name":"PWA",
```

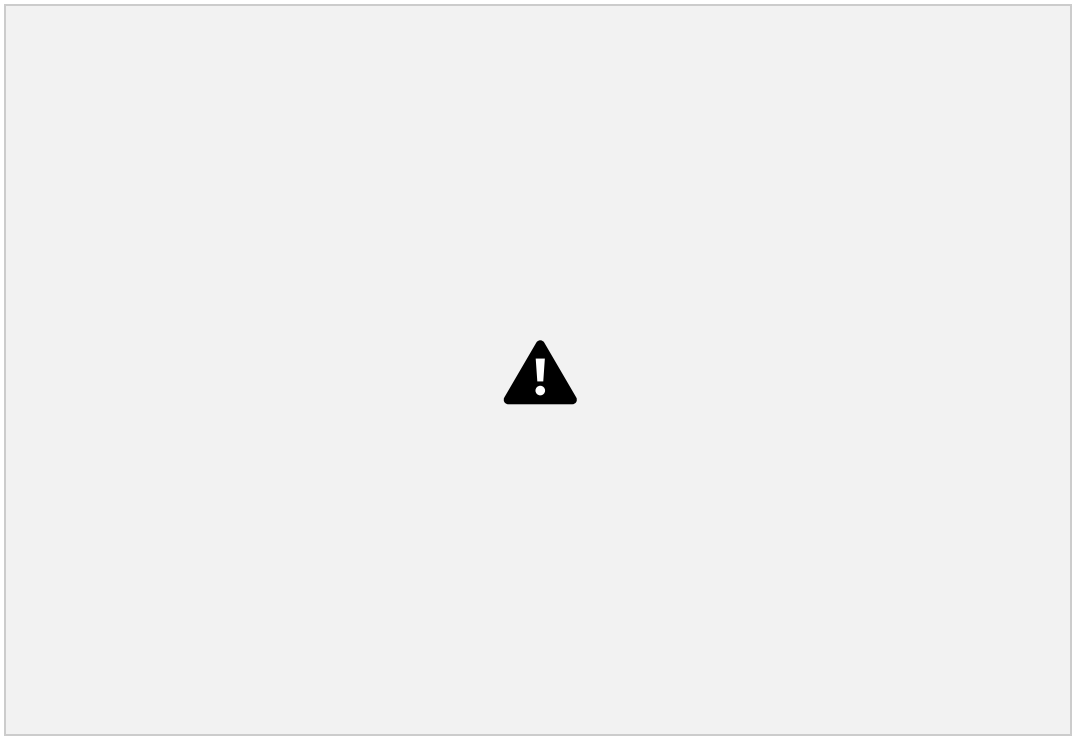
```
    "start_url":"index.html",
    "display":"standalone",
    "background_color":"#5900b3",
    "theme_color":"black",
    "scope": ".",
    "description":"This is a PWA tutorial.",
    "icons":[
    {
    "src":"icon-192x192.png",
    "sizes":"192x192",
    "type":"image/png",
    "purpose":"any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

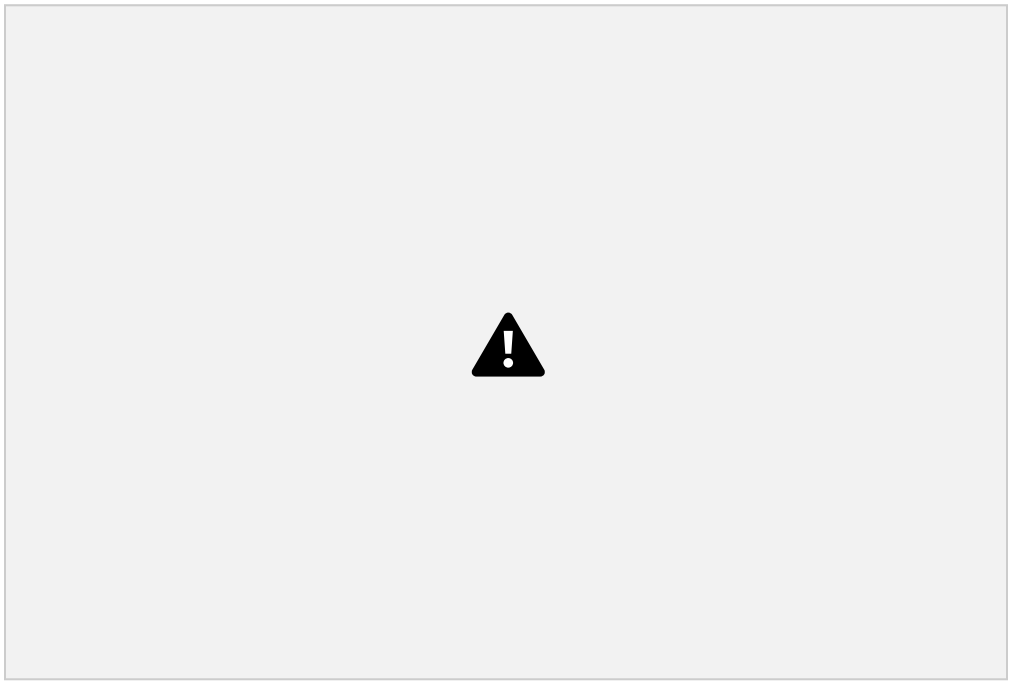
ScreenShot -

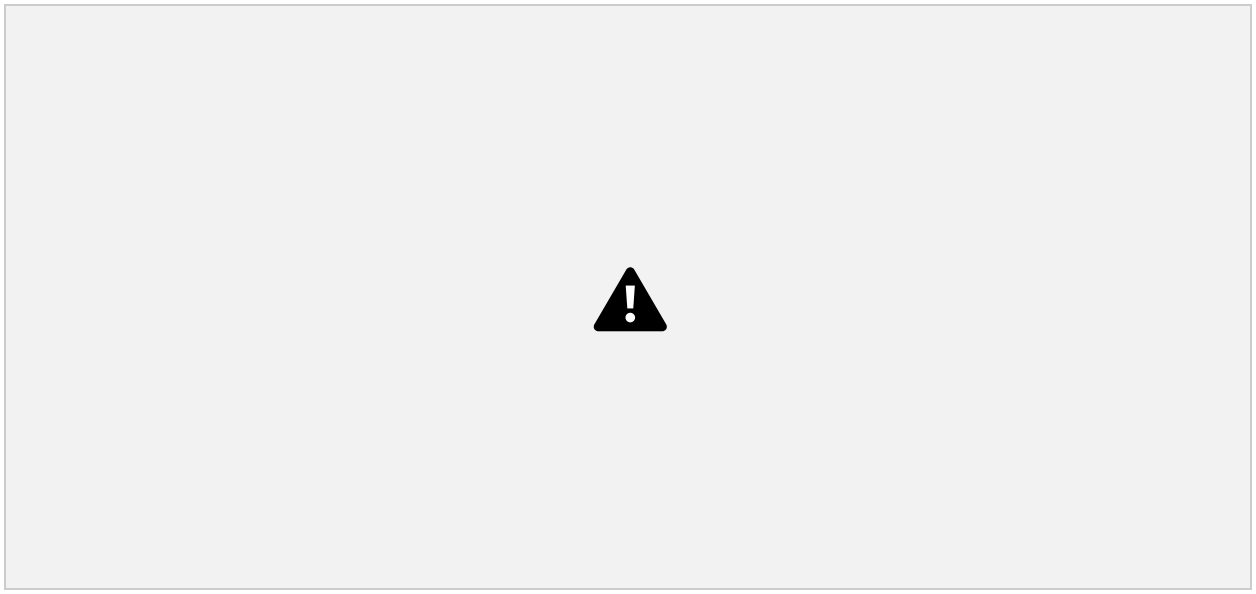


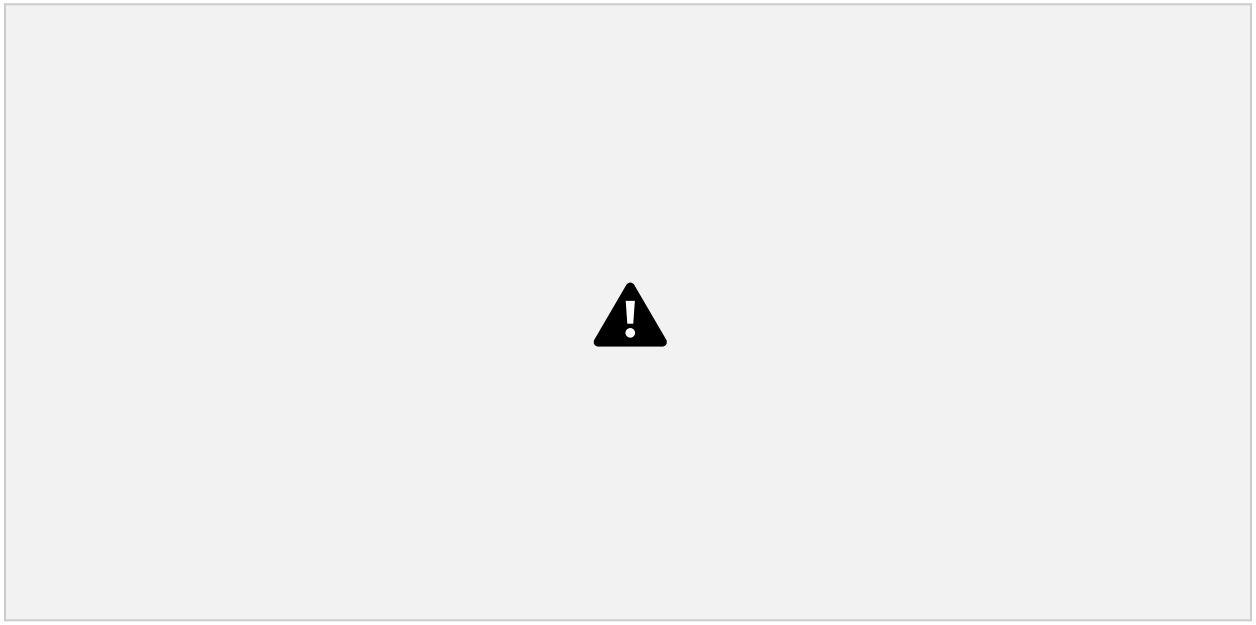


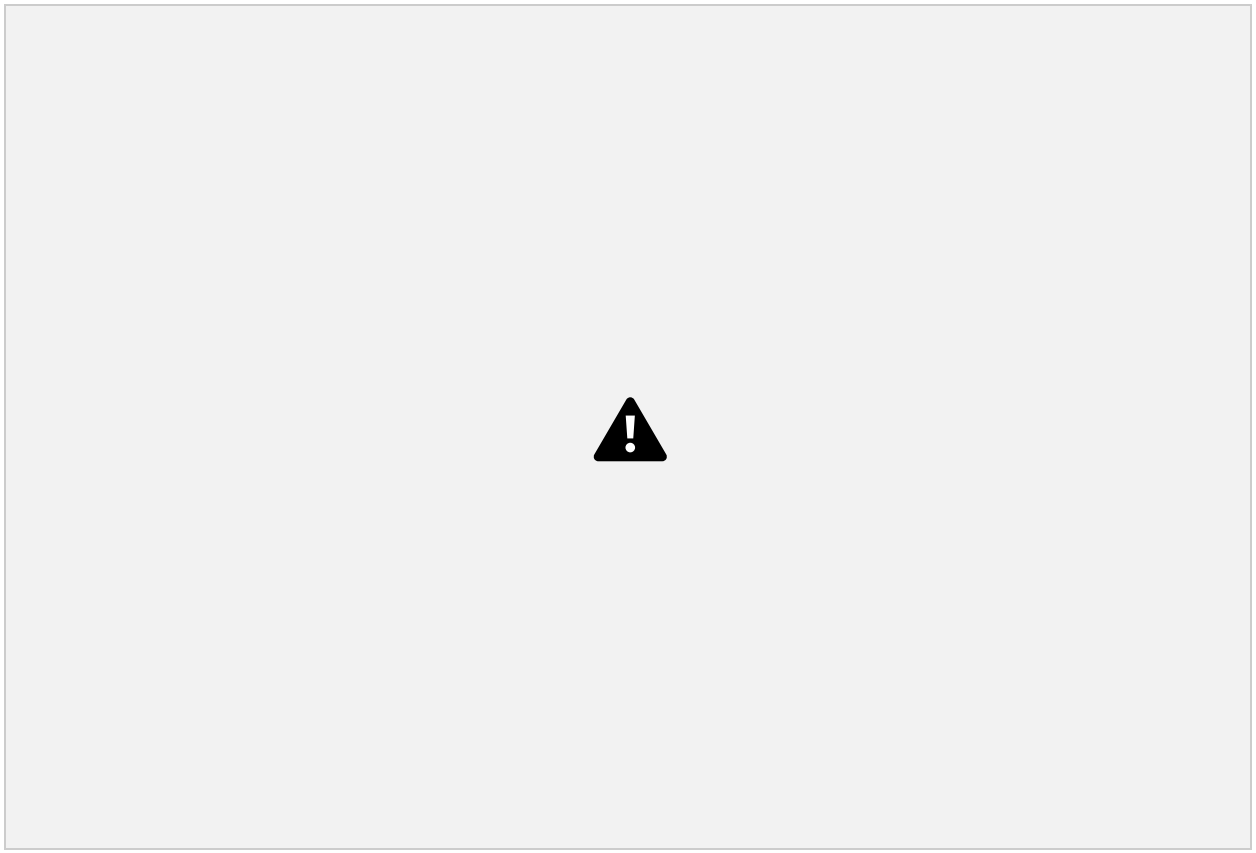


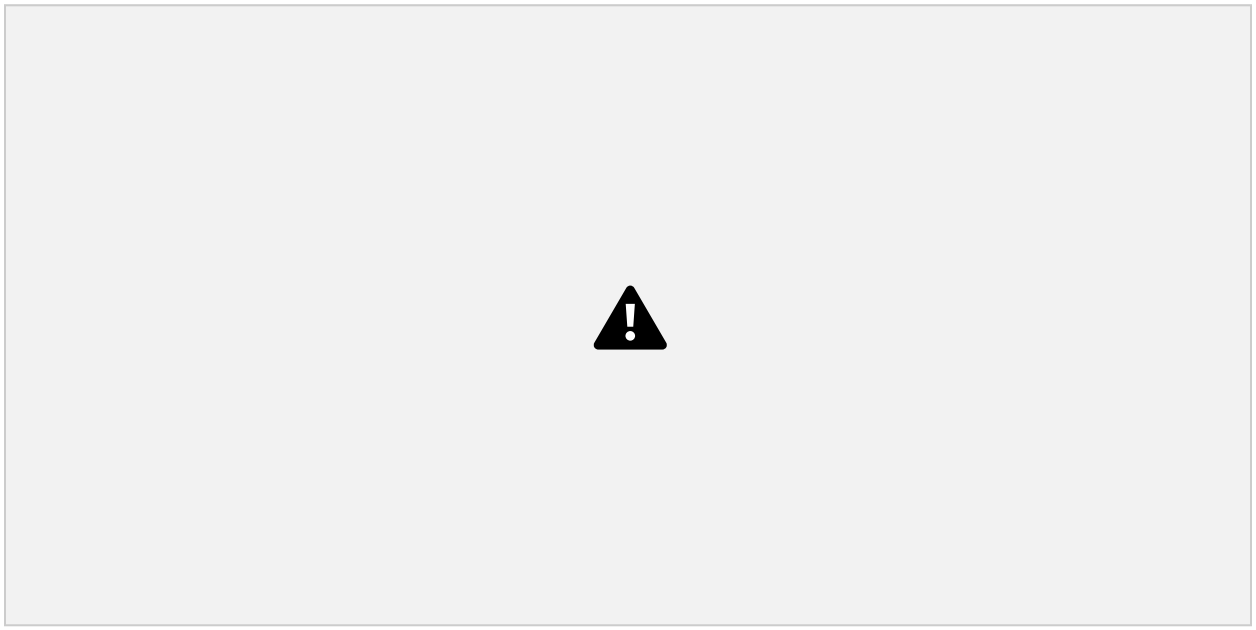












Conclusion -

In this experiment, we have successfully used Google Lighthouse PWA Analysis Tool for testing the PWA functioning.

MAD & PWA Lab
Journal

Experiment No.	Assignment-1
Assignment 1 Questions	1. Flutter Overview:Explain the key features and advantages of using Flutter for mobile app development. Discuss how the Flutter framework differs from traditional approaches and why it has gained popularity in the developer community. 2. Widget Tree and Composition: Describe the concept of the widget tree in Flutter. Explain how widget composition is used to build complex user interfaces. Provide examples of commonly used widgets and their roles in creating a widget tree. 3. State Management in Flutter: Discuss the importance of state management in Flutter applications. Compare and contrast the different state management approaches available in Flutter, such as setState, Provider, and Riverpod. Provide scenarios where each approach is suitable. 4. Firebase Integration in Flutter: Explain the process of integrating Firebase with a Flutter application. Discuss the benefits of using Firebase as a backend solution. Highlight the Firebase services commonly used in Flutter development and provide a brief overview of how data synchronization is achieved.
Roll No.	49
Name	Gaurang A Roarane
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:	

Name:- Gaurang A. Raorane
D15A 49

Flutter

Assignment 1

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.

Ans:- Fluttir is Google's software development kit for building iOS and Android apps. Flutter allows you to create cross-platform apps that provide native performance. Apps created with flutter feature beautiful and intuitive design and are able to run animations smoothly.

Key features:-

1) Single Codebase :- Code written once can be deployed to Android and iOS platforms.
2) Hot Reload :- allows developer to instantly see the impact of the changes made.
3) Rich Widget Library:- Fluttir provides comprehensive set of highly customizable widgets for building UI's.
4) Performance:- Flutter compiles to native ARM code, resulting in high-performance applications.
5) Widget-based Architecture:- Flutter's architecture revolves around widget making it modular and easy to organize code.
6) Open Source:- Flutter encourages collaboration and contributions from the community.
7) Cross Platform Development:- Flutter supports cross platform development for mobile, web and Desktop Applications.

Widget tree is a fundamental concept that represents the hierarchy of UI elements in applications.
Structure begins with root widget, typically representing entire application. This root widget can have child widget and each child widget can further have its own widget forming tree like structure.
Examples:-

1) Material App:- Represents root widget of the application.
2) Scaffold:- Provides a basic structure for the visual element of an app, such as app bar, body and bottom.
3) Column/ Row:- Arranges child widget vertically or horizontally.
4) Container - Basic box model that can contain other widgets and defines prop. like padding.

Q.3) State management in Flutter - Discuss the importance of state management in flutter application. Compare and contrast the different state management approaches available in Flutter, such as setState, Provider, and Riverpod. Provide scenarios where each approach is suitable.

→ State management is a critical aspect of flutter development, playing a pivotal role in ensuring a responsive and efficient user interface.
Effective state management strategies is key to building robust and maintainable applications.

Traditional Approach:-

1) Developers had to maintain different codebase for iOS and android.
2) Achieving a consistent UI across different platforms was a problem.
3) Changing in code required rebuilding and restarting of application.

Such traditional approaches have been tackled and these are some factors that makes flutter framework different from other approaches.

2) Widget tree and Composition - Describe the concept of widget tree in flutter. Explain how widget composition is used to build complex UI. Provide examples of commonly used widgets and their roles in creating a widget tree.

Ans:- Widget composition is a powerful concept in flutter that involves combining multiple smaller widgets to create more complex and sophisticated UI. Instead of creating large and monolithic UI components, developers can break down the UI into smaller components and compose them hierarchially to form a comprehensive UI. Some of the benefits are:-

1) Reusability of code.
2) Modularity
3) Readability
4) Flexibility
5) Scalability
6) Customization.

③ Riverpod - An extension of provider that introduces improvements in terms of readability, scalability and testability.
- Use case:- Offers improved syntax and advanced features compared to provider
- Pros:- Provides more readability and testability
- Cons:- Slightly deeper learning curve compared to provider.

Q. 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 service commonly used in flutter development and provide a brief overview of how data synchronization is achieved.

Ans:- Integrating firebase with flutter applications:-
1) Create firebase project
2) Register your app.
3) Add firebase SDK to flutter project.
4) Add flutter SDK dependencies to pubspec.yaml file
dependencies:
firebase_core: ^1.10.6
firebase_auth: ^4.10.6
4) Initialize firebase in your App:-
'firebase.initializeApp()' in the 'main()' function.
5) Use firebase services in your App

Benefits of using Firebase as backend solution:-

1) Real-time database
2) Authentication
3) Cloud firestore
4) Cloud functions
5) Cloud storage
6) Easy integration
7) Scalability
8) Authentication providers
9) Analytics and Crash Reporting
10) Cloud functions.

Data synchronization in Flutter:-

Firebase cloud firestore employs a real time synchronization model to ensure that data changes are instantly reflected across all connected devices. The process includes:-

1) Collections and documents:- Data in firestore is organized in collections, which can contain multiple documents. Each document is set of key value pair.
2) Listeners and Observable:- Flutter dependencies can set up listeners to changes in particular collection or document.
3) Real-time updates:- when the data-changes in firestore database, the changes are instantly pushed to all connected clients that have set up listeners for that particular data.
4) Offline support:- Firestore provides offline support, allowing users to interact with the app even when offline data modification made while offline are stored locally and synced with the server once the device is online.

Importance of state management in Flutter.

1. Reactivity and UI updates.
2. Performance Optimization.
3. Code Organization and Maintainability
4. Scalability
5. Testing

Comparison of State management approaches:-

(1) setState():- The simplest and built in way to manage local state with a state.

- Use case:- Suitable for small to medium sized application
  - Ideal for mangging simple UI specific states.

- Pros:- straightforward and easy to use
  - No external dependency
- Cons:- limited to scope of single widget
  - May lead to widget rebuilds higher in widget tree.

(2) Provider:- A popular state mangement solution that provides global & easily accessible solution.

- Use case:- Good for managing global state and sharing data b/w widgets.
- Pros:- Simple and easy to implement, offers good performance.
- Cons:- can be considered overkilled for small application.

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.	49
Name	Gaurang A Roarane
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:	

Gaurang. A. Raorane
D15A 49 C

PWA Assignment-2

Q.1) Define Progressive Web App (PWA) and explain its significance in Modern web development. Discuss the key characteristics that differentiate PWA's from ~~dtt~~ traditional mobile app.

Ans. A progressive web app (PWA) is a type of web application that uses modern web capabilities to deliver an app-like experience to users. PWA's are built using standard web technologies such as HTML, CSS, & JS but are designed to more native-like experience for users.

Key characteristics of Progressive Web Apps:-

1) Progressive Enhancement:- They are built using progressive enhancement principles, meaning they provide a basic principle for all users.

2) Responsive design:- PWA's are designed to feel and behave like native mobile apps. They include multiple features such as animations gestures and transitions to provide more immersive experience.

3) Reliability:- PWA's are designed to reliable, even when users are offline or has slow internet connection. They cache content or resource to ensure that app remains functional.

4) Fast Performance

5) Discoverability:- PWA's are easily discoverable and shareable, as they are built using standard web technologies and can be accessed via a URL.

6) Security:- PWA's are served over HTTPS to ensure security.

7) Cross-Platform compatibility.

The significance of PWA's in modern web technologies lies in their ability to bridge the gap between web and app native experience. By combining the reach and accessibility of the web with functionality and engagement of native apps. They allow developers to create fast, reliable and engaging experience that work seamlessly.

Q.2> Define responsive web design and explain its importance in the context of progressive web apps. Compare and contrast experience, fluid and adaptive web design approaches.

→ Importance for PWA:-

→ Foundation for app like experience - RWD ensures proper layout and navigation across devices, mimicking the adaptability of native apps.

→ Accessibility - A responsive design make PWA accessible to wider audience using diverse devices.

→ SEO benefits.

Feature	Responsive	Fluid	Adaptive
Layout	1) Flexible adapts to any screen size.	1) Continuously adjust based on relative units	1) Uses multiple fixed width layouts.
Development & efforts	1) Moderate	1) Less effort for basic layout	2) More efforts to create multiple layout
Control	3) Good control over overall layout	3) Less control over element appearances on extreme sizes	3) High control over layout on each device category.
Suitability for PWA's	4) Ideal foundation due to versatility	4) Can be used but may require adjustment for optimal experiences	4) Less common for PWA due to development overhead.

Q.34 Describe the lifecycle of service workers (including registration, installation and activation phases.

Ans. Service workers are crucial components of progressive web Apps (PWAs) that enable feature such as offline support, push notification and background synchronisation. The lifecycle of service worker includes several ways phases registration, installation and activation.

1) Registration:- 1st phase of lifecycle occurs in the main Javascript file of web Application.

→ Registration is done using the navigator service worker register() method, which takes the path to service worker file as parameter.

→ This method returns a promise that resolve to a service worker registration object, which can be used to interact with service worker.

Installation:-

→ Once service worker browser registered browser downloads and starts installation process of service worker script.

→ During installation the service worker script is executed, and the install event is fired. 'Install' event handler, developers can define what resources should be cached for offline access using technologies like caching strategy or pre-caching.

→ Installation process is critical for setting up the service worker's initial cache and preparing it to take control of web application.

Activation:-

→ After installation, Activation phase occurs when service worker is ready to take control of web application. During activation, 'activate' event is fired, giving service worker an opportunity to clean up any old caches or perform other necessary task for application functionality.

- Developers can use the 'activate' event handler to manage cache cleanup, versioning and other activation related tasks.
- service remains active until it is explicitly unregister or updated. [It's important to note that service workers run separately from main web page and operate on a different thread, allowing them to perform tasks like network interception and caching w/o blocking UI].

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

Ans. → IndexedDB is a browser API for storing large amounts of unstructured data on the client-side. It functions like a NoSQL database, allowing you to store key-value pairs and organize data in object stores with indexes for efficient retrieval.

Benefits ~~to~~ of IndexedDB with service-worker:-

→ offline data access:- Improve user experience by allowing interaction with the PWA even without an internet connection.

→ Faster load times:- cached data retrieval from indexed can be served much faster than fetching it from the service again.

→ Improved reliability:- Reduces dependencies on a constant network connection, making PWA more reliable.

→ security:- IndexedDB provides a secure storage mechanism within the browser's sandbox environment.

→ Data management:- PWA should have mechanism to manage storage space and avoid excessive data accumulation in IndexedDB.