

# **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 **Jatin Talreja** 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**

**Project Title: Chat-Gpt and Ecommerce PWA App Roll No. 61**

**Year/Sem/Class :** D15A A.Y.: 23-24 **Faculty Incharge :** Mrs. Kajal Joseph.

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

**Email :** [kajal.jewani@ves.ac.in](mailto: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.

## **Project Title: Chat-Gpt and Ecommerce PWA App Roll No. 61**

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

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

### **Program specific Outcomes**

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

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

**Lab Objectives:**

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

**Lab Outcomes:**

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

## **Index**

<b>Sr. No</b>	<b>Experiment Title</b>	<b>LO</b>	<b>DOP</b>	<b>DOS</b>	<b>Grade</b>
<b>1.</b>	To install and configure the Flutter Environment	LO1	19/1/24	2/2/24	10
<b>2.</b>	To design Flutter UI by including common widgets.	LO2	26/2/24	2/2/24	15
<b>3.</b>	To include icons, images, fonts in Flutter app	LO2	2/2/24	9/2/24	13
<b>4.</b>	To create an interactive Form using form widget	LO2	9/2/24	16/2/24	13
<b>5.</b>	To apply navigation, routing and gestures in Flutter App	LO2	16/2/24	23/2/24	13
<b>6.</b>	To Connect Flutter UI with fireBase database	LO3	23/2/24	8/3/24	13
<b>7.</b>	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.	LO4	15/3/24	22/3/24	14
<b>8.</b>	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	22/3/24	29/3/24	14
<b>9.</b>	To implement Service worker events like fetch, sync and push for E-commerce PWA	LO5	29/3/24	1/4/24	14
<b>10.</b>	To study and implement deployment of Ecommerce PWA to GitHub Pages.	LO5	29/3/24	1/4/24	14
<b>11.</b>	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.	LO6	29/3/24	1/4/24	14
<b>12.</b>	Assignment-1	LO1, LO2, LO3	28/1/24	5/2/24	4
<b>13.</b>	Assignment-2	LO4, LO5, LO6	14/3/24	21/3/24	4

## MAD & PWA Lab

### Journal

Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	61
Name	Jatin talreja
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	10

Name: Jatin talreja  
Division: D15A  
Roll No:61  
Batch: C

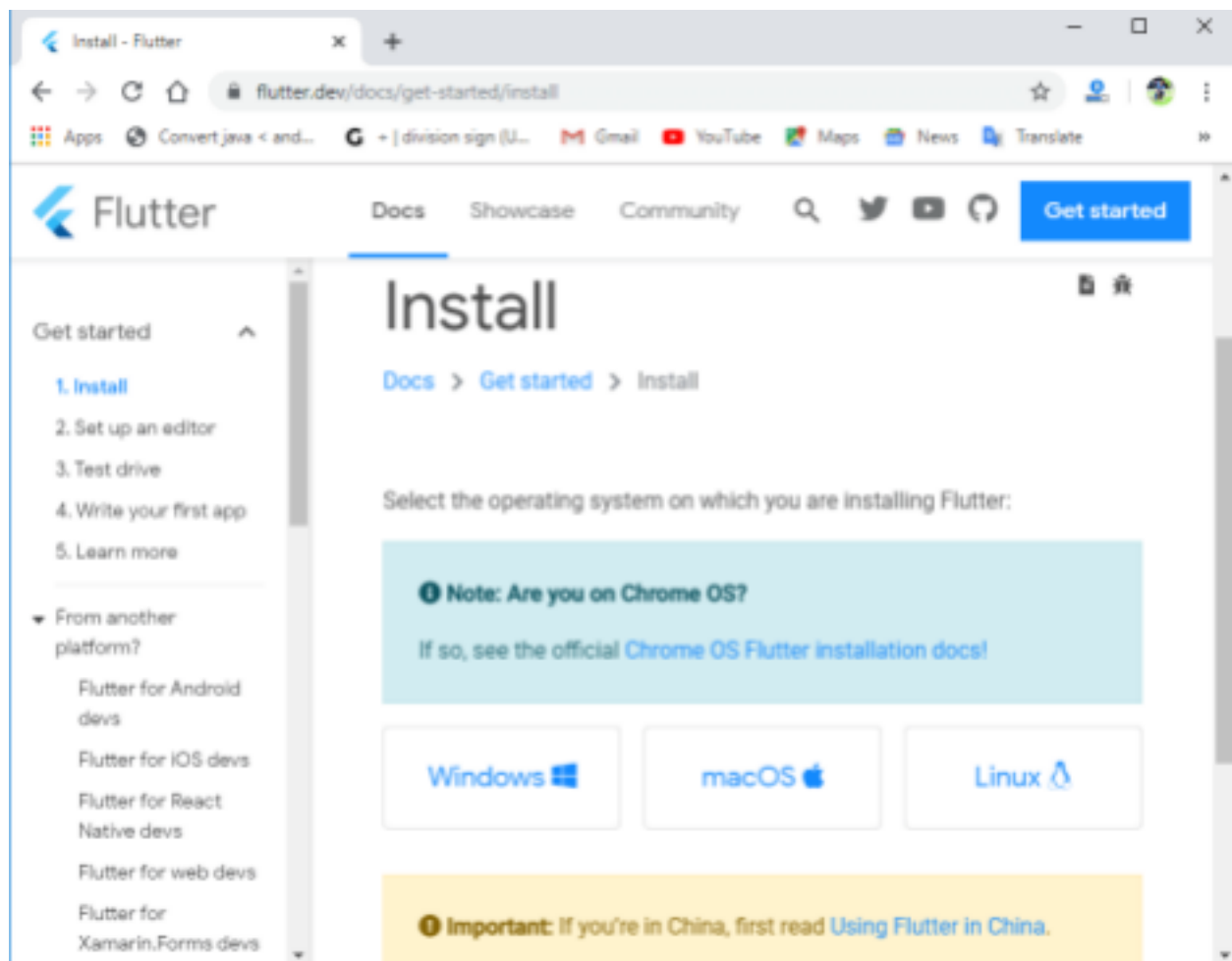
## Experiment No. 1

Aim: To Install and Configure Flutter Environment

Pre Requisites:

### Install the Flutter SDK

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

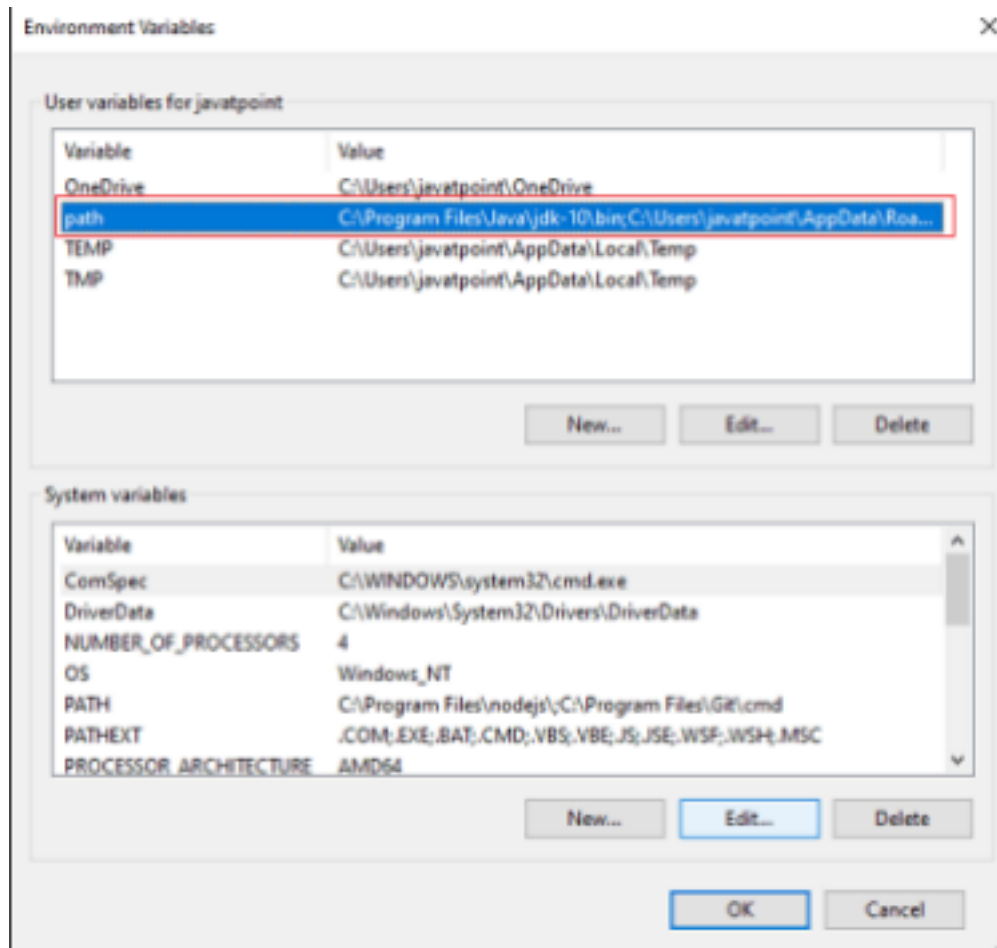


**Step 2:** Next, to download the latest Flutter SDK, click on the Windows **icon**. Here, you will find the download link for [SDK](#).

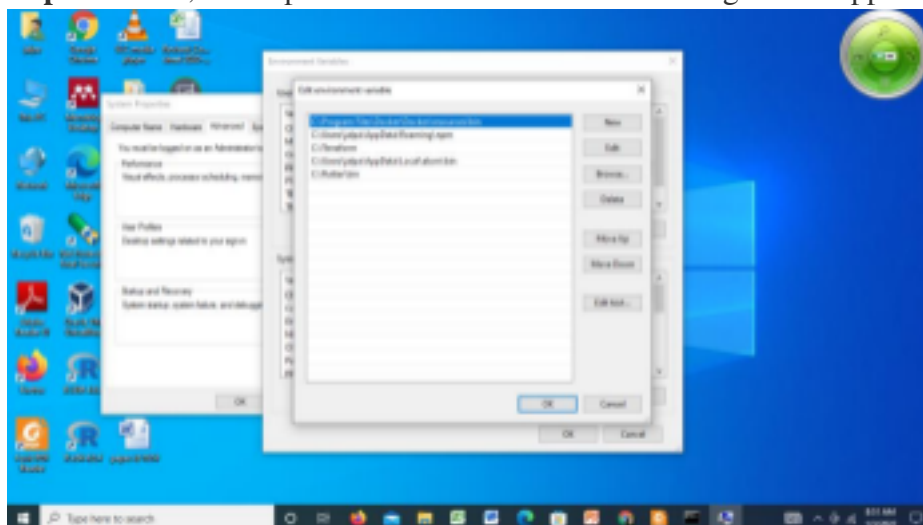
**Step 3:** When your download is complete, extract the **zip** file and place it in the desired installation folder or location, for example, C: /Flutter.

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

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



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



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

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



```
Microsoft Windows [Version 10.0.19042.1435]
(c) Microsoft Corporation. All rights reserved.

C:\Users\jalpa>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             Verbose logging, including all shell commands executed.
                           If used with "--help", shows hidden options. If used with "flutter doctor", shows additional
                           diagnostic information.
-o, (--no-verbose)        Do not force verbose logging in those cases.
-d, --device-id           Target device id or name (prefixed allowed).
--version                Reports the version of this tool.
--suppress-analytics      Suppress analytics reporting when this command runs.

Available commands:

Flutter SDK
  dart-completion         Output command line shell completion setup scripts.
  channel                 List or switch Flutter channels.
  config                  Configure Flutter settings.
  doctor                  Show information about the installed tooling.
  downgrade               Downgrade Flutter to the last active version for the current channel.
  precache                Repopulate the Flutter tool's cache of binary artifacts.
  upgrade                 Upgrade your copy of Flutter.

Project
  analyze                 Analyze the project's Dart code.
  assemble                Assemble and build Flutter resources.
  build                   Build an executable app or install bundle.
  clean                   Delete the build/ and .dart_tool/ directories.
  create                  Create a new Flutter project.
  drive                   Run integration tests for the project on an attached device or emulator.
  format                  Format one or more Dart files.
```

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

```
See Google's privacy policy:
https://policies.google.com/privacy

C:\Users\jalpa>Flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 2.8.1, on Microsoft Windows [Version 10.0.19042.1435], locale en-US)
[✓] Android toolchain - develop for Android devices (Android SDK version 33.0.0)
[✗] Unable to locate Android SDK.
    Install Android Studio from: https://developer.android.com/studio/index.html
    or visit https://flutter.dev/docs/get-started/install/windowsandroid-setup for detailed instructions.
    If the Android SDK has been installed to a custom location, please use
    'flutter config --android-sdk' to update to that location.
[✓] Chrome - develop for the web
[✓] Android Studio (not installed)
[✓] VS Code (version 1.55.2)
[✓] Connected device (2 available)

Doctor found issues in 2 categories.

C:\Users\jalpa>Flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 2.8.1, on Microsoft Windows [Version 10.0.19042.1435], locale en-US)
[✓] Android toolchain - develop for Android devices (Android SDK version 33.0.0)
[✗] cmdline-tools component is missing
    Run 'path/to/sdkmanager --install "cmdline-tools;latest"'
    See https://developer.android.com/studio/command-line for more details.
[✗] Android license status unknown.
    Run 'flutter doctor --android-licenses' to accept the SDK licenses.
    See https://flutter.dev/docs/get-started/install/windowsandroid-setup for more details.
[✓] Chrome - develop for the web
[✓] Android Studio (version 2020.3)
[✓] VS Code (version 1.55.2)
[✓] Connected device (2 available)

Doctor found issues in 1 category.
```

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

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

**Step 7.1:** Download the latest Android Studio executable or zip file from the [official site](https://developer.android.com/studio).

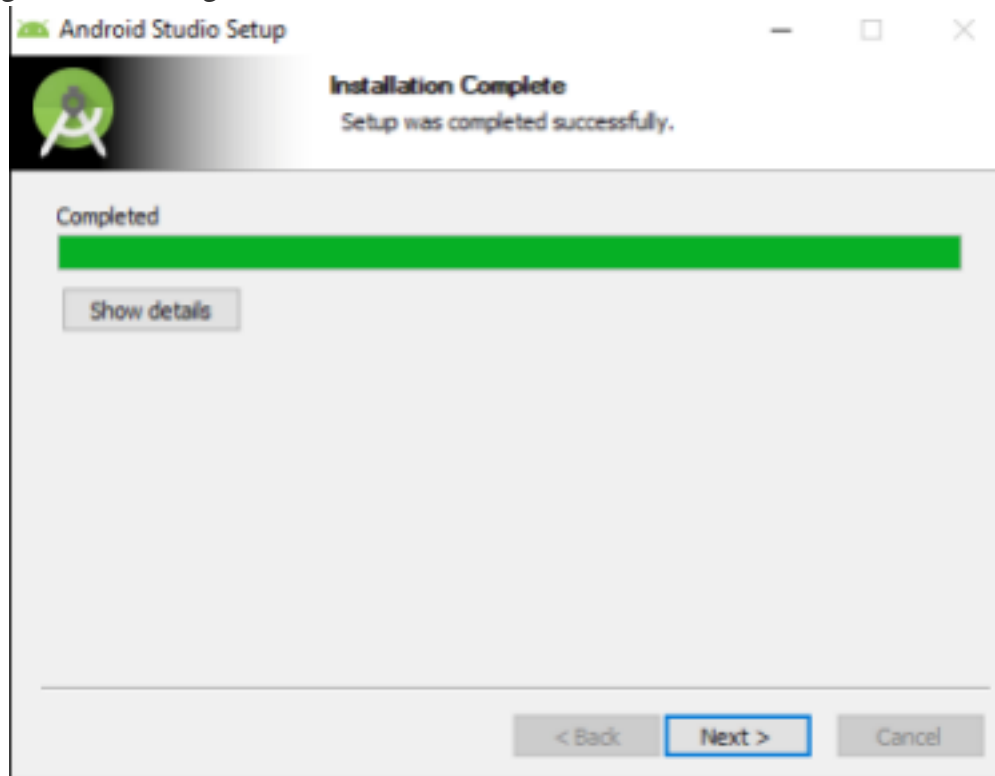
**Step 7.2:** When the download is complete, open the **.exe** file and run it. You will get the

following dialog box.



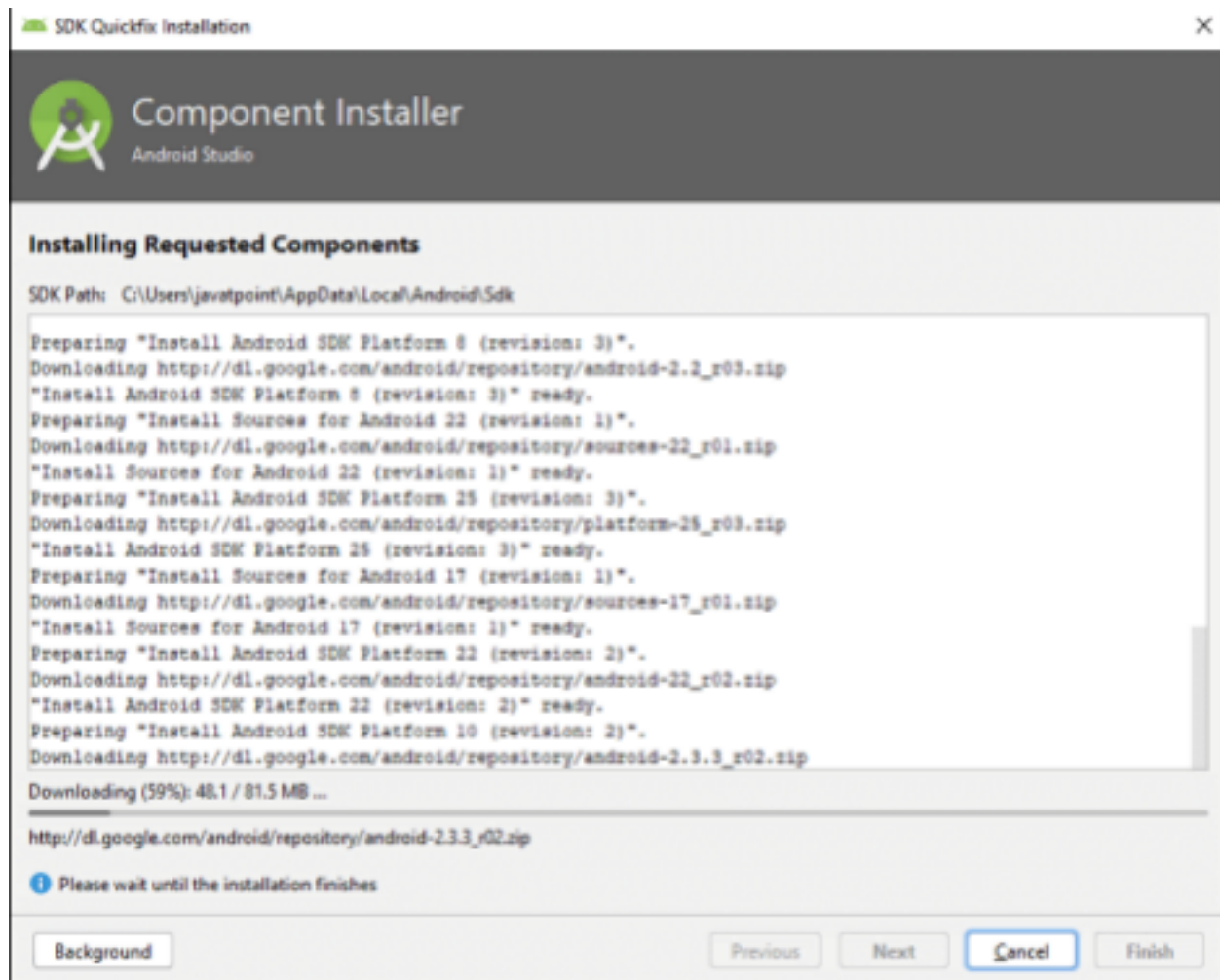
**Step 7.3:**

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

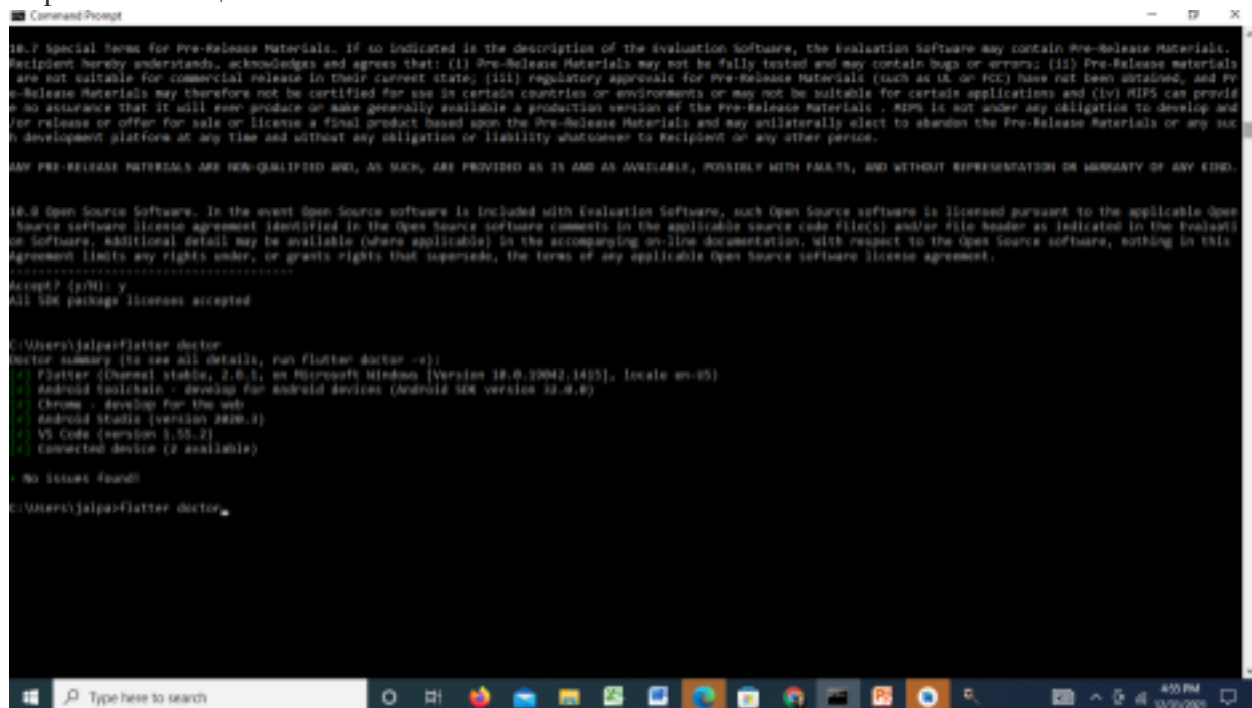


**Step 7.4:**

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



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



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

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



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

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

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



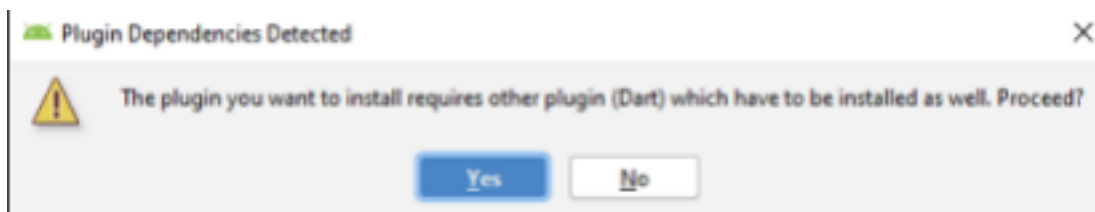
**Step 8.5:** Last, click on the icon pointed into the red color rectangle. The Android emulator displayed as below screen.



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

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

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



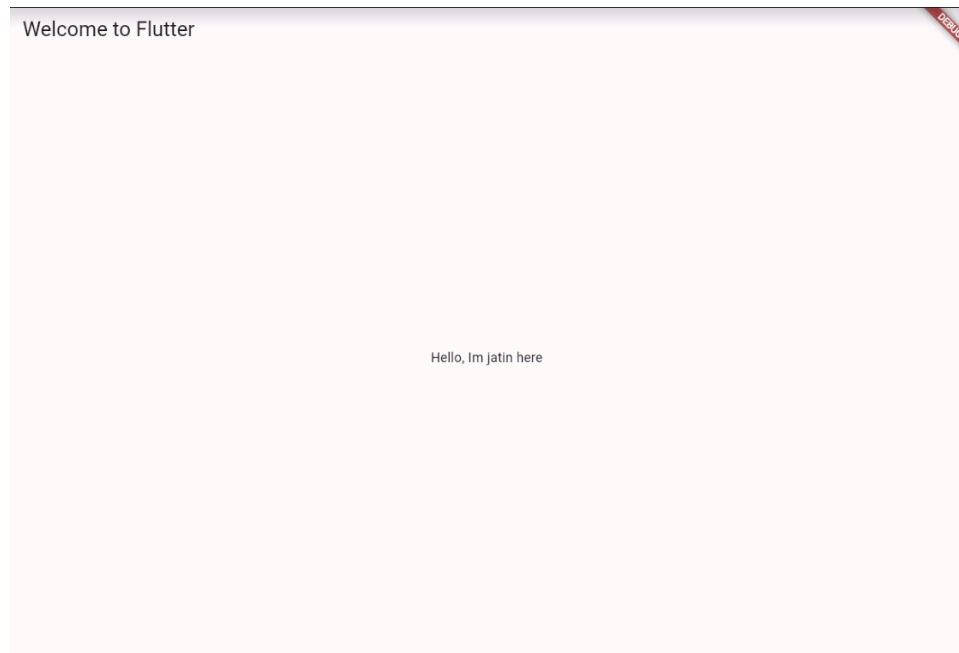
**Step 9.3:** Restart the Android Studio.

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('Jatin talreja'), ),  
    ),  
  );  
}
```

Output:



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

## MAD & PWA Lab

### Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	61
Name	Jatin talreja
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

Name: Jatin talreja  
Division: D15A  
Roll no: 61  
Batch: C

## Experiment No 2

Aim: To design flutter UI by including common widgets.

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

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



These are just a few of the many widgets available in Flutter, each serving a unique purpose in crafting dynamic and user-friendly interfaces.

Code:

```
import 'package:chatgpt/constants/constants.dart';
import 'package:chatgpt/services/assets_manager.dart';
import 'package:flutter/material.dart';

import 'text_widget.dart';

class ChatWidget extends StatelessWidget{
  const ChatWidget({super.key, required this.msg, required this.chatIndex});
  final String msg;
  final int chatIndex;
  @override
  Widget build(BuildContext context){
    return Column(
      children: [
        Material(
          color: chatIndex == 0 ? scaffoldBackgroundColor : cardColor,
          child: Padding(
            padding: const EdgeInsets.all(8.0),
            child: Row(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Image.asset(
                  chatIndex == 0
                    ? AssetsManager.userImage
                    : AssetsManager.botImage,
                  height: 30,
                  width: 30,
                ),
                const SizedBox(
                  width: 8,
                ),
                Expanded(
                  child: TextWidget(
                    label: msg,
                  ),
                ),
                chatIndex == 0
                  ? const SizedBox.shrink()
```



```
@override
Widget build(BuildContext context) {
  return Text(
    label,
    // textAlign: TextAlign.justify,
    style: TextStyle(
      color: color ?? Colors.white,
      fontSize: fontSize,
      fontWeight: fontWeight ?? FontWeight.w500,
    ),
  );
}
```

```
import 'package:chatgpt/services/api_service.dart';
import 'package:chatgpt/widgets/text_widget.dart';
import 'package:flutter/material.dart';

import '../constants/constants.dart';
import '../models/models_model.dart';
```

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

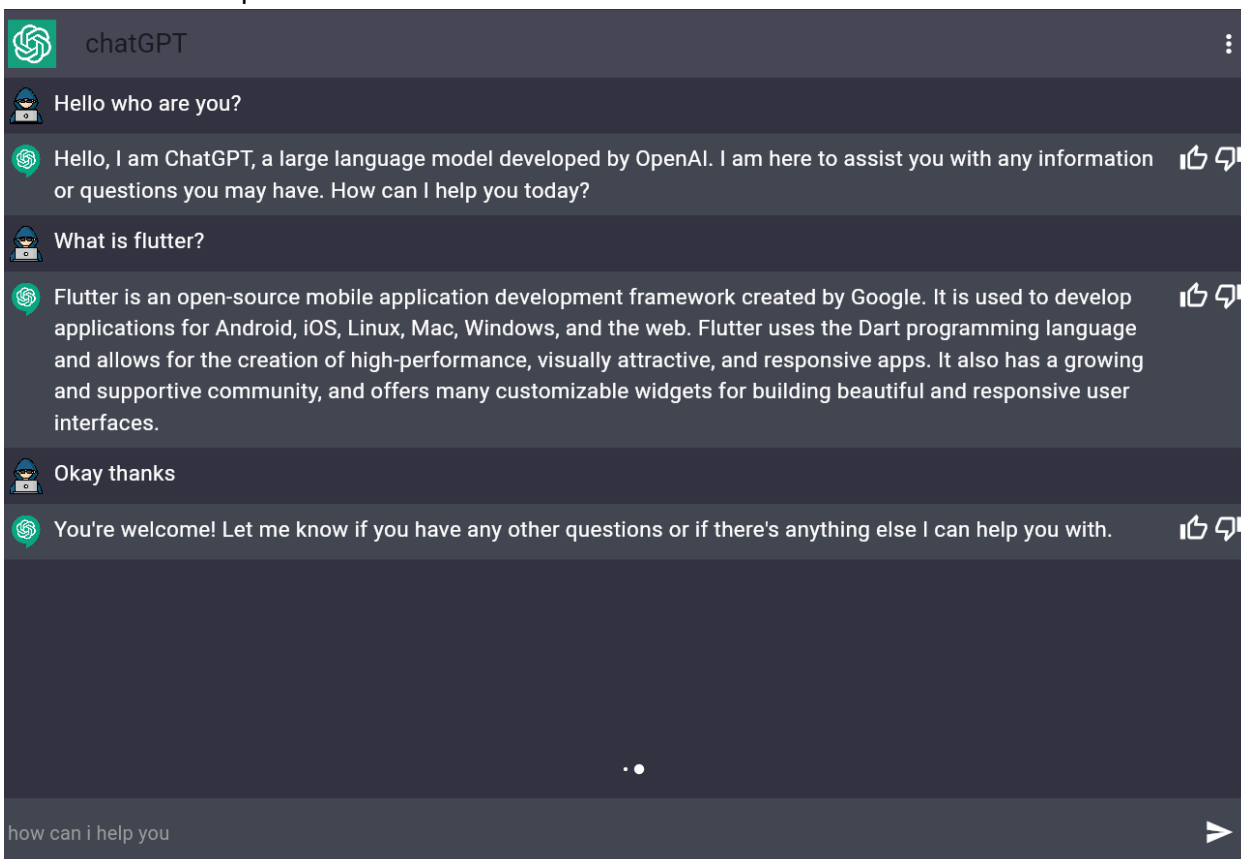
```

@override
State<ModelsDrowDownWidget> createState() => _ModelsDrowDownWidgetState();
}

class _ModelsDrowDownWidgetState extends State<ModelsDrowDownWidget> {
  String currentModel="gpt-3.5-turbo";
  @override
  Widget build(BuildContext context) {
    return FutureBuilder<List<ModelsModel>>(
      future: ApiService.getModels(),
      builder: (context, snapshot){
        if (snapshot.hasError) {
          return Center(child: TextWidget(label: snapshot.error.toString()),);
        }
        return snapshot.data == null || snapshot.data!.isEmpty
        ? const SizedBox.shrink()
        :DropdownButton(
          dropdownColor: scaffoldBackgroundColor,
          iconEnabledColor: Colors.white,
          items: List<DropdownMenuItem<String>>.generate(
            snapshot.data!.length,
            (index) => DropdownMenuItem(
              value: snapshot.data![index].id,
              child: TextWidget(
                label: snapshot.data![index].id,
                fontSize: 15,
              )),
            onChanged: (value) {
              setState(() {
                currentModel=value.toString();
              });
            },
          );
      });
  }
}

```

Output:



## MAD & PWA Lab Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	61
Name	Jatin talreja
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	13

Name: Jatin talreja  
Division: D15A  
Roll No: 61  
Batch: C

## Experiment No 3

Aim: To include images and fonts in flutter app

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

### 1. Import Libraries:

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

### 2. Adding Local Images:

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

### 3. Adding Network Images:

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

### 4. Image Widget:

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

### 5. ImageProvider:

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

### 6. CachedNetworkImage (Optional):

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

### 7. Image Loading and Error Handling:

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

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

Code:

```
import 'package:chatgpt/constants/constants.dart';
import 'package:chatgpt/services/assets_manager.dart';
import 'package:flutter/material.dart';

import 'text_widget.dart';

class ChatWidget extends StatelessWidget{
  const ChatWidget({super.key, required this.msg, required this.chatIndex});
  final String msg;
  final int chatIndex;
  @override
  Widget build(BuildContext context){
    return Column(
      children: [
        Material(
          color: chatIndex == 0 ? scaffoldBackgroundColor : cardColor,
          child: Padding(
            padding: const EdgeInsets.all(8.0),
            child: Row(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Image.asset(
                  chatIndex == 0
                    ? AssetsManager.userImage
                    : AssetsManager.botImage,
                  height: 30,
                  width: 30,
                ),
                const SizedBox(
                  width: 8,
                ),
                Expanded(
                  child: TextWidget(
                    label: msg,
                  ),
                ),
                chatIndex == 0
                  ? const SizedBox.shrink()
```





```
@override
Widget build(BuildContext context) {
  return Text(
    label,
    // textAlign: TextAlign.justify,
    style: TextStyle(
      color: color ?? Colors.white,
      fontSize: fontSize,
      fontWeight: fontWeight ?? FontWeight.w500,
    ),
  );
}
```

```
import 'package:chatgpt/services/api_service.dart';
import 'package:chatgpt/widgets/text_widget.dart';
import 'package:flutter/material.dart';

import '../constants/constants.dart';
import '../models/models_model.dart';
```

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

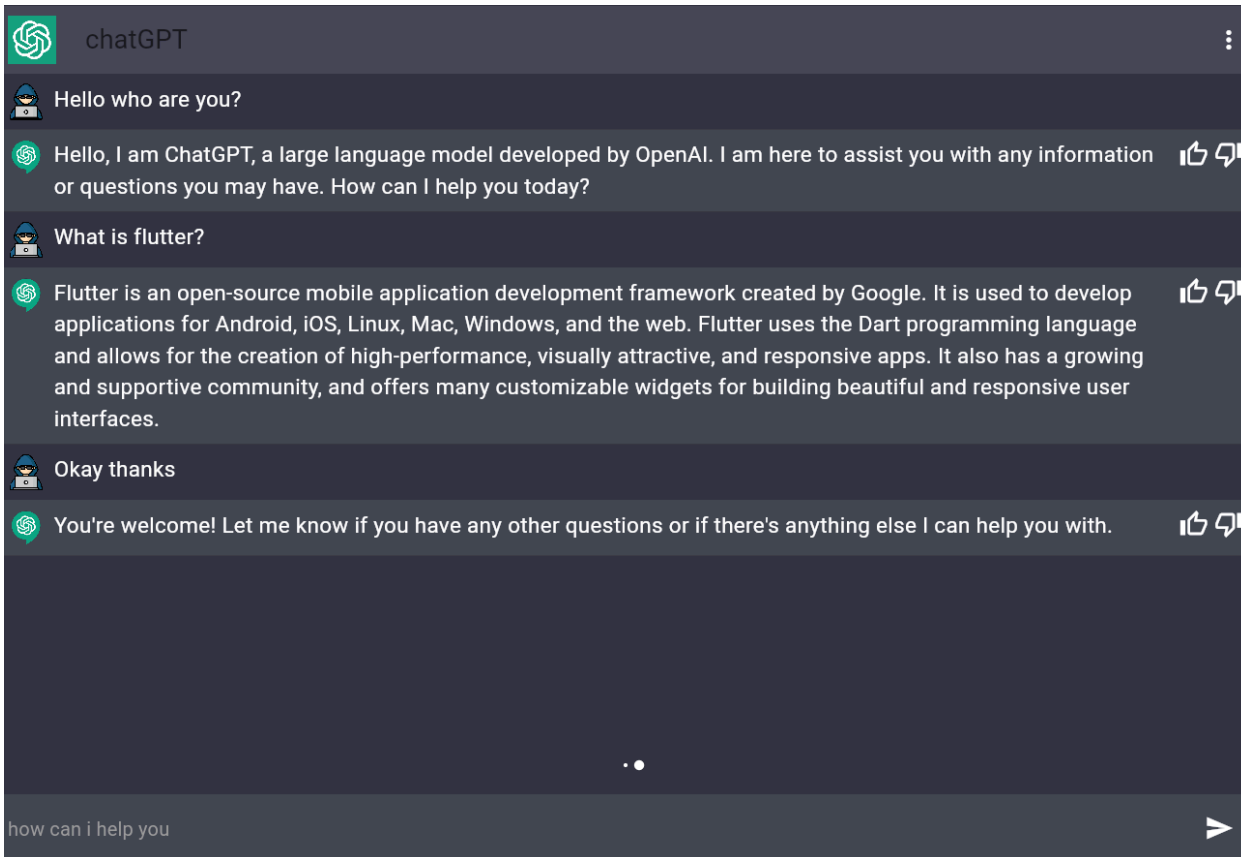
```

@override
State<ModelsDrowDownWidget> createState() => _ModelsDrowDownWidgetState();
}

class _ModelsDrowDownWidgetState extends State<ModelsDrowDownWidget> {
  String currentModel="gpt-3.5-turbo";
  @override
  Widget build(BuildContext context) {
    return FutureBuilder<List<ModelsModel>>(
      future: ApiService.getModels(),
      builder: (context, snapshot){
        if (snapshot.hasError) {
          return Center(child: TextWidget(label: snapshot.error.toString()),);
        }
        return snapshot.data == null || snapshot.data!.isEmpty
        ? const SizedBox.shrink()
        :DropdownButton(
          dropdownColor: scaffoldBackgroundColor,
          iconEnabledColor: Colors.white,
          items: List<DropdownMenuItem<String>>.generate(
            snapshot.data!.length,
            (index) => DropdownMenuItem(
              value: snapshot.data![index].id,
              child: TextWidget(
                label: snapshot.data![index].id,
                fontSize: 15,
              )),
            onChanged: (value) {
              setState(() {
                currentModel=value.toString();
              });
            },
          );
        });
  }
}

```

Output:



## MAD & PWA Lab

### Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	61
Name	Jatin talreja
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	13

Name: Jatin talreja  
Division: D15A  
Roll No: 61  
Batch: C

## Experiment No 4

Aim: To create interactive form using form widget

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

### 1. Widget Hierarchy:

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

### 2. Form State:

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

### 3. Validation:

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

### 4. Form Submission:

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

### 5. GlobalKey<FormState>:

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

### 6. Auto-validation:

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

### 7. Form Submission Lifecycle:

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

### 8. Focus Management:

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

### 9. FormKey and GlobalKey:

Using a `GlobalKey<FormState>` allows for more control over the form, such as triggering form validation or resetting the form. It is usually defined as a global key in the widget tree.

#### 10. Form Persistence:

Form data can be persisted across different screens or app sessions by passing the data down the widget tree or using state management solutions like Provider or Riverpod.

Forms play a crucial role in user interaction, data collection, and validation in Flutter applications, providing a structured and efficient way to handle user input.

Code:

```
import 'dart:convert';

import 'package:flutter/material.dart';
import 'package:intl/intl.dart';
import 'package:http/http.dart' as http;

import '../Home/home.dart';

final formatter = DateFormat.yMd();
var enteredname="";
var entereddate;

class SignUp extends StatefulWidget {
  const SignUp({super.key});
  @override
  State<StatefulWidget> createState() {
    return _SignUp();
  }
}

class DT{
  DT({required this.date});
  final DateTime date;
  String get formattefDate {
    return formatter.format(date);
  }
}
```

```

class _SignUp extends State<SignUp> {
  DateTime? _selectedDate=DateTime.now();
  void _presentDatePicker() async {
    final now = DateTime.now();
    final firstDate = DateTime(now.year - 1, now.month, now.day);
    // final lastDate = DateTime(now.year + 3, now.month, now.day);
    final pickedDate = await showDatePicker(
      context: context,
      initialDate: now,
      firstDate: firstDate,
      lastDate: now);
    if(_selectedDate!=null){
      setState(() {
        _selectedDate = pickedDate??_selectedDate;
      });
    }
  }
  String get formattedDate {
    return formatter.format(DateTime.now());
  }

  void savedata()async{

    enteredname=nameController.text;
    final url=Uri.https('flutter-prep-5b74d-default-rtdb.firebaseio.com','user-data.json');

    final response= await http.post(url,headers: {
      'Content-type':'application/json'
    },
    body: json.encode({
      'name': enteredname,
      'dateofbirth': formatter.format(_selectedDate!)
    })
  );
}

```



```
final _dateController = TextEditingController();
final nameController=TextEditingController();
```

```
@override
```

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

```
  DateTime? selectedDate=DateTime.now();
```

```
  // var selectedDate = __selectedDate;
```

```
  return Scaffold(
```

```
    backgroundColor: Theme.of(context).primaryColorLight,
```

```
    body: Center(
```

```
      child: Padding(
```

```
        padding: const EdgeInsets.all(12.0),
```

```
        child: Column(
```

```
          mainAxisAlignment: MainAxisAlignment.min,
```

```
          mainAxisAlignment: MainAxisAlignment.center,
```

```
          crossAxisAlignment: CrossAxisAlignment.center,
```

```
          children: [
```

```
            TextField(
```

```
              controller: nameController,
```

```
              keyboardType: TextInputType.name,
```

```
              decoration: InputDecoration(
```

```
                border: OutlineInputBorder(
```

```
                  borderRadius: BorderRadius.all(Radius.circular(8))),
```

```
                iconColor: Colors.black,
```

```
                label: Text(
```

```
                  "Add Your Name",
```

```
                  style: TextStyle(
```

```
                    color: Colors.black,
```

```
                    fontStyle: FontStyle.italic,
```

```
                ),
```

```
                ),
```

```
              ),
```

```
              expands: false,
```

```
            ),
```

```
            const SizedBox(height: 10,),
```

```
            const SizedBox(height: 10),
```

```
            Row(
```

```

children: [
  Expanded(
    child: TextField(
      controller: _dateController,
      decoration: InputDecoration(
        label: Text('Enter your D.O.B',style: TextStyle(
          color: Colors.black,
          fontStyle: FontStyle.italic,
        )),
      ),
    ),
  ),
  Expanded(
    child: Row(
      mainAxisAlignment: MainAxisAlignment.end,
      crossAxisAlignment: CrossAxisAlignment.center,
      children: [
        Text(

          selectedDate == null
            ? 'No Date Selected'
            : formatter.format(_selectedDate!),
          style: const TextStyle(color: Colors.black)
        ),
        const SizedBox(width: 10),
        IconButton(
          onPressed: () {
            _presentDatePicker();
          },
          icon: const Icon(
            Icons.calendar_month,
            color: Colors.black,
          )),
        // const SizedBox(width: 16),
        // Text(formattedDate),
      ],
    ),
  ),

```

),

],

),

const SizedBox(height: 10,),

ElevatedButton(onPressed: (){

savedata();

Navigator.of(context).push(MaterialPageRoute(builder: (ctx)=>

MyHomePage())

);

}, child: const Text("Next")),

],

),

),

),

);

}

}

Output:

**Create your account**

Please enter your Email

Please enter your Password

**SignUp**

Don't have an account ? [Login](#)

The image is a screenshot of a mobile application's account creation screen. It features a light pink background. At the top, the text 'Create your account' is centered in a bold, black font. Below this, there are two white input fields with rounded corners and thin grey borders. The first field contains the placeholder text 'Please enter your Email' and the second field contains 'Please enter your Password'. Below the input fields is a prominent green button with rounded corners and the text 'SignUp' in white. Underneath the button, the text 'Don't have an account ?' is followed by a green link labeled 'Login'. At the very bottom of the screen is a black navigation bar containing three white icons: a hamburger menu (three horizontal lines), a circle, and a left-pointing chevron.

## MAD & PWA Lab

### Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	61
Name	Jatin talreja
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	13

Name: Jatin talreja  
Division: D15A  
Roll No: 61  
Batch: C

## Experiment No 5

Aim: To apply routing in Flutter Application

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

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

Flutter also supports route arguments, allowing developers to pass data between screens. Additionally, the `Navigator` provides a flexible set of transitions, such as slide, fade, or custom animations, enhancing the user experience during navigation.

Overall, Flutter's routing system provides a structured and intuitive way to handle navigation within mobile and web applications.

Code:

login\_Screen.dart

```
import 'dart:async';
import 'dart:math';

import 'package:ChatGPT/views/Authentication/auth.dart';
import 'package:ChatGPT/views/home.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';

import 'firebase_options.dart';
import 'models/constants.dart';

void main() async{
  WidgetsFlutterBinding.ensureInitialized();

  await Firebase.initializeApp(
```

```
    options: DefaultFirebaseOptions.currentPlatform,  
  );  
  runApp(MyApp());  
}
```

```
class MyApp extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return MaterialApp(  
      title: 'Chat GPT',  
      theme: ThemeData.light(),  
      // darkTheme: ThemeData.dark(),  
      home: LandingPage(),  
    );  
  }  
}
```

```
class LandingPage extends StatefulWidget {  
  const LandingPage({super.key});  
  
  @override  
  State<LandingPage> createState() => _LandingPageState();  
}
```

```
class _LandingPageState extends State<LandingPage> {  
  List<String> texts = [  
    "Chat GPT",  
    "Let's Design",  
    "Let's Discover",  
    "Let's Collaborate",  
    "Let's Explore",  
    "Let's Chit-Chat"  
  ];  
  
  int currentIndex = 0;  
  String displayedText = "";  
  Color backgroundColor = Colors.blue; // Initial background color
```

```

@Override
void initState() {
    super.initState();
    startTypingLoop();
}

void startTypingLoop() {
    Timer.periodic(Duration(seconds: 3), (timer) {
        startTyping();
    });
}

void startTyping() {
    Timer.periodic(Duration(milliseconds: 100), (timer) {
        if (texts[currentIndex].length > displayedText.length) {
            setState(() {
                displayedText =
                    texts[currentIndex].substring(0, displayedText.length + 1);
            });
        } else {
            timer.cancel();
            Timer(Duration(seconds: 1), () {
                eraseText();
            });
        }
    });
}

void eraseText() {
    Timer.periodic(Duration(milliseconds: 100), (timer) {
        if (displayedText.isNotEmpty) {
            setState(() {
                displayedText = displayedText.substring(0, displayedText.length - 1);
            });
        } else {
            timer.cancel();
            setState(() {
                currentIndex = (currentIndex + 1) % texts.length;
            });
        }
    });
}

```



```

        backgroundColor = getRandomColor();
    });
    startTyping();
}
});
}

```

```

Color getRandomColor() {
    Random random = Random();
    return Color.fromARGB(
        255,
        random.nextInt(256),
        random.nextInt(256),
        random.nextInt(256),
    );
}

```

```

@override
Widget build(BuildContext context) {
    Size size = MediaQuery.of(context).size;
    return Scaffold(
        body: Column(
            mainAxisAlignment: MainAxisAlignment.start,
            children: [
                Container(
                    height: size.height * 0.7,
                    child: AnimatedContainer(
                        duration: Duration(seconds: 1),
                        color: backgroundColor,
                        child: Center(
                            child: Text(
                                displayedText,
                                style: TextStyle(fontSize: 30, color: Colors.white),
                            ),
                        ),
                    ),
                ),
            ],
        ),
    );
}

```

```

color: backgroundColor,
child: Container(
  height: size.height * 0.3,
  decoration: BoxDecoration(
    color: MyColors.black,
    borderRadius: BorderRadius.only(
      topLeft: Radius.circular(30.0),
      topRight: Radius.circular(30.0),
    ),
  ),
),
child: Column(
  children: [
    SizedBox(
      height: size.height*0.05,
    ),
    Padding(
      padding: const EdgeInsets.only(left: 10, right: 10),
      child: ElevatedButton(
        style: ElevatedButton.styleFrom(
          backgroundColor: MyColors.greyShadow,
          disabledBackgroundColor: MyColors.veryLightBlue,
          minimumSize: Size(double.infinity, 50),
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(40),
          ),
        ),
        onPressed: () {
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) => Home(),
            ),
          );
        },
      ),
    ),
    child: Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children:[
        Image.asset(

```

```

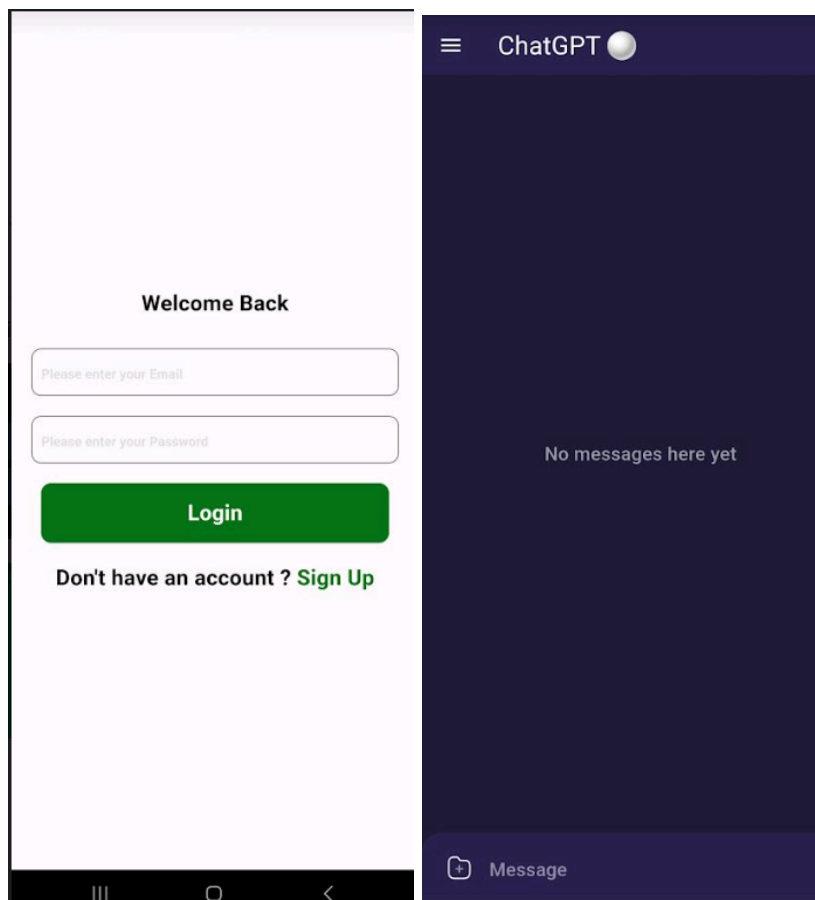
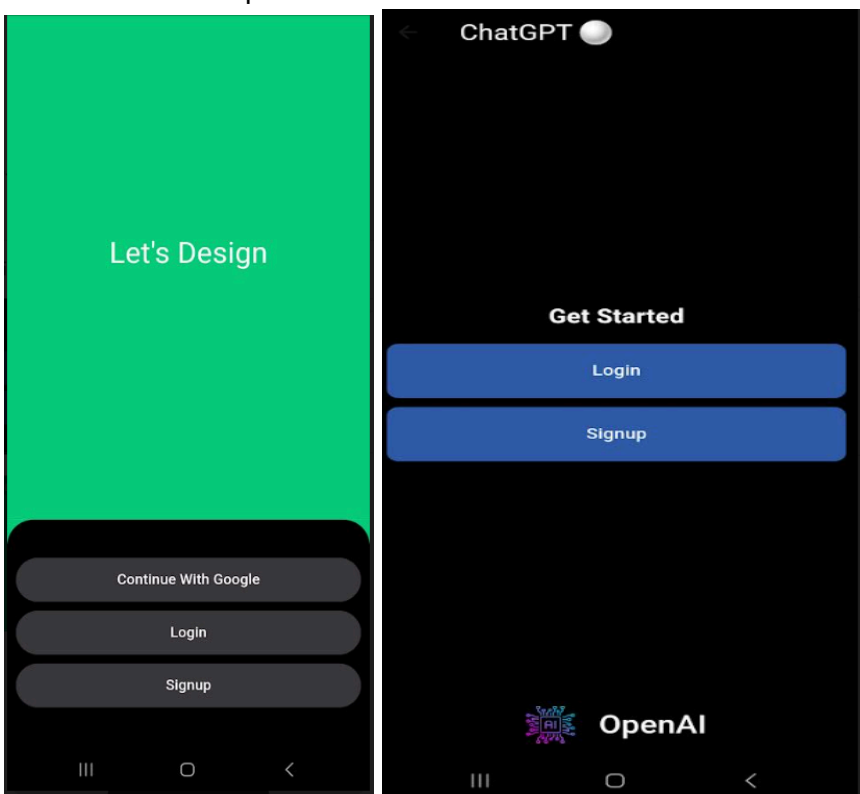
        'assets/icons/google.png',
        width: 50, // Set the desired width
        height: 50, // Set the desired height
        fit: BoxFit.cover,
      ),
      SizedBox(
        width: 20
      ),
      Text(
        'Continue with Google',
        style: TextStyle(
          color: MyColors.white,
          fontSize: 15,
          fontWeight: FontWeight.w500,
          fontFamily: 'Nunito',
        ),
      ),
    ],
  ),
),
),
SizedBox(
  height: 10,
),
Padding(
  padding: const EdgeInsets.only(left: 10, right: 10),
  child: ElevatedButton(
    style: ElevatedButton.styleFrom(
      backgroundColor: MyColors.greyShadow,
      disabledBackgroundColor: MyColors.veryLightBlue,
      minimumSize: Size(double.infinity, 50),
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(40),
      ),
    ),
  ),
  onPressed: () {
    Navigator.push(
      context,

```

[illegible]

```
child: Text(  
  'Signup',  
  style: TextStyle(  
    color: MyColors.white,  
    fontSize: 15,  
    fontWeight: FontWeight.w500,  
    fontFamily: 'Nunito',  
  ),  
,  
,  
,  
,  
  ],  
)),  
)  
],  
,  
);  
}  
}
```

Output:



## MAD & PWA Lab

### Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	61
Name	Jatin talreja
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	13

Name: Jatin talreja

Division: D15A

Roll No: 61

Batch: C

## Experiment No 6

Aim: To Connect Flutter UI with Firebase

Theory:

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

Key features of FlutterFire include:

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



feature toggling.

9. Firebase Crashlytics: FlutterFire includes support for Firebase Crashlytics, providing real-time crash reporting to help developers identify and fix issues quickly.

10. Firebase AdMob: FlutterFire includes AdMob plugins for integrating advertisements into Flutter applications using Firebase AdMob.

Firebase API code:

```
File generated by FlutterFire CLI.
// ignore_for_file: lines_longer_than_80_chars, avoid_classes_with_only_static_members
import 'package:firebase_core/firebase_core.dart' show FirebaseOptions;
import 'package:flutter/foundation.dart'
    show defaultTargetPlatform, kIsWeb, TargetPlatform;

/// Default [FirebaseOptions] for use with your Firebase apps.
///
/// Example:
/// ```dart
/// import 'firebase_options.dart';
/// // ...
/// await Firebase.initializeApp(
///   options: DefaultFirebaseOptions.currentPlatform,
/// );
/// ```
class DefaultFirebaseOptions {
  static FirebaseOptions get currentPlatform {
    if (kIsWeb) {
      return web;
    }
    switch (defaultTargetPlatform) {
      case TargetPlatform.android:
        return android;
      case TargetPlatform.iOS:
        return ios;
      case TargetPlatform.macOS:
        return macos;
      case TargetPlatform.windows:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for windows - '
          'you can reconfigure this by running the FlutterFire CLI again.',
        );
      case TargetPlatform.linux:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for linux - '
```

```
        'you can reconfigure this by running the FlutterFire CLI again.',
    );
    default:
      throw UnsupportedError(
        'DefaultFirebaseOptions are not supported for this platform.',
      );
  }
}
```

```
static const FirebaseOptions web = FirebaseOptions(
  apiKey: 'AIzaSyBU-aWG9s9UOl7n5MwsdUYUQE9ikRQrCSA',
  appId: '1:994530050638:web:f376bf4a92c4189fe7391f',
  messagingSenderId: '994530050638',
  projectId: 'chat-gpt-9ee62',
  authDomain: 'chat-gpt-9ee62.firebaseio.com',
  storageBucket: 'chat-gpt-9ee62.appspot.com',
  measurementId: 'G-5PKNLL5NVB',
);

static const FirebaseOptions android = FirebaseOptions(
  apiKey: 'AIzaSyAGUrrjNm0lDLiFfeR41aJno8NA85IBSiI',
  appId: '1:994530050638:android:8aaa72a65ed10148e7391f',
  messagingSenderId: '994530050638',
  projectId: 'chat-gpt-9ee62',
  storageBucket: 'chat-gpt-9ee62.appspot.com',
);
```

```
static const FirebaseOptions ios = FirebaseOptions(
  apiKey: 'AIzaSyBFCz46f-SRjMB9PcEOd1piTKulm9HKBVA',
  appId: '1:994530050638:ios:5c89c7487abcf9cde7391f',
  messagingSenderId: '994530050638',
  projectId: 'chat-gpt-9ee62',
  storageBucket: 'chat-gpt-9ee62.appspot.com',
  iosBundleId: 'com.example.chatGpt',
);
```

```
static const FirebaseOptions macos = FirebaseOptions(
  apiKey: 'AIzaSyBFCz46f-SRjMB9PcEOd1piTKulm9HKBVA',
  appId: '1:994530050638:ios:65caaaca61da6d36e7391f',
  messagingSenderId: '994530050638',
  projectId: 'chat-gpt-9ee62',
  storageBucket: 'chat-gpt-9ee62.appspot.com',
  iosBundleId: 'com.example.chatGpt.RunnerTests',
);
```

```
);  
}
```

## Pubspec.yaml

```
name: ChatGPT  
description: "Chat GPT Clone"  
  
publish_to: 'none' # Remove this line if you wish to publish to pub.dev  
  
version: 1.0.0+1  
  
environment:  
  sdk: '>=3.2.6 <4.0.0'  
  
dependencies:  
  flutter:  
    sdk: flutter  
  
  # The following adds the Cupertino Icons font to your application.  
  # Use with the CupertinoIcons class for iOS style icons.  
  cupertino_icons: ^1.0.2  
  dio: ^5.4.0  
  mobile_chat_ui: ^1.0.0  
  flutter_chat_ui: ^1.6.12  
  file_picker: ^6.1.1  
  open_filex: ^4.4.0  
  bubble: ^1.2.1  
  get_storage: ^2.1.1  
  firebase_core: ^2.24.2  
  firebase_auth: ^4.16.0  
  intl: ^0.19.0  
  google_fonts: ^6.1.0  
  http: ^1.2.0  
  uuid: ^4.3.3  
  cached_network_image: ^3.3.1  
  flutter_launcher_icons: ^0.13.1  
  url_launcher: ^6.0.9  
  flutter_svg:  
  
dev_dependencies:  
  flutter_test:  
    sdk: flutter
```

```
# The "flutter_lints" package below contains a set of recommended lints to
# encourage good coding practices. The lint set provided by the package is
# activated in the `analysis_options.yaml` file located at the root of your
# package. See that file for information about deactivating specific lint
# rules and activating additional ones.
flutter_lints: ^2.0.0

# For information on the generic Dart part of this file, see the
# following page: https://dart.dev/tools/pub/pubspec
flutter_icons:
  android: true
  ios: true
  image_path: "assets/icons/open-ai.png"

# The following section is specific to Flutter packages.
flutter:

  # The following line ensures that the Material Icons font is
  # included with your application, so that you can use the icons in
  # the material Icons class.
  uses-material-design: true

  # To add assets to your application, add an assets section, like this:
  assets:
    - assets/icons/
    - assets/images/bg2.png
    - assets/images/download.jpeg

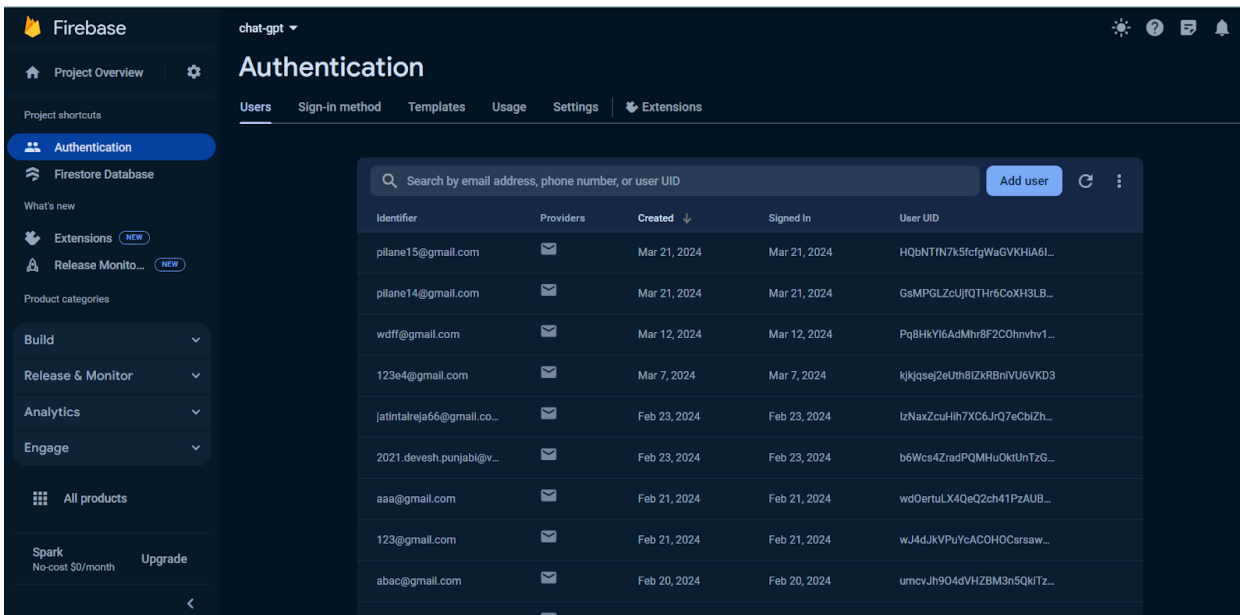
  # An image asset can refer to one or more resolution-specific "variants", see
  # https://flutter.dev/assets-and-images/#resolution-aware

  # For details regarding adding assets from package dependencies, see
  # https://flutter.dev/assets-and-images/#from-packages

  # To add custom fonts to your application, add a fonts section here,
  # in this "flutter" section. Each entry in this list should have a
  # "family" key with the font family name, and a "fonts" key with a
  # list giving the asset and other descriptors for the font. For
  # example:
  # fonts:
  #   - family: Schyler
  #     fonts:
```

```
# - asset: fonts/Schyler-Regular.ttf
# - asset: fonts/Schyler-Italic.ttf
# style: italic
# - family: Trajan Pro
# fonts:
# - asset: fonts/TrajanPro.ttf
# - asset: fonts/TrajanPro_Bold.ttf
# weight: 700
#
# For details regarding fonts from package dependencies,
# see https://flutter.dev/custom-fonts/#from-packages
```

Output:



The screenshot shows the Firebase Authentication console. The left sidebar contains navigation links for Project Overview, Authentication (selected), Firestore Database, and What's new. The main area displays the 'Users' tab with a search bar and an 'Add user' button. Below the search bar is a table listing users with columns for Identifier, Providers, Created, Signed In, and User UID.

Identifier	Providers	Created	Signed In	User UID
pilane15@gmail.com	📧	Mar 21, 2024	Mar 21, 2024	HQbNTfN7k5fcgWaGVKHIA6l...
pilane14@gmail.com	📧	Mar 21, 2024	Mar 21, 2024	GsMPGLZcUjfrQThr6CoXH3LB...
wdff@gmail.com	📧	Mar 12, 2024	Mar 12, 2024	Pq8HkYl6AdMhr8F2Cohnv1...
123e4@gmail.com	📧	Mar 7, 2024	Mar 7, 2024	kjkjqaej2eUth8IZxRBnIU6VKD3
jatintalreja66@gmail.co...	📧	Feb 23, 2024	Feb 23, 2024	IzNaxZcuHlr7XC6JrQ7eCbiZh...
2021.devesh.punjabi@v...	📧	Feb 23, 2024	Feb 23, 2024	b6Wcs4ZradPQMhuOktUnTzG...
aaa@gmail.com	📧	Feb 21, 2024	Feb 21, 2024	wdOertuLX4QeQ2ch41PzAU8...
123@gmail.com	📧	Feb 21, 2024	Feb 21, 2024	wJ4dJKVPuYcACOHOCrsaw...
abac@gmail.com	📧	Feb 20, 2024	Feb 20, 2024	umcvJh9O4dVHZBM3n5QkiTz...

## 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.	61
Name	Jatin talreja
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	14

Name: Jatin talreja  
Division: D15A  
Roll No: 61  
Batch: B

## Experiment No 7

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

Theory:-

### Regular Web App

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

### Progressive Web App

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

Difference between PWAs vs. Regular Web Apps:

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

#### 1. Native Experience

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

#### 2. Ease of Access

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

#### 3. Faster Services

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

#### 4. Engaging Approach

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

#### 5. Updated Real-Time Data Access

Another plus point of PWAs is that these apps get updated on their own. They do not demand the end-users to go to the App Store or other such platforms to download the update and wait until installed.

In this app type, the web app developers can push the live update from the server, which reaches the apps residing on the user's devices automatically. Therefore, it is easier for the mobile app developer to provide the best of the updated functionalities and services to the end-users without forcing them to update their app.

#### 6. Discoverable

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

#### 7. Lower Development Cost

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

#### Pros and cons of the Progressive Web App

The main features are:

**Progressive** — They work for every user, regardless of the browser chosen because they are built at the base with progressive improvement principles.

**Responsive** — They adapt to the various screen sizes: desktop, mobile, tablet, or dimensions that can later become available.

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

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

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

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

**Reactivable** — Make it easy to reactivate the application thanks to capabilities such as web notifications.

**Installable** — They allow the user to “save” the apps that he considers most useful with the corresponding icon on the screen of his mobile terminal (home screen) without having to face all



the steps and problems related to the use of the app store.

Linkable — Easily shared via URL without complex installations.

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

Weaknesses refer to:

IOS support from version 11.3 onwards;

Greater use of the device battery;

Not all devices support the full range of PWA features (same speech for iOS and Android operating systems);

It is not possible to establish a strong re-engagement for iOS users (URL scheme, standard web notifications);

Support for offline execution is however limited;

Lack of presence on the stores (there is no possibility to acquire traffic from that channel);

There is no “body” of control (like the stores) and an approval process;

Limited access to some hardware components of the devices;

Little flexibility regarding “special” content for users (eg loyalty programs, loyalty, etc.).

Folder Structure and icon size

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<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=Lato:wght@100;400;700&family=Poppins:wght@200
&display=swap"
rel="stylesheet">
```

```
<link href="https://fonts.googleapis.com/css2?family=Poppins:wght@200&display=swap"
rel="stylesheet">
<link rel="stylesheet" href="style.css">
<title>Portfolio</title>
</head>
<body>
  <header>
    <div class="container">
      <nav class="flex items-centre justify-between">
        <div class="left flex justify-right">
          <div class="logo">
            
          </div>
          <div>
            <a href="#">Home</a>
            <a href="#">About</a>
            <a href="#">Services</a>
            <a href="#">Blog</a></li>
            <a href="#">More</a>
          </div>
        </div>
        <div class="right">
          <button class="btn btn-primary">Contact</button>
        </div>
      </nav>
    </div>
    <div class="hero flex items-centre justify-between">
      <div class="left flex-1 justify-center">
        
      </div>
      <br><br><br><br><br><br><br><br>
      <div class="right flex-1">
        <h3>Jatin Talreja </h3>
        <h1>I'm a Web<br> <span>Developer</span></h1>
        <p>As a web developer, its not just someone who writes code; you are an
architect of the digital world, shaping the way people interact, communicate, and transact
online.
        </p>
        <div>
          <button class="btn btn-secondary">DOWNLOAD CV</button>
        </div>
      </div>
    </div>
```

```

</header>
<section class="about">
  <div class="container flex items-centre">
    <div class="left flex-1 justify-right">
      
    </div>
    <div class="right flex-1">
      <h1>About <span>Me</span></h1>
      <h3>Hello! I'm Jatin talreja.</h3>
      <p>hello everyone i am jatin talreja currently working on web development ,
        my aim is to keep exploring more on web development so that i can contribute
more in the open source
      </p>
      <div class="socials">

        <a href="#"></a>
        <a href="#"></a>
        <a href="#"></a>
        <a href="#"></a>
      </div>
    </div>
  </div>
</section>
<section class="services">
  <div class="container">
    <h1 class="services-head">Services</h1>
    <p>All your digital needs... covered.</p>
    <div class="card-grid">
      <div class="card">
        
        <h2>Graphic Desgin</h2>
        <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit. Nulla, debitis?</p>
      </div>
      <div class="card">
        
        <h2>Web Development</h2>
        <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit. Nulla, debitis?</p>
      </div>
      <div class="card">
        
        <h2>Content Writing</h2>
        <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit. Nulla, debitis?</p>
      </div>
    </div>
  </div>
</section>

```

```
        </div>
      </div>
    </section>
  </body>
</html>
```

## Style.css-

```
@import "utilities.css";

:root {
  --primary: rgb(29, 221, 189);
  --bgDark: rgb(12, 12, 12);
  --white: rgb(250, 250, 250);
  --secondary: rgb(0, 59, 50);
  --bgLight: rgb(190, 181, 181);
}

* {
  padding: 0;
  margin: 0;
  box-sizing: border-box;
  -webkit-font-smoothing: antialiased;
}

header {
  background-color: var(--bgDark);
  clip-path: polygon(0 0, 100% 0, 100% 100%, 73% 94%, 0 100%);
}

header nav .left a {
  color: var(--white);
  text-decoration: none;
  margin-right: 2rem;
  text-transform: uppercase;
  transition: all 0.2s ease;
}

header nav .left a:hover {
  color: var(--primary);
}

header nav {
  padding: 2rem 0;
}

header nav .logo {
  margin-right: 3rem;
```

```
}
body {
  font-family: "Poppins", sans-serif;
}
.container {
  max-width: 1152px;
  padding: 0 15px;
  margin: 0 auto;
}
.hero {
  padding-top: 2rem;
  padding-bottom: 3rem;
}
.hero .left img {
  width: 400px;
}
.hero .right {
  color: var(--white);
  margin-top: -7rem;
}
.hero .right h6 {
  font-size: 1.6rem;
  color: var(--primary);
  margin-bottom: 0.5rem;
}
.hero .right h1 {
  font-size: 4rem;
  font-weight: 100;
  line-height: 1.2;
  margin-bottom: 2rem;
}
.hero .right h1 span {
  color: var(--primary);
}
.hero .right p {
  line-height: 1.9;
  margin-bottom: 2rem;
}
section.services {
  background: rgb(17, 17, 17);
}
.services-head {
  color: rgb(10, 9, 9);
}
```

```
text-align: center;
margin-bottom: 1rem;
line-height: 0.5rem;
color: var(--primary);
}

.services-head + p {
color: var(--white);
font-family: "Lato", sans-serif;
margin-bottom: 1rem;
text-align: center;
margin-bottom: 6rem;
font-weight: 400;
}

.card img {
width: 50px;
background: white;
}

section.services .card-grid {
display: grid;
grid-template-columns: repeat(3, 1fr);
column-gap: 2rem;
}

section.services .card-grid .card {
background: var(--white);
padding: 3rem 2rem;
position: relative;
text-align: center;
transition: all 0.2s ease;
}

section.services .card-grid .card img {
position: absolute;
top: -1.5rem;
left: 50%;
transform: translateX(-50%);
color: var();
}

section.services .card-grid .card h2 {
font-weight: 600;
font-size: 1.2rem;
margin-bottom: 0.5rem;
}

section.services .card-grid .card p {
font-family: "Lato", sans-serif;
```

```

    color: var(--secondary);
    line-height: 1.6;
}

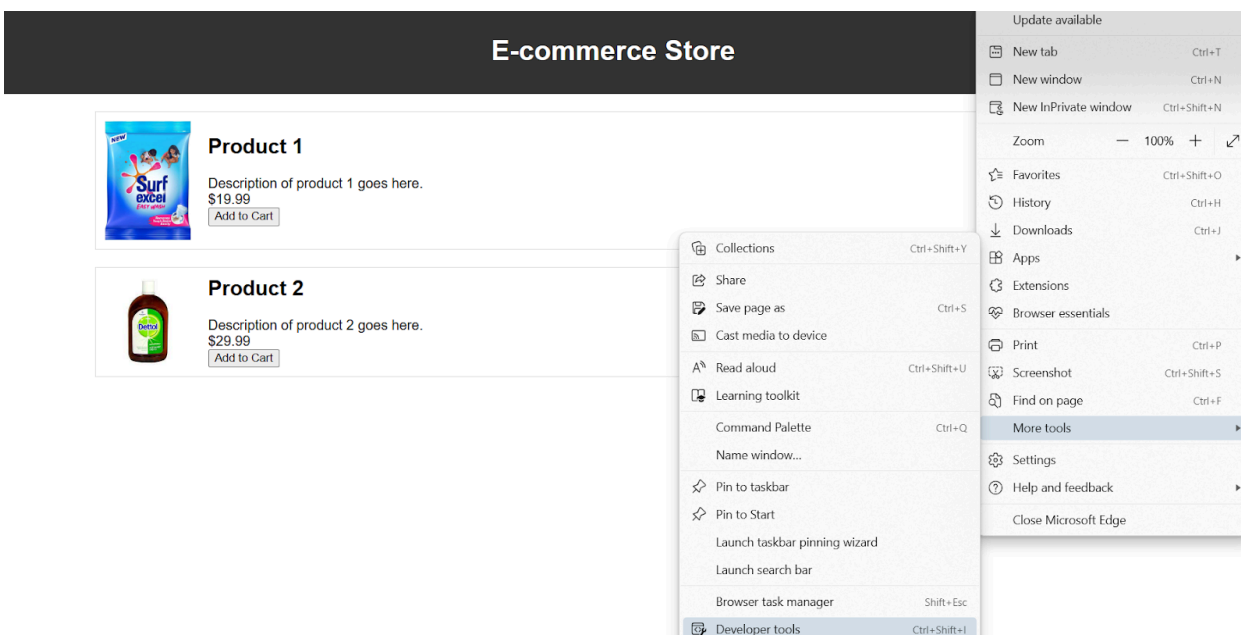
section.services .card-grid .card:hover {
    background: var(--primary);
}

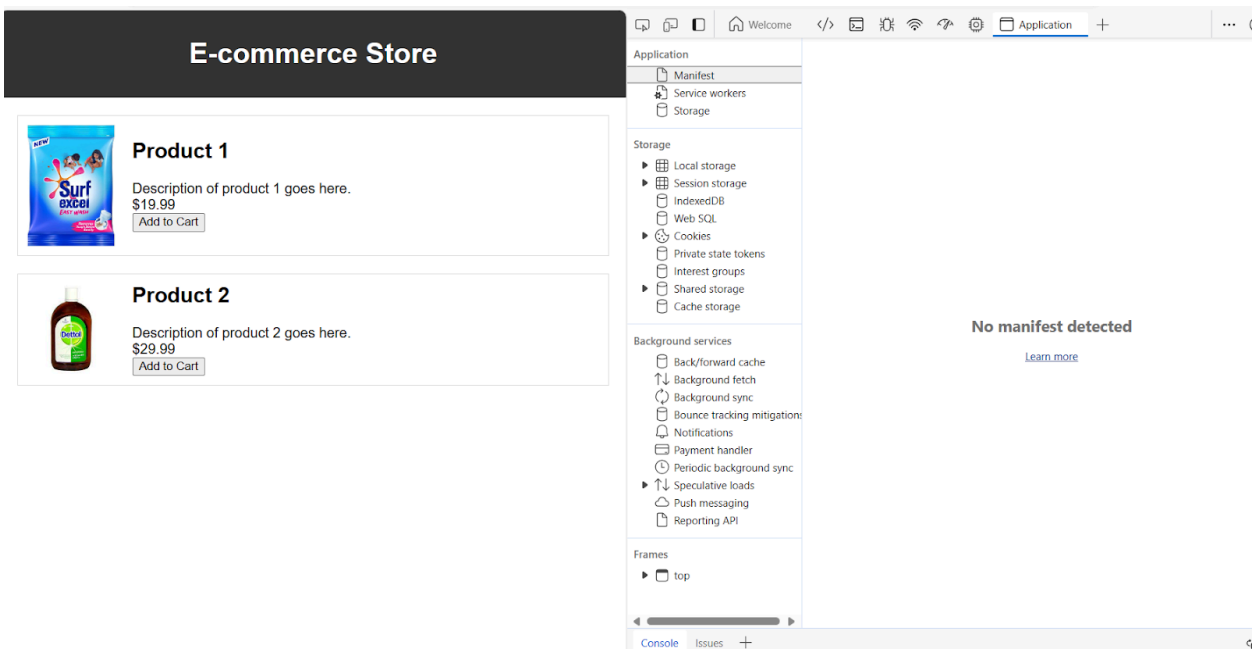
section.services .card-grid .card:hover h2 {
    color: var(--white);
}

section.services .card-grid .card:hover p {
    color: var(--white);
}

```

## Output-





# MAD & PWA Lab

## Journal

Experiment No.	08
Experiment Title.	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.
Roll No.	61
Name	Jatin talreja
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	14



**Name -jatin talreja div-D15A, roll no-61 , batch-c**

## **Experiment number 08 MAD and PWA**

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

### **Theory:**

#### **Service Worker**

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

Things to note about Service Worker:

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

#### **What can we do with Service Workers?**

- You can dominate **Network Traffic**  
You can manage all network traffic of the page and do any manipulations. For example, when the page requests a CSS file, you can send plain text as a response or when the page requests an HTML file, you can send a png file as a response. You can also send a true response too.
- You can **Cache**  
You can cache any request/response pair with Service Worker and Cache API and you can access these offline content anytime.
- You can manage **Push Notifications**  
You can manage push notifications with Service Worker and show any information message to the user.
- You can **Continue**  
Although Internet connection is broken, you can start any process with Background Sync of Service Worker.

#### **What can't we do with Service Workers?**

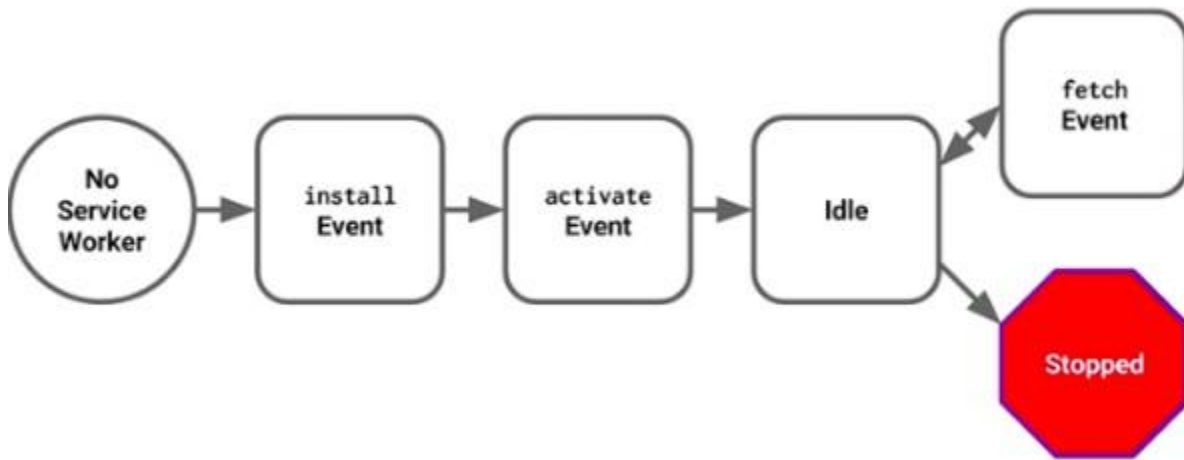
- You can't access the **Window**

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

- You can't work it on **80 Port**

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

## Service Worker Cycle



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

- Registration
- Installation
- Activation

## Registration

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

main.js

```

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

```

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

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

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

main.js

```

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

```

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

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

main.js

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

## Installation

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

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

service-worker.js

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

## Activation

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

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

service-worker.js

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

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

## Code

sw.js

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

var filesToCache = [
  '/',
  '/menu',
  '/contactUs',
  '/offline.html',
];

var preLoad = function () {
  return caches.open("offline").then(function (cache) {
    // caching index and important routes
    return cache.addAll(filesToCache);
  });
};

self.addEventListener("fetch", function (event) {
  event.respondWith(checkResponse(event.request).catch(function () {
    return returnFromCache(event.request);
  }));
  event.waitUntil(addToCache(event.request));
});

var checkResponse = function (request) { return
  new Promise(function (fulfill, reject) {
    fetch(request).then(function (response) { if
      (response.status !== 404) {
        fulfill(response);
```

```

    } else {
        reject();
    }
    }, reject);
    });
};

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

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;
            }
        });
    });
};

```

## Sample Code with Output

### index.html

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="theme-color" content="#4285f4" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>E-commerce Store</title>
    <link rel="stylesheet" href="style.css" />
    <link rel="manifest" href="manifest.json" />
    <script src="service-worker.js"></script>
  </head>
  <body>

```

```

<header>
  <h1>E-commerce Store</h1>
</header>
<div class="container">
  <div class="product">
    
    <div class="product-info">
      <h2>Product 1</h2>
      <p>Description of product 1 goes here.</p>
      <p>$19.99</p>
      <button>Add to Cart</button>
    </div>
  </div>
  <div class="product">
    
    <div class="product-info">
      <h2>Product 2</h2>
      <p>Description of product 2 goes here.</p>
      <p>$29.99</p>
      <button>Add to Cart</button>
    </div>
  </div>
  <!-- More products can be added here -->
</div>
<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("service-worker.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>
</body>
</html>

```

#### app.js

```

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

```

#### service-worker.js

```

// service-worker.js

const cacheName = 'ecommerce-pwa-v1';
const assetsToCache = [
    '/',

```



```

    '/index.html',
    '/main.css',
    '/app.js'
    // Add more files and assets here as needed
];

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

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

```

#### main.css

```

body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
}

header {
    background-color: #333;
    color: #fff;
    padding: 10px 20px;
}

```

```
    text-align: center;
}
.container {
    max-width: 1200px;
    margin: 0 auto;
    padding: 20px;
}
.product {
    border: 1px solid #ddd;
    margin-bottom: 20px;
    padding: 10px;
    display: flex;
    align-items: center;
}
.product img {
    max-width: 100px;
    margin-right: 20px;
}
.product-info {
    flex: 1;
}
.product-info h2 {
    margin-top: 0;
}
.product-info p {
    margin: 0;
}
```

### Steps for Execution

Create a folder and put all 4 files main.css , service-worker.js, app.js, index.html

open visual studio

install extension Live server

open folder in visual studio open index.html

on bottom right corner click go Live

it will open html page in browser

go to developer tools and take following screenshots

# E-commerce Store



## Product 1

Description of product 1 goes here.  
\$19.99

Add to Cart

Application

ManifestService workersStorage

Storage

Local storageSession storageIndexedDBWeb SQLCookiesPrivate state tokensInterest groups

Service workers

☐ Offline☐ Update on reload☐ Bypass for network

https://jatintalreja0510.github.io/MPL-PWA-APP/ Network requests

Source service-worker.js

Received 4/2/2024, 12:06:48 PM

Status ● #6410 activated and is running stop

Clients https://jatintalreja0510.github.io/MPL-PWA-APP/ focus

view-source:https://jatintalreja0510.github.io/MPL-PWA-APP/

# MAD & PWA Lab

## Journal

Experiment No.	09
Experiment Title.	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.
Roll No.	61
Name	Jatin talreja
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	14

## EXPERIMENT NO. 9

### MAD and PWA Lab

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

#### Theory: Service Worker

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

Things to note about Service Worker:

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

#### Fetch Event

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

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

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

```

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

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

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

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

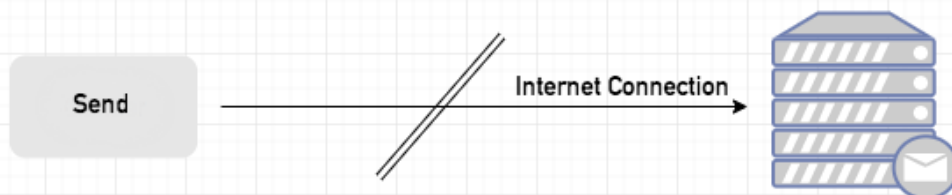
```

## Sync Event

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

Here is a job for the Background Sync.

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



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

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

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

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

#### Event Listener for Background Sync Registration

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

#### Event Listener for sw.js

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

### Push Event

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

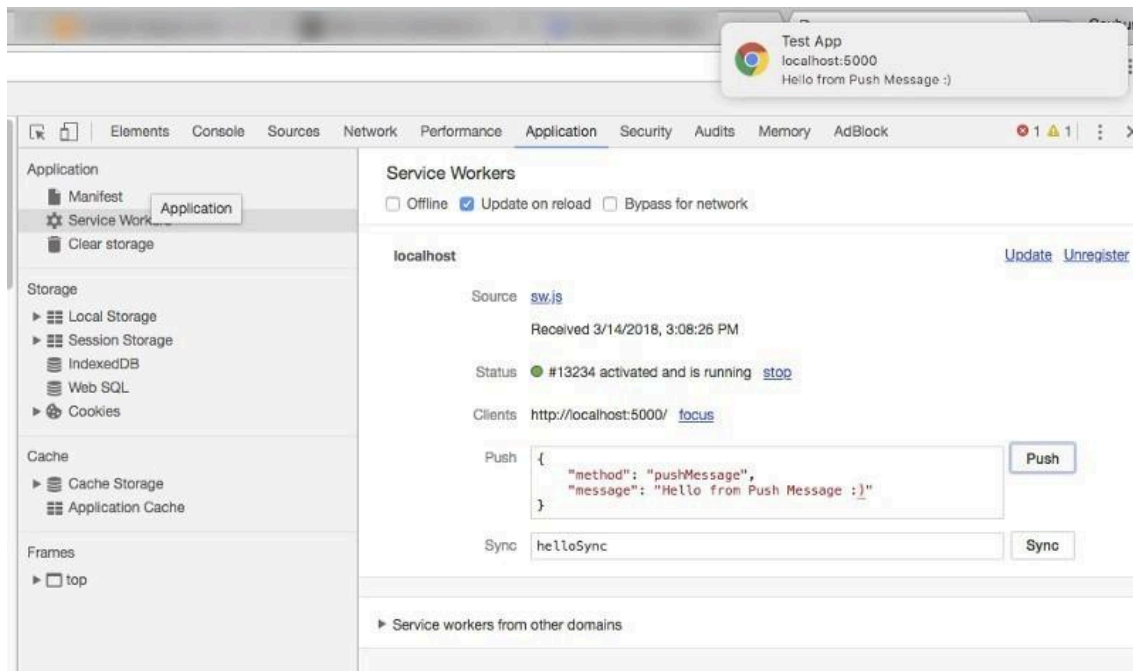
We can check in the following example.

“Notification.requestPermission();” is the necessary line to show notification to the user. If you don’t want to show any notification, you don’t need this line.

In the following code block is in sw.js file. You can handle push notifications with this event. In this example, I kept it simple. We send an object that has “method” and “message” properties. If the method value is “pushMessage”, we open the information notification with the “message” property.

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

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



**Code:**

sw.js

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

self.addEventListener("fetch", function (event) {
  event.respondWith(checkResponse(event.request).catch(function () {
console.log("Fetch from cache successful!") return
    returnFromCache(event.request);
  }));
console.log("Fetch successful!") event.waitUntil(addToCache(event.request));
});

self.addEventListener('sync', event => { if
  (event.tag === 'syncMessage') {
```



```

        console.log("Sync successful!")
    }
});

self.addEventListener('push', function (event) {
    if (event && event.data) {
        var data = event.data.json();
        if (data.method === "pushMessage") {
            console.log("Push notification sent");
            event.waitUntil(self.registration.showNotification("Omkar Sweets Corner", { body:
                data.message
            })))
        }
    }
})

```

```

var filesToCache = [
    '/',
    '/menu',
    '/contactUs',
    '/offline.html',
];

```

```

var preLoad = function () {
    return caches.open("offline").then(function (cache) {
        // caching index and important routes
        return cache.addAll(filesToCache);
    });
};

```

```

var checkResponse = function (request) { return
    new Promise(function (fulfill, reject) {
        fetch(request).then(function (response) {
            if (response.status !== 404) {
                fulfill(response);
            } else {
                reject();
            }
        });
    });
};

```

```

    }
    }, reject);
  });
};

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


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;
      }
    });
  });
};

```

**Output:**

Fetch event

# E-commerce Store




## Product 1

Description of product 1 goes here.

\$19.99

Add to Cart



## Product 2

Description of product 2 goes here.

\$29.99

Add to Cart

Application

Manifest

Service workers

Storage

Storage

Background services

Service workers

Offline

Update on reload

Bypass for network

http://127.0.0.1:5500/

Network requests

Update

Unregister

Source

Received 2/4/2024, 11:22:03 am

Status

Clients

Push

Sync

Periodic Sync

Update Cycle

Console

Issues

Filter

Default levels

4

Notification permission granted.


Fetch successful!

Push notification sent

Sync successful!

Sync event

# E-commerce Store




## Product 1

Description of product 1 goes here.

\$19.99

Add to Cart



## Product 2

Description of product 2 goes here.

\$29.99

Add to Cart

Application

Manifest

Service workers

Storage

Storage

Background services

Service workers

Offline

Update on reload

Bypass for network

http://127.0.0.1:5500/

Network requests

Update

Unregister

Source

Received 2/4/2024, 11:22:03 am

Status

Clients

Push

Sync

Periodic Sync

Update Cycle

Console

Issues

Filter

Default levels

4

Notification permission granted.


Fetch successful!

Push notification sent

Sync successful!

Push event

E-commerce Store




Product 1

Description of product 1 goes here.

\$19.99

Add to Cart



Product 2

Description of product 2 goes here.

\$29.99

Add to Cart

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

Service workers

Offline

Update on reload

Bypass for network

http://127.0.0.1:5500/

Network requests

Update

Unregister

Source

service-worker.js

Received 2/4/2024, 11:22:03 am

Status

#1879 activated and is running

stop

Clients

http://127.0.0.1:5500/index.html

focus

Push

{ "method": "pushMessage", "mes

Push

Sync

syncMessage

Sync

Periodic Sync

test-tag-from-devtool!

Periodic Sync

Update Cycle

Version

Update Activity

Timeline

#1879 Install

#1879 Wait

#1879 Activate

Console

Issues

+

top

Filter

Default levels

4

Notification permission granted.

index.html:60

Fetch successful!

service-worker.js:50

Push notification sent

service-worker.js:67

Sync successful!

service-worker.js:57

## MAD & PWA Lab

### Journal

Experiment No.	10
Experiment Title.	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.
Roll No.	61
Name	Jatin talreja
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	14

Name -Jatin talreja

div-D15a roll no-61

## Experiment No. 10 MAD &PWA Lab

### Aim:

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

### Theory:

#### GitHub Pages

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

GitHub Pages provides the following key features:

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

Reasons for favoring this over Firebase:

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

GitHub Pages is used by Lyft, CircleCI, and HubSpot.

GitHub Pages is listed in 775 company stacks and 4401 developer stacks.

#### Pros

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

#### Cons

1. The code of your website will be public, unless you pay for a private repository.
2. Currently, there is no support for HTTPS for custom domains. It's probably coming soon though.
3. Although Jekyll is supported, plug-in support is rather spotty.

#### Firebase

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

milliseconds.

Some of the features offered by Firebase are:

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

Reasons for favoring over GitHub Pages:

1. Realtime backend made easy
2. Fast and responsive

Instacart, 9GAG, and Twitch are some of the popular companies that use Firebase

Firebase has a broader approval, being mentioned in 1215 company stacks & 4651 developers stacks

Pros












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

Cons

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

**Link to our GitHub repository: <https://github.com/Jatintalreja0510/MPL-PWA-APP>**


# Github Screenshot:

<div> <b>Jatintalreja0510</b> project full ✓</div>	
<div> images</div>	project full
<div> app.js</div>	project full
<div> dettol.jpg</div>	project full
<div> index.html</div>	project full
<div> manifest.json</div>	project full
<div> maskable_icon.png</div>	project full
<div> offline.html</div>	project full
<div> service-worker.js</div>	project full
<div> style.css</div>	project full
<div> surf excel.jpg</div>	project full


←

→

🔄

 jatintalreja0510.github.io/MPL-PWA-APP/

E-commerce Store




Product 1

Description of product 1 goes here.

\$19.99

Add to Cart



Product 2

Description of product 2 goes here.

\$29.99

Add to Cart



# MAD & PWA Lab

## Journal

Experiment No.	11
Experiment Title.	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.
Roll No.	61
Name	Jatin talreja
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	14

Name -Jatin talreja

div-D15a roll no-61

## Mpl Experiment 11

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

**Theory :**

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

### Google Lighthouse :

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

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

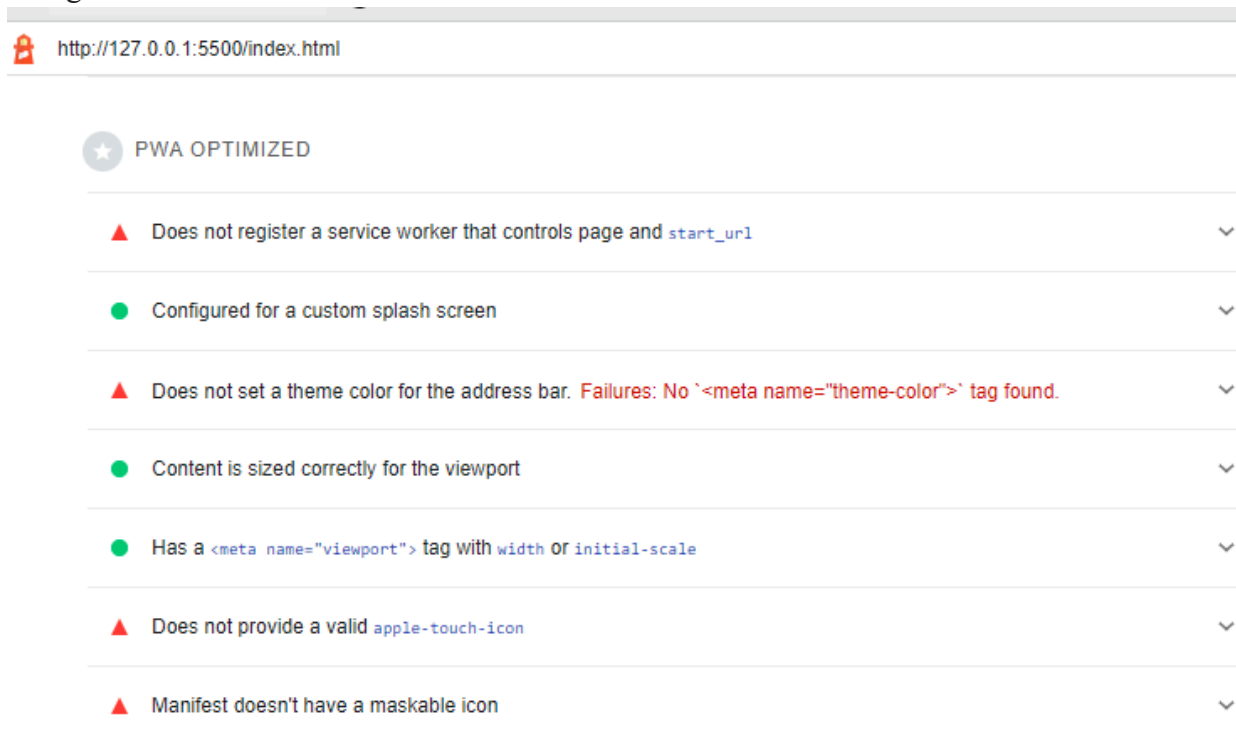
### Key Features and Audit Metrics

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

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

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

Changes made to the code :



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

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

For a maskable icon add "purpose": "any maskable" to the icons in manifest.json file

For apple touch icon add the following meta tag in index.html-

```
<link rel="apple-touch-icon" href="">
```

### Changes in manifest.json

```
{
  "name": "flower shop website",
  "short_name": "flowers",
  "start_url": "index.html", "scope":
  "./",
  "theme_color": "#ffd31d",
  "background_color": "#333",
  "display": "standalone",
  "icons": [
    {
      "src": "icon-1.png",
      "sizes": "192x192", "type":
      "image/png"
      "purpose": "any maskable"


    },
    {
      "src": "icon-2.png",
      "sizes": "512x512", "type":
      "image/png"
      "purpose": "any maskable"

    }
  ]
}
```

The Lighthouse tool provides links to content hosted on third-party websites. [Don't show again](#) [Learn more](#)

2:07:49 PM - 127.0.0.1:5500

http://127.0.0.1:5500/



## PWA

These checks validate the aspects of a Progressive Web App. [Learn more](#).

+

INSTALLABLE

●

Web app manifest and service worker meet the installability requirements

★

PWA OPTIMIZED

●

Registers a service worker that controls page and `start_url`

●

Configured for a custom splash screen

●

Sets a theme color for the address bar.

●

Content is sized correctly for the viewport

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

