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 **Mr Chetan Kashinath Pilane** of **D15A/D15B** 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: MAD/Whatsapp Clone/Pwa/Lifestyle Store **Roll no:44**

Name of the Course : MAD & PWA Lab **Course Code :** ITL604

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

Faculty Incharge : Mrs. Kajal Joseph.

Lab Teachers : Mrs. Kajal Jewani.

Email : kajal.jewani@ves.ac.in

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

Program specific Outcomes

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

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

Project Title: MAD/Whatsapp Clone/Pwa/Lifestyle Store **Roll no:44**

Lab Objectives:

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

Lab Outcomes:

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

Index

Sr. No	Experiment Title	LO	DOP	DOS	Grade
1.	To install and configure the Flutter Environment	LO1	17/1	24/1	
2.	To design Flutter UI by including common widgets.	LO2	24/1	31/1	
3.	To include icons, images, fonts in Flutter app	LO2	31/1	7/2	
4.	To create an interactive Form using form widget	LO2	7/2	14/2	
5.	To apply navigation, routing and gestures in Flutter App	LO2	14/2	21/2	
6.	To Connect Flutter UI with fireBase database	LO3	21/2	6/3	
7.	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.	LO4	6/3	13/3	
8.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA	LO5	13/3	20/3	
9.	To implement Service worker events like fetch, sync and push for E-commerce PWA	LO5	20/3	27/3	
10.	To study and implement deployment of Ecommerce PWA to GitHub Pages.	LO5	27/3	27/3	
11.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.	LO6	27/3	27/3	
12.	Assignment-1	LO1,LO2 ,LO3	4/2	5/2	
13.	Assignment-2	LO4,LO5 ,LO6	20/3	21/3	

MAD & PWA Lab
Journal

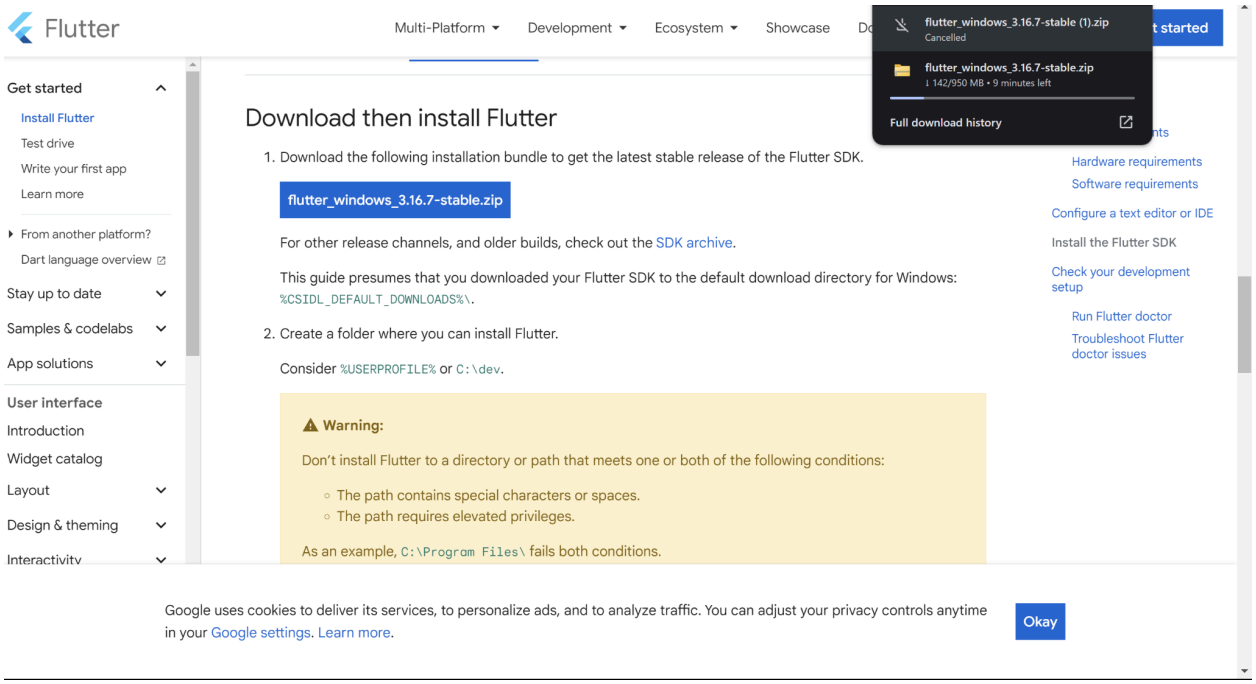
Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	44
Name	Chetan K Pilane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	15

Project Title: MAD/Whatsapp Clone/Pwa/Lifestyle Store **Roll no:44**

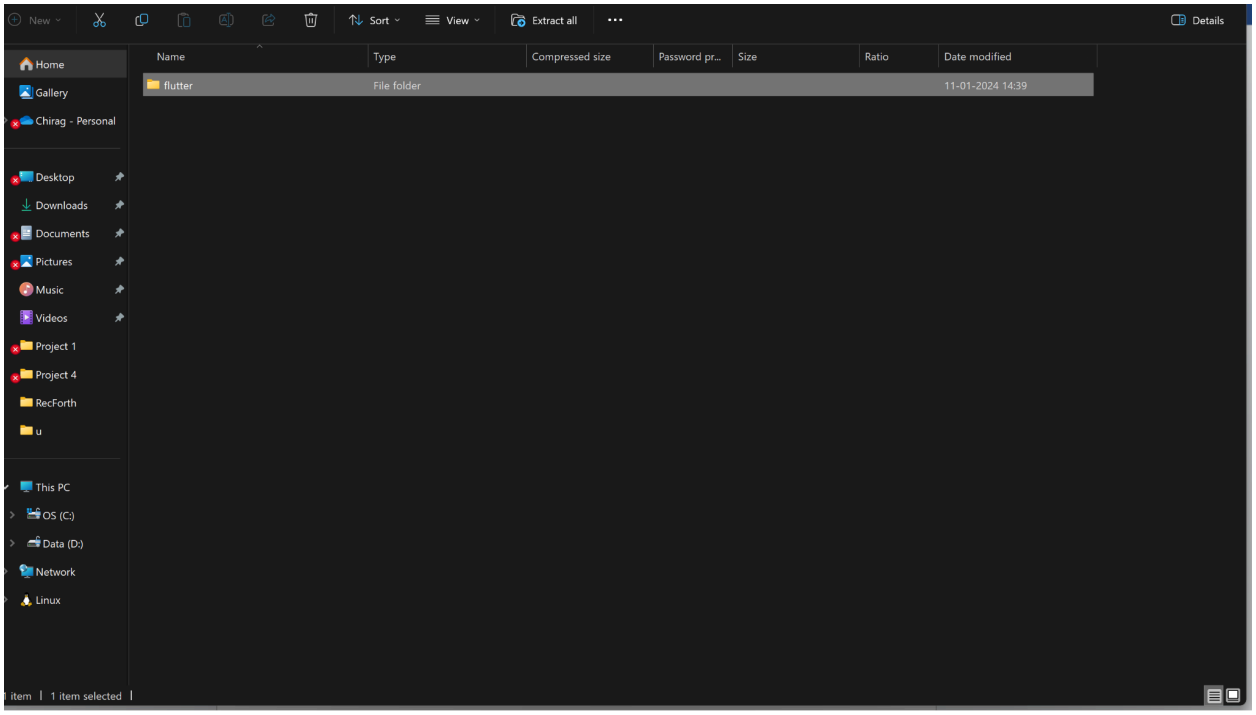
EXP 1 MAD PWD LAB

Aim: Flutter and android studio code installation along with sample program.

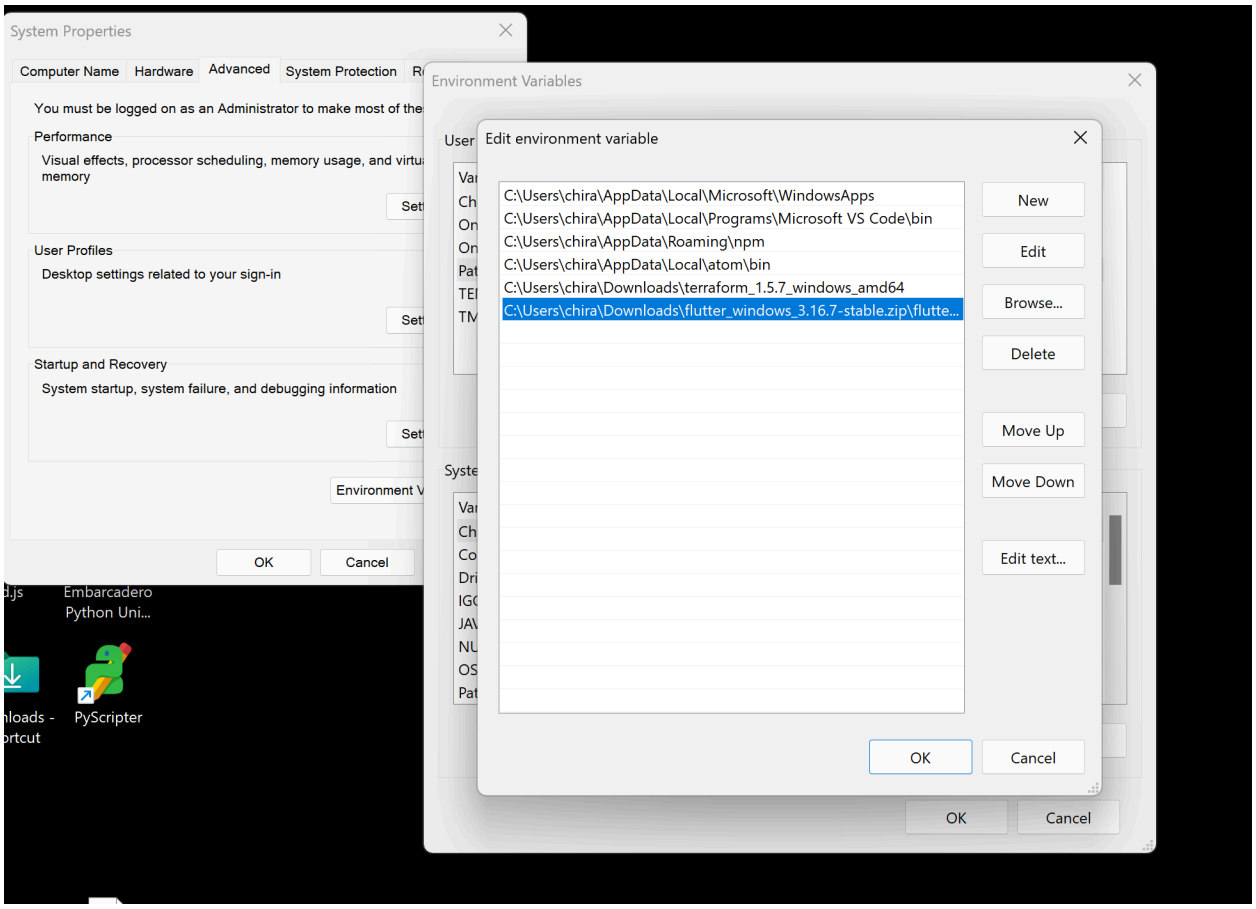
Step 1:Install flutter and android sdk from https://docs.flutter.dev/get-started/install



Step 2: extract the zip file in folder



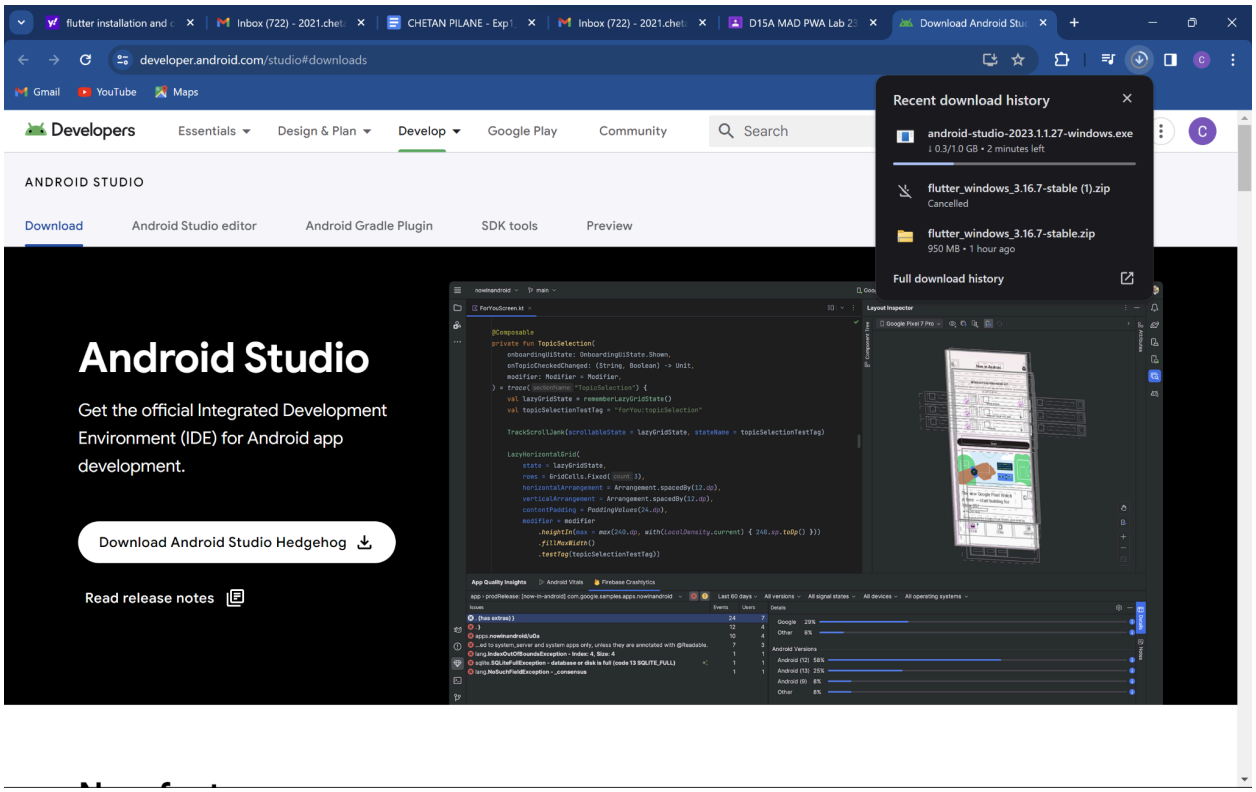
Step 3: Change the environment variables for flutter configuration

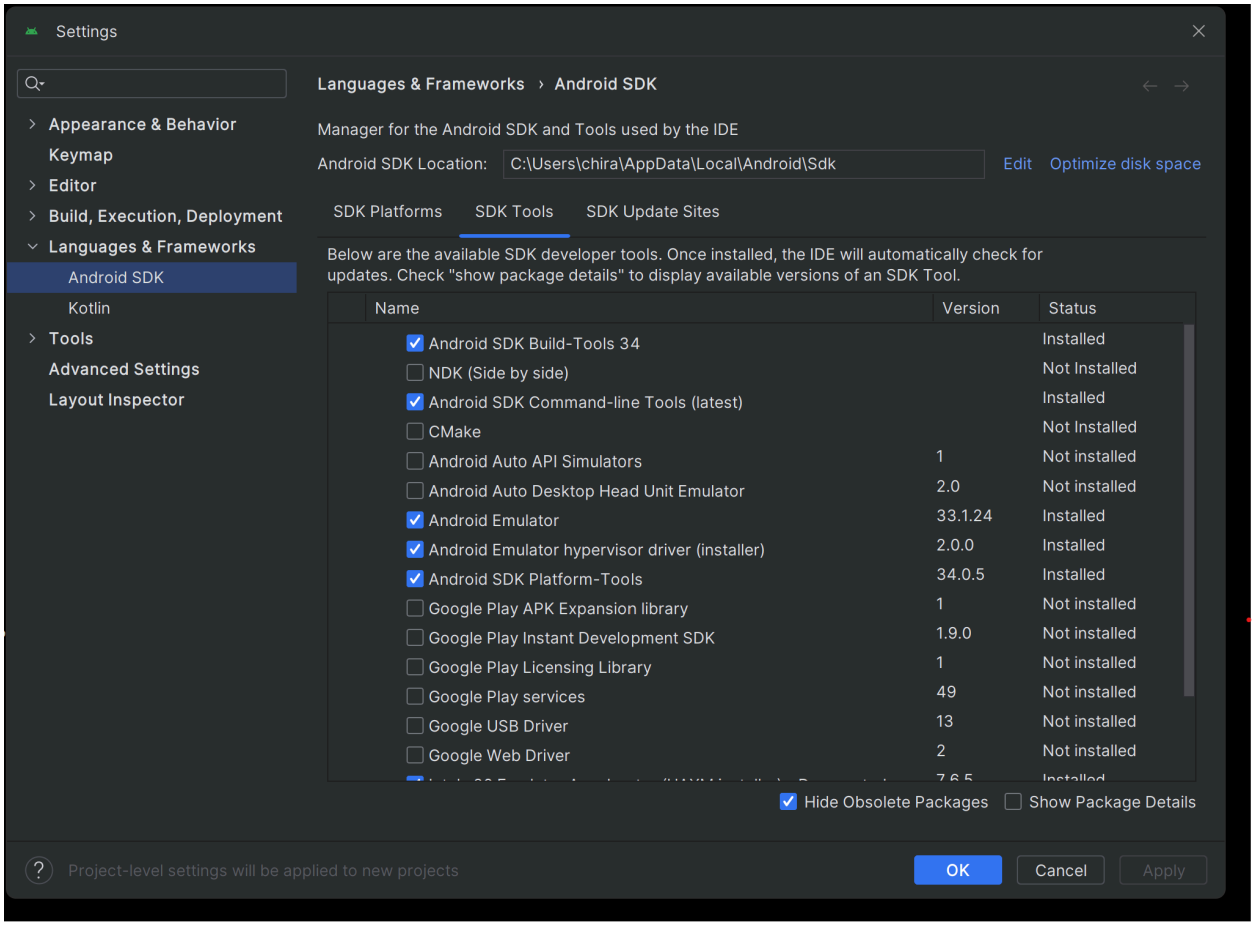


Step 4: run flutter and flutter doctor commands on command prompt

```
C:\Users\chira>flutter doctor
Terminate batch job (Y/N)? y

C:\Users\chira>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.16.7, on Microsoft Windows [Version 10.0.22621.2861], locale en-IN)
[√] Windows Version (Installed version of Windows is version 10 or higher)
[X] Android toolchain - develop for Android devices
    X Unable to locate Android SDK.
      Install Android Studio from: https://developer.android.com/studio/index.html
      On first launch it will assist you in installing the Android SDK components.
      (or visit https://flutter.dev/docs/get-started/install/windows#android-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
[!] Visual Studio - develop Windows apps (Visual Studio Build Tools 2019 16.11.29)
    X The current Visual Studio installation is incomplete.
      Please use Visual Studio Installer to complete the installation or reinstall Visual Studio.
[!] Android Studio (not installed)
[√] VS Code (version 1.85.1)
[√] Connected device (3 available)
[√] Network resources

! Doctor found issues in 3 categories.

C:\Users\chira>
```

Step 5: Install android studio code





Step 6:write flutter code

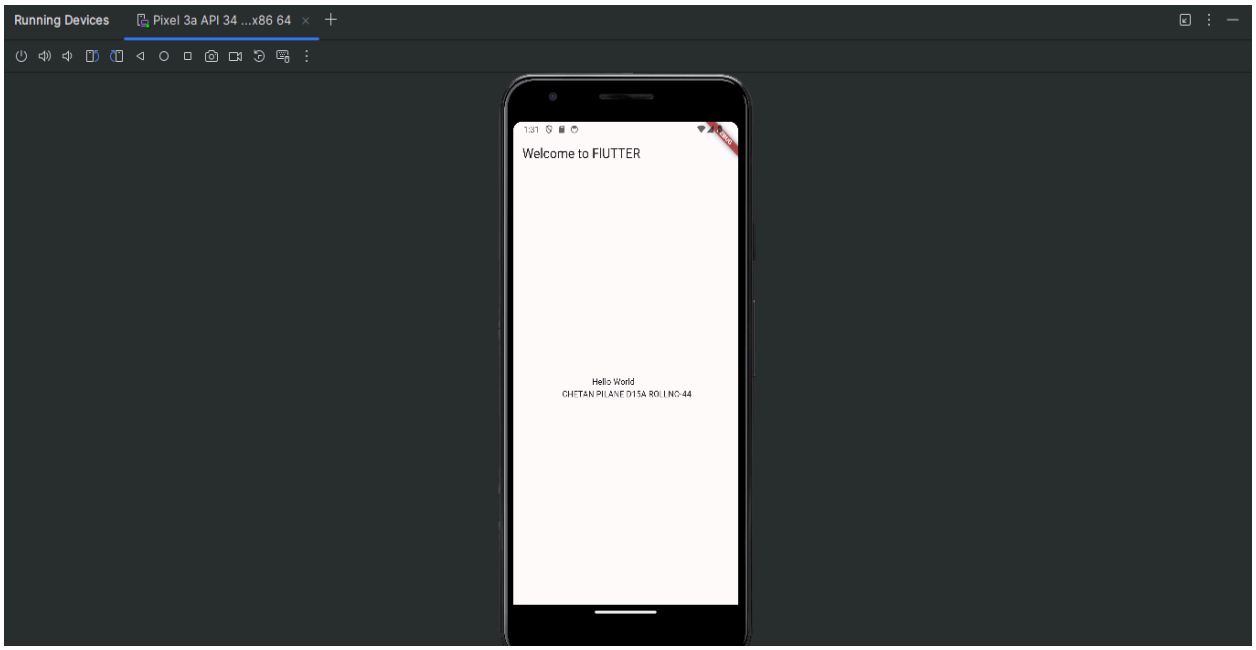
On android studio code after installing dart plugin.

Flutter code:

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

```
        ),
        body: const Center(
          child: Text('              Hello World \n CHETAN PILANE D15A
ROLLNO-44'),

        ),
      ),
    );
  }
}
```

Flutter output:Hello world program



Conclusion: Successfully installed android studio code,flutter and configured flutter to write sample program.

MAD & PWA Lab Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	44
Name	Chetan K Pilane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Project Title: MAD/Whatsapp Clone/Pwa/Lifestyle Store **Roll no:44**

AIM:To design Flutter UI by including common widgets.
Theory:
Common flutter widgets are as follows:
1.Text: The Text widget is used to display a piece of text in a Flutter application. It allows you to customize the text's style, such as font size, color, weight, alignment, and more.

2.Container: The Container widget is a versatile widget used to create visual elements, such as boxes, padding, margins, and borders. It's widely used for layout purposes and can contain other widgets within it.

3.Row and Column: These widgets are used to arrange child widgets horizontally (Row) or vertically (Column). They are fundamental for creating layouts in Flutter, allowing you to build complex UI designs by nesting them within each other.

4.Image: The Image widget is used to display images in a Flutter application. It supports various image formats and can load images from different sources, such as assets, network URLs, and memory.

5.Button Widgets: Flutter provides several button widgets for user interaction, such as ElevatedButton, TextButton, OutlinedButton, and IconButton. These widgets allow users to trigger actions by tapping or clicking on them.

6.TextField: The TextField widget is used to accept user input as text. It provides a text input field where users can type, edit, and submit text data. You can customize its appearance and behavior, including input validation and keyboard type.

7.ListView: The ListView widget is used to display a scrollable list of widgets. It's commonly used for displaying large sets of data or dynamic content. Flutter offers various types of list views, such as ListView, GridView, and ListView.builder, each with its own use cases and performance optimizations.

8.Stack: The Stack widget allows you to stack multiple widgets on top of each other. It's often used to overlay widgets or create complex layouts where widgets can overlap and be positioned relative to each other.

9.AppBar: The AppBar widget represents the top app bar in a Flutter application. It typically contains a title, leading and trailing actions, and can be customized with various properties to achieve different designs.

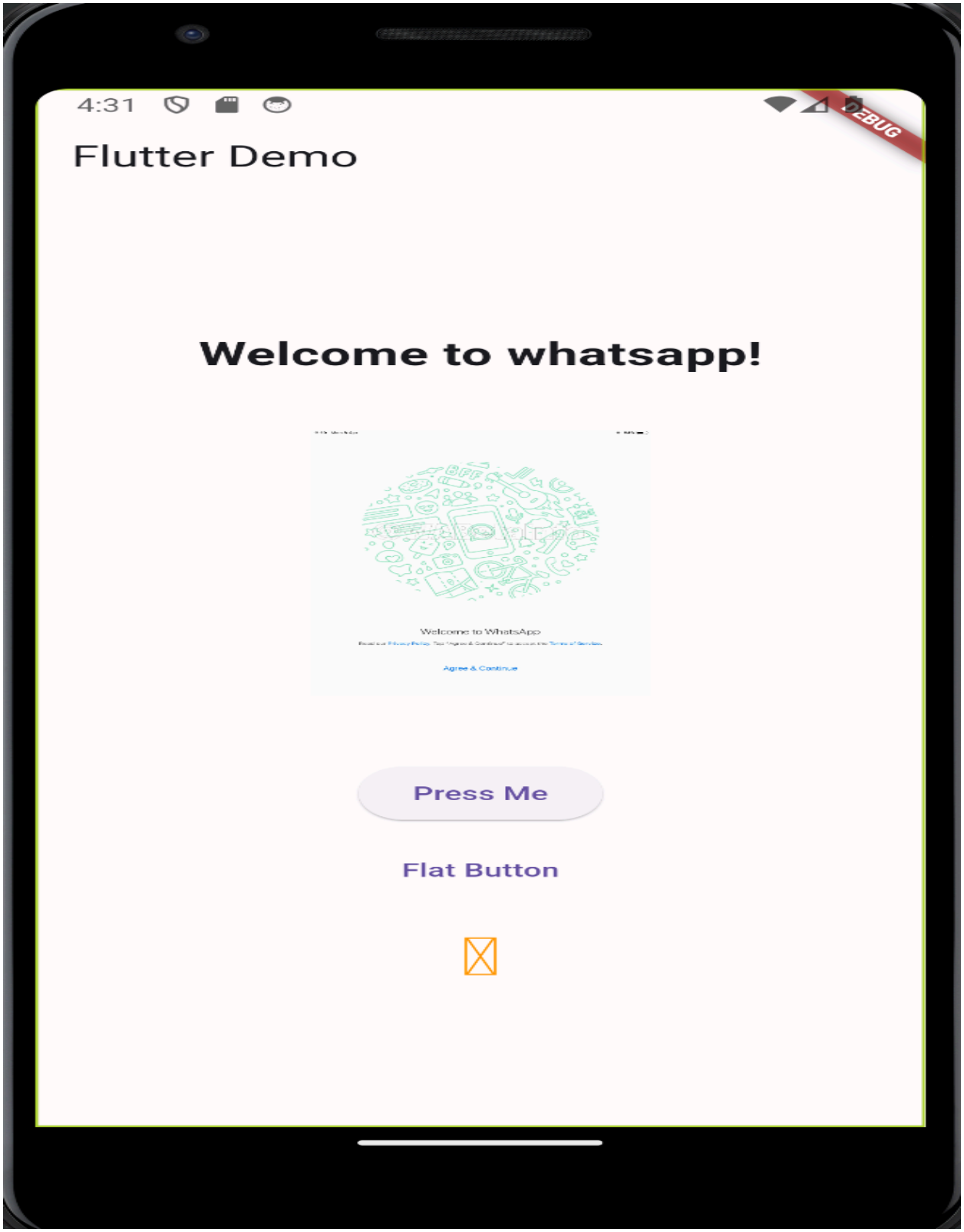
10.Scaffold: The Scaffold widget is a fundamental building block for Flutter applications. It provides a layout structure for the app, including support for app bars, navigation drawers, bottom sheets, and more.
Code:

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

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
```

```
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: MyHomePage(),
    );
  }
}

class MyHomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Flutter Demo'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text(
              'Welcome to whatsapp!',
              style: TextStyle(
                fontSize: 24.0,
                fontWeight: FontWeight.bold,
              ),
            ),
            SizedBox(height: 40.0),
            Image.asset(
              'assets/whatsapp image.jpg', // Change this to your image path
              width: 150.0,
            ),
            SizedBox(height: 50.0),
            ElevatedButton(
              onPressed: () {
                // Add your onPressed functionality here
              },
              child: Text('Press Me'),
            ),
            SizedBox(height: 10.0),
            TextButton(
              onPressed: () {
                // Add your onPressed functionality here
              },
              child: Text('Flat Button'),
            ),
            SizedBox(height: 10.0),
            IconButton(
              onPressed: () {
```

```
              // Add your onPressed functionality here
            },
            icon: Icon(Icons.star),
            iconSize: 40.0,
            color: Colors.orange,
          ),
        ],
      ),
    ),
  );
 }
}
```

Output:



Conclusion:Successfully understood and demonstrated the use of basic widgets in flutter.

MAD & PWA Lab Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	44
Name	Chetan Kashinath Pilane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

AIM: To make use icons,fonts and images in flutter UI

THEORY:

Icons, fonts, and images are essential elements in UI design, providing visual representation, branding, and enhancing user experience.

1.Icons:

Icons are small, graphical representations of actions, objects, or concepts.

They serve as visual cues to convey meaning or functionality in an application.

Icons help users quickly recognize and interact with various elements within the user interface.

Code for adding icons in flutter ui:

```
SizedBox(height: 20.0),
Icon(
  Icons.favorite,
  size: 56.0,
  color: Colors.purpleAccent,
),  // Icon
SizedBox(height: 20.0),
```

2.Fonts:

Fonts are sets of typefaces or styles used to display text in a consistent and visually appealing manner.

They contribute to the overall aesthetics and readability of the UI.

Different fonts evoke different emotions and can reflect the brand's identity or theme of the application.

Code for adding fonts:

```
Text(
  'Hello, Flutter!',
  style: TextStyle(
    fontFamily: 'Roboto',
    fontSize: 24.0,
    fontWeight: FontWeight.bold,
  ),  // TextStyle
),  // Text
```

3.Images

Images are visual representations of objects, scenes, or graphics used to enhance the UI.

They provide context, branding, and visual appeal to the application.

Images can be in various formats, such as PNG, JPEG, GIF, SVG, etc.

In Flutter, images can be loaded from local assets, network URLs, or memory.

Local images are typically stored in the assets folder of the Flutter project and can be displayed using the Image.asset widget.

Code for adding images:

```
SizedBox(height: 20.0),
Image.asset(
  'assets/th.jpeg',
  width: 400,
  height: 400,
),  // Image.asset
  // <Widget>[]
```

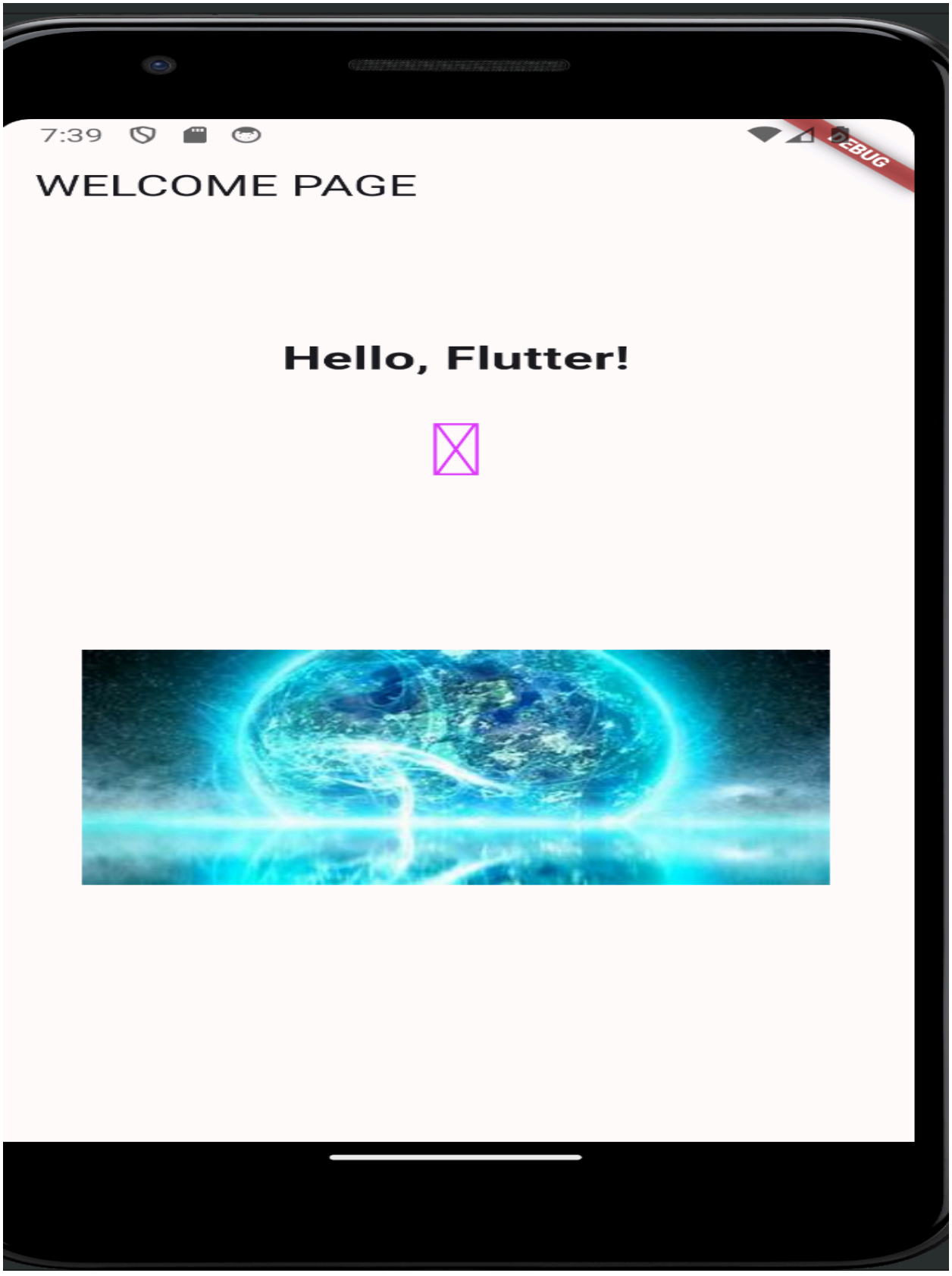
icons, fonts, and images play crucial roles in UI design by providing visual cues, enhancing readability, and contributing to the overall aesthetics and user experience of the application.

Code:

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

void main() {
 runApp(MyApp());
}

class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     home: Scaffold(
       appBar: AppBar(
         title: Text('WELCOME PAGE'),
       ),
       body: Center(
         child: Column(
           mainAxisAlignment: MainAxisAlignment.center,
           children: <Widget>[
             Text(
               'Hello, Flutter!',
               style: TextStyle(
                 fontFamily: 'Roboto',
                 fontSize: 24.0,
                 fontWeight: FontWeight.bold,
               ),
```

```
            ),             SizedBox(height: 20.0),
            Icon(
              Icons.favorite,
              size: 56.0,
              color: Colors.purpleAccent,
            ),
            SizedBox(height: 20.0),
            Image.asset(
              'assets/th.jpeg',
              width: 400,
              height: 400,
            ),
          ],
        ),
      ),
    ),
  );
 }
}
```

Output:


7:39
DEBUG
WELCOME PAGE
Hello, Flutter!

Conclusion: Successfully made use of flutter widgets like fonts, icons and images in Flutter UI.

MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	44
Name	Chetan Pilane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

AIM:**To create an interactive Form using form widget.**
THEORY:

In Flutter, form widgets and input widgets play a crucial role in creating interactive user interfaces, especially when dealing with user input and data submission.
1.Form Widgets:
The Form widget is a container that holds form fields for gathering user input.
It helps in managing form state, validation, and submission.
A GlobalKey<FormState> is required to uniquely identify the form and access its state.
2.Text Input Widgets:
Flutter provides several text input widgets for gathering textual user input:
1.TextField: Basic text input field.
2.TextFormField: A form field that integrates with the Form widget and provides features like validation, error handling, and saving form data.
These widgets allow users to input text using the device's keyboard.
3.Input Validation:
Input validation ensures that user input meets certain criteria or constraints before proceeding.
Each input widget typically includes a validator property where you can provide a validation function.

4.Form Submission:
Form submission involves validating user input and handling the form data.
After validating all form fields, it calls the save method on the FormState to save the form data.
Upon submission, it can perform actions like sending data to a server, updating local state, or navigating to another screen.
Steps for creating interactive form:

1. Creating a global form key

```
final _formKey = GlobalKey<FormState>();
String _name = '';
```

```
String _email = '';
```

2.Adding text form field with validation logic

```
SizedBox(height: 10),
TextFormField(
 style: TextStyle(color: Colors.black),
 decoration: InputDecoration(
   labelText: 'Email',
   filled: true,
   fillColor: Colors.white.withOpacity(0.7),
   border: OutlineInputBorder(
     borderRadius: BorderRadius.circular(10.0),
   ),
 ),
 validator: (value) {
   if (value == null || value.isEmpty) {
     return 'Please enter your email';
   }
   // This is a basic email validation
   if (!value.contains('@')) {
     return 'Please enter a valid email';
   }
   return null;
```

3.Adding submit button for validation and form submission

```
SizedBox(height: 10),
Padding(
 padding: const EdgeInsets.symmetric(vertical: 16.0),
 child: ElevatedButton(
   onPressed: () {
     _submitForm();
   },
   child: Text('Submit'),
```

Code:

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

void main() {
 runApp(MyApp());
}

class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     title: 'Interactive Form',
     theme: ThemeData(
       primarySwatch: Colors.blue,
```

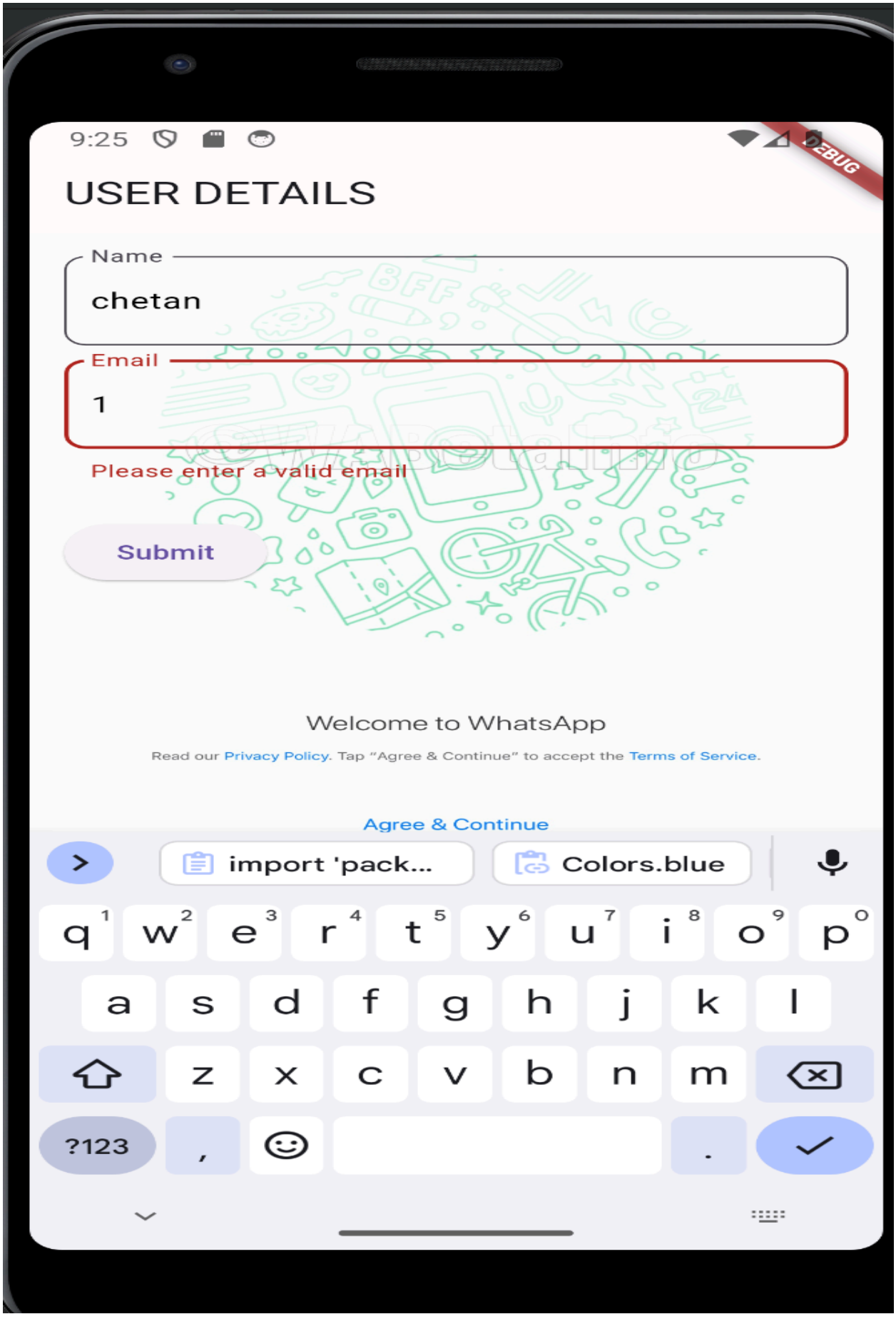
```
    ),
    home: MyForm(),
   );
 }
}

class MyForm extends StatefulWidget {
 @override
 _MyFormState createState() => _MyFormState();
}

class _MyFormState extends State<MyForm> {
 final _formKey = GlobalKey<FormState>();
 String _name = '';
 String _email = '';

 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       title: Text('USER DETAILS'),
     ),
     body: Stack(
       children: [
         // Background Image
         Image.asset(
           'assets/whatsapp image.jpg',
           width: MediaQuery.of(context).size.width,
           height: MediaQuery.of(context).size.height,
           fit: BoxFit.cover,
         ),
         // Form
         Padding(
           padding: EdgeInsets.all(16.0),
           child: Form(
             key: _formKey,
             child: Column(
               crossAxisAlignment: CrossAxisAlignment.start,
               children: <Widget>[
                 TextFormField(
                   style: TextStyle(color: Colors.black),
                   decoration: InputDecoration(
                     labelText: 'Name',
                     filled: true,
                     fillColor: Colors.white.withOpacity(0.7),
                     border: OutlineInputBorder(
                       borderRadius: BorderRadius.circular(10.0),
                     ),
                   ),
```

```
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return 'Please enter your name';
                  }
                  return null;
                },
                onSaved: (value) {
                  _name = value!;
                },
              ),
              SizedBox(height: 10),
              TextFormField(
                style: TextStyle(color: Colors.black),
                decoration: InputDecoration(
                  labelText: 'Email',
                  filled: true,
                  fillColor: Colors.white.withOpacity(0.7),
                  border: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(10.0),
                  ),
                ),
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return 'Please enter your email';
                  }
                  // This is a basic email validation
                  if (!value.contains('@')) {
                    return 'Please enter a valid email';
                  }
                  return null;
                },
                onSaved: (value) {
                  _email = value!;
                },
              ),
              SizedBox(height: 10),
              Padding(
                padding: const EdgeInsets.symmetric(vertical: 16.0),
                child: ElevatedButton(
                  onPressed: () {
                    _submitForm();
                  },
                  child: Text('Submit'),
                ),
              ),
            ],
          ),
        ),
      ),
```

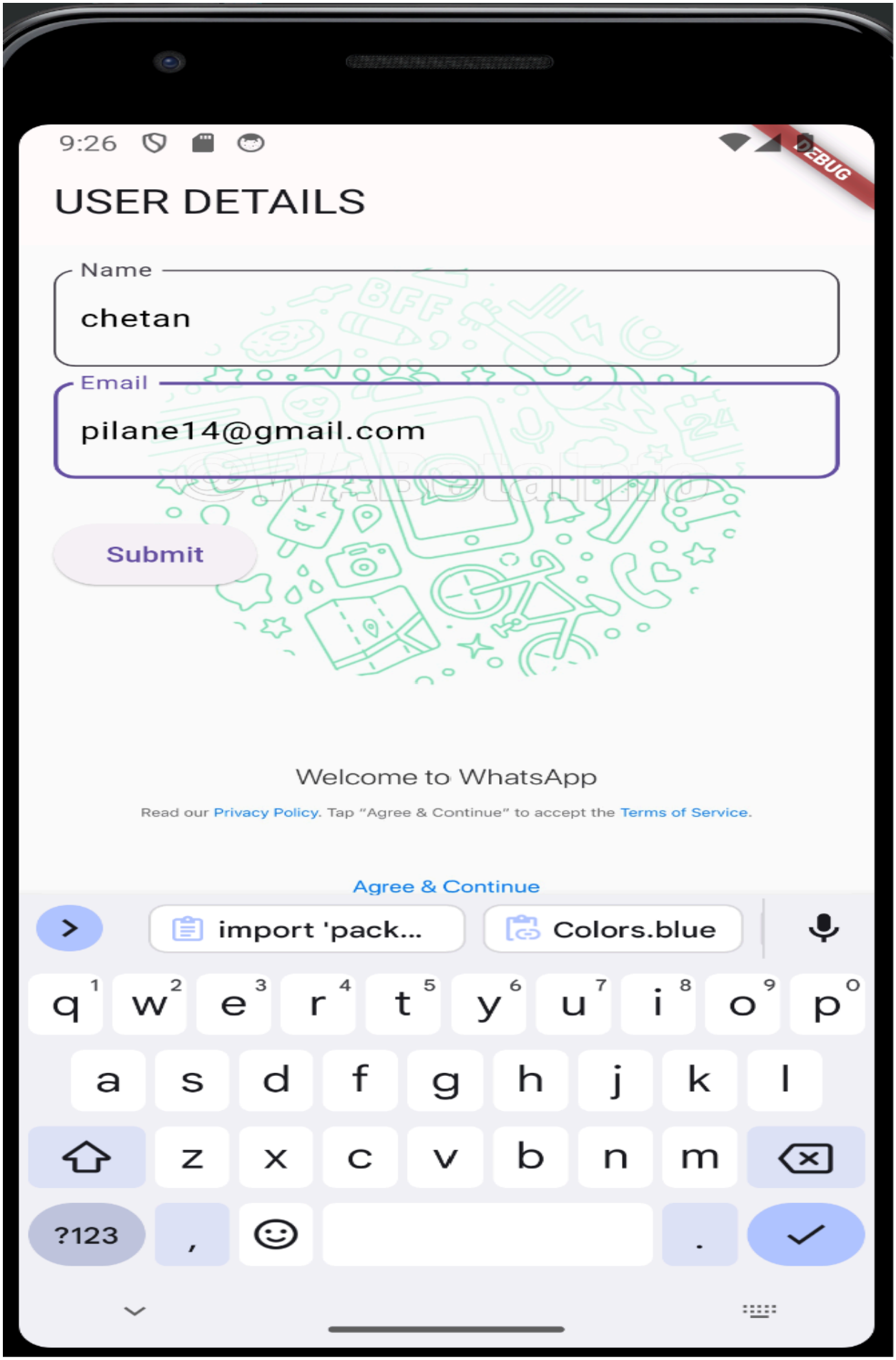
```
        ],
      ),
    );
  }

  void _submitForm() {
    if (_formKey.currentState!.validate()) {
      _formKey.currentState!.save();
      // You can handle form submission here
      print('Name: $_name, Email: $_email');
    }
```

OUTPUT:

1.Incorrect details


9:25
DEBUG
USER DETAILS
Name
chetan
Email
1
Please enter a valid email
Submit
Welcome to WhatsApp
Read our Privacy Policy. Tap "Agree & Continue" to accept the Terms of Service.
Agree & Continue
import 'pack...
Colors.blue

2.Successful submission


9:26
DEBUG
USER DETAILS
Name
chetan
Email
pilane14@gmail.com
Submit
Welcome to WhatsApp
Read our Privacy Policy. Tap "Agree & Continue" to accept the Terms of Service.
Agree & Continue
import 'pack...
Colors.blue
?123

MAD & PWA Lab Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	44
Name	Chetan.k.Pilane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

AIM:TO APPLY NAVIGATION,ROUTING AND GESTURES IN FLUTTER APP.

Theory:

In Flutter, navigation and routing are essential for building dynamic and interactive user interfaces.
Navigation refers to the process of moving between different screens or "routes" within your application.
Routing, on the other hand, is the mechanism that manages the navigation flow, determining which screen to display based on user interactions or application logic.
Following are the methods to handle routing in flutter

1.Navigator class: Flutter provides the Navigator class, which manages a stack of Route objects. You can push new routes onto the stack to navigate forward, pop routes to navigate backward, or replace routes to update the current screen.

2.Named routes: Named routes allow you to define routes with unique names and associated widgets. This approach is particularly useful for managing navigation in larger applications. You can define named routes in your app's MaterialApp or CupertinoApp widget using the routes parameter.
3.Navigation methods: Flutter provides various methods for navigating between screens:

1.Navigator.push(): Pushes a new route onto the navigator's stack.
2.Navigator.pop(): Pops the current route off the navigator's stack.
3.Navigator.pushNamed(): Pushes a named route onto the navigator's stack.
4.Navigator.popAndPushNamed(): Pops the current route off the stack and pushes a named route.
5.Route generation: You can implement custom logic for generating routes dynamically based on certain conditions. This is often done using the onGenerateRoute callback in MaterialApp or CupertinoApp.

6.Passing data between screens: You can pass data between screens when navigating using constructor arguments or using the ModalRoute static methods to access the route's settings.

Code:

```
static Route<dynamic> onGenerateRoute(RouteSettings settings) {
 switch (settings.name) {
   case welcome:
     return MaterialPageRoute(
       builder: (context) => const WelcomePage(),
     );

   case login:
     return MaterialPageRoute(
       builder: (context) => const LoginPage(),
     );

   case verification:
     final Map args = settings.arguments as Map;
     return MaterialPageRoute(
       builder: (context) => VerificationPage(phoneNumber: '',
           //VerificationId: args['verification'],
           //PhoneNumber: args['phone Number'],
           ),
     );
```

OUTPUT:

SCREEN 1

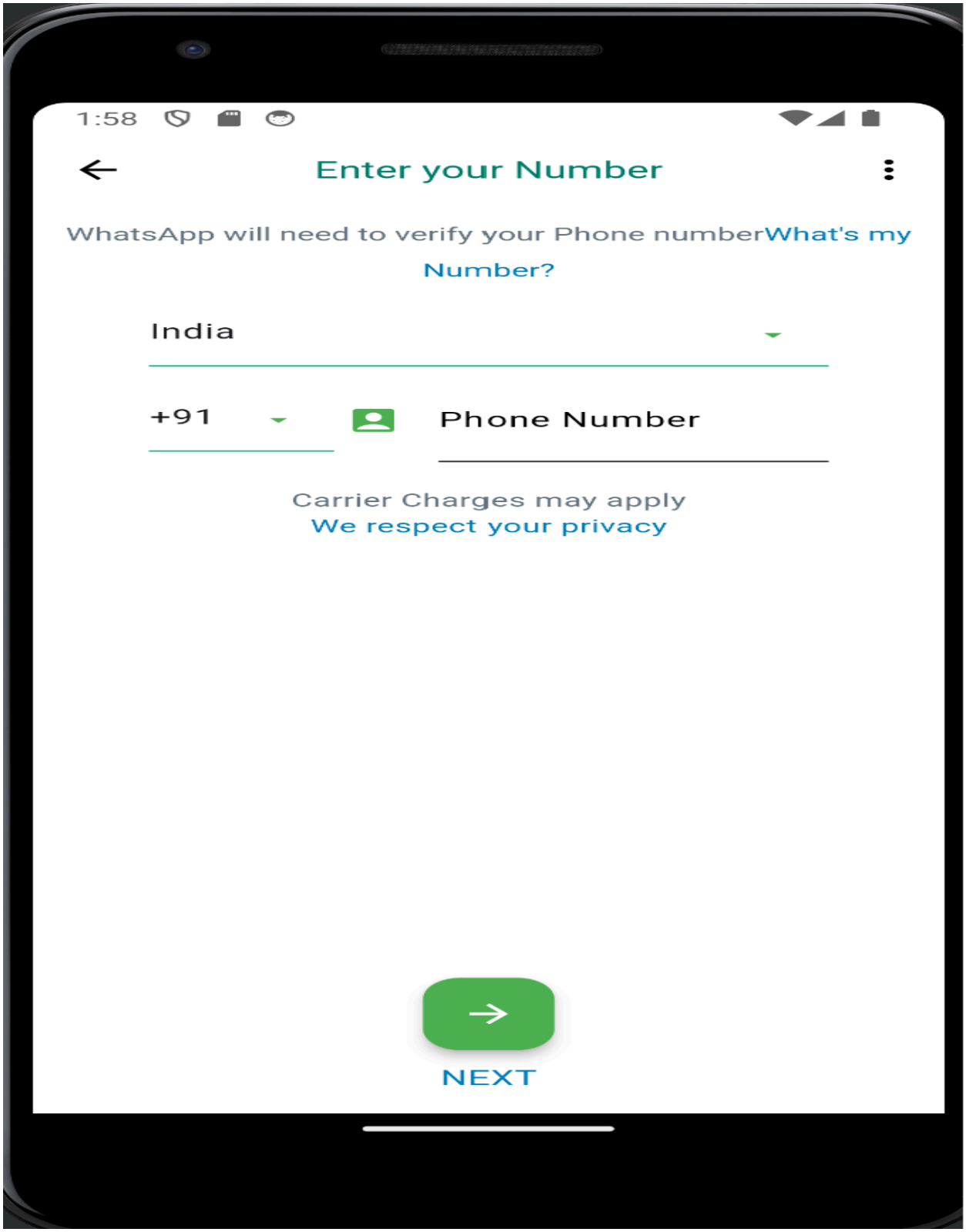
Welcome to WhatsApp

Read our Privacy Policy. Tap " Agree and continue"
to accept our Terms of Services

AGREE AND CONTINUE

English

SCREEN 2



Conclusion:Successfully studied the concept of routing in flutter web app.

MAD & PWA Lab Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	44
Name	Chetan .k .Pilane
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:	

AIM:To setup Firebase in Flutter Project

Theory:

Setting up flutter Firebase in IOS app:

Firebase is a great backend solution for anyone that wants to use authentication, databases, cloud functions, ads, and countless other features within an app.
In this article, you will create a Firebase project for iOS and Android platforms using Flutter.
Prerequisites for Firebase setup:
1.A Google account to use Firebase.
2.Developing for iOS will require XCode.
3.To download and install Flutter.
4.To download and install Android Studio and Visual Studio Code.
5.It is recommended to install plugins for your code editor:
6.Flutter and Dart plugins installed for Android Studio.
7.Flutter extension installed for Visual Studio Code.

Code:

Firebase.options file

```
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:
      throw UnsupportedError(
        'DefaultFirebaseOptions have not been configured for macos - '
        'you can reconfigure this by running the FlutterFire CLI again.',
      );
    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: 'AIzaSyD-_VN8edivHrRv5xd9QVgnYv-pV1Rl0WI',
  appId: '1:724141443535:web:eada6c562a8f679026eaba',
  messagingSenderId: '724141443535',
  projectId: 'whatsapp-new-clone-3fa2b',
  authDomain: 'whatsapp-new-clone-3fa2b.firebaseapp.com',
  storageBucket: 'whatsapp-new-clone-3fa2b.appspot.com',
  measurementId: 'G-L3LR8LLGB7',
);

static const FirebaseOptions android = FirebaseOptions(
  apiKey: 'AIzaSyCNJou1wH1xCqCPu-9yUpVcXuMOOduCShY',
  appId: '1:724141443535:android:5ef3e32a07637f1926eaba',
  messagingSenderId: '724141443535',
  projectId: 'whatsapp-new-clone-3fa2b',
  storageBucket: 'whatsapp-new-clone-3fa2b.appspot.com',
);

static const FirebaseOptions ios = FirebaseOptions(
  apiKey: 'AIzaSyClZNaP9NVYF18N3gVMCqW81ElLsds8mLk',
  appId: '1:724141443535:ios:ac7cca22ed2f848926eaba',
  messagingSenderId: '724141443535',
  projectId: 'whatsapp-new-clone-3fa2b',
  storageBucket: 'whatsapp-new-clone-3fa2b.appspot.com',
  iosBundleId: 'com.example.whatsapp',
```

```
 );
}
```

Pubspeck.yaml

```
dependencies:
 flutter:
   sdk: flutter
 # The following adds the Cupertino Icons font to your application.
 # Use with the CupertinoIcons class for iOS style icons.
 cloud_firestore: ^4.9.2
 country_picker: ^2.0.21
 cupertino_icons: ^1.0.2
 firebase_auth: ^4.10.0
 firebase_core: ^2.16.0
 firebase_storage: ^11.2.7
 flutter_native_splash: ^2.3.2
 flutter_hooks: ^0.18.0
 hooks_riverpod: ^2.4.1
 cached_network_image: ^3.3.1
 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
 intl: ^0.19.0
 google_fonts: ^6.1.0
 http: ^1.2.0
 uuid: ^4.3.3
 flutter_launcher_icons: ^0.13.1
 url_launcher: ^6.0.9
 image_picker: ^1.0.7
 fluttertoast: ^8.2.4
 story_view: ^0.16.5

dev_dependencies:
 flutter_test:
   sdk: flutter

 # The "flutter_lints" package or other dev dependencies go here if needed

flutter_native_splash:
 color: "#FFFFFF"
 color_dark: "#111b21"
 image: assets/images/splash_light.png
 image_dark: assets/images/splash_dark.png
 android_12:
   color1: "#FFFFFF"
   color_dark1: "#111b21"
```

```
  image1: assets/images/splash_light.png
  image_dark1: assets/images/splash_dark.png
  icon_background_color: "#FFFFFF"
  icon_background_color_dark: "#111B21"
```

OUTPUT:

Firebase for authentication:

Users using whatsapp.

Conclusion:

Successfully implemented the Firebase setup in flutter project.

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.	44
Name	Chetan Pilane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	15

Experiment No 7

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

Theory:-
Regular Web App
A regular web app is a website that is designed to be accessible on all mobile devices such that the content gets fit as per the device screen. It is designed using a web technology stack (HTML, CSS, JavaScript, Ruby, etc.) and operates via a browser. They offer various native-device features and functionalities. However, it entirely depends on the browser the user is using. In other words, it might be possible that you can access a native-device feature on Chrome but not on Safari or Mozilla Firefox because the browsers are incompatible with that feature.
Progressive Web App
Progressive Web App (PWA) is a regular web app, but some extras enable it to deliver an excellent user experience. It is a perfect blend of desktop and mobile application experience to give both platforms to the end-users.

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

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

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

4. Engaging Approach

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

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

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

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

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

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

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

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

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

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

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

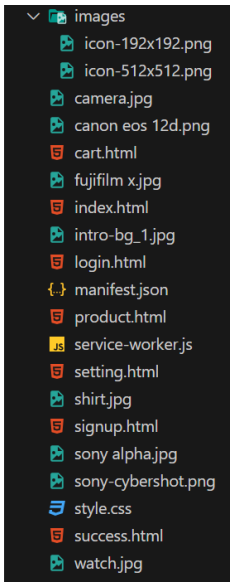
Linkable — Easily shared via URL without complex installations.

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

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

Project Title: MAD/Whatsapp Clone/Pwa/Lifestyle Store **Roll no:44**



Index.html

```
<!DOCTYPE html>
<html>

<head>
    <meta name="apple-mobile-web-app-status-bar" content="#aa7700">
    <meta name="theme-color" content="black">
    <link rel="manifest" href="manifest.json">
    <script src="service-worker.js"></script>
    <title>
        Index
    </title>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.cs
s">

    <!--jQuery library-->
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></
script>

    <!--Latest compiled and minified JavaScript-->
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js">
</script>
    <meta name="viewport" content="width=device-width, initial-scale=1">
```

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

<body>

    <nav class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle"
data-toggle="collapse" data-target="#mynavbar">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <a class="navbar-brand" href="index.html">Lifestyle
Store</a>

            </div>
            <div class="collapse navbar-collapse" id="mynavbar">
                <ul class="nav navbar-nav navbar-right">
                    <li>
                        <a href="signup.html">
                            <span class="glyphicon glyphicon-user" />
Sign-Up </a>
                    </li>
                    <li>
                        <a href="login.html">
                            <span class="glyphicon glyphicon-log-in" />
Login </a>
                    </li>
                </ul>
            </div>

        </div>

    </nav>
    <div class="banner-image">
        <div class="container">
            <div class="banner-content" style="margin-left :25%">
                <h1>We sell lifestyle</h1>
```

```
                <p>Flat 40% OFF on premium brands</p> <br>
                <a href="product.html" class="btn btn-danger btn-lg
active">Shop Now</a>
            </div>
        </div>
    </div>
    <footer>
        <div class="container">
            <p style="text-align:center;">Copyright © Lifestyle Store. All
Rights Reserved and Contact Us: +91 90000
                00000 </p>
        </div>
    </footer>
    <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);
                }
            }
        }
    </script>
```

```
</body>

</html>
```

Manifest.json

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

Service-worker.json

```
// service-worker.js

const CACHE_NAME = 'my-ecommerce-app-cache-v1';
const urlsToCache = [
  '/',
  'cart.html',
```

```
  'index.html',
  'product.html',
  'shop.html',
  'style.css',
  'success.html',
  'service-worker.js',
  'manifest.json'

  // Add more files to cache as needed
];

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

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

Product.html

```
<!DOCTYPE html>
<html>

    <head>
    <title>
        product
    </title>
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.cs
s" >

        <!--jQuery library-->
        <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></
script>

        <!--Latest compiled and minified JavaScript-->
        <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js">
</script>
        <meta name="viewport" content="width=device-width,
initial-scale=1">
        <link rel = "stylesheet" href = "style.css">
    </head>
        <body>
            <nav class = "navbar navbar-inverse navbar-fixed-top">
                    <div class ="container">
                        <div class ="navbar-header">
                            <button type="button" class ="navbar-toggle"
data-toggle="collapse" data-target="#mynavbar">
                                <span class="icon-bar"></span>
                                <span class="icon-bar"></span>
                                <span class="icon-bar"></span>
                            </button>
                            <a class="navbar-brand"
href="index.html">Lifestyle Store</a>

                        </div>
                        <div class="collapse navbar-collapse"
id="mynavbar">
```

```
                        <ul class="nav navbar-nav navbar-right">
                            <li>
                                <a href="cart.html">
                                    <span class="glyphicon
glyphicon-shopping-cart">  Cart </span> </a>
                            </li>
                            <li>
                                <a href="setting.html">
                                    <span class="glyphicon
glyphicon-user">  Setting</span>  </a>
                            </li>
                            <li>
                                <a href="index.html">
                                    <span class="glyphicon
glyphicon-log-out">    Logout</span></a>
                            </li>
                        </ul>
                    </div>

                </div>

            </nav>

            <div class =" container" style="margin-top: 5%;">

            <div class ="jumbotron">

            <h1> Welcome  to our Lifestyle Store!  </h1>
            <p>We have the best cameras, watches and shirts for you.
No need to hunt
            around, we have all in one place.  </p>
            </div>

            <div class="row text-center">
            <div class="  col-md-3 col-sm-6  thumbnail " >
            <img src="canon eos 12d.png">
            <div class="caption">
            <h2>Canon EOS 12D</h2>
            <p>Full Zoom Quality</p>
```

```
                </div>
                <div class=" btn btn-primary  btn-block btn-md"> 29000
                </div>
                </div>


                <div class="  col-md-3 col-sm-6  thumbnail " >
                <img src="sony-cybershot.png">
                <div class="caption">
                <h2>Sony CyberShot</h2>
                <p>48MP Camera</p>

                </div>
                <div class=" btn btn-primary  btn-block btn-md"> 19000
                </div>
                </div>


                <div class="  col-md-3 col-sm-6  thumbnail " >
                <img src="sony alpha.jpg">
                <div class="caption">
                <h2>Sony Alpha</h2>
                <p>Flagship Camera</p>

                </div>
                <div class=" btn btn-primary  btn-block btn-md"> 50000
                </div>
                </div>

                <div class="  col-md-3 col-sm-6  thumbnail " >
                <img src="fujifilm x.jpg">
                <div class="caption">
                <h2>Fujifilm X</h2>
                <p>Best off all</p>
                </div>

                <div class=" btn btn-primary  btn-block btn-md"> 29000
                </div>

                </div>
```

```
</div>


<div class="row text-center">

<div class="  col-md-3 col-sm-6  thumbnail " >
<img src="watch.jpg">
<div class="caption">
<h2>ROLEX</h2>
<p>Best Watch</p>
</div>
<div class=" btn btn-primary  btn-block btn-md"> 200000
</div>
</div>


<div class="  col-md-3 col-sm-6  thumbnail " >
<img src="watch.jpg">
<div class="caption">
<h2>Sonata</h2>
<p>Cheapest Watch</p>
</div>
<div class=" btn btn-primary  btn-block btn-md"> 2000
</div>
</div>


<div class="  col-md-3 col-sm-6  thumbnail " >
<img src="watch.jpg">
     <div class="caption">
    <h2>TIMEX by Titan</h2>
    <p>Premium watch</p>
    </div>
    <div class=" btn btn-primary  btn-block btn-md"> 20000
</div>

</div>

<div class="  col-md-3 col-sm-6  thumbnail " >
```

```
                <img src="watch.jpg">
                <div class="caption">
                <h2>JACOB&CO. WATCHES </h2>
                <p>Most Expensive</p>
                </div>
                <div class=" btn btn-primary  btn-block btn-md"> 2000000
                </div>

                </div>
                </div>



                <div class="row text-center">

                <div class="  col-md-3 col-sm-6  thumbnail " >
                <img src="shirt.jpg">
                  <div class="caption">
                <h2>Raymond</h2>
                <p>Blue Shirt</p>
                </div>
                <div class=" btn btn-primary  btn-block btn-md">2000
</div>

                </div>

                <div class="  col-md-3 col-sm-6  thumbnail " >
                <img src="shirt.jpg">
                 <div class="caption">
                <h2>Levi</h2>
                <p>Formals</p>
                </div>
                <div class=" btn btn-primary  btn-block btn-md">3000</div>
                </div>

                <div class="  col-md-3 col-sm-6  thumbnail " >
                <img src="shirt.jpg">
                  <div class="caption">
                <h2>Denim</h2>
                <p>Blue Shirt</p>
```

```
                </div>
                <div class=" btn btn-primary  btn-block btn-md">SHIRTS
</div>
                </div>

                <div class="  col-md-3 col-sm-6  thumbnail " >
                <img src="shirt.jpg">
                  <div class="caption">
                <h2>Cambridge</h2>
                <p>Original Shirts and Formals</p>
                </div>
                <div class=" btn btn-primary  btn-block btn-md">SHIRTS
</div>
                </div>
                </div>
                </div>

                <footer style="margin-top: 5%; margin-bottom:.5%;">
                                <div class="container" >
                                    <p
style="text-align:center;">Copyright © Lifestyle Store. All Rights
Reserved and Contact Us: +91 90000 00000 </p>
                                </div>
                    </footer>



        </body>
    </html>
```

Style.css

```
.banner-image
{padding-top: 75px;
 padding-bottom: 50px;
text-align: center;
color: #f8f8f8;
 background: url(intro-bg_1.jpg) no-repeat center center;
background-size: cover;
}
```

```
.banner-content{
  position: relative;

 padding-top: 6%;
 padding-bottom: 6%;

 margin-top: 12%;
 margin-bottom: 12%;
 background-color: rgba(0, 0, 0, 0.7);
 width: 50%;
 text-align:center;
}
footer
{
padding: 10px 0;
background-color: #101010;
color:#9d9d9d;
bottom: 0;
width: 100%;
}
.container{
  width:90%;
 margin:auto;
 overflow:hidden;

}
```

Starting the Server:

Project Title: MAD/Whatsapp Clone/Pwa/Lifestyle Store **Roll no:44**

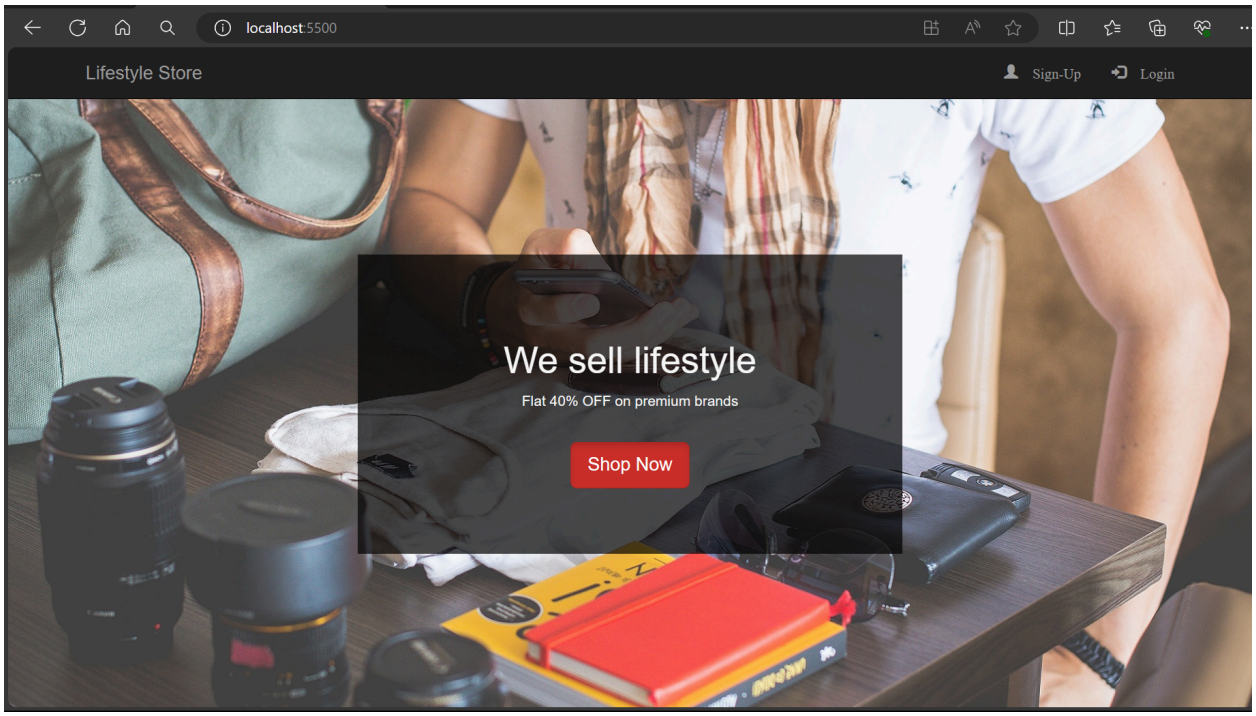
index.html | manifest.json | service-worker.js | style.css

index.html > html > head

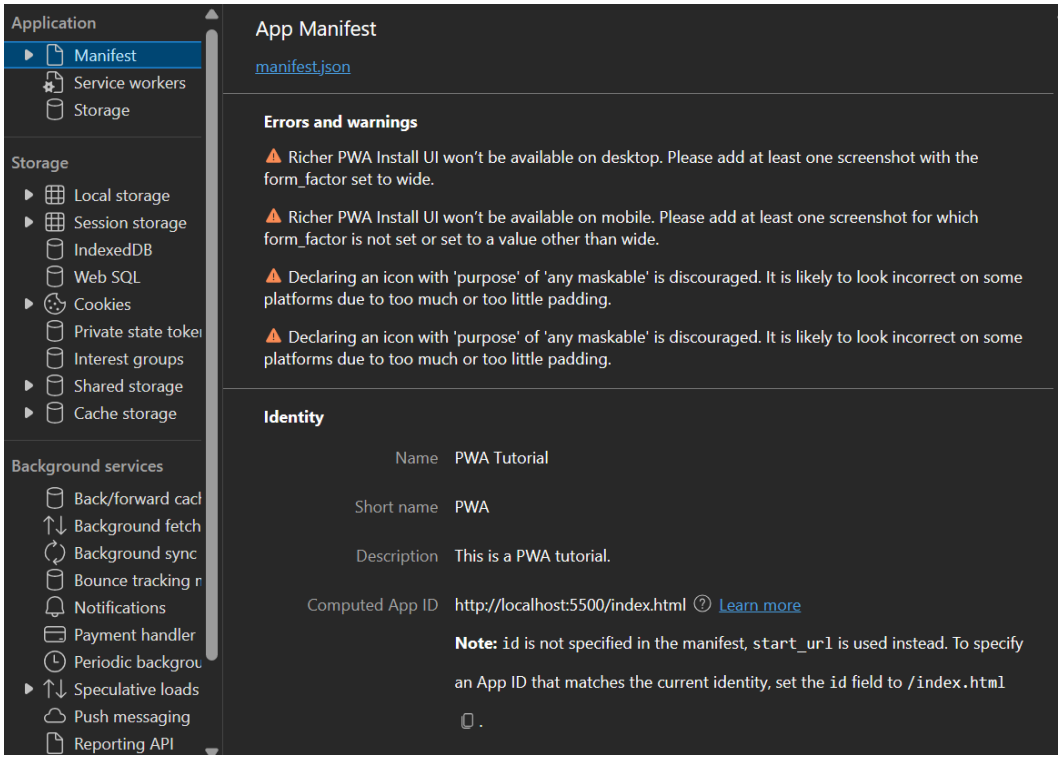
Click here to ask Blackbox to help you code faster

```
<!DOCTYPE html>
<html>

<head>
    <meta name="apple-mobile-web-app-status-bar" content="#aa7700">
    <meta name="theme-color" content="black">
    <link rel="manifest" href="manifest.json">
    <script src="service-worker.js"></script>
    <title>
        Index
    </title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

    <!--jQuery library-->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

    <!--Latest compiled and minified JavaScript-->
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="style.css">
</head>

<body>

    <nav class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#mynavbar">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
```

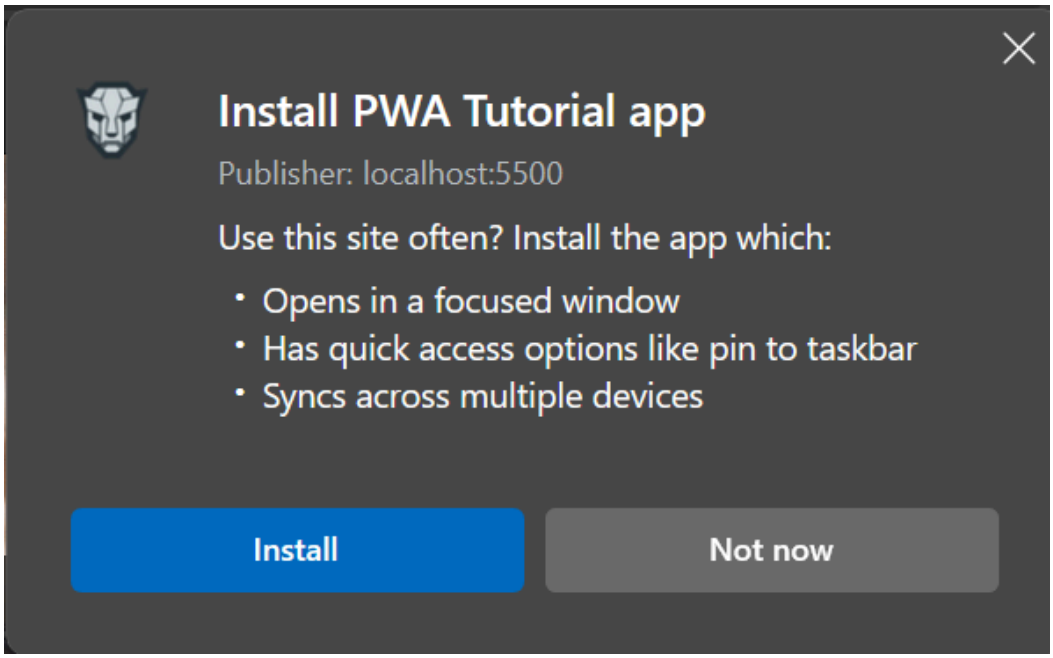
re Code Link | Explain Code | Comment Code | Find Bugs | Code Chat | Blackbox | Ln 17, Col 51 | Tab Size: 4 | UTF-8 | CRLF | HTML | Port : 5500 | Blackbox



Now go to developer options -> Application->Manifest

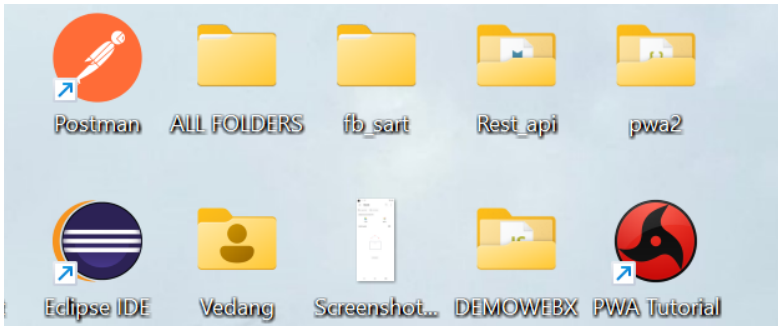


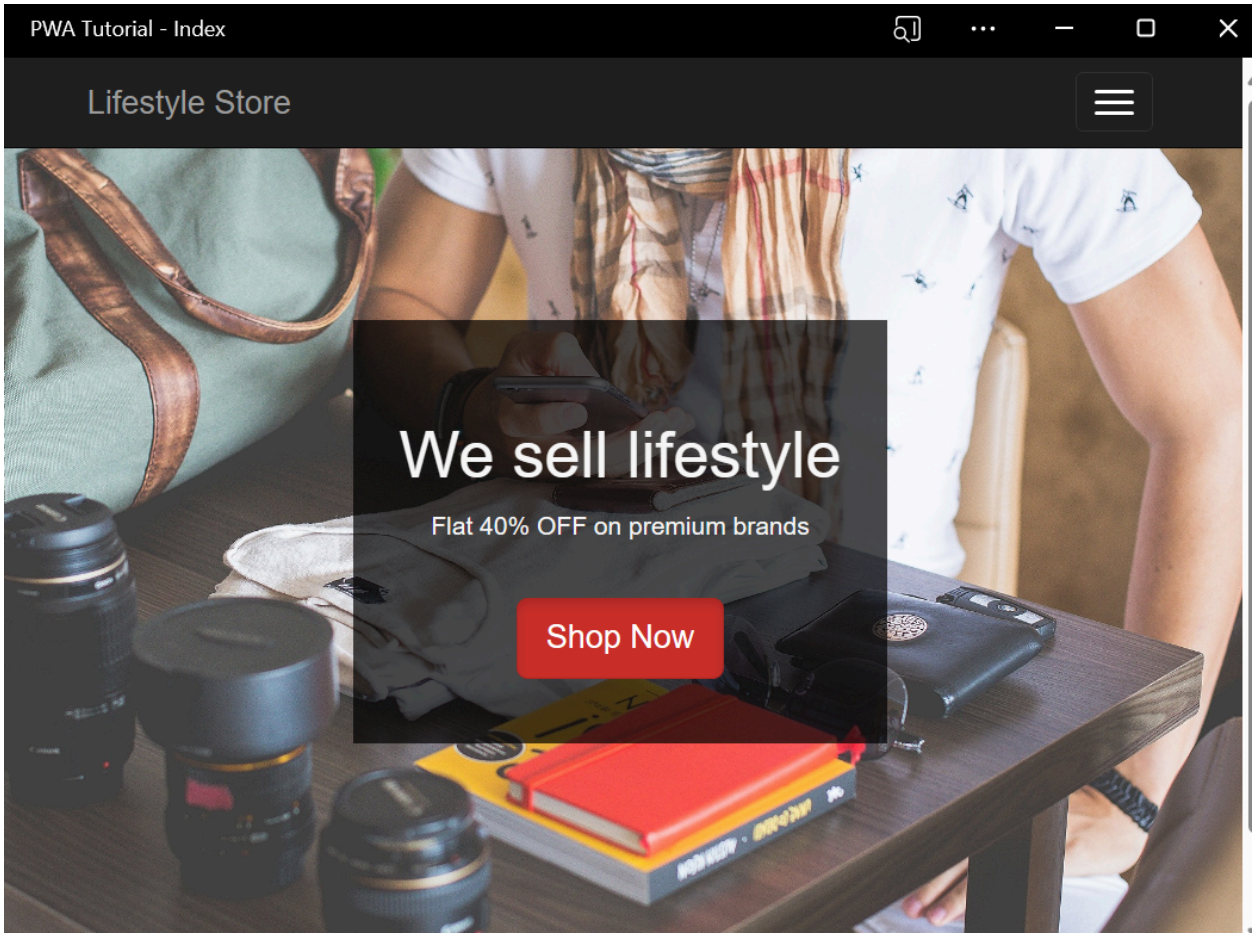
Now to install PWA , click on Three dots(...) -> Apps -> Install PWA



Now On the Desktop

Project Title: MAD/Whatsapp Clone/Pwa/Lifestyle Store **Roll no:44**





Conclusion: Hence We wrote meta data of our Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”. And it is currently added Successfully on the Desktop

MAD & PWA Lab
Journal

Experiment No.	08
Experiment Title.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA
Roll No.	44
Name	Chetan .k.Pilane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

Experiment No 8

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

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.

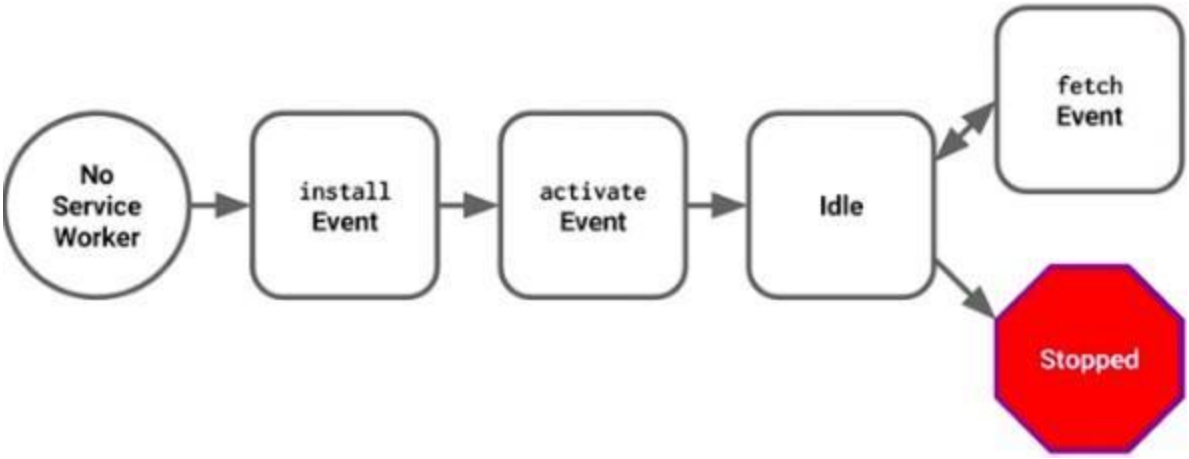
- You can manage **Push Notifications**
  You can manage push notifications with Service Worker and show any information message to the user.
- You can **Continue**
  Although Internet connection is broken, you can start any process with Background Sync of Service Worker.

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('/ser
  vice-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 in service-worker.js

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

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

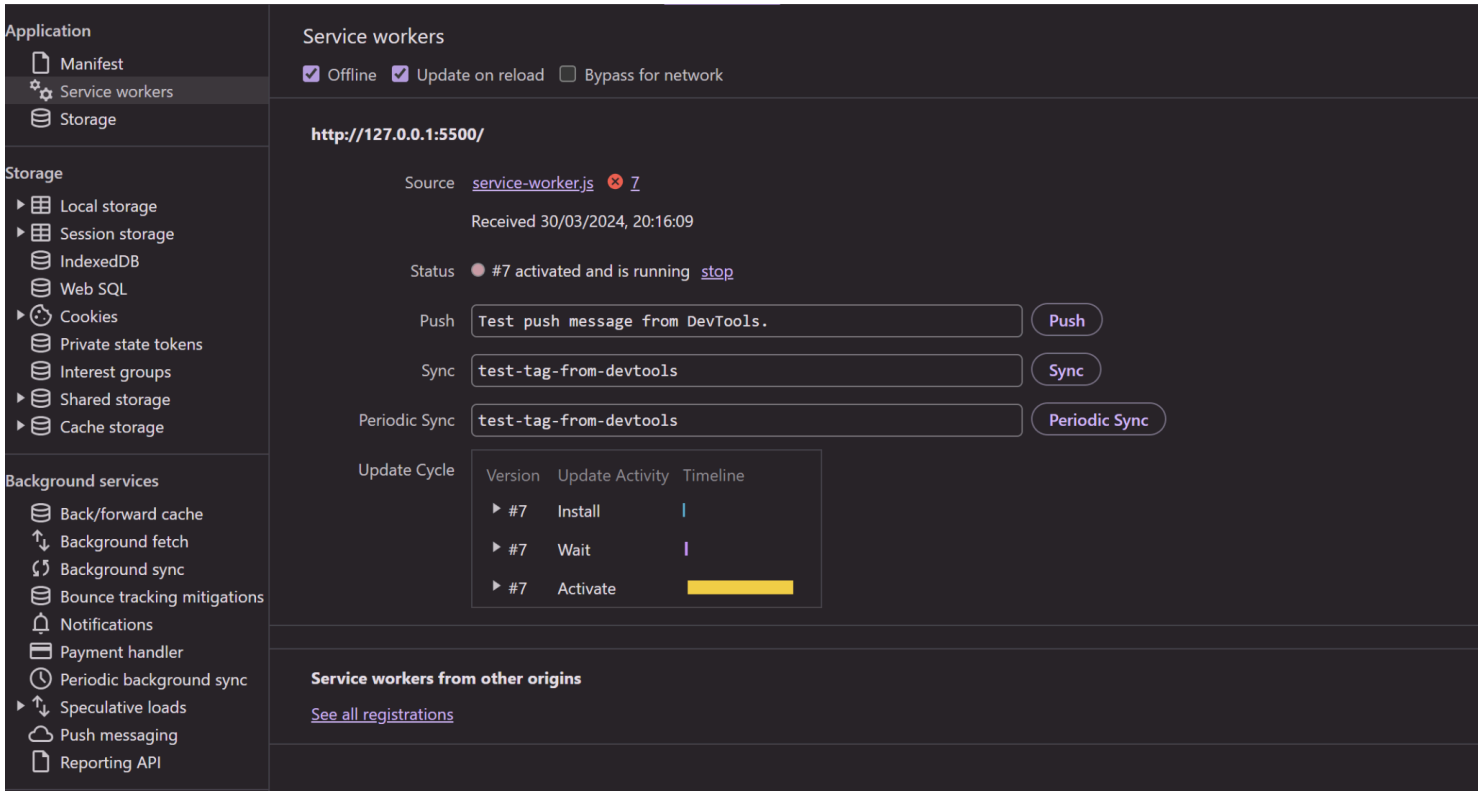
Index.html

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

Project Title: MAD/Whatsapp Clone/Pwa/Lifestyle Store **Roll no:44**

```
        // 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);
                }
            }
        }
    </script>
```

Output:



Conclusion: Hence,Service worker is successfully registered in progressive web app and is actively running.

MAD & PWA Lab Journal

Experiment No.	09
Experiment Title.	To implement Service worker events like fetch, sync and push for E-commerce PWA
Roll No.	44
Name	Chetan.k.Pilane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

Experiment No 9

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

Theory:

Service Worker

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

Things to note about Service Worker:

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

Fetch Event

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

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

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

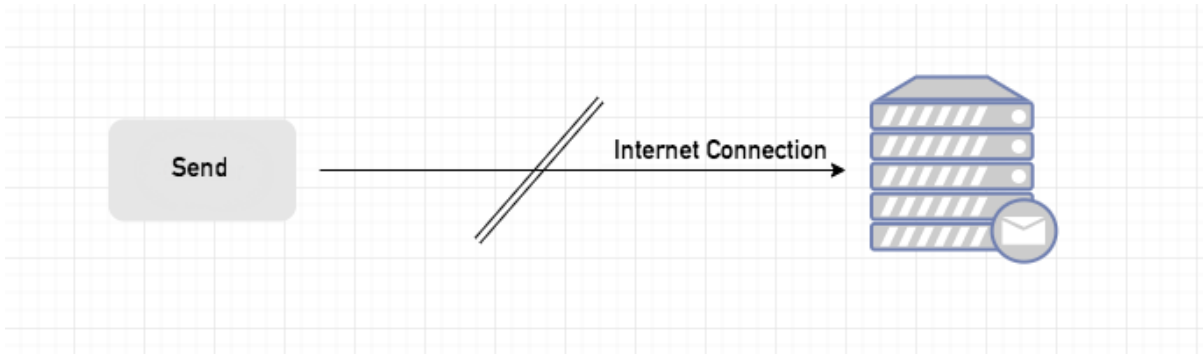
```
self.addEventListener("fetch", function (event) {
    const req = event.request;
    const url = new URL(req.url);

    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

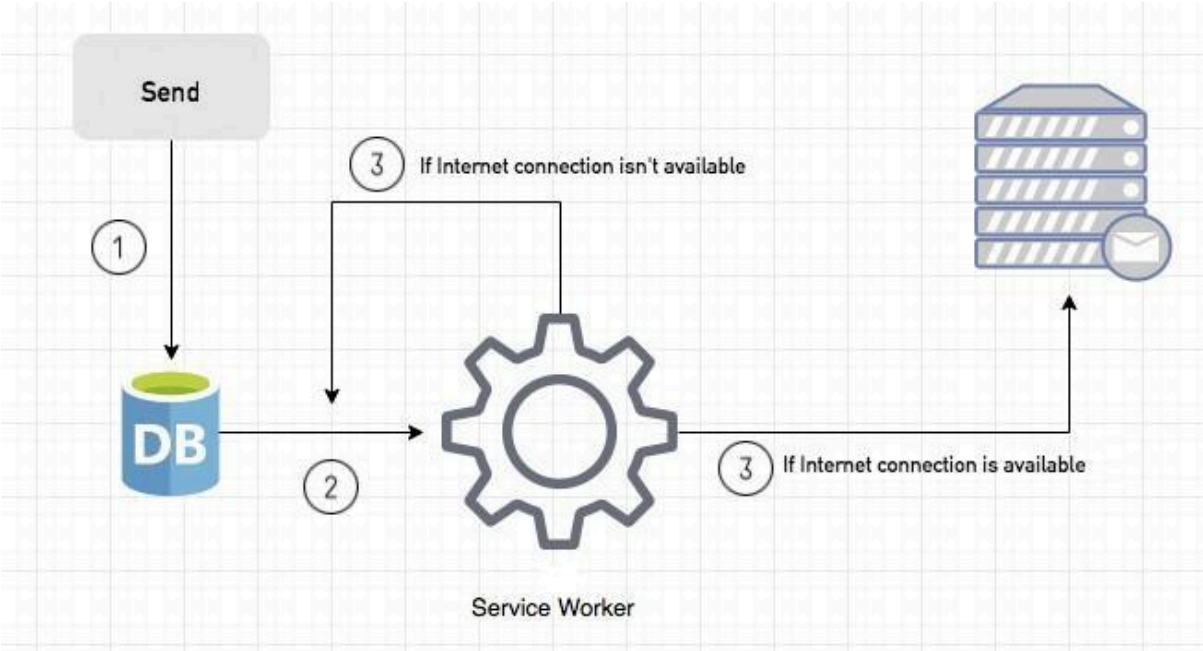
Background Sync is a Web API that is used to delay a process until the Internet connection is stable. We can adapt this definition to the real world; there is an e-mail client application that works on the browser and we want to send an email with this tool. Internet connection is broken while we are 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
            }));
        }
    }
});
```

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:

In index.html

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

This code sends notification permission to your Device , and click on allow to send push notification

service-worker.js

```
// service-worker.js

const CACHE_NAME = 'my-ecommerce-app-cache-v1';
const urlsToCache = [
  '/',
  'cart.html',
  'index.html',
  'product.html',
  'shop.html',
  'style.css',
  'success.html',
  'service-worker.js',
  'manifest.json',
  'offline.html'
  // Add more files to cache as needed
];

self.addEventListener('install', function(event) {
  event.waitUntil(
    caches.open(CACHE_NAME)
```

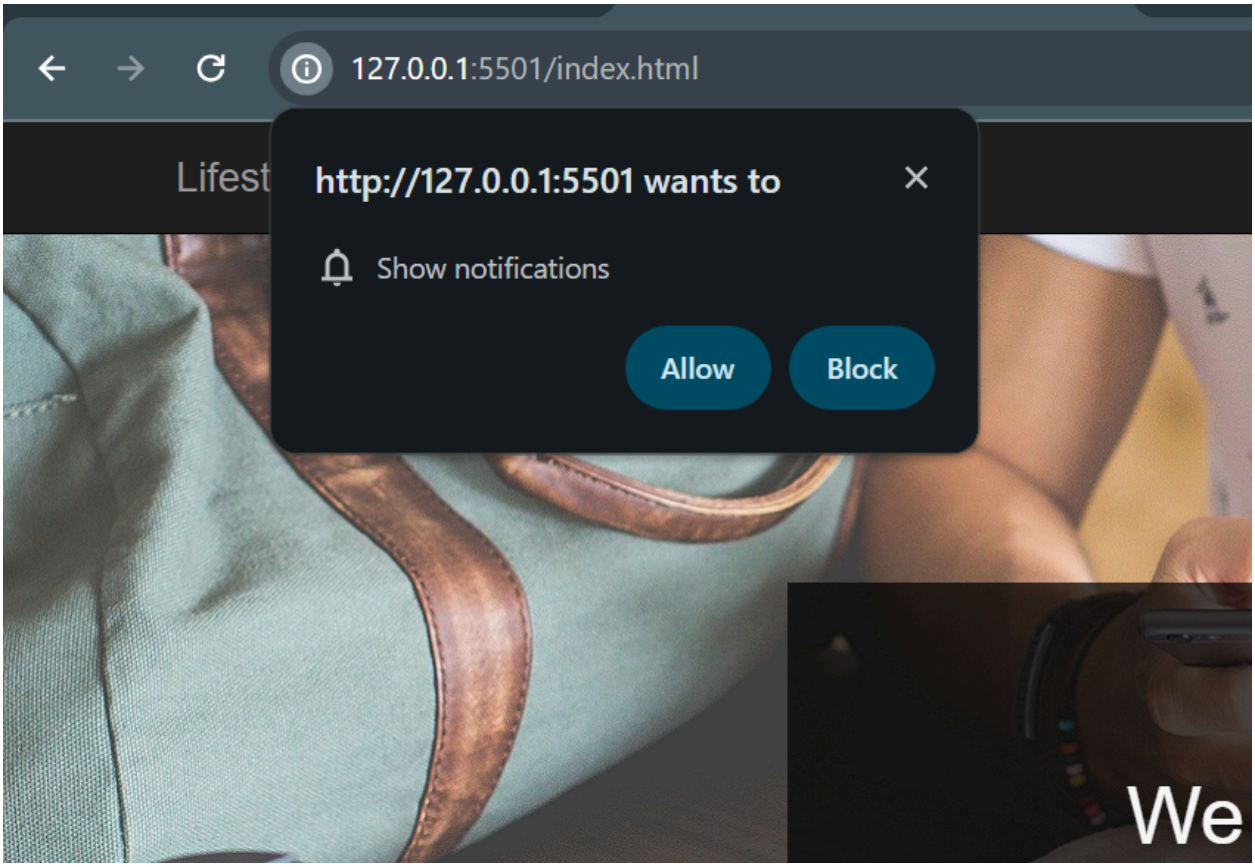
```
      .then(function(cache) {
        console.log('Opened cache');
        return cache.addAll(urlsToCache)
          .catch(function(error) {
            console.error('Cache.addAll error:', error);
          });
      })
  );
});

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

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

```
});

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

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

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

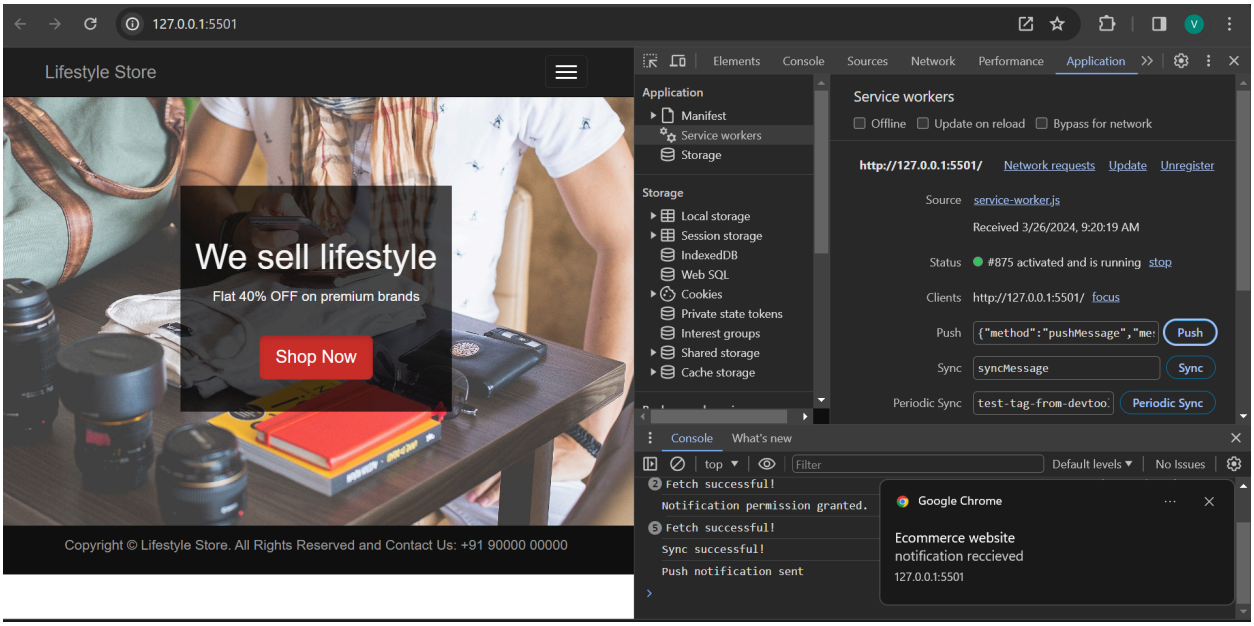
```
    } catch (error) {
      console.error('Failed to request notification permission:', error);
    }
  }
});

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

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

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

Output:



Project Title: MAD/Whatsapp Clone/Pwa/Lifestyle Store **Roll no:44**



Conclusion: Hence we implemented methods like fetch,sync,and push on the service worker , and if we push the message, it says “notification recieved” on the desktop. So the push, sync , and fetch method is implemented successfully

MAD & PWA Lab Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	44
Name	Chetan.k.Pilane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

Experiment No 10

Aim:

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

Theory:

GitHub Pages

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

GitHub Pages provides the following key features:

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

Reasons for favoring this over Firebase:

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

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

Pros

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

Cons

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

Firebase

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

Some of the features offered by Firebase are:

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

Reasons for favoring over GitHub Pages:

1. Realtime backend made easy
2. Fast and responsive

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

Pros

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

Cons

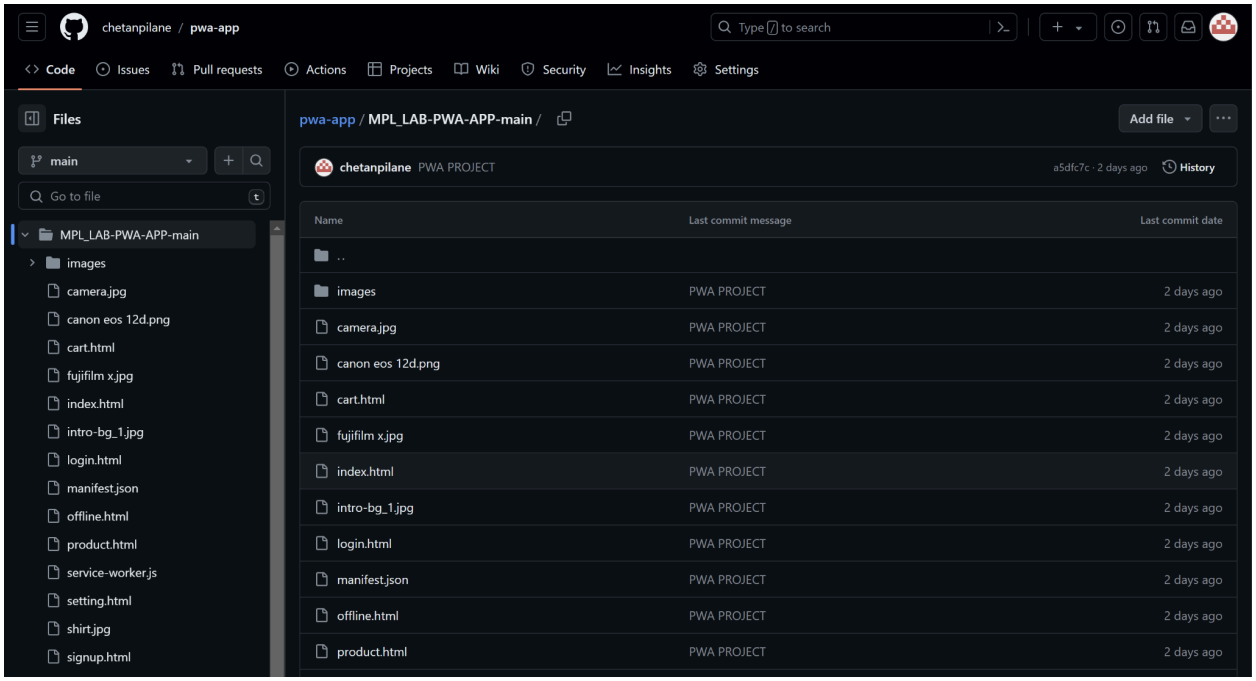
1. Only 10 GB of data transfer is allowed per month. But this is not really a big problem, if you use a CDN or AMP.
2. Command-line interface only.
3. No in-built support for any static site generator.

Project Title: MAD/Whatsapp Clone/Pwa/Lifestyle Store **Roll no:44**

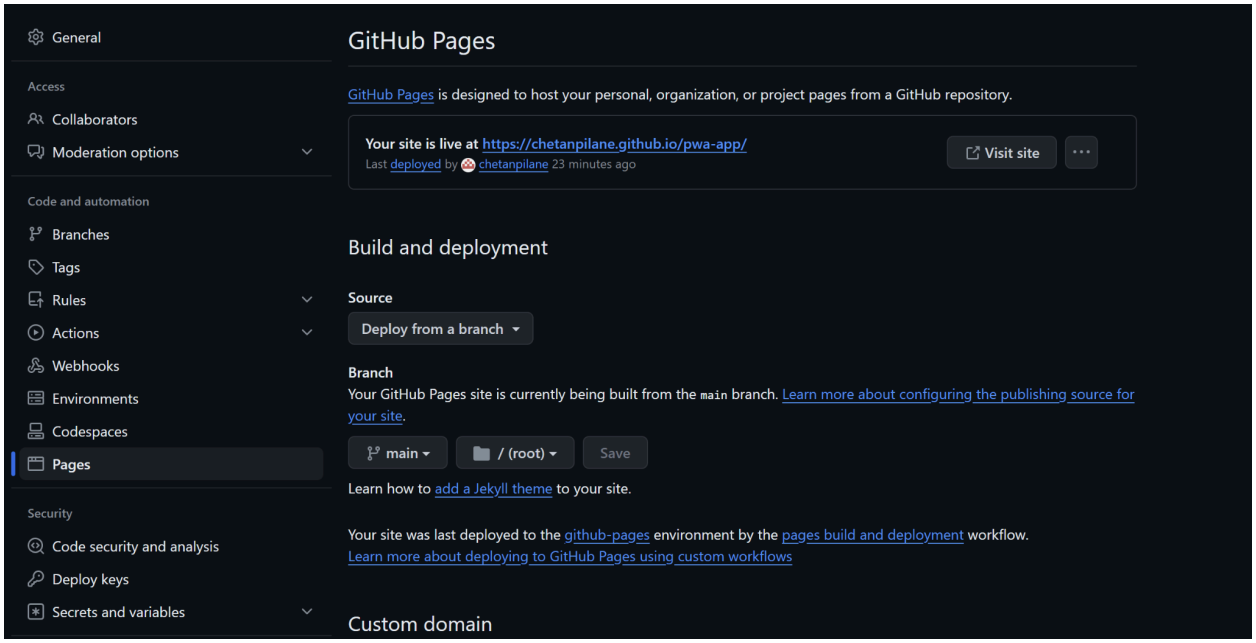
Link for GitHub:https://github.com/chetanpilane/pwa-app/tree/main/MPL_LAB-PWA-APP-main

GitHub ScreenShots

Step1: Make your GitHub Repository public and push your PWA into the repository

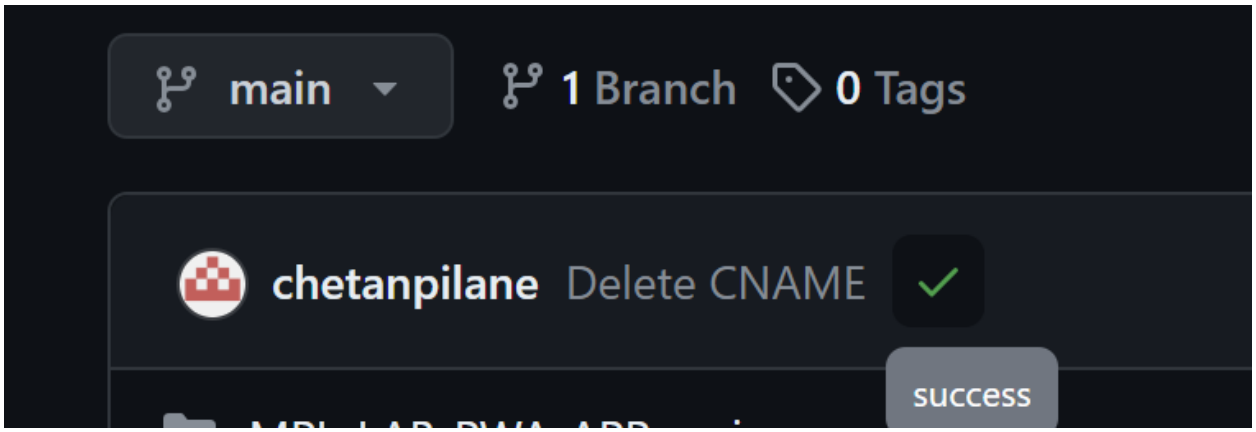


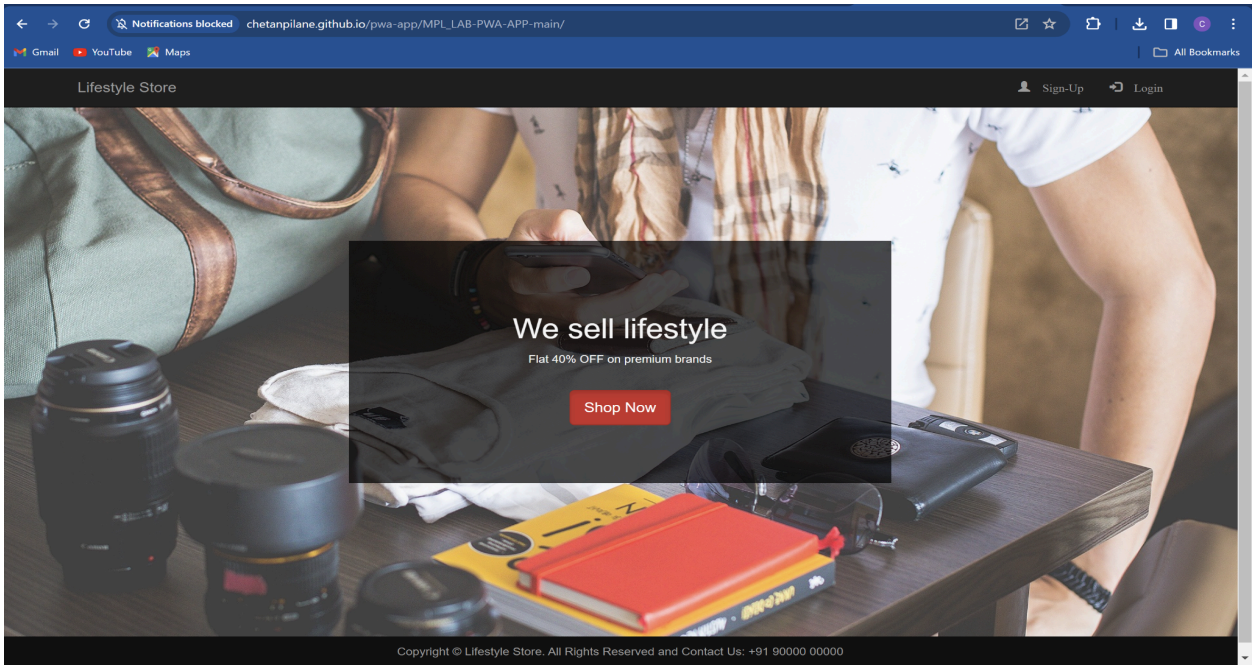
Step 2: Go to settings -> pages and choose your root directory and save it



Step 3: Now go to your Code and you will see a small circle near your recent commit(Mine is finished deploying so i am getting a tick-mark sign)

On clicking Logs of all the deployment is shown for convenience





Conclusion: Hence we deployed our E-Commerce Progressive Web App Successfully on GitHub Pages

MAD & PWA LabJournal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	44
Name	Chetan.k.Pilane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	15

Experiment No 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.

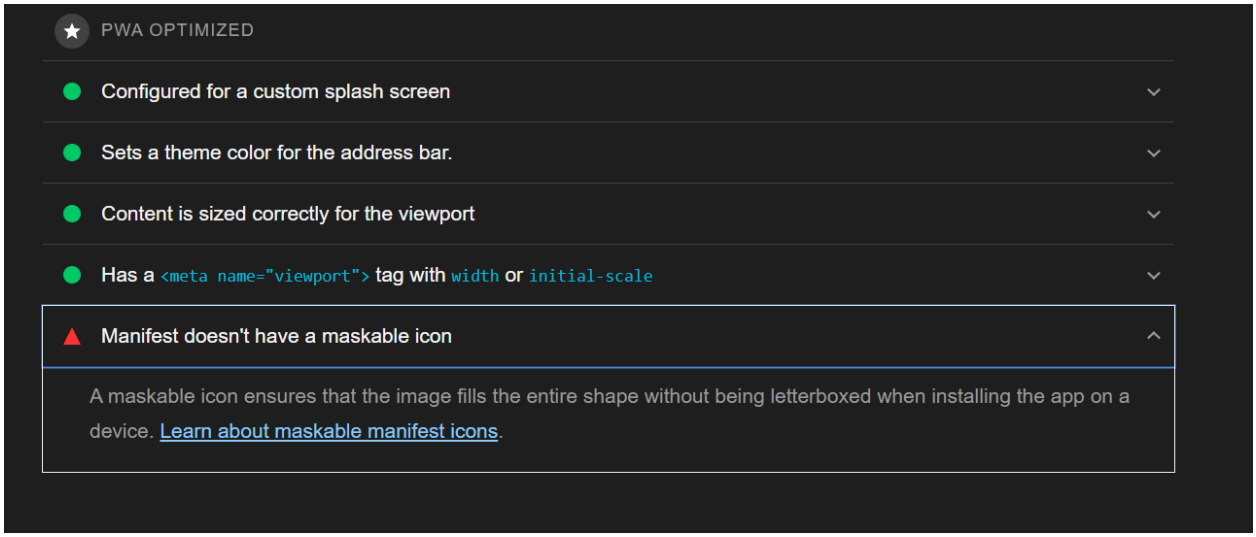
2. **PWA Score (Mobile):** Thanks to the rise of Service Workers, app manifests, etc., a lot of modern web applications are moving towards the PWA paradigm, where the objective is to make the application behave as close as possible to native mobile applications. Scoring points are based on the Baseline 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.

Output:

Before

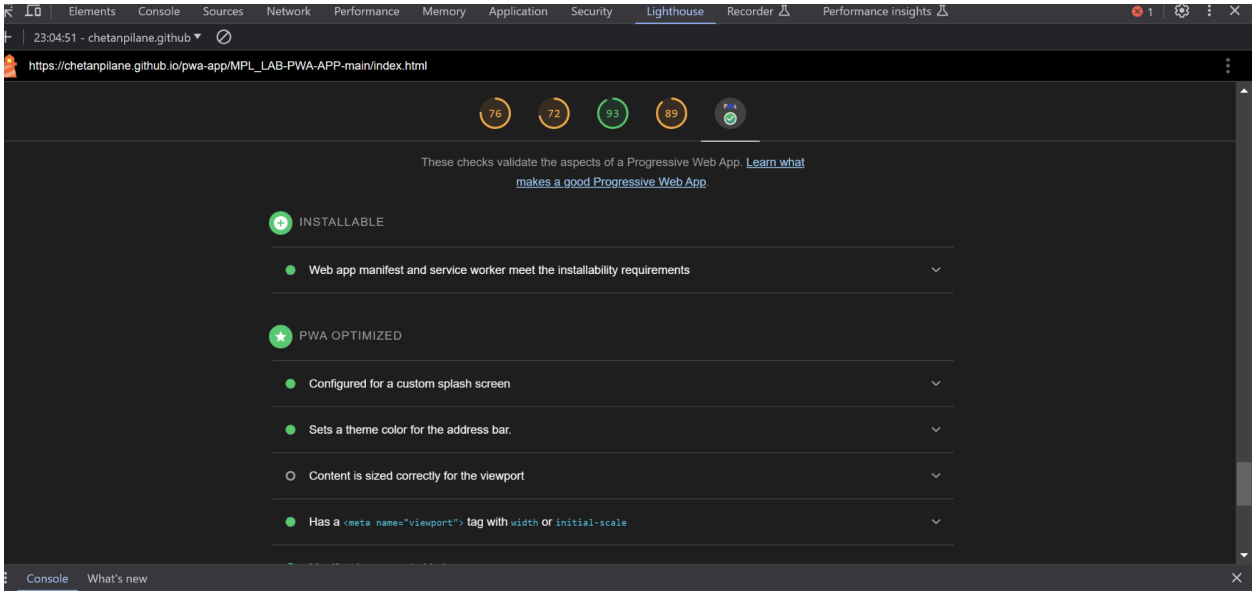


We encountered an issue here , it says “Manifest does not have a maskable icon”
Changes made to the code:

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

```
    "purpose":"any maskable"
    }
]
}
```

After:



Conclusion: Hence by making some changes to the code , we did google lighthouse analysis and our PWA is Fully Optimized and ready to go

MAD & PWA Lab
Journal

Experiment No.	Assignment-1
Assignment 1 Questions	1. Flutter Overview:Explain the key features and advantages of using Flutter for mobile app development. Discuss how the Flutter framework differs from traditional approaches and why it has gained popularity in the developer community. 2. Widget Tree and Composition: Describe the concept of the widget tree in Flutter. Explain how widget composition is used to build complex user interfaces. Provide examples of commonly used widgets and their roles in creating a widget tree. 3. State Management in Flutter: Discuss the importance of state management in Flutter applications. Compare and contrast the different state management approaches available in Flutter, such as setState, Provider, and Riverpod. Provide scenarios where each approach is suitable. 4. Firebase Integration in Flutter: Explain the process of integrating Firebase with a Flutter application. Discuss the benefits of using Firebase as a backend solution. Highlight the Firebase services commonly used in Flutter development and provide a brief overview of how data synchronization is achieved.
Roll No.	44
Name	Chetan.k.Pilane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	4

Chetan Pilare D15A RN 0:44

MAD LAB assignment - 1

Q1) flutter overview

Ans (1) flutter is an open source UI software development toolkit created by google for building natively compiled application for mobile, web and desktop from single codebase.

(2) Characteristics of flutter

1] Single codebase:
flutter offers a single codebase that can be used to create applications through multiple platforms.

2] Hot reload:
Developers can see the changes made in the code instantly reflected in the app.

3] High performance:
flutter uses SKIA graphics engine to render visuals contributing to smooth experience.

4] Access to naive features:
flutter provides seamless access to naive features and API's

Q2) Widget tree composition

Ans ① Widget tree is a hierarchical structure of UI elements represented by widgets.

② Widgets are the building blocks of flutter application and the widget tree organizes them in a parent child relationship widget

③ commonly widgets include
1) container
2) Stack
3) List View
4) Text
5) Image

④ Widget column involves combining simple widgets to create more complex UIs

Q3) State management in flutter

Ans ① State management in flutter is crucial to handle changes in application data and UI

② Proper state management ensures that the UI ~~ensures~~ that stays in sync with underlying data and contributes to a scalable and maintainance codebase

Some of the state management modules include:

1) Setstate : Used for small to medium size applications where state is localized and doesn't need to be shared between multiple widgets.

2) Provider : Well suitable for medium to large size applications where you need to share state across different parts of app.

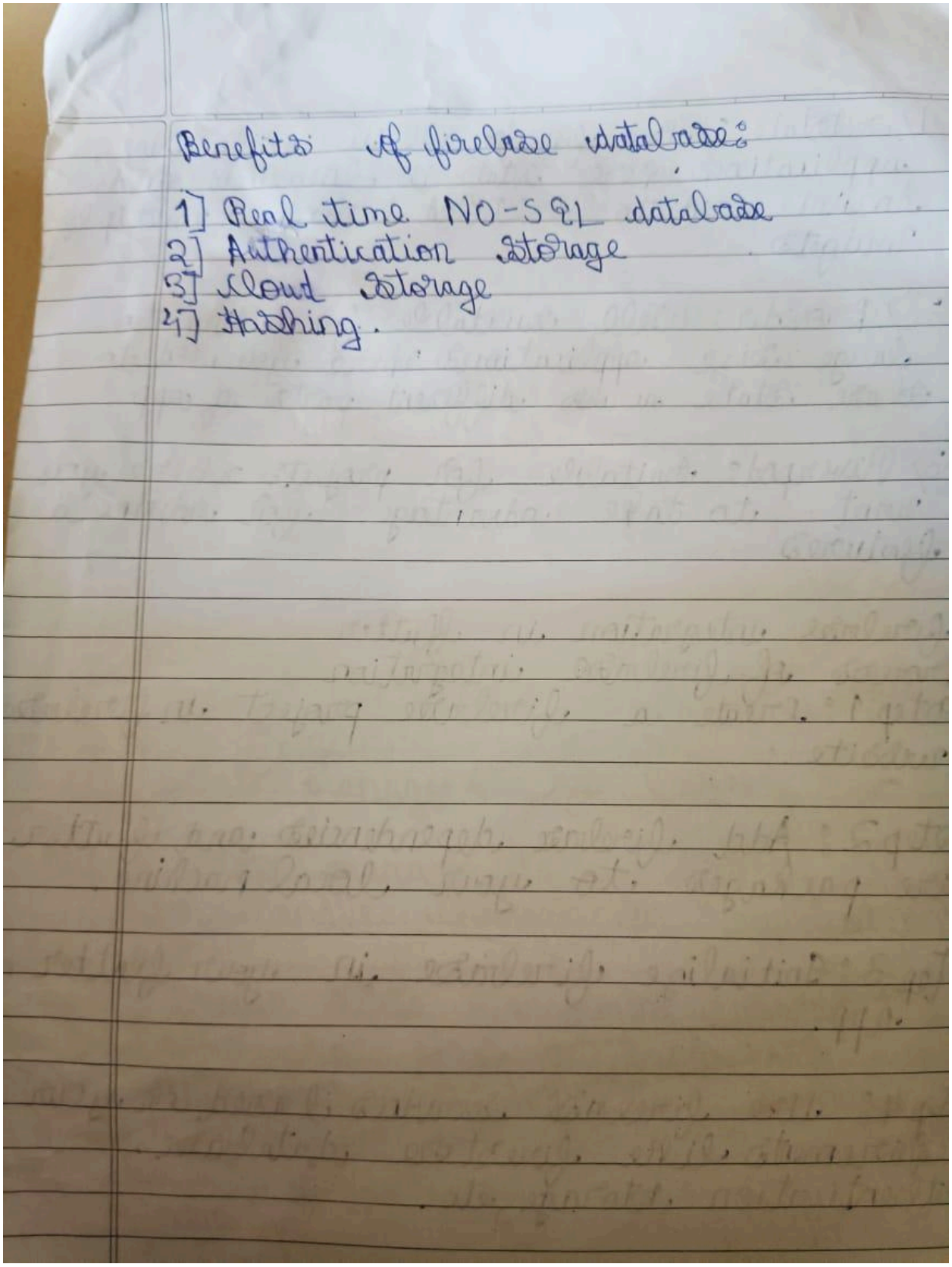
3) Riverpod: Suitable for projects where you want to take advantage for advanced features

4) firebase integration in flutter
Process of firebase integration
Step 1 : Create a firebase project in firebase website.

Step 2 : Add firebase dependencies and flutter fire packages to your local machine.

Step 3 : Initialize firebase in your flutter app.

Step 4 : Use firebase services based on your requirements like firestore database, Authentication storage, etc.



Benefits of firebase database:

1] Real time NO-SQL database
2] Authentication storage
3] Cloud storage
4] Hosting.

MAD & PWA Lab
Journal

Experiment No.	Assignment-2
Assignment 2 Questions	1. Define Progressive Web App (PWA) and explain its significance in modern web development.Discuss the key characteristics that differentiate PWAs from traditional mobile apps 2. Define responsive web design and explain its importance in the context of Progressive Web Apps.Compare and contrast responsive, fluid, and adaptive web design approaches. 3. Describe the lifecycle of Service Workers, including registration, installation, and activation phases. 4. Explain the use of IndexedDB in the Service Worker for data storage.
Roll No.	44
Name	Chetan.k.Pilane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4:Understand various PWA frameworks and their requirements LO5: Design and Develop a responsive User Interface by applying PWA Design techniques LO6:Develop and Analyze PWA Features and deploy it over app hosting solutions
Grade:	5

Chetan Pilane
ROLL NO: 44

PWA assignment 2.

Q1) Define PWA

Ans ① A progressive web app is a type of web application that uses Modern web capabilities to provide a user experience similar to that of traditional web app.

② They are designed to work across different devices and browsers offering an app like experience without the need for installation through app stores.

③ Key characteristics include:
1) Responsiveness
2) Offline functionality
3) App like interactions
4) Installability
5)

Q2) Define Responsive web pages

Ans ① Responsive web design is an approach to web development that aims to make web pages render well on various devices and windows or screen sizes.

② It uses flexible grids and layouts along with media queries to adapt the content and design.

③ Responsive web design is crucial for PWA's as it ensures that the app interface

adapts seamlessly across different devices maintaining a consistent and user friendly experience.

Responsive Web design	Fluid Web design	Adaptive Web design
Uses flexible grids and layouts adapts to any screen size	Emphasizes proportional resizing of elements.	Provides specific layouts for pre-determined break-points.
Adapts to flexible grids and media queries.	Primarily relies on percentage-based widths for fluidity	Requires seperate layouts for different device categories

Q3)

Ans A] Registration: ① Initiation through inclusion of a Javascript file.

② Scope definition to specify controlled pages.

③ Registration using navigator, service worker register().

B] Installation: ① Triggered by a new or updated service worker script.
② Caches essential resources during installation.
③ Enters the waiting phase were worker is busy.

C] Activation: ① Triggered after installation when service worker is fine.
② Cleans up resources from ~~pre~~ previous service worker
③ Take control of clients and become the active service worker.

Q4)
Ans ① Indexed DB is a web API that allows code applications to asynchronously store and retrieve large amount of data.

It is utilized in the following context:

1) offline data storage
2) caching strategies
3) Background synchronization.
4) Improved performance
5) complex data structures.