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 **GOMATI MANIKANDAN IYER** of **D15A** semester **VI,** have successfully completed necessary experiments in the **MAD & PWA Lab** under my supervision in **VES Institute of Technology** during the academic year **2023-2024.**

Lab Assistant

Subject Teacher

Mrs. Kajal Joseph

Principal

Head of Department

Dr. Mrs. Shalu Chopra

Project Title: Flutter : Trivia App / PWA : Cafe Website **Roll No.: 25**

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

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

Faculty Incharge : Mrs. Kajal Joseph.

Lab Teachers : Mrs. Kajal Jewani.

Email : kajal.jewani@ves.ac.in

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

Program specific Outcomes

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

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

Project Title: Flutter : Trivia App / PWA : Cafe Website **Roll No.: 25**

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	15
2.	To design Flutter UI by including common widgets.	LO2	24/1	31/1	15
3.	To include icons, images, fonts in Flutter app	LO2	31/1	07/2	15
4.	To create an interactive Form using form widget	LO2	07/2	14/2	15
5.	To apply navigation, routing and gestures in Flutter App	LO2	14/2	21/2	15
6.	To Connect Flutter UI with fireBase database	LO3	21/2	06/3	15
7.	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.	LO4	06/3	13/3	15
8.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA	LO5	13/3	20/3	15
9.	To implement Service worker events like fetch, sync and push for E-commerce PWA	LO5	20/3	27/3	15
10.	To study and implement deployment of Ecommerce PWA to GitHub Pages.	LO5	27/3	27/3	15
11.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.	LO6	06/3	13/3	15
12.	Assignment-1	LO1,LO2 ,LO3	02/2	05/2	5
13.	Assignment-2	LO4,LO5 ,LO6	19/3	21/3	4

MAD & PWA Lab
Journal

Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	25
Name	Gomati Iyer
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	15

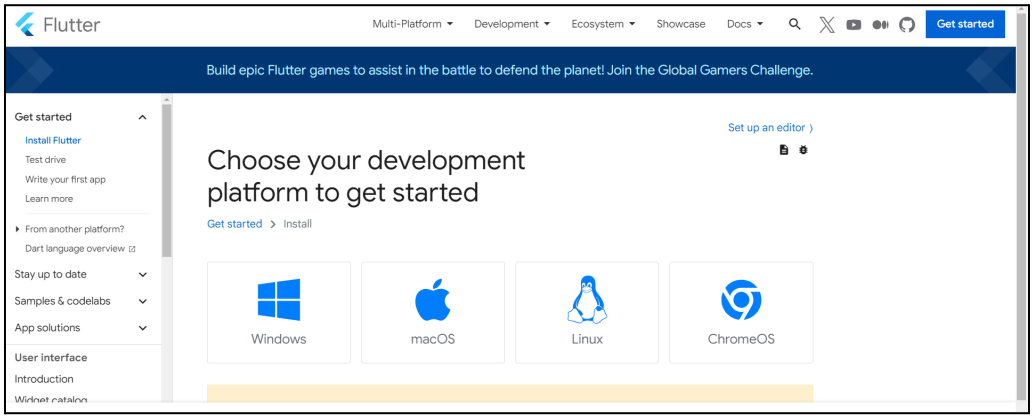
Project Title: Flutter : Trivia App / PWA : Cafe Website **Roll No.: 25**

EXPERIMENT - 01

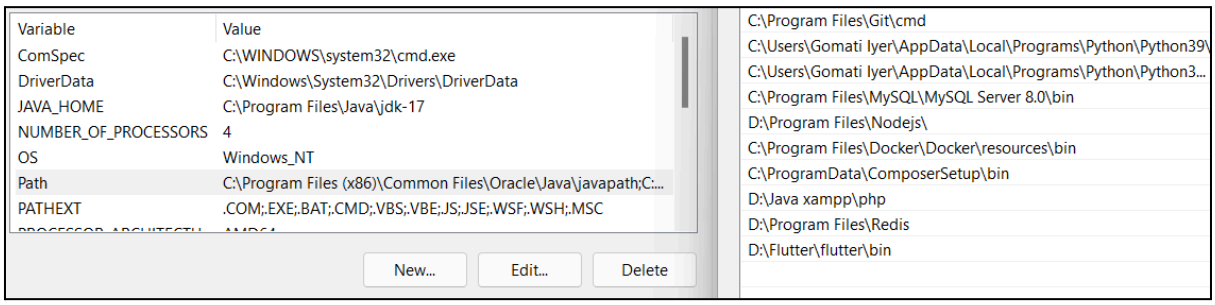
AIM : Installation and Configuration of Flutter Environment.

CODE :

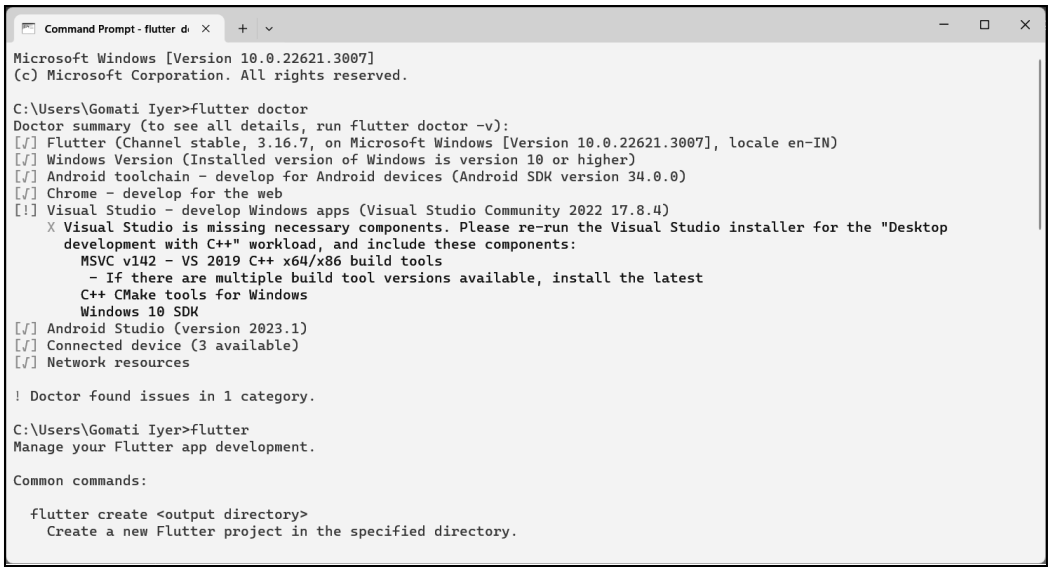
Install Flutter



Set Environment variables



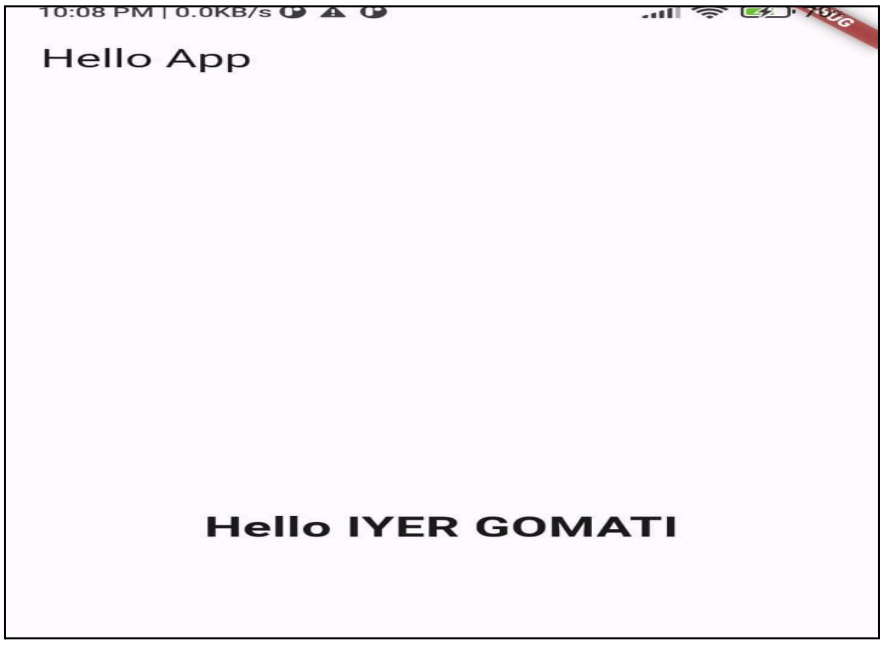
Run flutter and flutter doctor on cmd.



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('Hello App'),
        ),
        body: Center(
          child: Text(
            'Hello IYER GOMATI',
            style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold),
          ),
        ),
      ),
    );
  }
}
```

OUTPUT :



CONCLUSION : Thus we have installed and configured the flutter environment.

MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	25
Name	Gomati Iyer
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

EXPERIMENT - 02

AIM : To design Flutter UI by including common widgets.

THEORY :

To build any application we start with widgets – The building block of flutter applications. Widgets describe what their view should look like given their current configuration and state. It includes a text widget, row widget, column widget, container widget, and many more.
Widgets: Each element on a screen of the Flutter app is a widget. The view of the screen completely depends upon the choice and sequence of the widgets used to build the apps. And the structure of the code of an apps is a tree of widgets.

Row and Column are the two most important and powerful widgets in Flutter. These widgets let you align children horizontally and vertically as per the requirement
There are broadly two types of widgets in the flutter:

1. Stateless Widget
2. Stateful Widget

Description of the widgets used are as follows:

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

Some of the commonly used widgets are :

1. Container:
   - A box model that can contain other widgets, providing customization for size, padding, margin, and decoration.

2. Row:
   - A widget that displays its children in a horizontal array.

3. Column:
   - A widget that displays its children in a vertical array.

4. ListView:
   - A scrollable list of widgets.

5. Stack:
   - A widget that overlays multiple children widgets.

6. AppBar:
   - A material design app bar that typically contains the page title and some action buttons.

7. Scaffold:
   - Implements the basic material design layout structure, providing a top bar (app bar), a bottom bar, and a body.

8. Text:
   - A widget for displaying a short piece of text.

9. Image:
   - A widget for displaying images.

10. Button Widgets (E.g., ElevatedButton, TextButton, OutlinedButton):
    - Widgets for creating interactive buttons with different styles.

11. TextField:
    - A widget for user input of text.

12. Card:
    - A material design card. A card has slightly rounded corners and a shadow.

CODE :

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

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

class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return MaterialApp(
    title: 'Quiz App',
    theme: ThemeData(
      primarySwatch: Colors.blue,
    ),
    home: UserLandingPage(),
  );
 }
}

class UserLandingPage extends
StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('User Landing Page'),
      actions: [
        IconButton(
          icon: Icon(Icons.search),
          onPressed: () {
            // Expand search bar
          },
        ),
      ],
```

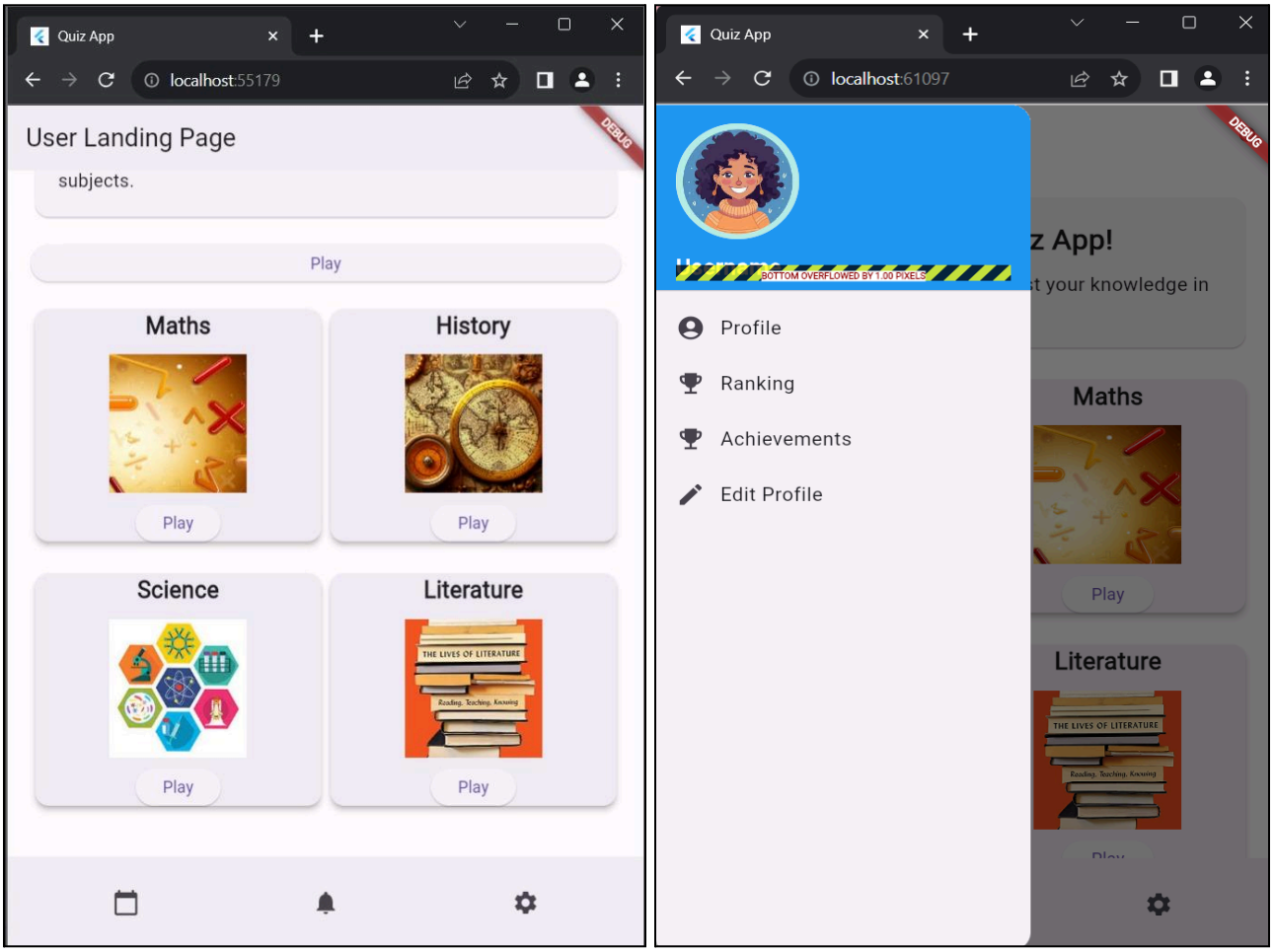
```
    ),
    body: SingleChildScrollView(
      child: Padding(
        padding: EdgeInsets.all(20),
        child: Column(
          crossAxisAlignment:
CrossAxisAlignment.stretch,
          children: [
            // First Card
            Card(
              child: Padding(
                padding: EdgeInsets.all(20),
                child: Column(
                  children: [
                    Text(
                      'Welcome to Quiz App!',
                      style: TextStyle(
                        fontSize: 24,
                        fontWeight: FontWeight.bold,
                      ),
                    ),
                    SizedBox(height: 10),
                    Text(
                      'This app is designed to help you
test your knowledge in various subjects.',
                      style: TextStyle(fontSize: 16),
                    ),
                  ],
                ),
              ),
            ),
            SizedBox(height: 20),
            // Quiz Categories
            Row(
              mainAxisAlignment:
MainAxisAlignment.spaceBetween,
              children: [
                // Science Quiz
                _buildCategoryBox(context,
'Science', 'assets/science.jpg'),
                // Maths Quiz
                _buildCategoryBox(context,
'Maths', 'assets/math.jpg'),
              ],
            ),
            SizedBox(height: 20),
            Row(
              mainAxisAlignment:
MainAxisAlignment.spaceBetween,
              children: [
                // History Quiz
                _buildCategoryBox(context,
'History', 'assets/history.png'),
                // Literature Quiz
                _buildCategoryBox(context,
'Literature', 'assets/literature.jpg'),
              ],
            ),
            SizedBox(height: 20),
          ],
        ),
      ),
    ),
    bottomNavigationBar: BottomAppBar(
      child: Row(
        mainAxisAlignment:
MainAxisAlignment.spaceAround,
        children: [
          IconButton(
            icon: Icon(Icons.calendar_today),
            onPressed: () {
              // Calendar icon pressed
            },
          ),
          IconButton(
            icon: Icon(Icons.notifications),
            onPressed: () {
              // Notifications icon pressed
            },
          ),
          IconButton(
            icon: Icon(Icons.settings),
            onPressed: () {
              // Settings icon pressed
            },
          ),
        ],
      ),
```

```
    ),
    drawer: Drawer(
      child: ListView(
        padding: EdgeInsets.zero,
        children: <Widget>[
          DrawerHeader(
            decoration: BoxDecoration(
              color: Colors.blue,
            ),
            child: Column(
              crossAxisAlignment:
CrossAxisAlignment.start,
              children: [
                Padding(
                  padding: EdgeInsets.only(bottom:
10),
                  child: CircleAvatar(
                    radius: 50,
                    backgroundImage:
AssetImage('assets/profile_image.jpg'),
                  ),
                ),
                Text(
                  'Username',
                  style: TextStyle(
                    color: Colors.white,
                    fontSize: 18,
                    fontWeight: FontWeight.bold,
                  ),
                ),
              ],
            ),
          ),
          ListTile(
            leading: Icon(Icons.account_circle),
            title: Text('Profile'),
            onTap: () {
              // Profile tapped
            },
          ),
          ListTile(
            leading: Icon(Icons.emoji_events),
            title: Text('Ranking'),
            onTap: () {
              // Ranking tapped
            },
          ),
          ListTile(
            leading: Icon(Icons.emoji_events),
            title: Text('Achievements'),
            onTap: () {
              // Achievements tapped
            },
          ),
          ListTile(
            leading: Icon(Icons.edit),
            title: Text('Edit Profile'),
            onTap: () {
              // Edit Profile tapped
            },
          ),
        ],
      ),
    ),
  );
}

 Widget _buildCategoryBox(BuildContext
context, String categoryName, String
imagePath) {
  return Expanded(
    child: GestureDetector(
      onTap: () {
        // Navigate to quiz page for the selected
category (question.dart)
        Navigator.push(
          context,
          MaterialPageRoute(builder: (context)
=> QuizPage(categoryName)),
        );
      },
      child: Card(
        elevation: 4,
        child: Column(
          mainAxisAlignment:
MainAxisAlignment.center,
          children: [
            Text(
```

```
            categoryName,
            style: TextStyle(
              fontSize: 20,
              fontWeight: FontWeight.bold,
            ),
          ),
          SizedBox(height: 10),
          Image.asset(
            imagePath,
            height: 120,
            width: 120,
            fit: BoxFit.cover,
          ),
          SizedBox(height: 10),
          ElevatedButton(
            onPressed: () {
              // Navigate to quiz page for the
selected category (question.dart)
              Navigator.push(
                context,
                MaterialPageRoute(builder:
(context) => QuizPage(categoryName)),
              );
            },
            child: Text('Play'),
          ),
                ],
              ),
            ),
          ),
        );
      }
    }

    class QuizPage extends StatelessWidget {
      final String category;

      QuizPage(this.category);

      @override
      Widget build(BuildContext context) {
        return Scaffold(
          appBar: AppBar(
            title: Text('Quiz - $category'),
          ),
          body: Center(
            child: Text('Quiz Page for $category'),
          ),
        );
      }
    }
```

OUTPUT :

Project Title: Flutter : Trivia App / PWA : Cafe Website **Roll No.: 25**




```
This app is linked to the debug service: ws://127.0.0.1:56071/D_Cd2-tSAKA=/ws
Debug service listening on ws://127.0.0.1:56071/D_Cd2-tSAKA=/ws
Debug service listening on ws://127.0.0.1:56071/D_Cd2-tSAKA=/ws
You pressed Profile Button
You pressed Blocked Profile Button
You pressed Notifications Button
You pressed Help Button
```

CONCLUSION : Thus, we have designed Flutter UI by including common widgets.

MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	25
Name	Gomati Iyer
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

Experiment - 03

AIM : To include icons ,images and fonts in flutter app.

THEORY :

A flutter app when built has both assets (resources) and code. Assets are available and deployed during runtime. The asset is a file that can include static data, configuration files, icons, and images. The Flutter app supports many image formats, such as JPEG, WebP, PNG, GIF, animated WebP/GIF, BMP, and WBMP.

To display an image in Flutter, do the following steps:

Step 1: First, we need to create a new folder inside the root of the Flutter project and named it assets. We can also give it any other name if you want.
Step 2: Next, inside this folder, add one image manually.
Step 3: Update the pubspec.yaml file. Suppose the image name is tablet.png, then pubspec.yaml file is:

assets:
- assets/tablet.png
- assets/background.png

Displaying images from the internet or network is very simple. Flutter provides a built-in method Image.network to work with images from a URL. The Image.network method also allows you to use some optional properties, such as height, width, color, fit, and many more.

```
Image.network(
  'https://picsum.photos/250?image=9',
)
```

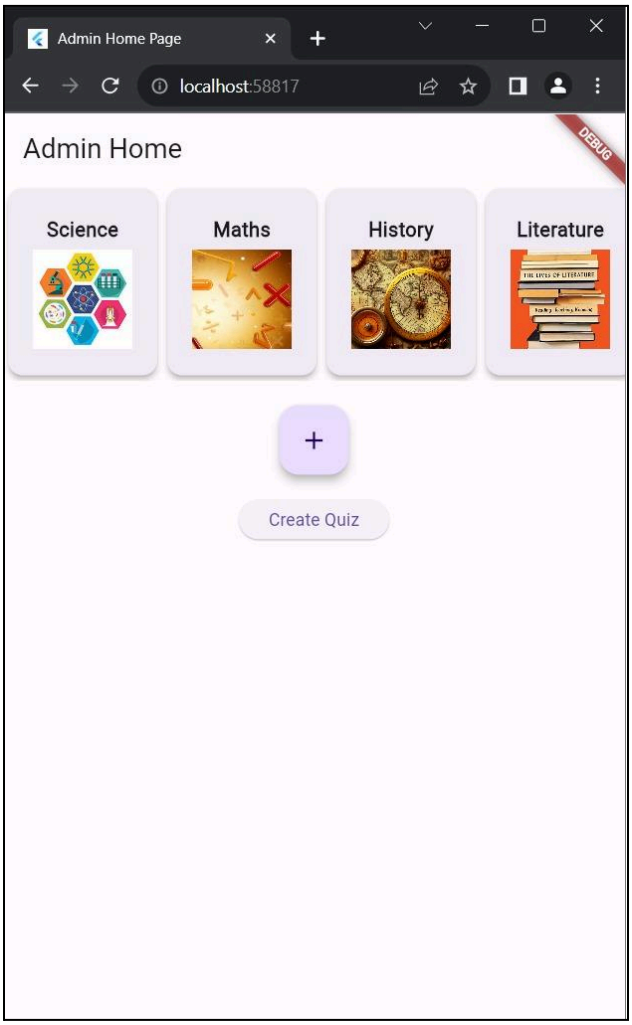
CODE :

```
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: 'Flutter Demo',
    home: Scaffold(
```

```
      body: SafeArea(
        child: Stack(
          children: [
            // Background Image
            Positioned.fill(
              child: Image.asset(
                'assets/background_img.jpg', // Replace with your image path
                fit: BoxFit.cover,
              ),
            ),
            // Signup Buttons
            Positioned(
              left: 0,
              right: 0,
              bottom: 100,
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  ElevatedButton.icon(
                    onPressed: () {
                      // Handle Google signup
                    },
                    icon: Icon(Icons.login),
                    label: Text('Sign up with Google'),
                  ),
                  SizedBox(height: 20),
                  GestureDetector(
                    onTap: () {
                      // Handle email signup
                    },
                    child: Text(
                      'Sign up with Email',
                      style: TextStyle(
                        color: Colors.white,
                        decoration: TextDecoration.underline,
                      ),
                    ),
                  ),
                ],
              ),
            ),
          ],
        ),
      ),
    );
  }
```

Project Title: Flutter : Trivia App / PWA : Cafe Website **Roll No.: 25**

OUTPUT :



CONCLUSION : Thus , we have included icons, images and fonts in our flutter app.

MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	25
Name	Gomati Iyer
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

EXPERIMENT - 04

AIM : To create an interactive Form using a form widget.

THEORY :

Flutter provides a Form widget to create a form. The form widget acts as a container, which allows us to group and validate the multiple form fields. When you create a form, it is necessary to provide the GlobalKey. This key uniquely identifies the form and allows you to do any validation in the form fields.
The form widget uses child widget TextFormField to provide the users to enter the text field. This widget renders a material design text field and also allows us to display validation errors when they occur

Some Properties of Form Widget

- key: A GlobalKey that uniquely identifies the Form. You can use this key to interact with the form, such as validating, resetting, or saving its state.
- child: The child widget that contains the form fields. Typically, this is a Column, ListView, or another widget that allows you to arrange the form fields vertically.
- autovalidateMode: An enum that specifies when the form should automatically validate its fields.

Some Methods of Form Widget

- validate(): This method is used to trigger the validation of all the form fields within the Form. It returns true if all fields are valid, otherwise false. You can use it to check the overall validity of the form before submitting it.
- save(): This method is used to save the current values of all form fields. It invokes the onSaved callback for each field. Typically, this method is called after validation succeeds.
- reset(): Resets the form to its initial state, clearing any user-entered data.
    - currentState: A getter that returns the current FormState associated with the Form.

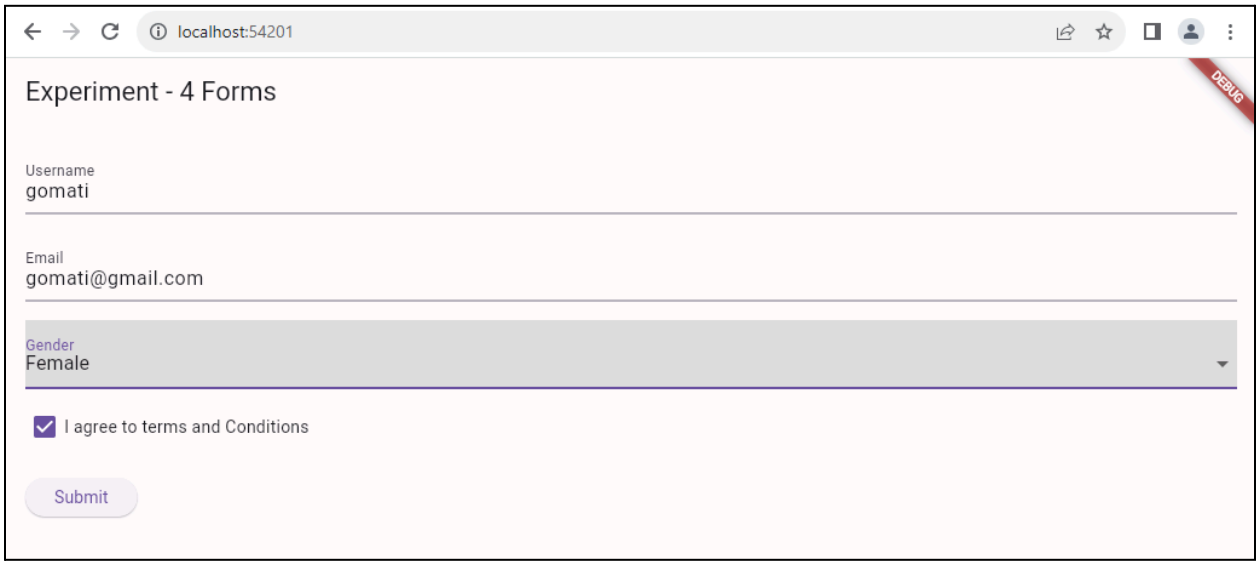
CODE :

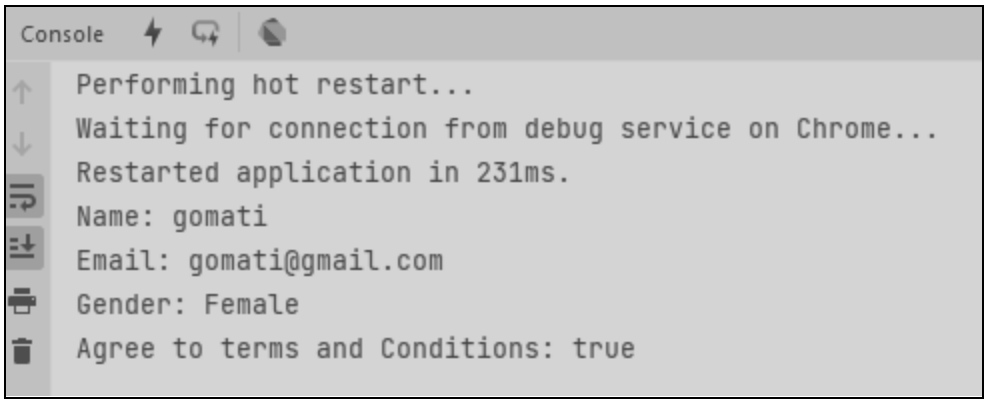
```
import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return MaterialApp(
```

```
    home: Scaffold(
      appBar: AppBar(
        title: Text('Experiment - 4 Forms'),
      ),
      body: MyForm(),
    ),
  );
 }
}


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


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


 @override
 Widget build(BuildContext context) {
  return Padding(
    padding: const EdgeInsets.all(16.0),
    child: Form(
      key: _formKey,
      child: Column(
        crossAxisAlignment:
CrossAxisAlignment.start,
        children: [
          TextFormField(
            decoration: InputDecoration(labelText:
'Username'),
            validator: (value) {
              if (value == null || value.isEmpty) {
                return 'Please enter your name';
              }
              return null;
            },
            onSaved: (value) {
              _name = value!;
            },
          ),
          SizedBox(height: 16),
          TextFormField(
            decoration: InputDecoration(labelText:
'Email'),
            validator: (value) {
              if (value == null || value.isEmpty ||
!value.contains('@')) {
                return 'Please enter a valid email
address';
              }
              return null;
            },
            onSaved: (value) {
              _email = value!;
            },
          ),
          SizedBox(height: 16),
          DropdownButtonFormField<String>(
            value: _selectedGender,
            items: ['Male', 'Female', 'Prefer Not to
say']
                .map((gender) =>
DropdownMenuItem(
              value: gender,
              child: Text(gender),
            ))
                .toList(),
            onChanged: (value) {
              setState(() {
                _selectedGender = value!;
              });
            },
            decoration: InputDecoration(labelText:
'Gender'),
          ),
          SizedBox(height: 16),
          Row(
            children: [
              Checkbox(
```

```
            value: _subscribeToNewsletter,
            onChanged: (value) {
              setState(() {
                _subscribeToNewsletter = value ??
false;
              });
            },
          ),
          Text('I agree to terms and Conditions'),
        ],
      ),
      SizedBox(height: 26),
      ElevatedButton(
        onPressed: () {
          if (_formKey.currentState!.validate()) {
            _formKey.currentState!.save();
                // Process the form data, e.g., send it
to a server
                print('Name: $_name');
                print('Email: $_email');
                print('Gender: $_selectedGender');
                print('Agree to terms and Conditions:
$_subscribeToNewsletter');
                print('');
              }
            },
            child: Text('Submit'),
          ),
        ],
      ),
    ),
  );
 }
}
```

OUTPUT :




Console
Performing hot restart...
Waiting for connection from debug service on Chrome...
Restarted application in 231ms.
Name: gomati
Email: gomati@gmail.com
Gender: Female
Agree to terms and Conditions: true

MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	25
Name	Gomati Iyer
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

EXPERIMENT - 05

AIM : To apply navigation, routing and gestures in Flutter App.

THEORY :

Gestures are an interesting feature in Flutter that allows us to interact with the mobile app (or any touch-based device). Generally, gestures define any physical action or movement of a user in the intention of specific control of the mobile device. Flutter divides the gesture system into two different layers, which are given below:

1. Pointers
2. Gestures

Pointers

Pointers are the first layer that represents the raw data about user interaction. It has events, which describe the location and movement of pointers such as touches, mice, and style across the screens.

1. **PointerDownEvents**: It allows the pointer to contact the screen at a particular location.
2. **PointerMoveEvents**: It allows the pointer to move from one location to another location on the screen.
3. **PointerUpEvents**: It allows the pointer to stop contacting the screen.
4. **PointerCancelEvents**: This event is sent when the pointer interaction is canceled

Gestures

It is the second layer that represents semantic actions such as tap, drag, and scale, which are recognized from multiple individual pointer events.

1. **Tap**: It means touching the surface of the screen from the fingertip for a short time and then releasing them.
2. **Double Tap**: It is similar to a Tap gesture, but you need to tapping twice in a short time.
3. **Drag**: It allows us to touch the surface of the screen with a fingertip and move it from one location to another location and then releasing them.
4. **Long Press**: It means touching the surface of the screen at a particular location for a long time.

Gesture Detector

Flutter provides a widget that gives excellent support for all types of gestures by using the GestureDetector widget. The basic idea of the gesture detector is a stateless widget that contains parameters in its constructor for different touch events.

CODE :

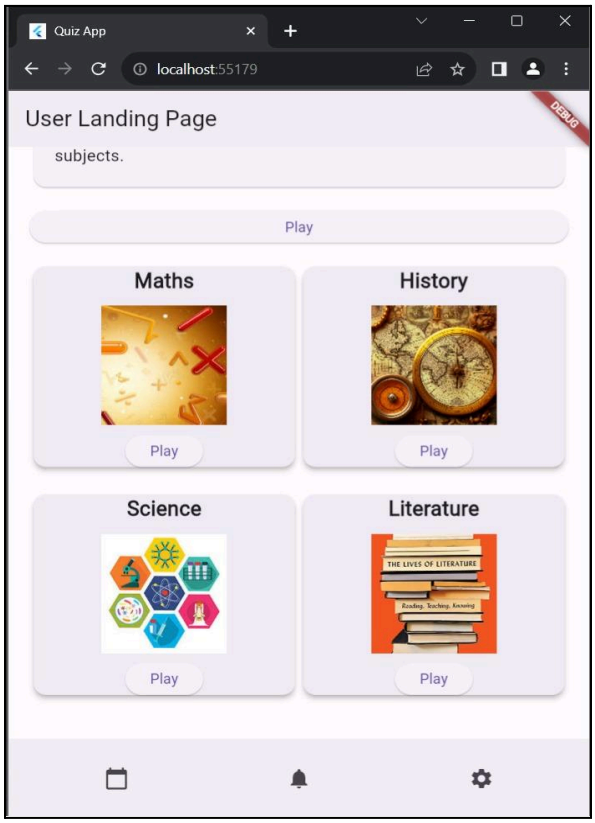
```
import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   title: 'Quiz App',
   theme: ThemeData(
    primarySwatch: Colors.blue,
   ),
   home: UserLandingPage(),
  );
 }
}

class UserLandingPage extends StatelessWidget
{
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
    title: Text('User Landing Page'),
    actions: [
     IconButton(
      icon: Icon(Icons.search),
      onPressed: () {
       // Expand search bar
      },
     ),
    ],
   ),
   body: SingleChildScrollView(
    child: Padding(
     padding: EdgeInsets.all(20),
     child: Column(
      crossAxisAlignment:
CrossAxisAlignment.stretch,
      children: [
       // First Card
       Card(
        child: Padding(
         padding: EdgeInsets.all(20),
         child: Column(
          children: [
           Text(
            'Welcome to Quiz App!',
            style: TextStyle(
             fontSize: 24,
             fontWeight: FontWeight.bold,
            ),
           ),
           SizedBox(height: 10),
           Text(
            'This app is designed to help you
test your knowledge in various subjects.',
            style: TextStyle(fontSize: 16),
           ),
          ],
         ),
        ),
       ),
       SizedBox(height: 20),
       // Quiz Categories
       Row(
        mainAxisAlignment:
MainAxisAlignment.spaceBetween,
        children: [
         // Science Quiz
```

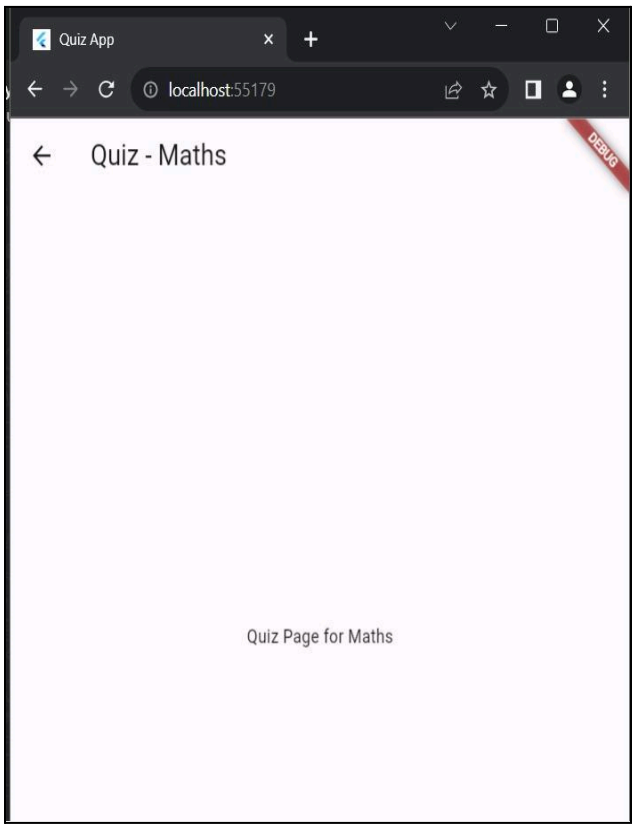
```
            _buildCategoryBox(context, 'Science',
'assets/science.jpg'),
            // Maths Quiz
            _buildCategoryBox(context, 'Maths',
'assets/math.jpg'),
           ],
          ),
          SizedBox(height: 20),
          Row(
           mainAxisAlignment:
MainAxisAlignment.spaceBetween,
           children: [
            // History Quiz
            _buildCategoryBox(context, 'History',
'assets/history.png'),
            // Literature Quiz
            _buildCategoryBox(context,
'Literature', 'assets/literature.jpg'),
           ],
          ),
          SizedBox(height: 20),
         ],
        ),
       ),
      ),
    bottomNavigationBar: BottomAppBar(
     child: Row(
      mainAxisAlignment:
MainAxisAlignment.spaceAround,
      children: [
       IconButton(
        icon: Icon(Icons.calendar_today),
        onPressed: () {
         // Calendar icon pressed
        },
       ),
       IconButton(
        icon: Icon(Icons.notifications),
        onPressed: () {
         // Notifications icon pressed
        },
       ),
       IconButton(
        icon: Icon(Icons.settings),
        onPressed: () {
         // Settings icon pressed
        },
       ),
      ],
     ),
    ),
    drawer: Drawer(
     child: ListView(
      padding: EdgeInsets.zero,
      children: <Widget>[
       DrawerHeader(
        decoration: BoxDecoration(
         color: Colors.blue,
        ),
        child: Column(
         crossAxisAlignment:
CrossAxisAlignment.start,
         children: [
          Padding(
           padding: EdgeInsets.only(bottom:
10),
           child: CircleAvatar(
            radius: 50,
            backgroundImage:
AssetImage('assets/profile_image.jpg'),
           ),
          ),
          Text(
           'Username',
           style: TextStyle(
            color: Colors.white,
            fontSize: 18,
            fontWeight: FontWeight.bold,
           ),
          ),
         ],
        ),
       ),
       ListTile(
        leading: Icon(Icons.account_circle),
        title: Text('Profile'),
        onTap: () {
         // Profile tapped
```

```
          },
        ),
        ListTile(
          leading: Icon(Icons.emoji_events),
          title: Text('Ranking'),
          onTap: () {
            // Ranking tapped
          },
        ),
        ListTile(
          leading: Icon(Icons.emoji_events),
          title: Text('Achievements'),
          onTap: () {
            // Achievements tapped
          },
        ),
        ListTile(
          leading: Icon(Icons.edit),
          title: Text('Edit Profile'),
          onTap: () {
            // Edit Profile tapped
          },
        ),
      ],
    ),
  ),
 );
}

 Widget _buildCategoryBox(BuildContext
context, String categoryName, String
imagePath) {
  return Expanded(
    child: GestureDetector(
      onTap: () {
        // Navigate to quiz page for the selected
category (question.dart)
        Navigator.push(
          context,
          MaterialPageRoute(builder: (context) =>
QuizPage(categoryName)),
        );
      },
      child: Card(
        elevation: 4,
        child: Column(
          mainAxisAlignment:
MainAxisAlignment.center,
          children: [
            Text(
              categoryName,
              style: TextStyle(
                fontSize: 20,
                fontWeight: FontWeight.bold,
              ),
            ),
            SizedBox(height: 10),
            Image.asset(
              imagePath,
              height: 120,
              width: 120,
              fit: BoxFit.cover,
            ),
            SizedBox(height: 10),
            ElevatedButton(
              onPressed: () {
                // Navigate to quiz page for the
selected category (question.dart)
                Navigator.push(
                  context,
                  MaterialPageRoute(builder:
(context) => QuizPage(categoryName)),
                );
              },
              child: Text('Play'),
            ),
          ],
        ),
      ),
    ),
  );
 }
}

class QuizPage extends StatelessWidget {
 final String category;

 QuizPage(this.category);
```

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Quiz - $category'),
    ),
    body: Center(
      child: Text('Quiz Page for $category'),
    ),
  );
}
```

OUTPUT :





CONCLUSION : Thus , we have studied navigation, routing and gestures in flutter.

MAD & PWA Lab
Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	25
Name	Gomati Iyer
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	15

EXPERIMENT - 06

AIM : To connect Flutter UI with Firebase database.

THEORY :

Firebase helps developers to manage their mobile app easily. It is a service provided by Google. Firebase has various functionalities available to help developers manage and grow their mobile apps.

Steps to Add firebase to our Flutter app using Firebase CLI

1. Install the Firebase CLI and log in (run firebase login)
2. From any directory, run this command:
    a. dart pub global activate flutterfire_cli

3. Then, at the root of your Flutter project directory, run this command:
    a. flutterfire configure --project=questitnextjs

4. This automatically registers your per-platform apps with Firebase and adds a lib/firebase_options.dart configuration file to your Flutter project.

5. To initialise Firebase, call Firebase.initializeApp from the firebase_core package with the configuration from your new firebase_options.dart file:

```
import 'package:firebase_core/firebase_core.dart';

import 'firebase_options.dart';
await Firebase.initializeApp(

   options: DefaultFirebaseOptions.currentPlatform,
);
```

6. Add the dependencies in the pubspec.yaml file

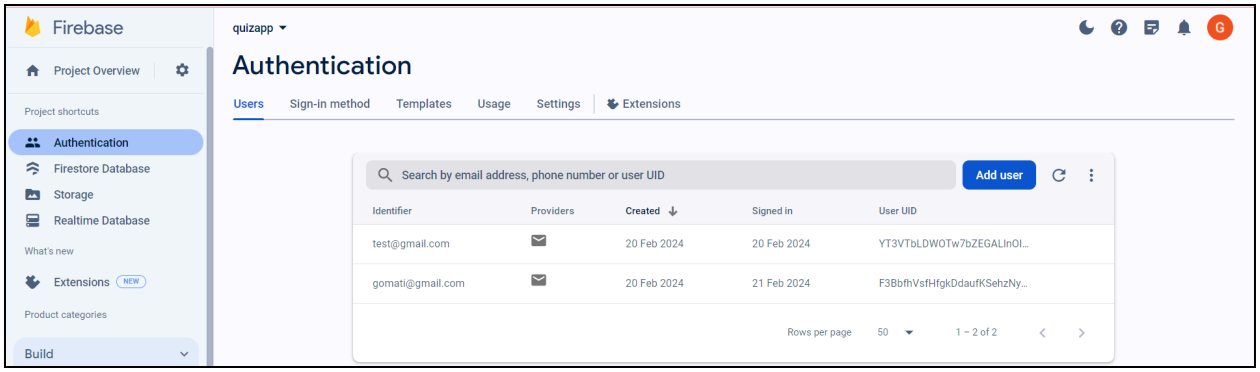
```
Firebase_core : ^version
Firebase_auth : ^version
Cloud_firestore : ^version
```

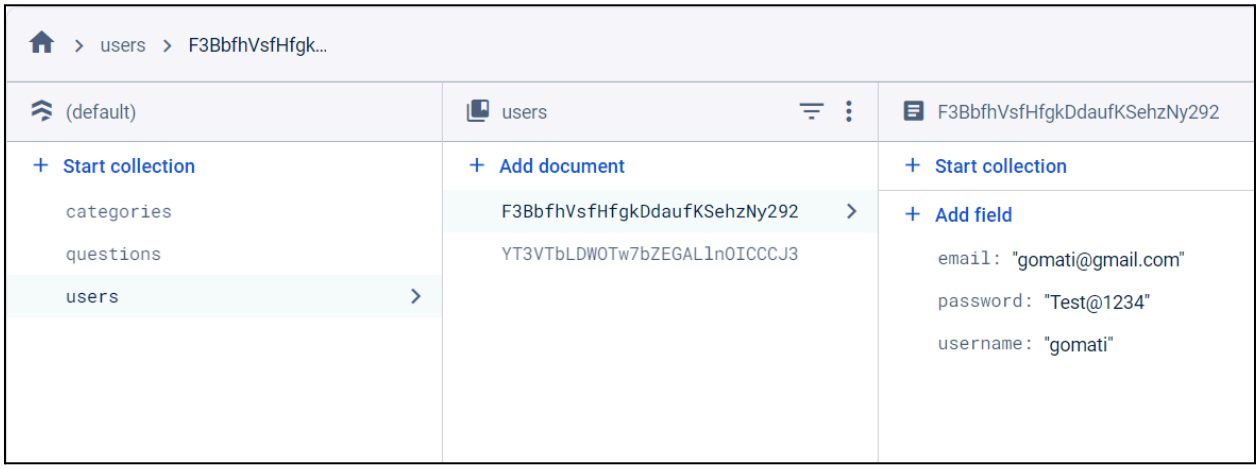
CODE :

```
void main() async {
 // Initialize Firebase
 WidgetsFlutterBinding.ensureInitialized();
 await Firebase.initializeApp(
   options: DefaultFirebaseOptions.currentPlatform,
 );
 runApp(const MyApp());
}

Adding user :
UserCredential userCredential = await
FirebaseAuth.instance.createUserWithEmailAndPassword(
 email: email,
 password: password,
);

// Add user details to Firestore
await FirebaseFirestore.instance.collection('users').doc(userCredential.user!.uid).set({
 'username': username,
 'email': email,
 'password': password,
});

// Navigate to the home page
Navigator.push(
 context,
 MaterialPageRoute(builder: (context) => UserLandingPage()),
);
```

OUTPUT :

Project Title: Flutter : Trivia App / PWA : Cafe Website **Roll No.: 25**





Conclusion : Thus , we have successfully connected firebase to our flutter app.

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.	25
Name	Gomati Iyer
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
2. Ease of Access
3. Faster Services
4. Engaging Approach
5. Updated Real-Time Data Access
6. Discoverable
7. Lower Development Cost

CODE:-

//Manifest.json :

```
{
   "name":"Language Tutorl",
   "short_name":"PWA",
   "start_url":"index.html",
   "display":"standalone",
   "background_color":"#5900b3",
   "theme_color":"black",
   "scope": ".",
   "description":"This is a PWA tutorial.",
   "icons":[
```

```
    {
    "src":"images/icon-192x192.png",
    "sizes":"192x192",
    "type":"image/png"
    },
    {
    "src":"images/icon-512x512.png",
    "sizes":"512x512",
    "type":"image/png"
    }
]
}
```

//**serviceworker.js**

```
var staticCacheName = "pwa";
self.addEventListener("install", function (e) {
e.waitUntil(
    caches.open(staticCacheName).then(function (cache) {
    return cache.addAll(["/"]);
    })
);
});
self.addEventListener("fetch", function (event) {
console.log(event.request.url);
event.respondWith(
    caches.match(event.request).then(function (response) {
    return response || fetch(event.request);
    })
);
});
```

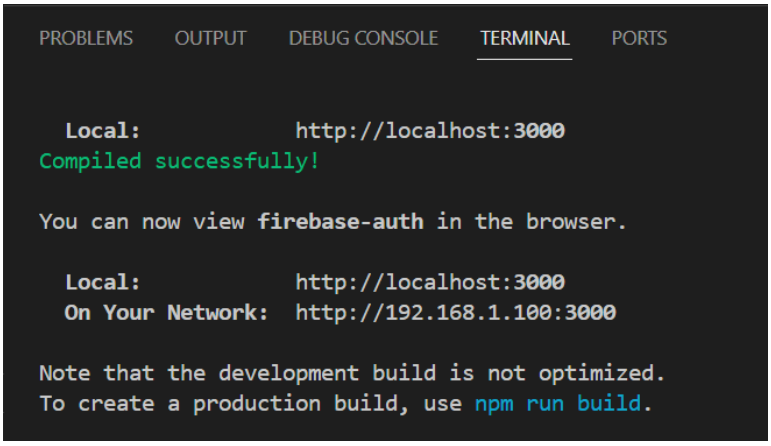
//**Index.html**

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
```

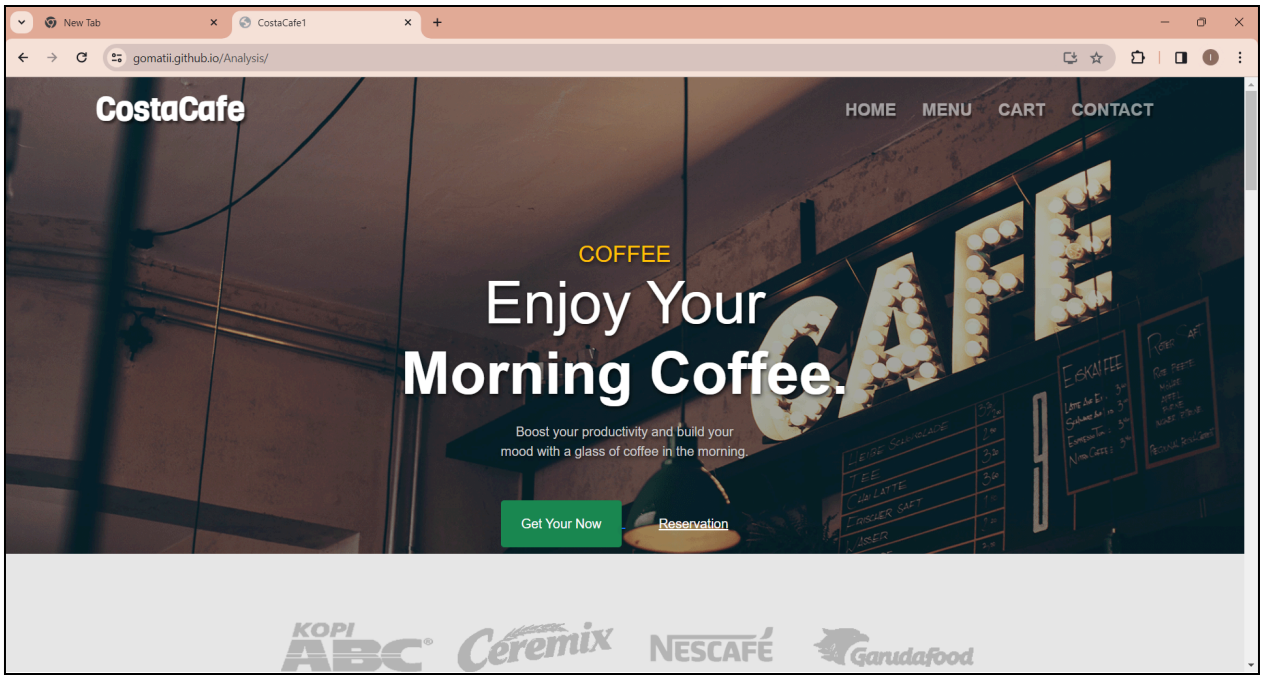
```
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
<meta name="apple-mobile-web-app-status-bar" content="#aa7700">
<meta name="theme-color" content="black">

<!-- Manifest File link -->
<link rel="manifest" href="manifest.json">
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>

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

OUTPUT :

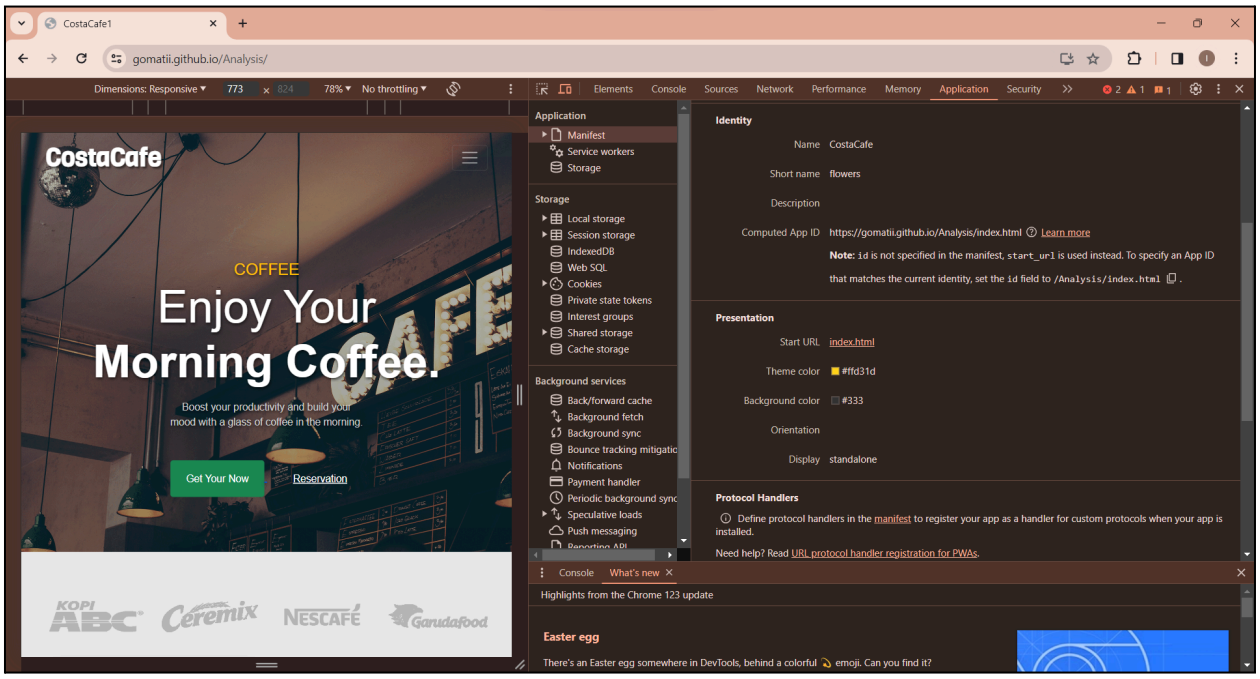
//Open folder in VS code and run the website.



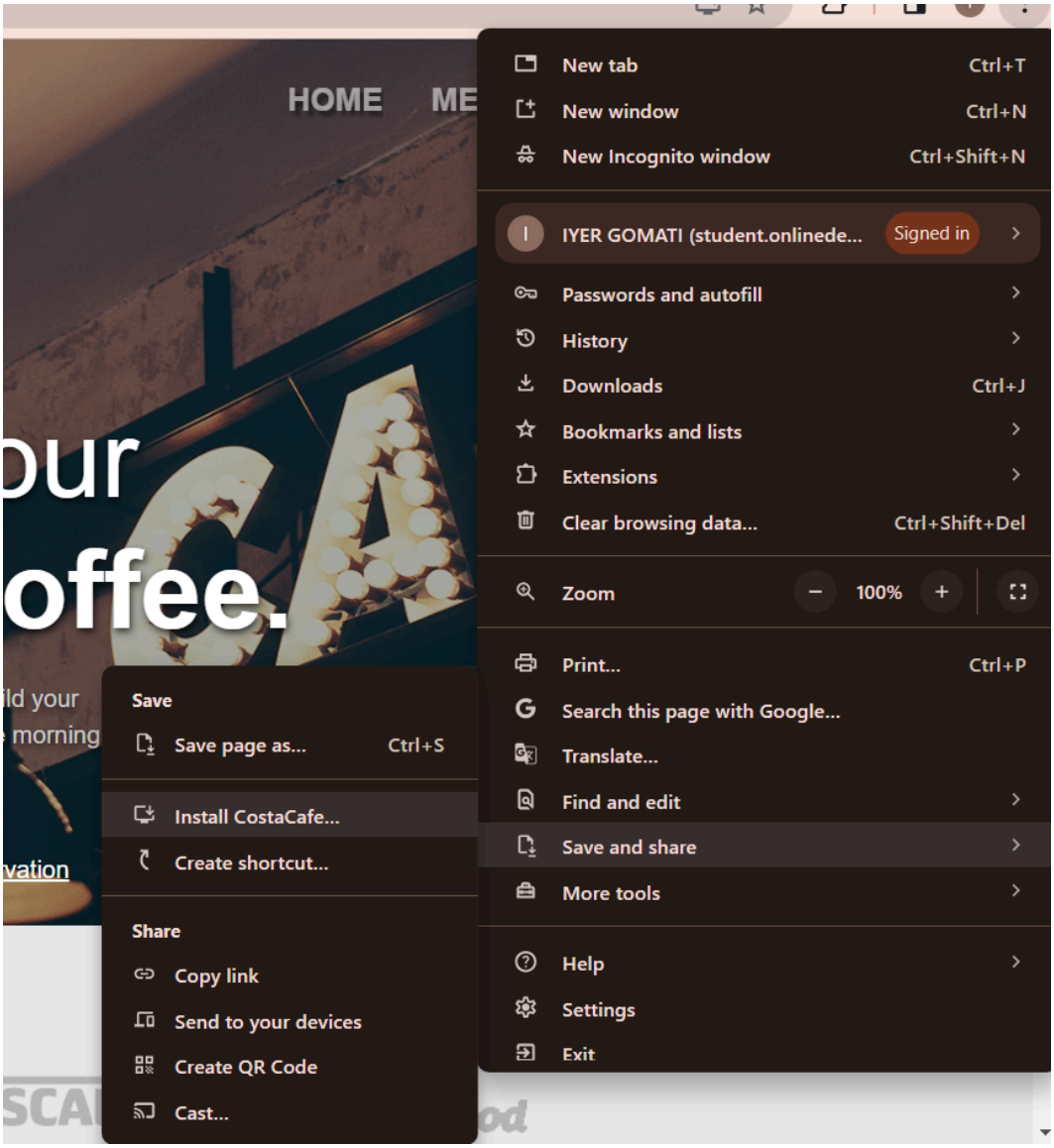
Project Title: Flutter : Trivia App / PWA : Cafe Website **Roll No.: 25**



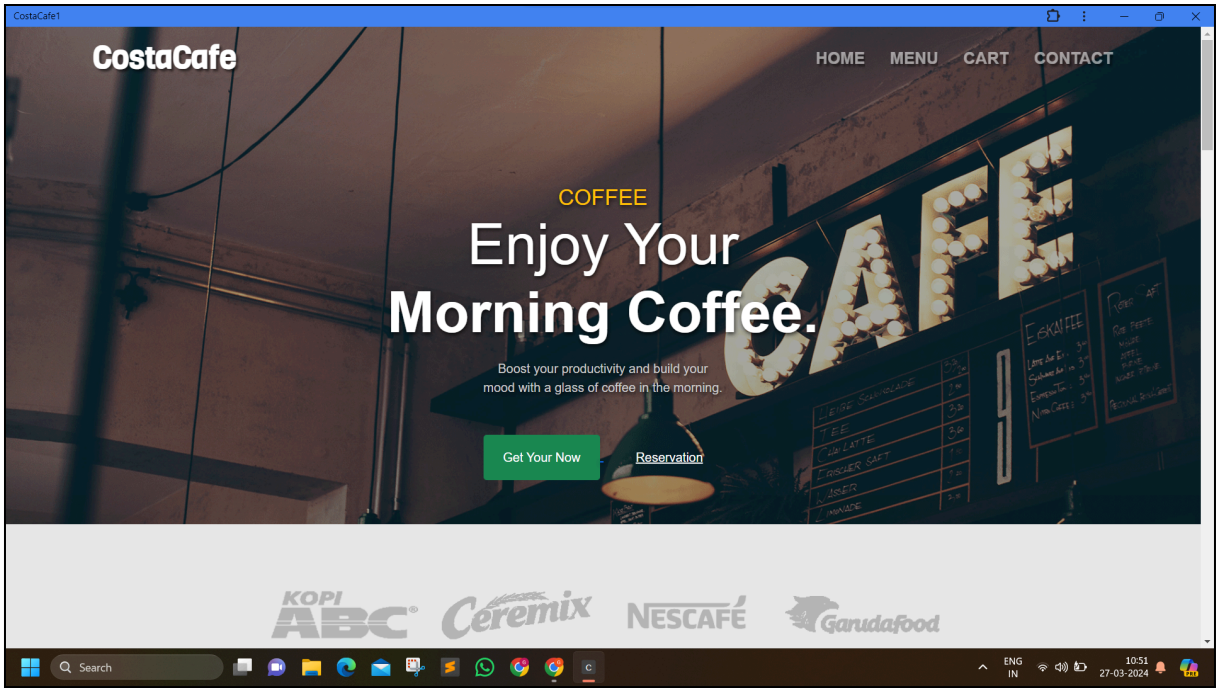
//open Developer tools options -> Applications



//click on 3 dots on top right corner of browser from app option install this site as an app



//App icon will appear at the bottom



Conclusion:-

Hence, we learnt how to write a metadata of our E-commerce website PWA in a Web App Manifest File to enable add to homescreen feature.

MAD & PWA Lab

Journal

Experiment No.	08
Experiment Title.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA
Roll No.	25
Name	Gomati Iyer
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

Experiment - 08

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

Theory:

Service Worker

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

Things to note about Service Worker:

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

What can we do with Service Workers?

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

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

- You can manage **Push Notifications**
  You can manage push notifications with Service Worker and show any information message to the user.
- You can **Continue**

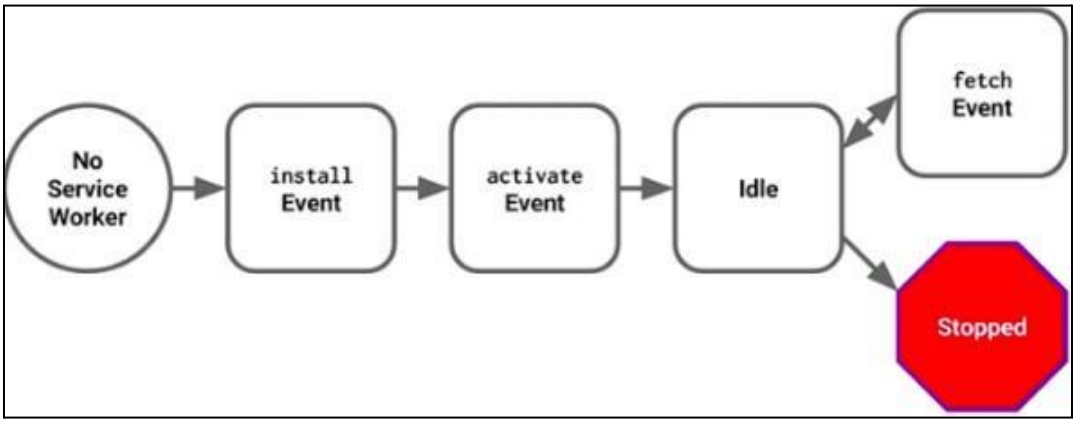
Although Internet connection is broken, you can start any process with Background Sync of Service Worker.

What can't we do with Service Workers?

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

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

Service Worker Cycle



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

- Registration
- Installation
- Activation

Registration

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

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-w
  orker.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.

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).

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).

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

script.js

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

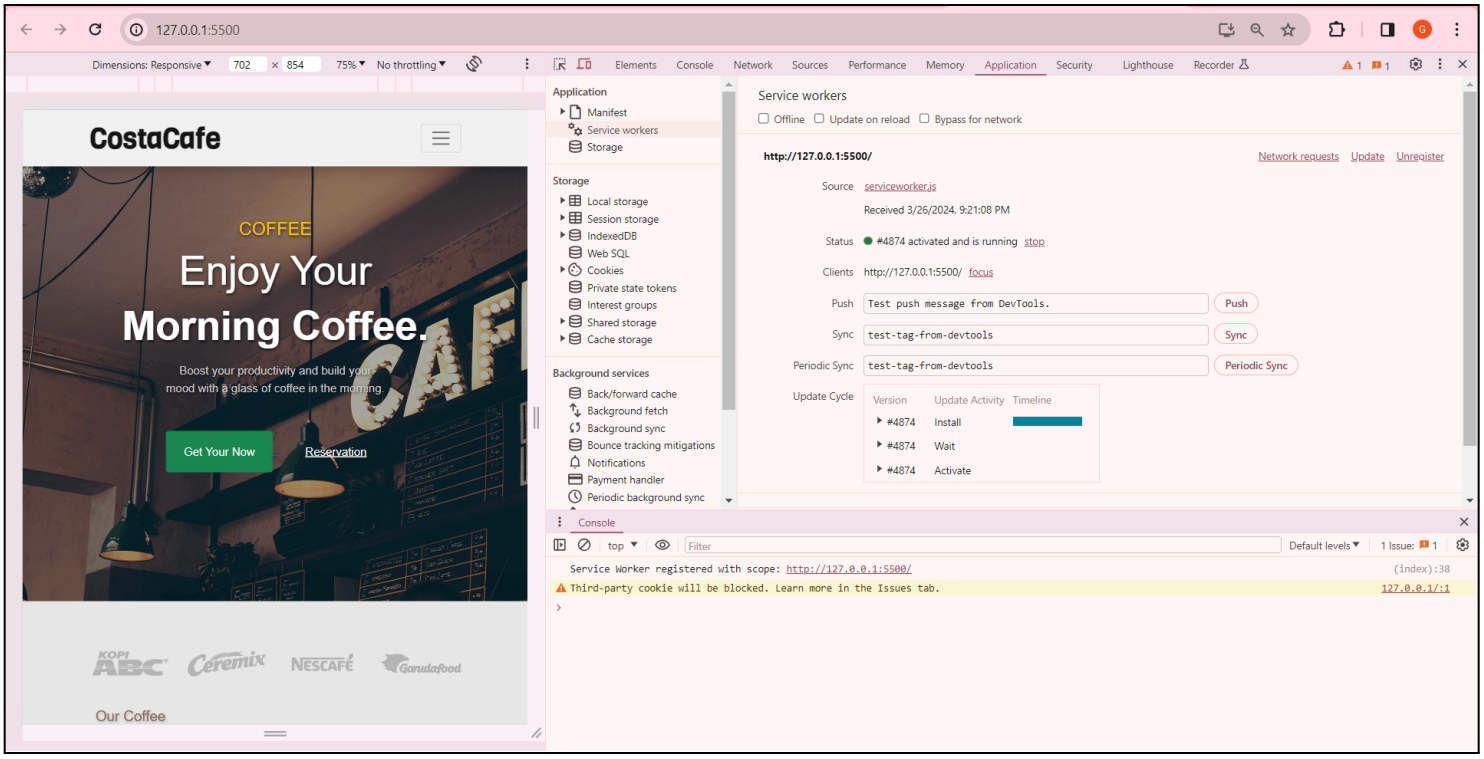
service-worker.js

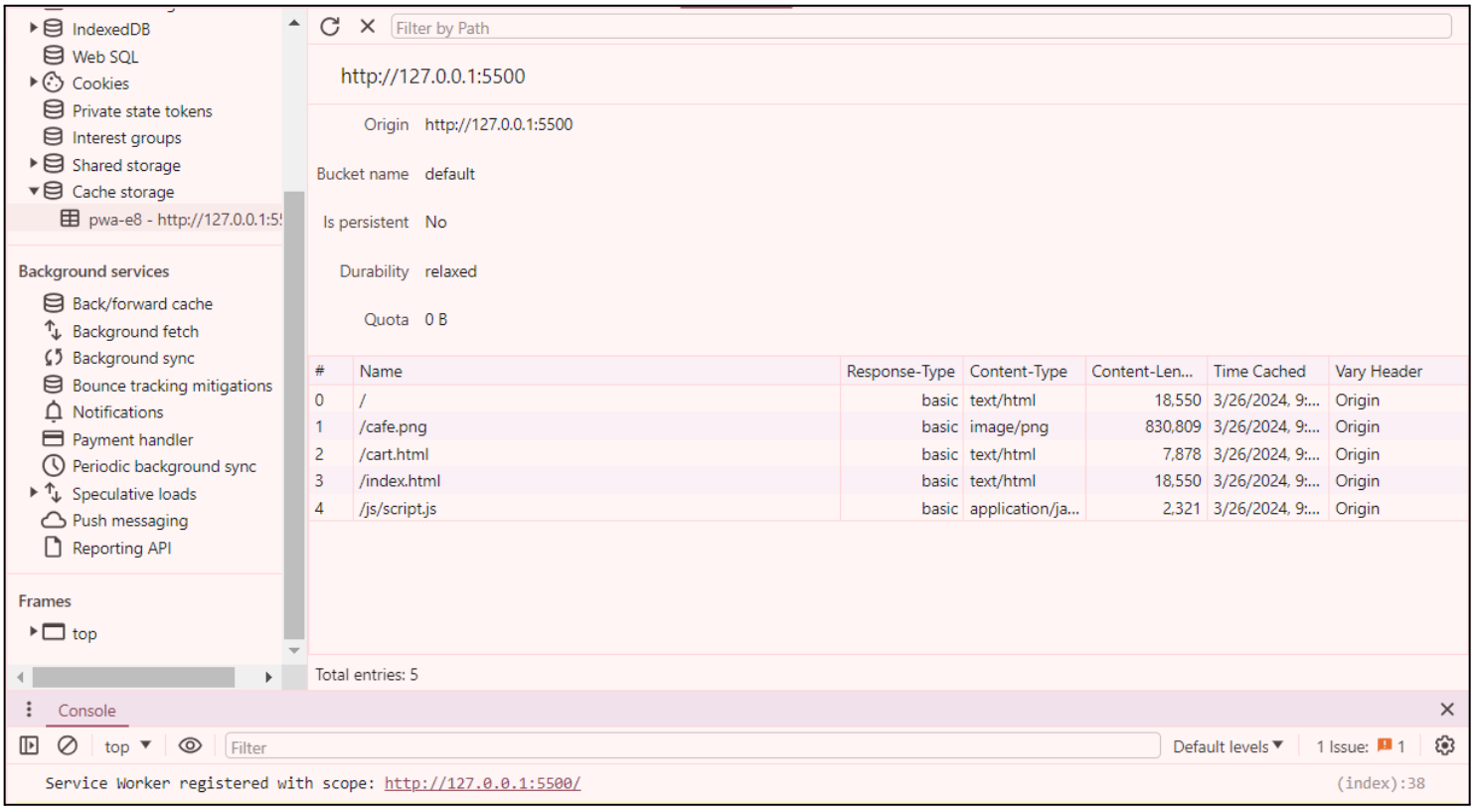
```
const cacheName = "pwa-e8";
const assetsToCache = [
  "/",
  "/index.html",
  "/cart.html",
  "/cafe.png",
  "/js/script.js",
  "/nature.png"
];

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

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

OUTPUT :

In developer tools -> Application -> service -workers

Project Title: Flutter : Trivia App / PWA : Cafe Website **Roll No.: 25**





CONCLUSION : Thus we have registered a service worker, and completed the installation and activation process for a new service worker for the E-commerce PWA.

MAD & PWA Lab
Journal

Experiment No.	09
Experiment Title.	To implement Service worker events like fetch, sync and push for E-commerce PWA
Roll No.	25
Name	Gomati Iyer
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

Experiment 9

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

Theory:

Service Worker

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

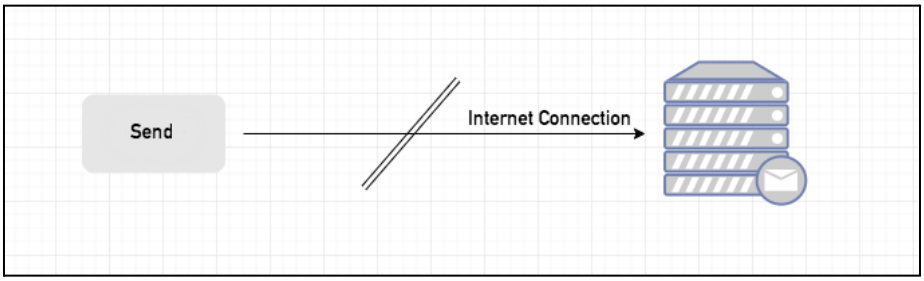
or not. If a cache exists, it is returned to the page, but if not, this is up to you. You can return dummy content or information messages to the page.

Sync Event

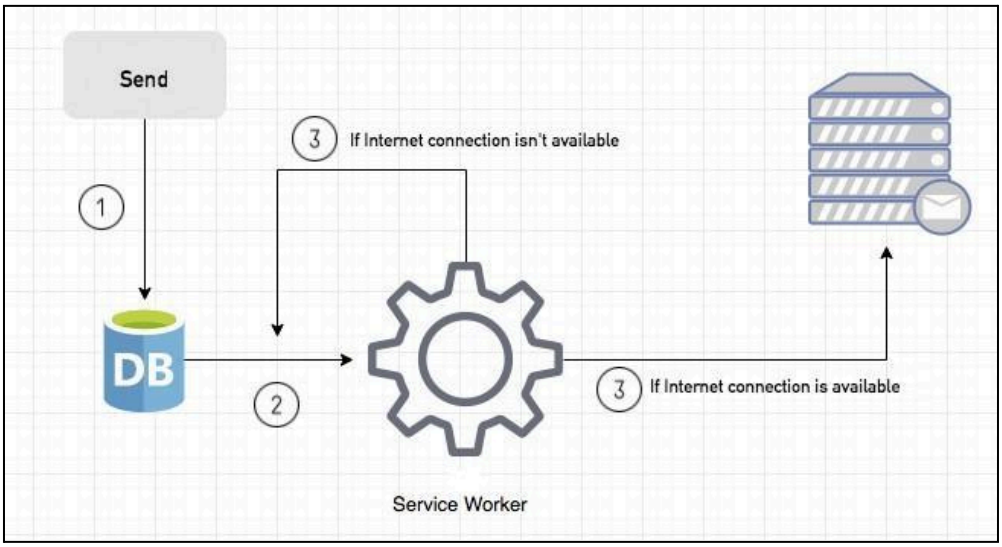
Background Sync is a Web API that is used to delay a process until the Internet connection is stable. We can adapt this definition to the real world; there is an e-mail client application that works on the browser and we want to send an email with this tool. Internet connection is broken while we are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

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

the method value is “pushMessage”, we open the information notification with the “message” property.

Code:

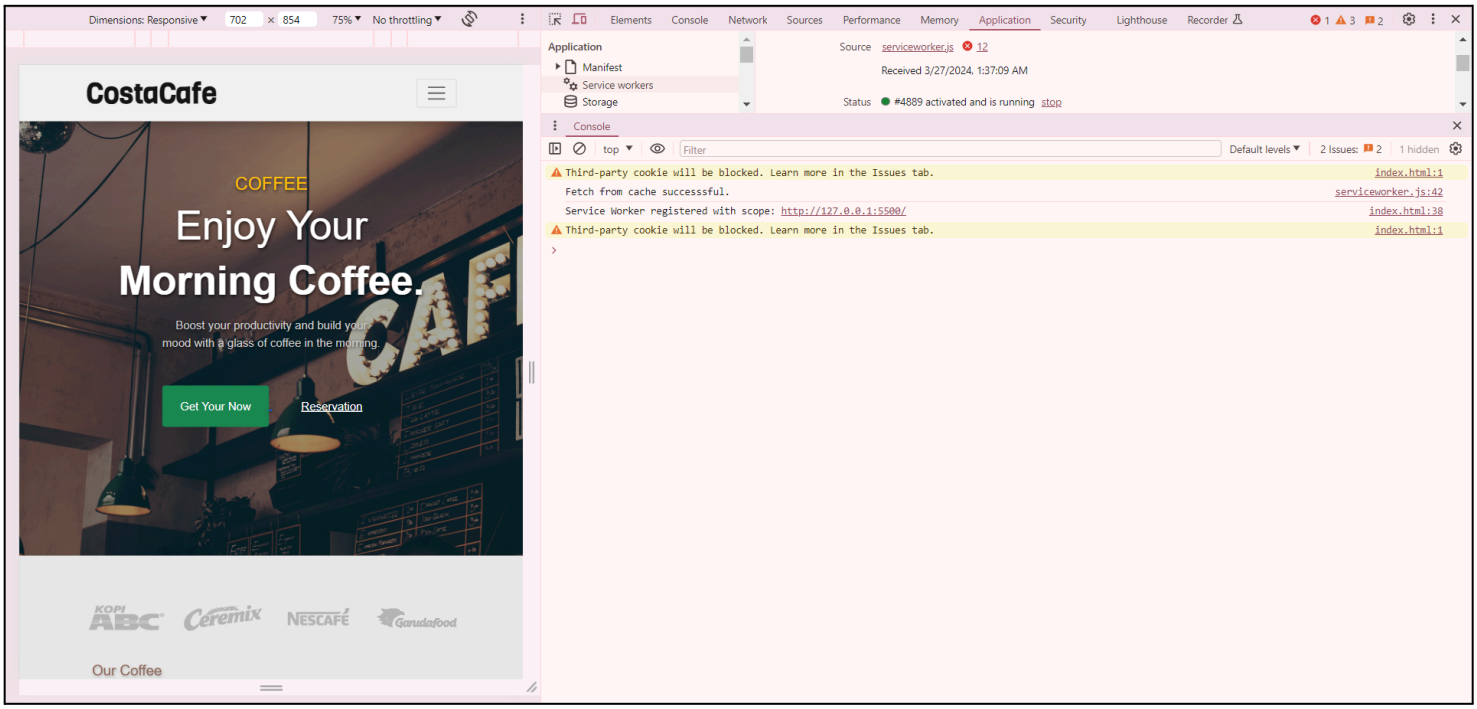
```
self.addEventListener("install", (event) => {
  event.waitUntil(preLoad());
});

self.addEventListener("activate", (event) => {
  event.responWith(checkResponse(event.request).catch(function () {
    console.log("Fetch from cache successsful.");
    return returnFromCache(event.request);
  }));

  console.log("Fetch Sucessful !!");
  event.waitUntil(addToCache(event.request))
});

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

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

var preLoad = function () {
  return caches.open("offline").then(function (cache) {
    // caching index and important routes
    return cache.addAll(["/", "/index.html", "/cart.html", "/menu.html", "/contact.html",
```

```
    "/nature.jpg", "/cafe.png"]);
  });
};

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

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

};

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

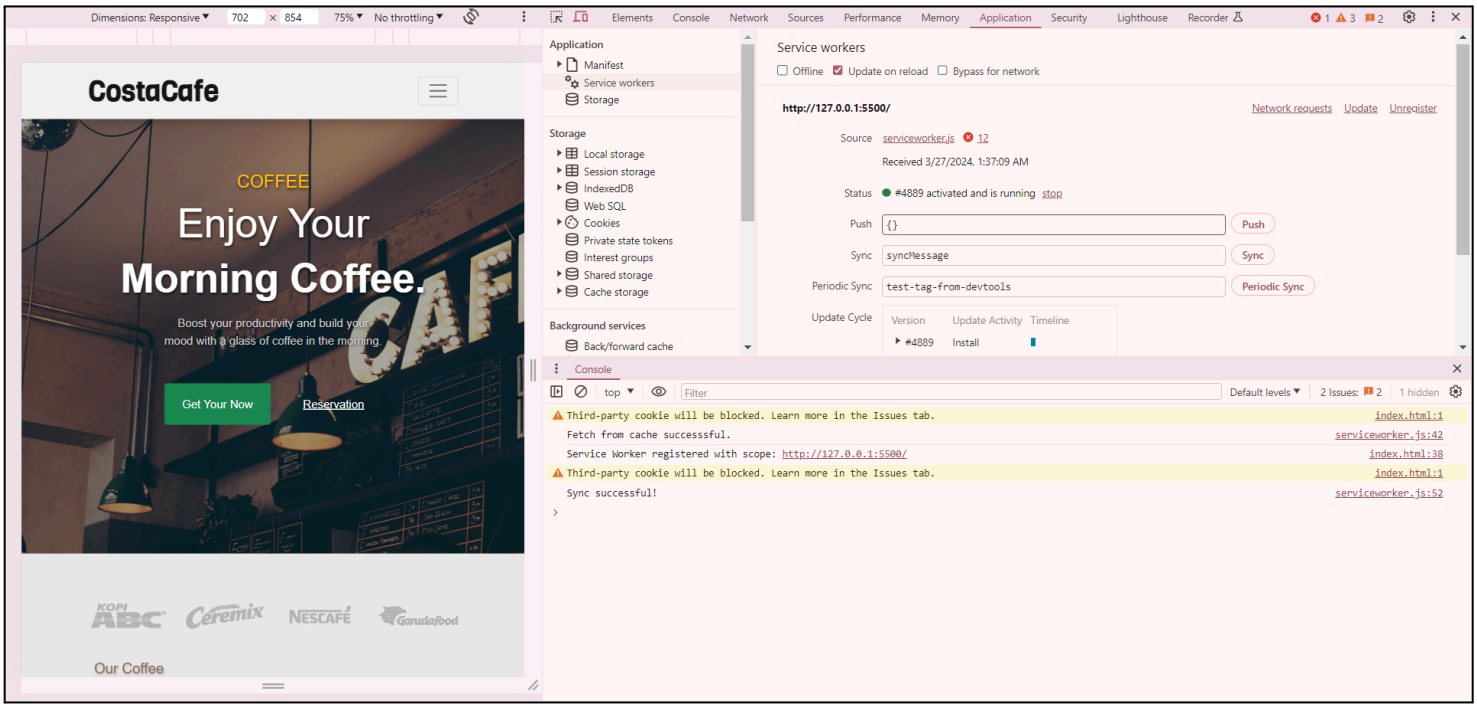
Project Title: Flutter : Trivia App / PWA : Cafe Website **Roll No.: 25**

Output:

Fetch event

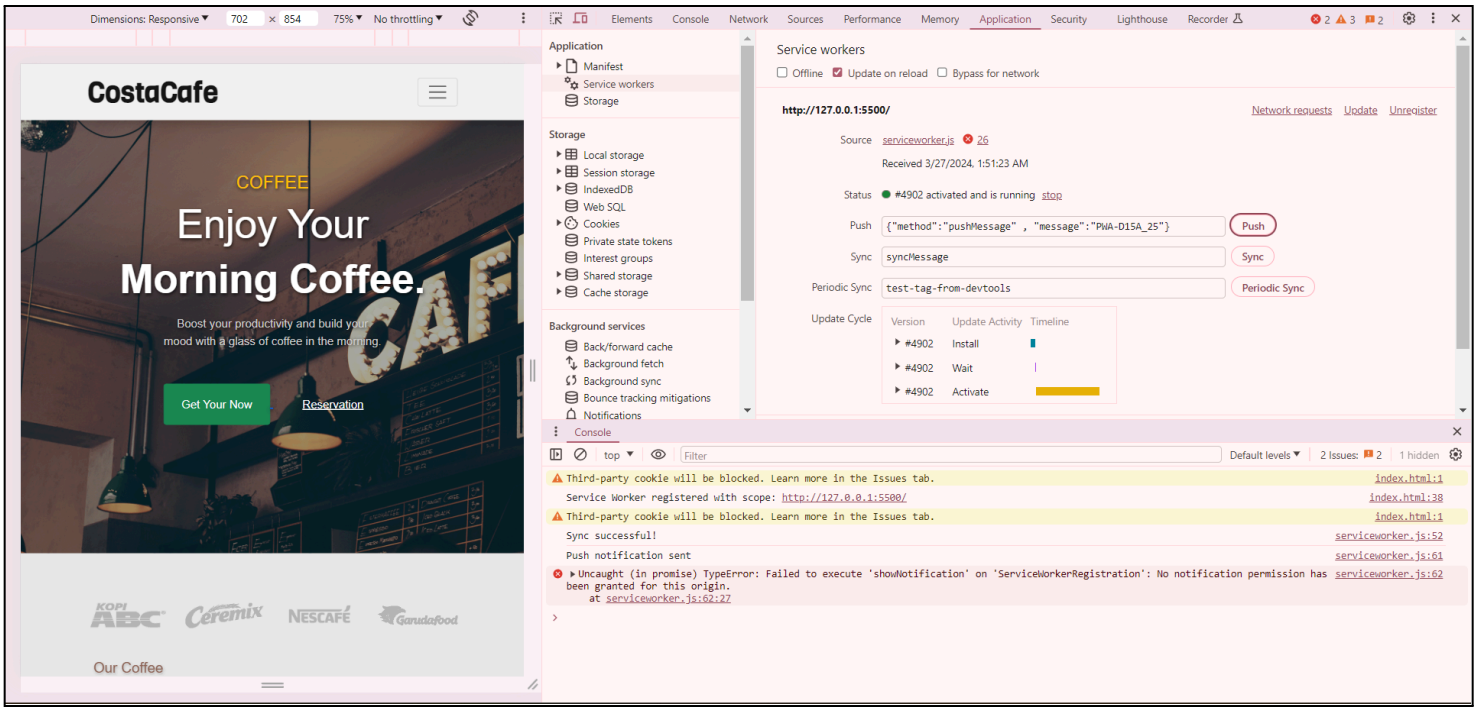


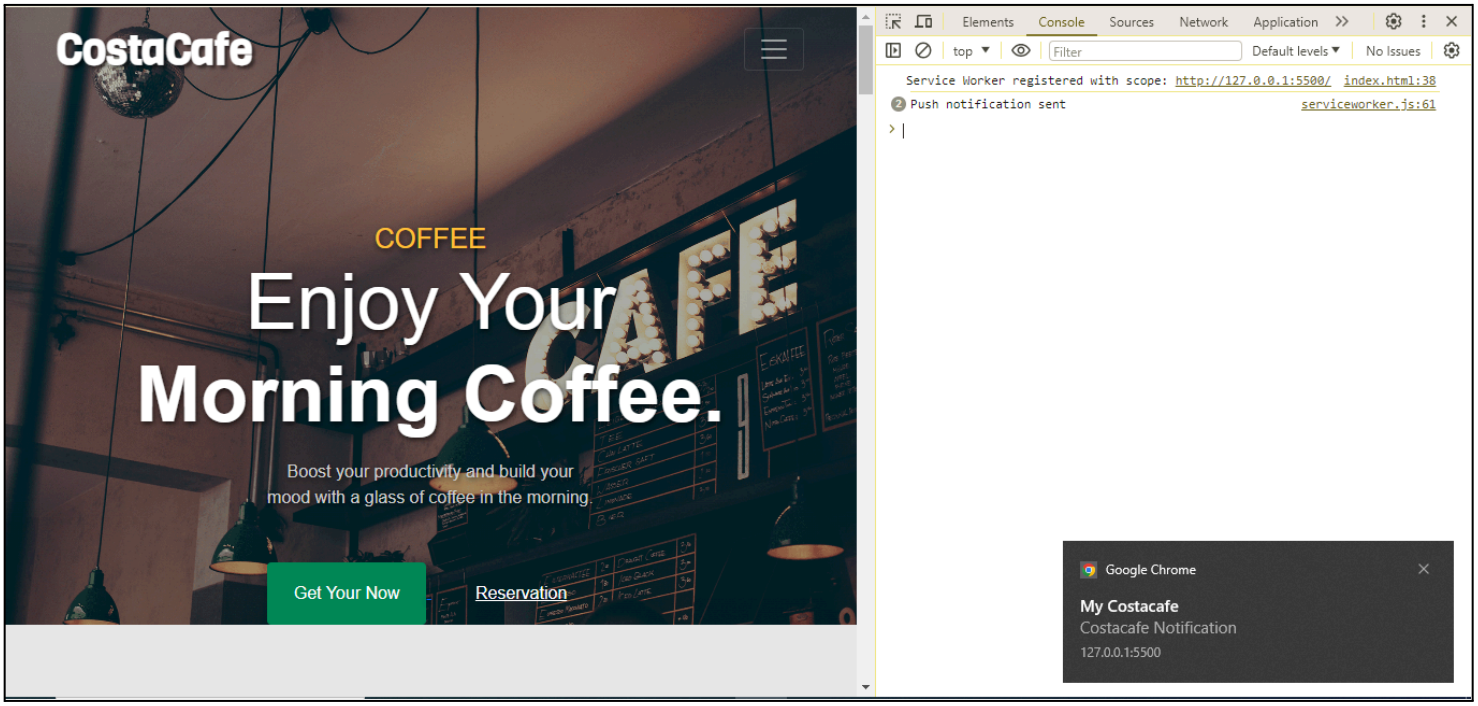
Sync event



Push event

Project Title: Flutter : Trivia App / PWA : Cafe Website **Roll No.: 25**





CONCLUSION : Thus , we have implemented Service worker events like fetch, sync and push for E-commerce PWA

MAD & PWA Lab
Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	25
Name	Gomati Iyer
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

Experiment 10

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

Theory:

GitHub Pages

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

GitHub Pages provides the following key features:

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

Reasons for favoring this over Firebase:

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

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

Pros

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

Cons

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

Firebase

The Realtime App Platform. Firebase is a cloud service designed to power real-time, collaborative applications. Simply add the Firebase library to your application to gain

access to a shared data structure; any changes you make to that data are automatically synchronized with the Firebase cloud and with other clients within milliseconds.

Some of the features offered by Firebase are:

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

Reasons for favoring over GitHub Pages:

1. Realtime backend made easy
2. Fast and responsive

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

Pros

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

Cons

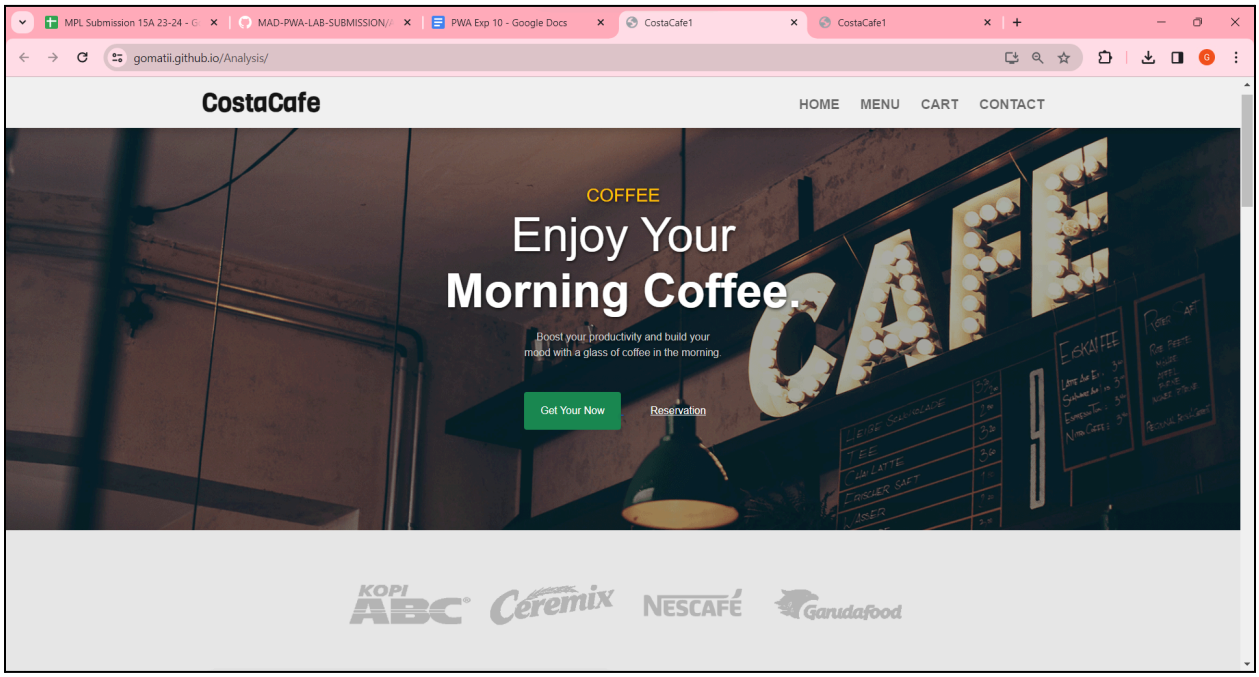
1. Only 10 GB of data transfer is allowed per month. But this is not really a big problem, if you use a CDN or AMP.
2. Command-line interface only.
3. No in-built support for any static site generator.

Link to my GitHub repository:https://github.com/Gomatii/Analysis

Github Pages Link: https://gomatii.github.io/Analysis/

Project Title: Flutter : Trivia App / PWA : Cafe Website **Roll No.: 25**

Screenshot :



CONCLUSION : Thus , the PWA application is hosted on github.

MAD & PWA Lab
Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	25
Name	Gomati Iyer
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	15

EXPERIMENT 11

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

Theory :

Google Lighthouse :

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

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

Key Features and Audit Metrics

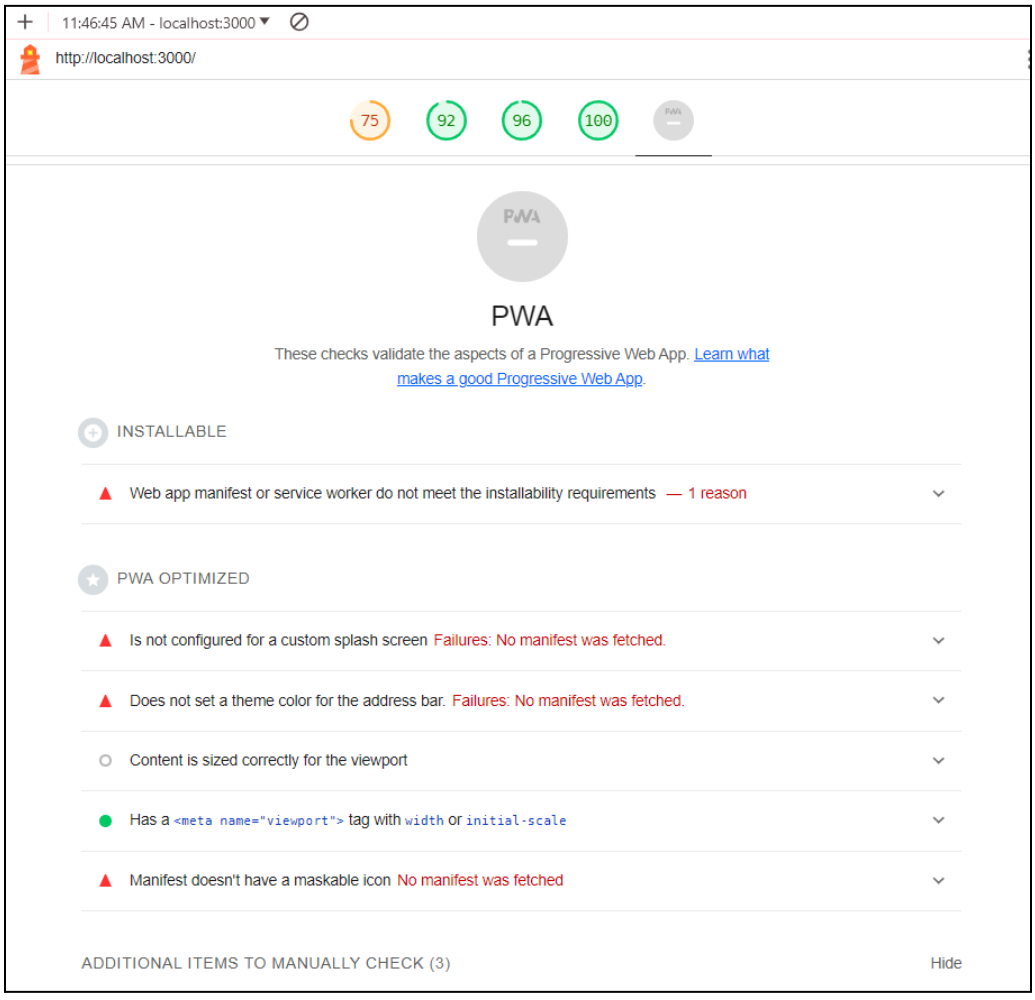
Google Lighthouse has the option of running the Audit for Desktop as well as mobile version of your page(s). The top metrics that will be measured in the Audit are:

1. **Performance:** This score is an aggregation of how the page fared in aspects such as (but not limited to) loading speed, time taken for loading for basic frame(s), displaying meaningful content to the user, etc. To a layman, this score is indicative of how decently the site performs, with a score of 100 meaning that you figure in the 98th percentile, 50 meaning that you figure in the 75th percentile and so on.
2. **PWA Score (Mobile):** Thanks to the rise of Service Workers, app manifests, etc., a lot of modern web applications are moving towards the PWA paradigm, where the objective is to make the application behave as close as possible to native mobile applications. Scoring points are based on the Baseline PWA checklist laid down by Google which includes Service Worker implementation(s), viewport handling, offline functionality, performance in script-disabled environments, etc.
3. **Accessibility:** As you might have guessed, this metric is a measure of how accessible your website is, across a plethora of accessibility features that can be implemented in your page (such as the 'aria-' attributes like aria-required, audio captions, button names, etc.). Unlike the other metrics though, Accessibility metrics score on a

pass/fail basis i.e. if all possible elements of the page are not screen-reader friendly (HTML5 introduced features that would make pages easy to interpret for screen readers used by visually challenged people like tag names, tags such as <section>, <article>, etc.), you get a 0 on that score. The aggregate of these scores is your Accessibility metric score.

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

Before making changes :



Changes made to the code :

For theme color add a meta tag in index.html-
<meta name="theme-color" content="#4285f4">

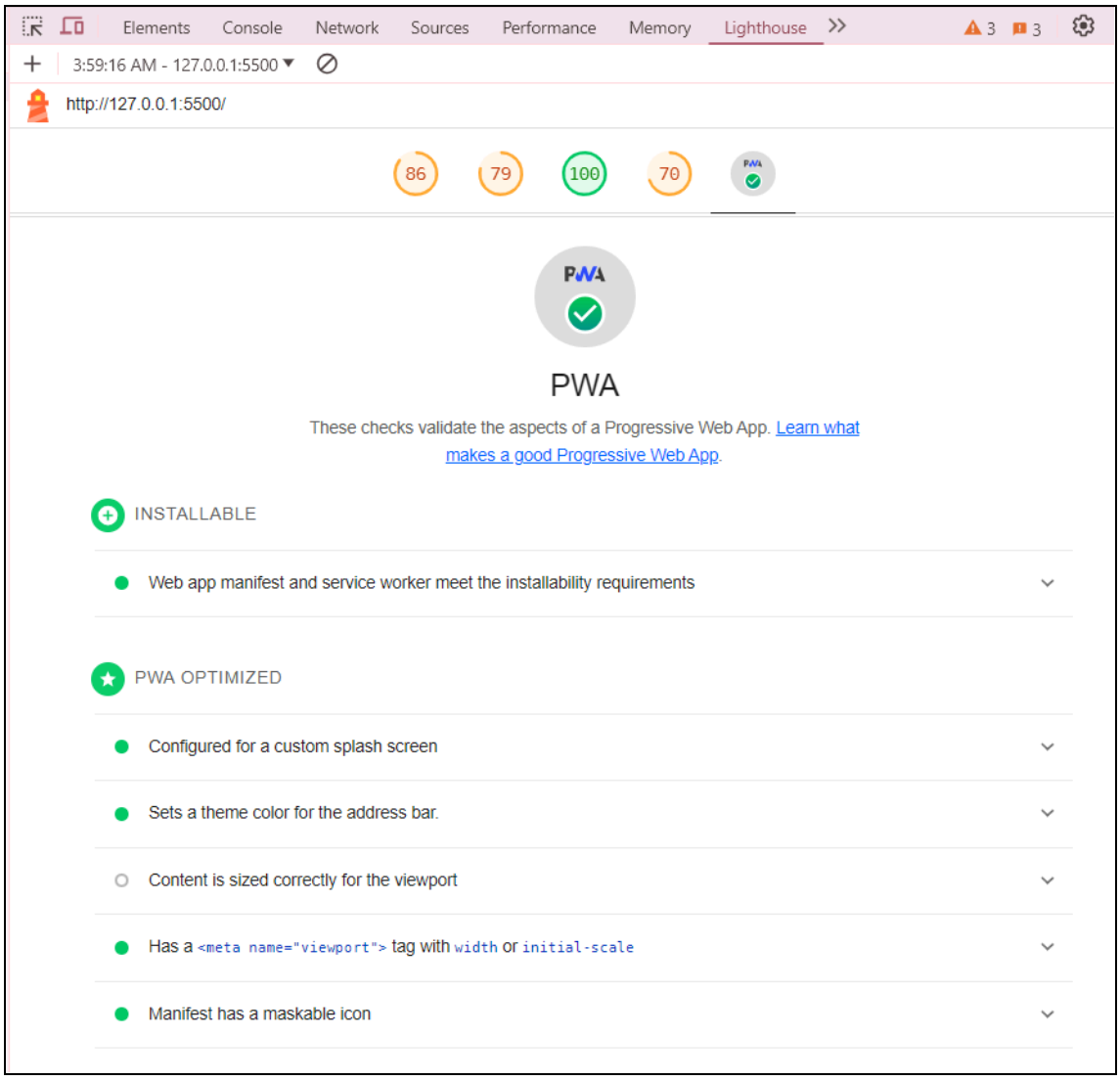
For a maskable icon add "purpose": "any maskable" to the icons in manifest.json file For apple touch icon add the following meta tag in index.html-
<link rel="apple-touch-icon" href="">

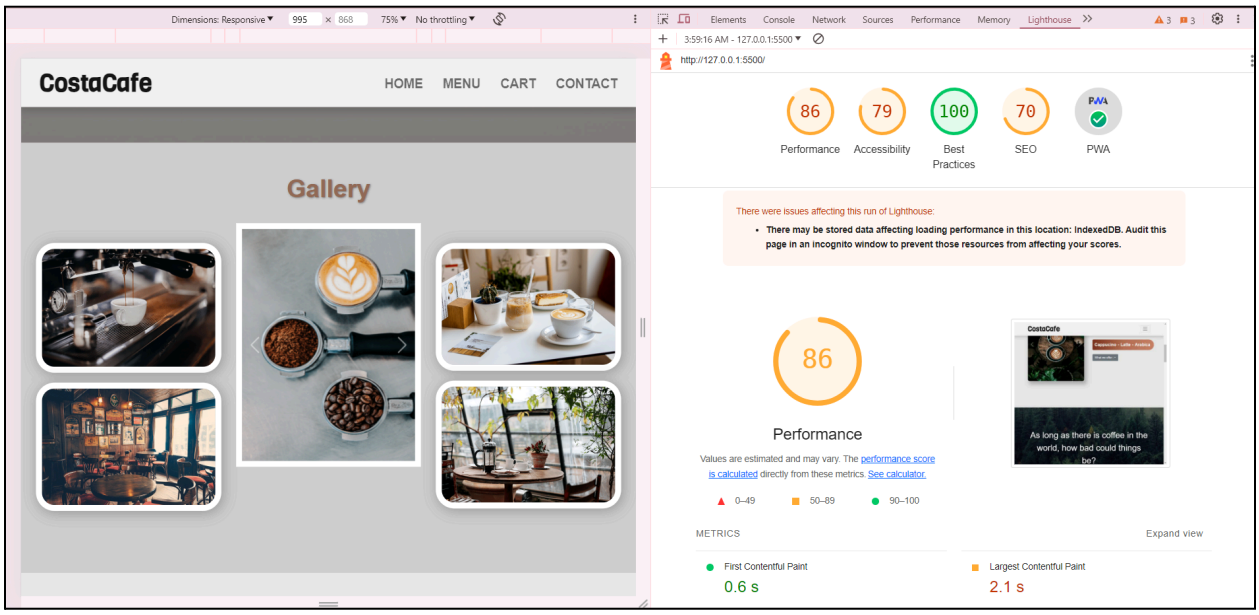
Changes in manifest.json

```
{
"name": "Language Tutor", "short_name": "Lang",
"start_url": "index.html", "scope": "./",
"theme_color": "#ffd31d",
"background_color": "#333",
"display": "standalone",
"icons": [
{
"src": "icon-1.png",
"sizes": "192x192", "type": "image/png"
“purpose”:"any maskable"
},

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

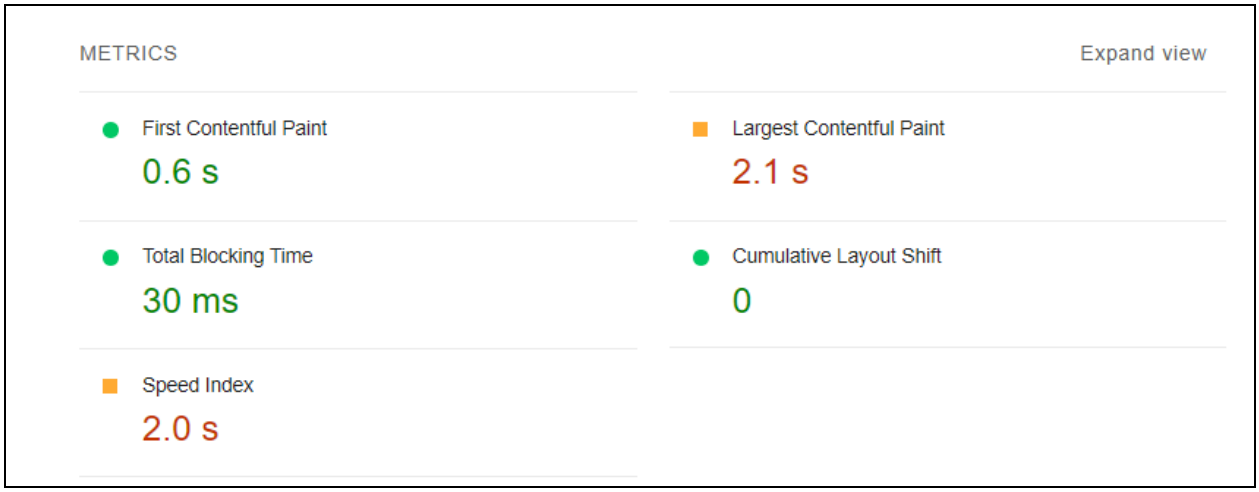
PWA enabled after making the above changes.

Project Title: Flutter : Trivia App / PWA : Cafe Website **Roll No.: 25**

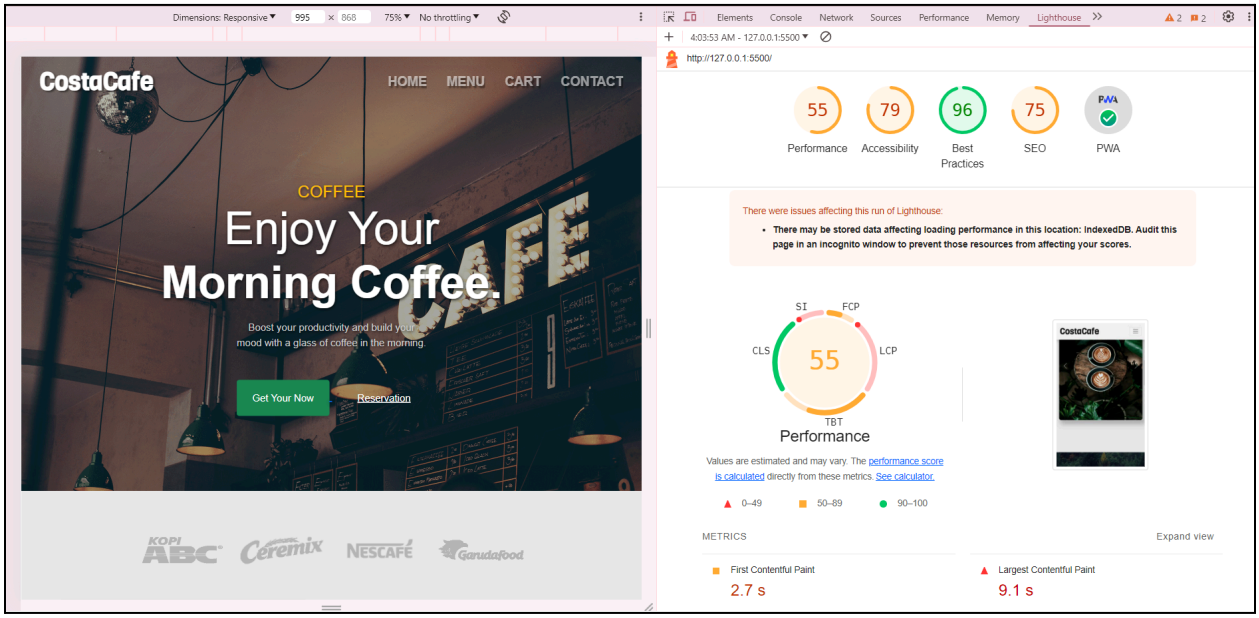


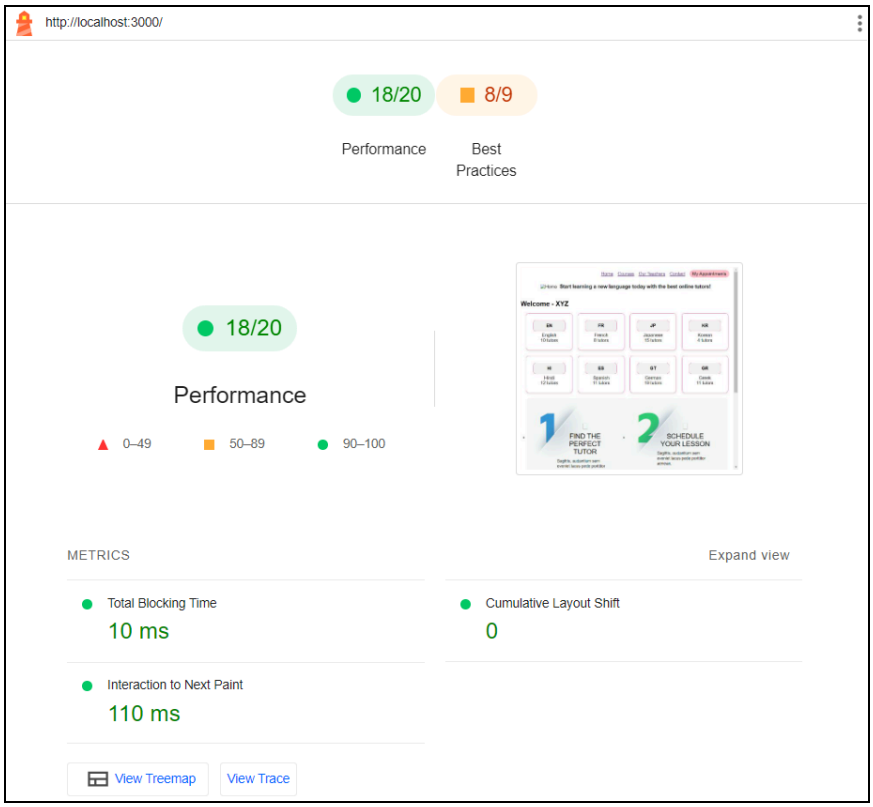


Project Title: Flutter : Trivia App / PWA : Cafe Website **Roll No.: 25**



For Mobile :





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

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.	25
Name	Gomati Iyer
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:	5

T

Date
Page

MAD / PWA Lab — Gomati Iyer
~~Fro~~ Assignment - 01 — DISA 25

Q) Flutter overview.

A) • Flutter is an open source UI development software toolkit developed by Google. It allows developers to build natively combined applications for mobile web and desktop ~~fr~~

• Key features :

1. Sigle codebase for Multiple Platforms:
- Flutter enables development for iOS, Android, Web and desktop from a single codebase.
- 'Write once run anywhere policy'.

2. Hot reload
- Instantly view changes speeds process.

3. Expressive and flexible UI
- It provides a rich set of customizable widgets.
- ~~Fro~~ Reactive programming model for smooth animation and transition.

4. Dart Longuge.
- Simple and productive.

5. Native Performance
- Flutter apps are compiled through native ARM code, providing near-native performance.

Advantages :

1. Productivity : Hot reload and single codebase increase developer productivity.

Teacher's Sign.: _____

2. Fast development : Pre-designed widgets speeds up the development cycle.
3. Consistent UI across platforms : Flutter reduces the effort required to maintain seperate codebases for Android and iOS.
4. Open source and free.

Different from traditional platforms.

- Flutter enables cross-platform development with a unified codebase enabling simultaneous deployment on iOS, Andriod, web and desktop.
- Before flutter, development was platform, specific, seperate, code base, different UIs and slower development cycle.
- The framework's popularity stems from its single codebase efficiency, rapid development iteration and a visually cohesive user experience across diverse platforms.

Q2] Widget Tree and composition.

=) • In flutter, a widget tree is fundamental to building user interfaces
- It represents a heirarchial structure of those widgets reflecting the layout and organization of UI.
- A widget is a basic building block that represents a part of entire screen, ranging from buttons to complex structures.

Teacher's Sign.: _____

1. Heirarchy : The heirarchial organization allows for the composition of complex UIs.
2. Composition : Widgets are composed together to create more significant and complex UI elements.
3. Immutable : Widgets are immutable. Efficient rendering and ensures consistent UI.
4. Reactive UI : When the state of widget changes, it automatically rebuilds widget tree.

Commonly used widgets :

1. Container
- Versatile box model that can contain other widgets
- Commonly used for layout and styling.

2. Row and column
- Used to arrange child widgets horizontally and vertical

3. ListView
- Scrollable lists to display large amount of data.

4. Stack
- Allows widgets to be overlaid on top of each other.

5. AppBar
- Used for creating top App-bar.
- Typically contains title, actions, navigation

6. TextField - capturing user input.

Teacher's Sign.: _____

7 Image -
- Used to display image, it supports various formats

8. FlatButton, RaisedButton & ElevateButton
- Used to create interactive buttons with different visual data
- Responds to user-input

Q3) State management in Flutter

=) • In flutter, state manggement is a crucial aspect of developing applications, espeacially when dealing with dynamic and changing data.
• Some of the common state management approaches are :

1. SetState
- Involves the 'setState' method provided by 'StatefulWidge'. It is suitable for local state management
- Small to moderately complex applications
- Simple UI components with local state.

2. Provider
- Based on 'InheritedWidget' and is simple but powerful, it allows for dependancy injection and easy access to provided objects
- Used for sharing data across widget tree.
- Medium - sized application with moderate state management requirements.

Teacher's Sign.: _____

3. Bloc Pattern or Riverpod
- It involves seperating the business logic from the UI
- It is used in situation where a clear seperation of concerns is crucial

5. GetX
- It provides state management dependency, injection and route management.
- It is known for simplicity and performance
- It is used in medium sized applications.

6. ReduX
- Redux is a predictable state container that is often used with Flutter through packages like flutter_redux
- It is based on unidirectional data flow pattern
- Large scale applications with extensive state manggement needs

Q4) Firebase Integration.

=) - Firebase is a comprehensive mobile and web application development platform provided by Google.
- Steps :

1. Create a Firebase project
2. Add flutter app to Project.
3. Add Firebase configuration to Flutter.
4. Add dependencies.
5. Initialize firebase

Teacher's Sign.: _____

Advantages of using Firebase :

1. Real time database
2. Authentication
3. Cloud Firestore
4. Cloud Functions
5. Cloud Storage
6. Cloud Messaging
7. Performance Monitoring
8. Analytics

Common Used Firebase Services :

1. Firebase Auth
2. Cloud Firestore
3. Firebase Functions
4. Firebase Storage
5. Performance Monitoring
6. Firebase Anglytics

Teacher's Sign.: _____

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.	25
Name	Gomati Iyer
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:	4

Gomati tyagi
DISA
(25)

PWA

Assignment - 02

Q1] Define PWA and explain its significance in modern web development. Discuss the key components that differentiate PWAs from traditional mobile apps.

=) • A Progressive Web App is a type of web application that utilizes modern web capabilities to provide a user experience similar to that of native mobile apps.
• They are built using HTML, CSS and Javascript and they are designed to work across multiple platforms and devices, including desktops, smartphones and tablets.

• Significance of PWA in Modern Wed Development:

1. Cross platform Compatability
- Can run on any device with a compatible web browser.
- No need for seperate codebase.

2. Offline functionality
- uses service workers to leverage service workers to enable offline functionality.

3 Responsive Design
- Adaptable to different screen sizes and orientation

4 App like experience
- Features like push notifications, home screen installat. and full screen mode

Teacher's Sign.: _____

5 Fast Performance
- Quick load times and smooth interactions.

- PWA vs. traditional Mobile Apps.

1. Platform Independence.
- PWAs are not tied to a specific system or device platform.
- Traditional mobile apps are performed for specific OS.

2. Installation
- Direct installation from browser to home-screen.
- Traditional mobile apps need an appstore.

3 Updates
- PWA - automatic updation through the web.
- Traditional Mobile Apps - manual updates through app store.

4. Offline functionality
- PWAs can work offline or with minimal network connectivity because of caching mechanisms & service cor
- Traditional mobile apps have limited offline functionality or require internet for full functionality

5. Distribution
- PWAs can be distributed via URLs, making them easily shareable and accessible to users.
- Traditional Mobile apps have to be distributed through centralized app stores, which may involve processes and restrictions.

Teacher's Sign.:

DISA 25

(Q2)

=) • Responsive web design is an approach to web develapma aimed at creating websites that provide optimal viewing experience across a wide range of devices and screen sizes.
- It involves using flexible layouts, images and CSS media queries to automatically adjust the layout and content of a website based on the device's screen, orientation & resolution.

Importance of Responsive web design in PWAs:

1. Cross-device compatibility
- PWAs work on different devices.
- Responsive web design ensures PWA's layout and content adapt fluidly to the screen

2. Enhanced User Experience.
- Users can easily navigate and interact with the app leading to higher user engagement and satisfaction.

3. Mobile First Approach
- Responsive web design prioritizes mobile experience and optimizing small screens given the increasing number of mobile usage.

Responsive vs Fluid vs Adaptive approaches.

1. Responsive web design.
- Approach: Uses flexible layouts, CSS media queries

Teacher's Sign.: _____

and fluid grids to adapt the layout to different screen sizes and devices.
- Key feature - Adopts dynamically based on the device's viewport size & orientation.
- Example : Navigation bar changes from horizontal layout to collapsible menu on small screens.

2. Fluid Web design
- Approach : Focuses on fluid layouts that use percentages instead of fixed units for sizing elements, allowing them to stretch or shrink.
- Key Feature - Emphasizes flexibility and scalability depending upon viewport size
- Example : Text or images expand or contract smoothly to fill available spacing without causing layout breaks.

3 Adaptive Web design.
- Approach - Involves creating multiple fixed layouts designed for specific screen sizes, with the server delivering the appropriate layout based on the device characteristics
- Key Feature - Provides predifened layouts based on the device's screen size
- Example - Seperate layouts of websites designed for desktop and mobile versions; tailored specifically.

Teacher's Sign.: _____

DISA 25

Date
Page

Q3] Lifecycle

=> 1. Registration

- First step is to register it within the web-application.
- It occurs in the main javascript file of the application such as index.js or app.js
- Method used - 'navigator.serviceWorker.register()' passing the path to the service worker file as an argument.
- Example :

```
if ('serviceWorker' in navigator) {
    navigator.serviceWorker.register('/sw.js')
    .then(registration => {
        console.log("Successfully registered");
    })
    .catch(error => {
        // handle error
    });
}
```

2. Installation

- Once registered, the service worker goes through an installation phase.
- During installation, the browser downloads the service worker script file specified using registration.
- The installation process also triggers the 'install' event within the service worker script, allowing developers to perform tasks like caching static assets & resources.

Teacher's Sign.: _____

- Developers listen for 'install' event using the self.addEventListener()' method within the service worker script.
- For example:

```
self.addEventListener('install', event => {
    event.waitUntil(
        caches.open('my-cache')
        .then(cache => {
            return cache.addAll([
                '/index.html',
                '/styles.html',
            ]);
        })
    );
});
```

3. Activation,
- After installation, the service worker enters an activation phase.
- During activation, the browser activates the new service worker and starts controlling pages within its scope.
- The activation process also triggers the 'activate' event within the service worker script, allowing developers to perform tasks such as cleaning up outdated caches or updating service worker logic.
- Developers listen for the 'activate' event using the self.addEventListener()' method within the service worker script.

Teacher's Sign.: _____

DISA (25)

Q4) Explain the use of IndexedDB in the Service Worker for data storage.

⇒ • IndexedDB is a robust, asynchronous, transactional database system built directly into the browser, allowing web applications to store large amounts of structured data locally.

Uses

1. Persistent storage.
- Data stored in database persists across page reloads and browser sessions.
- It makes it suitable to access data stored offline or across different parts of application.

2. Asynchronous Operation.
- Database operations can be performed without blocking the main thread of application.

3. Transaction based.
- Uses transactions to ensure data integrity and consistency.
- Operations are performed within traction to ensure ACID properties.

4. Indexed queries.
- Allows developers to create indexes on specific object properties enabling efficient querying and retrieval of data.

Teacher's Sign.: _____

5. Versioning
- Supports database versioning, allowing developers to define and manage database schemas over time.

6. Caching dynamic content
- Service workers can intercept network requests and cache responses in IndexedDB for offline access.

7. Background Data Sync.
- Service workers can periodically synchronize data with a remote server in the background storing the synchronized data in IndexedDB for later retrieval.

8. Storing User Preferences
- It can be used to store various client side data like preferences and settings locally providing personalized experience for users across different devices and sessions.

***

Teacher's Sign.: _____