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

Lab Assistant

Subject Teacher

Mrs. Kajal Joseph

Principal

Head of Department

Dr. Mrs. Shalu Chopra

Name of the Course : MAD & PWA Lab

Course Code : ITL604

Project Title:Fiver(flutter) / Jwellery site(Pwa) **Roll No.27**

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:Fiver(flutter) / Jwellery site(Pwa) **Roll No.27**

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

MAD & PWA Lab
Journal

Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	27
Name	Param Keswani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	

Name - Param Keswani

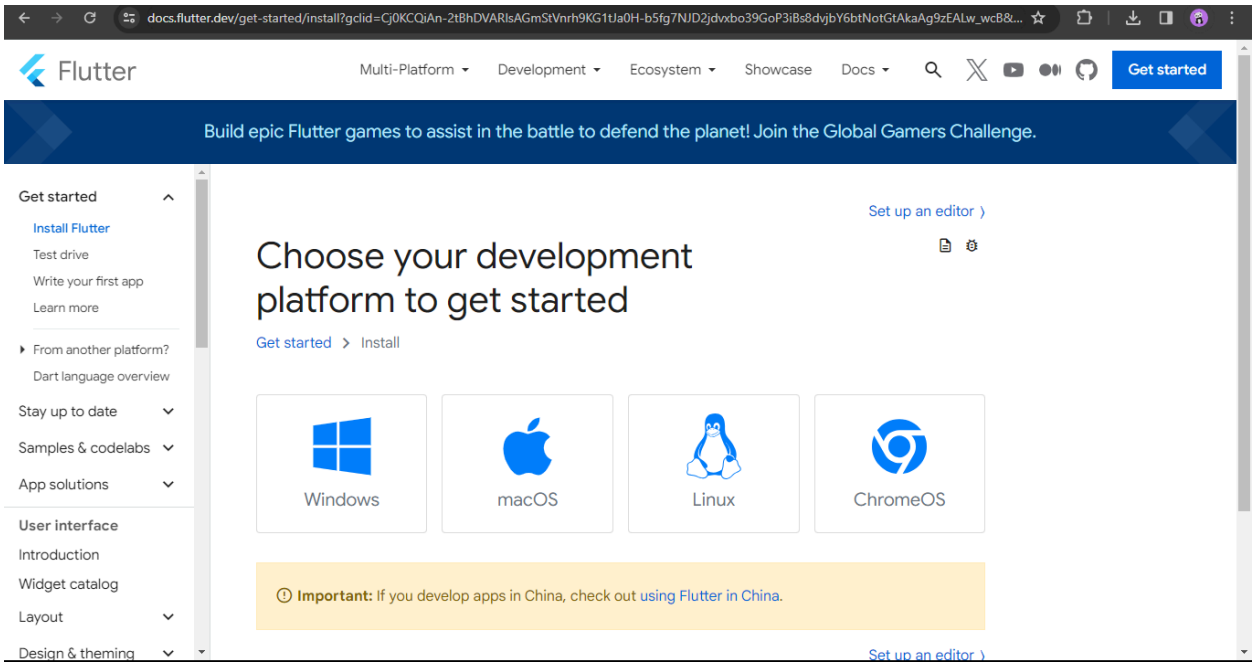
Div - D15A Roll no - 27
Batch - C

Experiment - 1

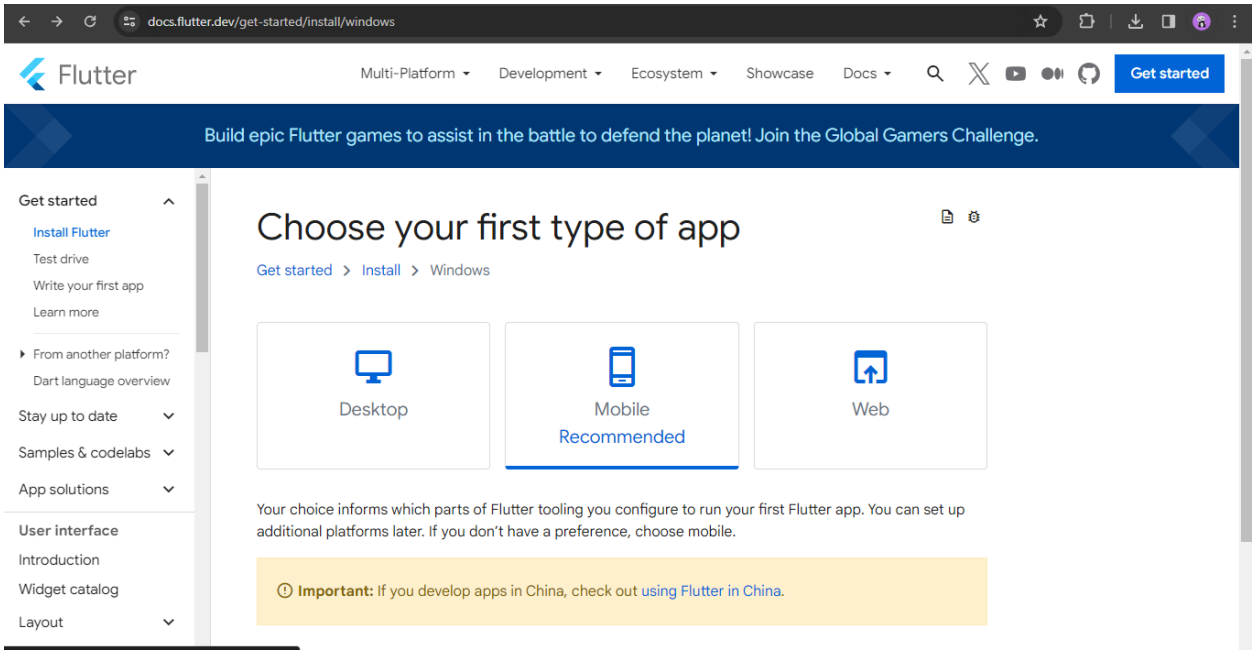
Aim - Installation and Configuration of Flutter Environment.

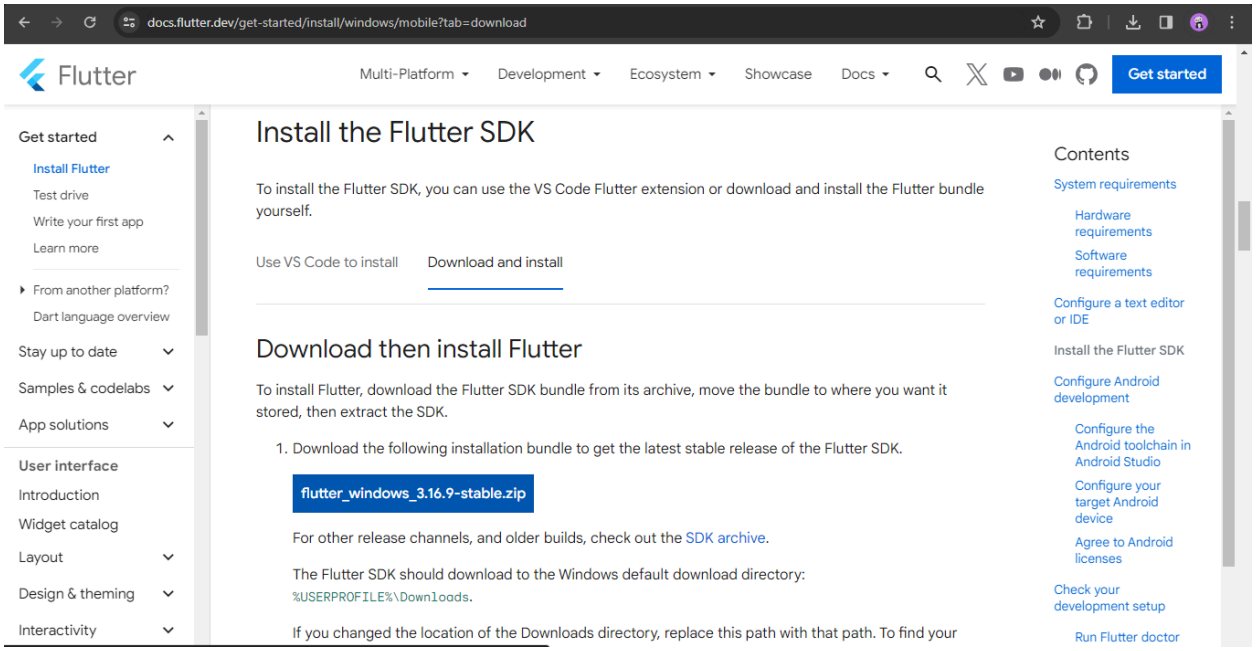
Pre Requisites: Android Studio Hedgehog, Visual Studio Code and Flutter package.I highly recommend watching this video, for Setting up your Flutter Environment and your Virtual Device: https://youtu.be/ZSWfgxrxN0M?feature=shared

Installation -

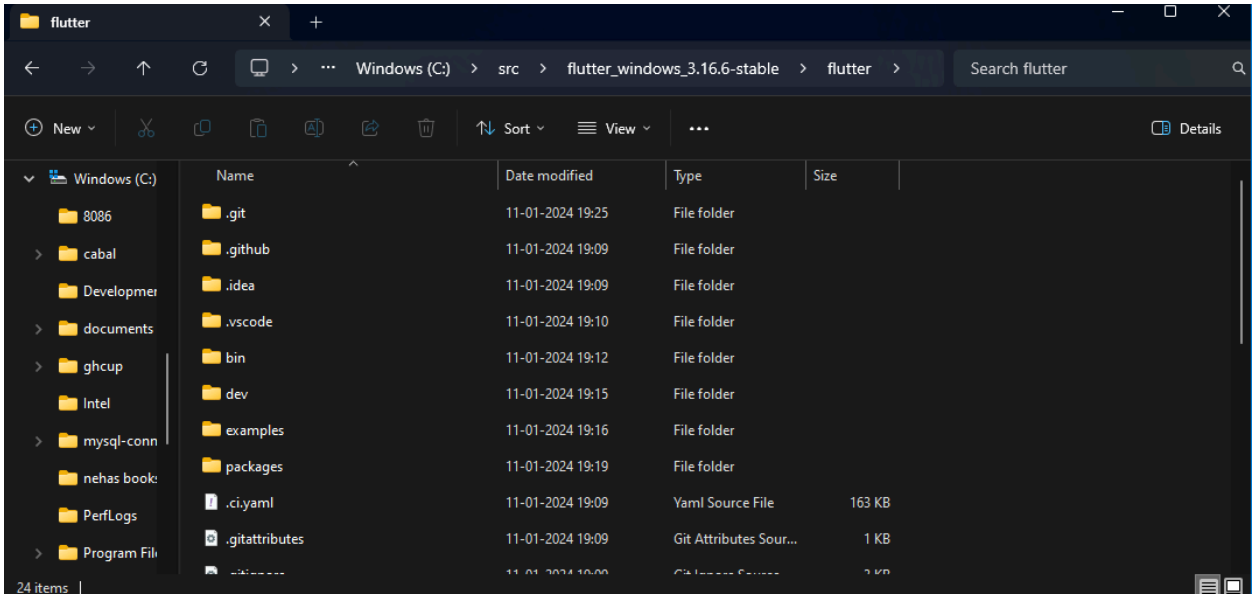


Project Title:Fiver(flutter) / Jwellery site(Pwa) **Roll No.27**

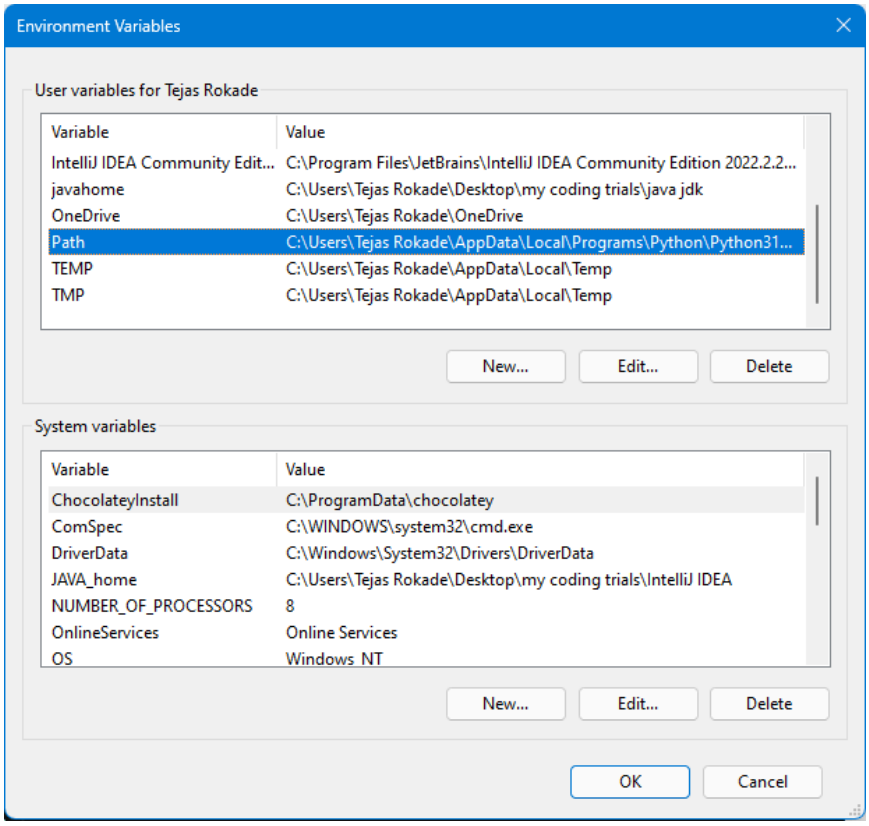




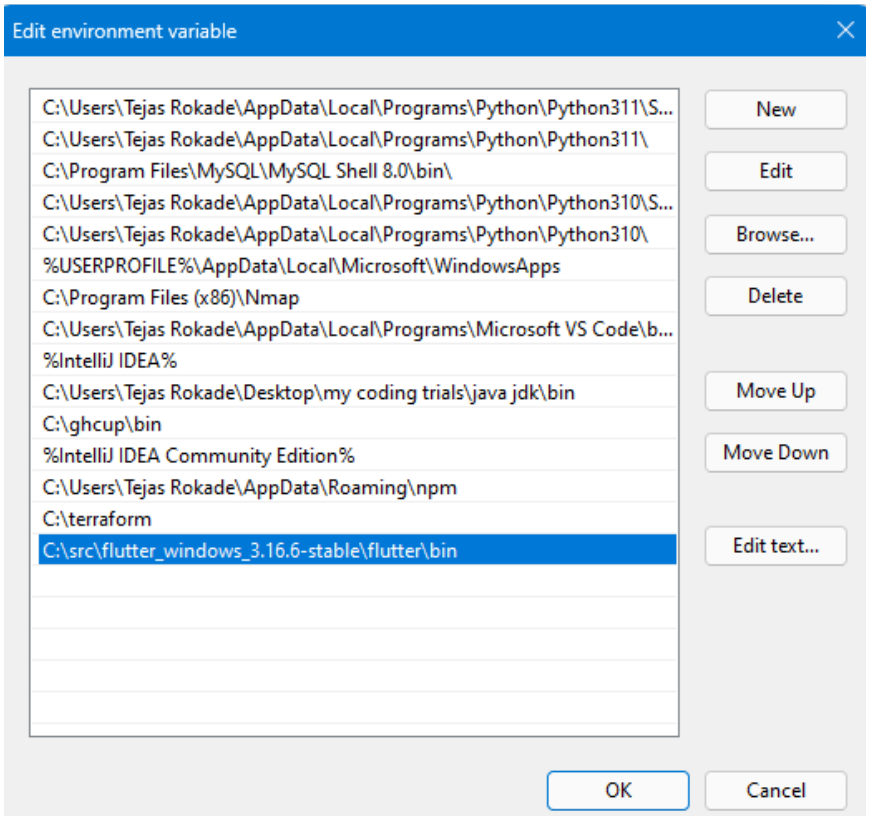
Extract the files in the directory -



Setup path in Environment variables -



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



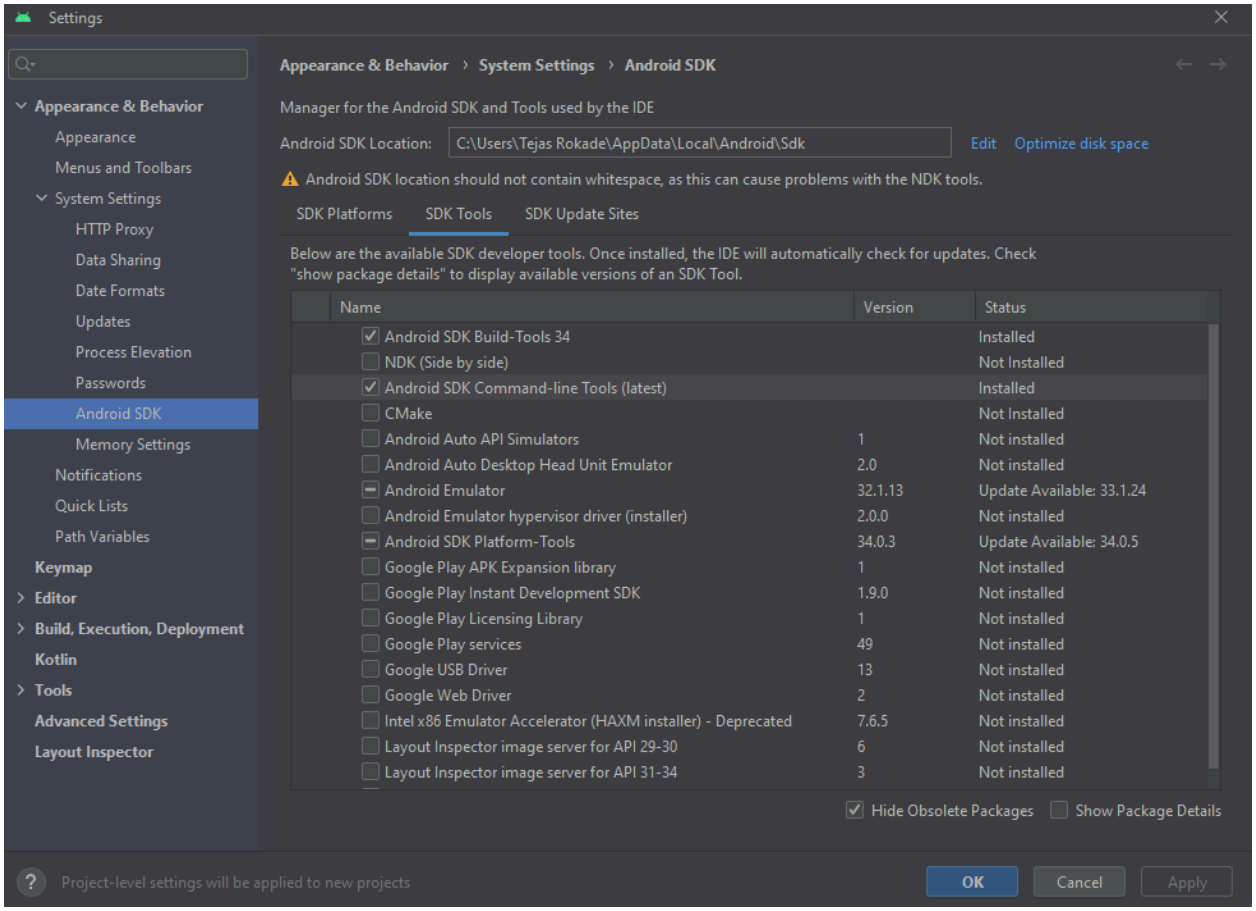
Then on CMD -

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

Common commands:

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

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

Usage: flutter <command> [arguments]

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

```
C:\Users\Tejas Rokade>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.16.6, on Microsoft Windows [Version 10.0.22621.3007], locale en-IN)
[√] Windows Version (Installed version of Windows is version 10 or higher)
[√] Android toolchain - develop for Android devices (Android SDK version 34.0.0)
[√] Chrome - develop for the web
[√] Visual Studio - develop Windows apps (Visual Studio Build Tools 2019 16.11.24)
[√] Android Studio (version 2022.2)
[√] VS Code (version 1.85.2)
[√] Connected device (3 available)
[√] Network resources

• No issues found!

C:\Users\Tejas Rokade>
```

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

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



Code -

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

```
void main() {
runApp(const MyApp());
}
class MyApp extends StatelessWidget {
const MyApp({Key? key}) : super(key: key);
@override
Widget build(BuildContext context) {
return MaterialApp(
title: 'Welcome to Flutter',
home: Scaffold(
appBar: AppBar(
title: const Text('Welcome to Flutter'),
),
body: const Center(
child: Text('Param Keswani''),
),
),
);
}
}
```

Output -

MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	27
Name	Param Keswani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Param keswani
D15A 27
Batch B

Experiment No 2

AIM: To design Flutter UI by including common widgets.

Theory:

In summary, Flutter widgets are fundamental components in constructing the user interface of a Flutter application.They can be broadly categorized into two types: `StatelessWidget` representing immutable parts of the UI and `StatefulWidget` representing mutable components that can change over time.

Some key Flutter widgets include:

1. Scaffold: The basic structure for a Flutter app, providing layout elements such as AppBar, BottomNavigationBar, and a body for main content.
2. Container: A versatile box model used for layout, padding, margin, decoration, and constraints, capable of containing other widgets.
3. Row & Column: Widgets for arranging child widgets horizontally (Row) or vertically (Column), essential for creating flexible and responsive layouts.
4. Text: Used for displaying text on the screen with support for various styling options like font size, color, and alignment.
5. TextField: Captures user input, such as text, numbers, or passwords, with the `onChanged` property for dynamic updates based on user input.
6. Buttons: Various button widgets like `ElevatedButton` or `TextButton` trigger actions when pressed, providing a means for user interaction.
7. Forms: The `Form` widget manages a group of `TextFormField` widgets, facilitating input validation and submission.
8. Icons: The `Icon` widget displays icons from libraries, enhancing visual elements and conveying meaning through symbols.

Key Design Principles highlighted include:

● Consistency: Common widget usage fosters a consistent design language throughout the app.
● Responsive Layouts: Widgets like `Row` and `Column` aid in creating responsive and flexible layouts, adapting to different screen sizes.
● User Input Handling: `TextField` and `Form` widgets facilitate proper handling, ensuring data integrity and validation.
● Interactive Elements: Buttons and icons contribute to interactivity and user engagement within the app.
● Visual Styling: The `Container` widget and styling properties of other widgets allow for visual customization and theming.

Hero Card: A hero card typically refers to a prominent or featured card within a user interface, often used in web design or mobile app design. A hero card is usually larger in size compared to other cards and is placed prominently on a page or screen to draw attention to a specific piece of content, product, or feature. It may contain a title, description, image, and call-to-action button, among other elements.

Card One: "Cardone" could potentially refer to a variety of things depending on the

context. In software development or user interface design, it might refer to the first card in a series of cards, or it could be a specific component or module named "CardOne" within a codebase or design system.

Code :

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

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

  @override
  State<n_search> createState() => _n_searchState();
}

class _n_searchState extends State<n_search> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
          title: TextField(
            decoration: InputDecoration(
              hintText: 'Search',
            ),
          ),
          backgroundColor: Colors.amber,
          automaticallyImplyLeading: false,
          leading: IconButton(
            onPressed: () {
              Navigator.of(context).pop();
            },
            icon: const Icon(Icons.arrow_back_ios),
          ),
          actions: <Widget>[
            Padding(
                padding: EdgeInsets.only(right: 20.0),
                child: GestureDetector(
                  onTap: () {},
                  child: Icon(
                    Icons.search,
                    size: 26.0,
                  ),
                )),
          ],
```

```
),
body: Column(
  children: [
    Card(
      shadowColor: Colors.black,
      elevation: 10,
      child: Container(
        child: Flex(
          direction: Axis.horizontal,
          children: [
            Container(
              margin: EdgeInsets.fromLTRB(20, 10, 10, 10),
              padding: EdgeInsets.fromLTRB(0, 0, 10, 0),
              decoration: BoxDecoration(
                  border: Border(
                right: BorderSide(
                  //                   <--- right side
                  color: Colors.black,
                  width: 3.0,
                ),
              )),
              child: Image.network(
                  "https://i.pinimg.com/736x/23/ea/0c/23ea0c17068f4e290bada3457c5fef0b.jpg",
                  height: 700,
                  width: 80),
            ),
            Container(
              padding: EdgeInsets.fromLTRB(25, 5, 0, 0),
              margin: EdgeInsets.fromLTRB(0, 20, 90, 0),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text("Apple",
                      style: TextStyle(
                          color: Colors.black,
                          fontSize: 30,
                          fontWeight: FontWeight.bold)),
                  Text("Visit profile",
                      style: TextStyle(
                          color: Colors.black,
                          fontSize: 15,
                          fontWeight: FontWeight.bold)),
                ],
              ),
```

```
                ),
                Container(
                  child: Icon(Icons.email),
                  padding: EdgeInsets.fromLTRB(20, 0, 0, 10),
                ),
              ],
            ),
            color: Color.fromARGB(255, 194, 63, 53),
            height: 100,
            width: double.infinity,
            margin: EdgeInsets.all(10),
          ),
        ),
      ],
    ));
  }
}
```

Screenshot :

MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	27
Name	Param Keswani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Param keswani
D15A 27
Batch B

Experiment no 3

Aim:To include images and fonts in flutter app.

Theory:

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

1. Import Libraries:

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

2. Adding Local Images:

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

3. Adding Network Images:

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

4. Image Widget:

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

5. ImageProvider:

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

6. CachedNetworkImage (Optional):

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

7. Image Loading and Error Handling:

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

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

Code:

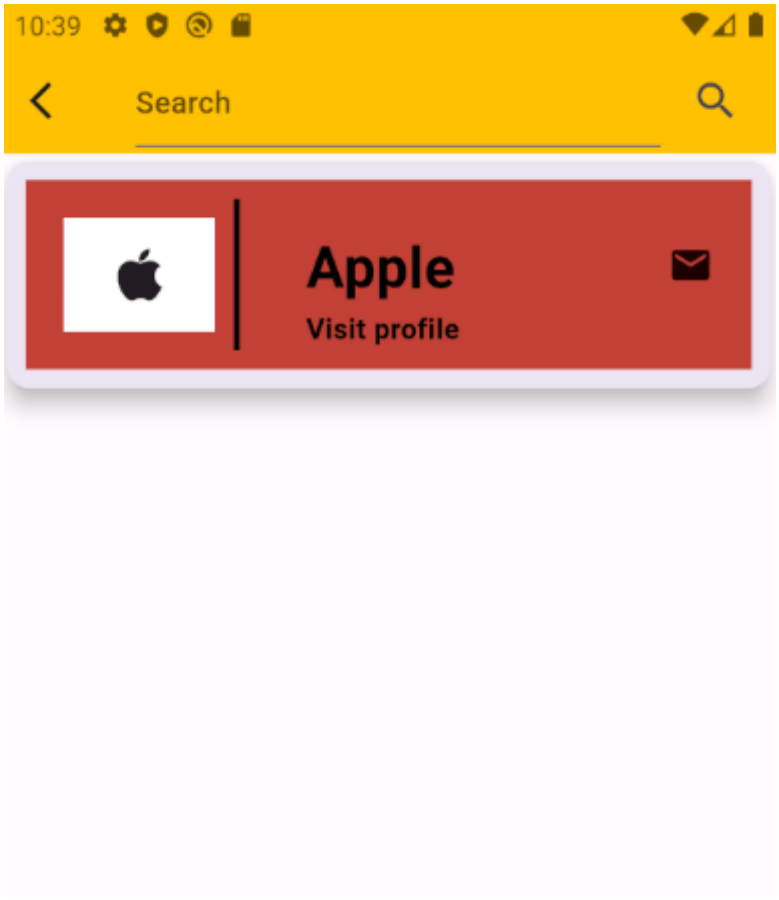
For Image and card creation -

```
class _CategoryListState extends State<CategoryList> {
 final List<String> categories = ['Category 1', 'Category 2', 'Category 3'];
 String selectedCategory = 'Category 1'; // Default category

 @override
 Widget build(BuildContext context) {
  return Container(
    padding: EdgeInsets.symmetric(horizontal: 16.0),
    child: Column(
```

```
crossAxisAlignment: CrossAxisAlignment.start,
children: [
  Text(
    'Category List',
    style: TextStyle(
      fontSize: 20.0,
      fontWeight: FontWeight.bold,
    ),
  ),
  SizedBox(height: 8.0),
  Row(
    children: [
      Text('Select a category: '),
      DropdownButton<String>(
        value: selectedCategory,
        onChanged: (String? newValue) {
          // Update the selected category when the dropdown changes
          if (newValue != null) {
            setState(() {
              selectedCategory = newValue;
            });
          }
        },
        items: categories.map((String category) {
          return DropdownMenuItem<String>(
            value: category,
            child: Text(category),
          );
        }).toList(),
      ),
    ],
  ),
  SizedBox(height: 50.0),
  // Updated section with 4 cards and images
  Container(
    height: 200.0,
    child: ListView.builder(
      scrollDirection: Axis.horizontal,
      itemCount: 4, // Number of cards
      itemBuilder: (context, index) {
        return Card(
          margin: EdgeInsets.all(8.0),
          child: Container(
            width: 150.0, // Customize the card width as needed
```

```
                child: Column(
                  children: [
                    // Add an Image widget with the image path
                    Image.asset(
                      'assets/card_image_${index + 1}.jpeg',
                      height: 120.0,
                      width: 150.0,
                      fit: BoxFit.cover,
                    ),
                    ListTile(
                      title: Text('Card ${index + 1}'),
                      subtitle: Text('Description of card ${index + 1}'),
                      // Add onTap callback if needed
                    ),
                  ],
                ),
              ),
            );
          },
        ),
      ),
    ],
  ),
 );
}
}
```

Project Title:Fiver(flutter) / Jwellery site(Pwa) **Roll No.27**



MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	27
Name	Param Keswani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Param Keswani
D15A 27
Batch B

Experiment no 4

Aim - To create an interactive Form using form widget

Theory -

Interactive Form Creation in Flutter

Flutter offers an array of powerful widgets and techniques to build adaptable and user-friendly forms that seamlessly integrate with your app's design and functionality. This guide delves into the key concepts and strategies involved in crafting interactive forms.

Fundamental Widgets:

● Form: The overarching container that encompasses form fields and manages validation. Create a unique GlobalKey<FormState> for global access and validation.

● FormField: A base class for input widgets, providing styling, validation, and error handling. Choose the appropriate concrete widget based on the input type (e.g., TextField, Checkbox, DropdownButton).

● TextInputAction: Set the textInputAction property on TextField to control the on-screen keyboard's behavior (e.g., TextInputAction.next, TextInputAction.done).

● FocusNode: Control keyboard focus navigation between form fields or widgets within a field (e.g., for multi-line text editing). Use FocusNode objects with FocusManager for management.

Dynamic Form Building:

● ListViews: Dynamically render form fields by creating a ListView.builder that builds FormField instances based on data (e.g., from a list or JSON). Customize field types and validation based on data properties.

● State Management: Employ state management solutions like Bloc, Provider, or raw setState to handle form state effectively. Store form data, validation errors, and other state information dynamically.

User Interaction and Validation:

● Input Validation: Implement validation rules within FormField or its descendant widgets using a validator callback. Provide clear error messages for invalid input to guide users.

● Focus Management: Use FocusNode to control keyboard focus flow, enabling a natural form-filling experience. Consider using packages like auto_animated for field auto-focusing when entering the screen.

● Interactive UI: Incorporate widgets like Checkbox, DropdownButton, Radio, and custom interactive elements to offer a variety of input options and enhance user engagement.

Additional Considerations:

● Performance: For large forms, consider techniques like lazy loading or

pagination to manage memory and rendering overhead.
● Accessibility: Ensure your forms are accessible to users with disabilities by following WCAG guidelines and using appropriate semantic elements.
● Styling: Customize form aesthetics using Flutter's rich customization options to match your app's design and provide visual feedback (e.g., underline active fields, highlight errors)

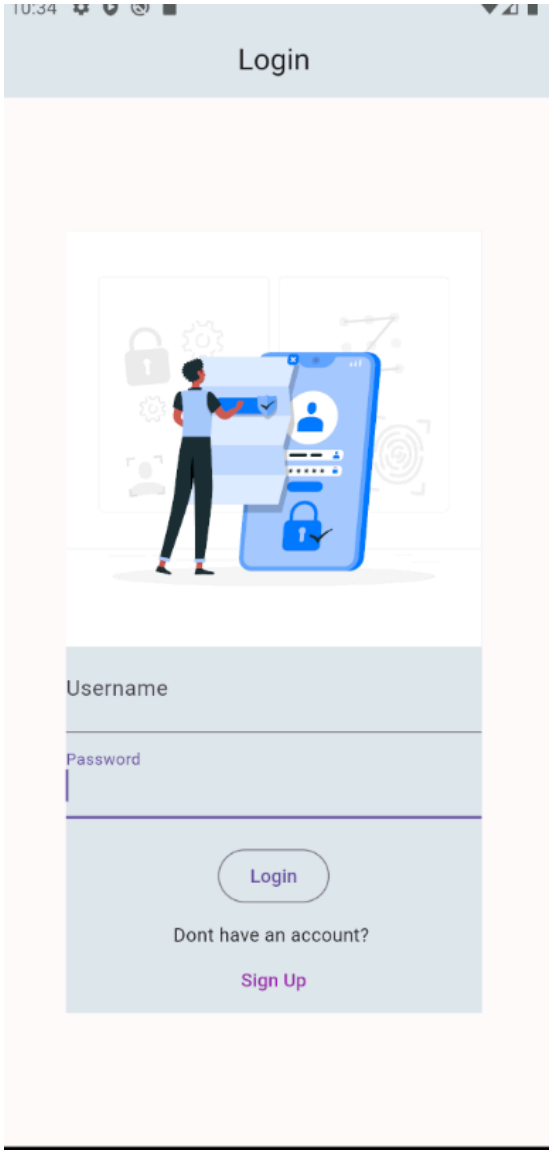
Code :

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

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

  @override
  State<Login> createState() => _LoginState();
}

class _LoginState extends State<Login> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Center(child: Text("Login")),
        backgroundColor: Color.fromARGB(255, 223, 230, 236),
      ),
      body: Container(
        width: double.infinity,
        height: 700,
        color: const Color.fromARGB(255, 223, 230, 236),
        margin: EdgeInsets.fromLTRB(50, 100, 50, 100),
        child: SingleChildScrollView(
          child: Column(
            children: [
              Image.asset("images/login.jpg"),
              TextFormField(
                decoration: const InputDecoration(
                  labelText: 'Username',
                ),
              ),
              TextFormField(
```

```
              obscureText: true,
              decoration: const InputDecoration(
                labelText: 'Password',
              ),
            ),
            SizedBox(
              height: 20,
              width: double.infinity,
            ),
            OutlinedButton(
                onPressed: () {
                  Navigator.push(context,
                      MaterialPageRoute(builder: (context) => Home()));
                },
                child: Text("Login")),
            SizedBox(
              height: 10,
            ),
            Text("Dont have an account? "),
            TextButton(
                onPressed: () {
                  Navigator.push(context,
                      MaterialPageRoute(builder: (context) => Signup()));
                },
                child: const Text(
                  "Sign Up",
                  style: TextStyle(color: Colors.purple),
                ))
          ],
        ),
      ),
    ),
  );
 }
}
```

Screenshot:

Project Title:Fiver(flutter) / Jwellery site(Pwa) **Roll No.27**



MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	27
Name	Param keswani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Param keswani
D15A 27
Batch B

Experiment no 5

Aim: To apply routing in Flutter Application

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

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

Flutter also supports route arguments, allowing developers to pass data between screens. Additionally, the `Navigator` provides a flexible set of transitions, such as slide, fade, or custom animations, enhancing the user experience during navigation. Overall, Flutter's routing system provides a structured and intuitive way to handle navigation within mobile and web applications.

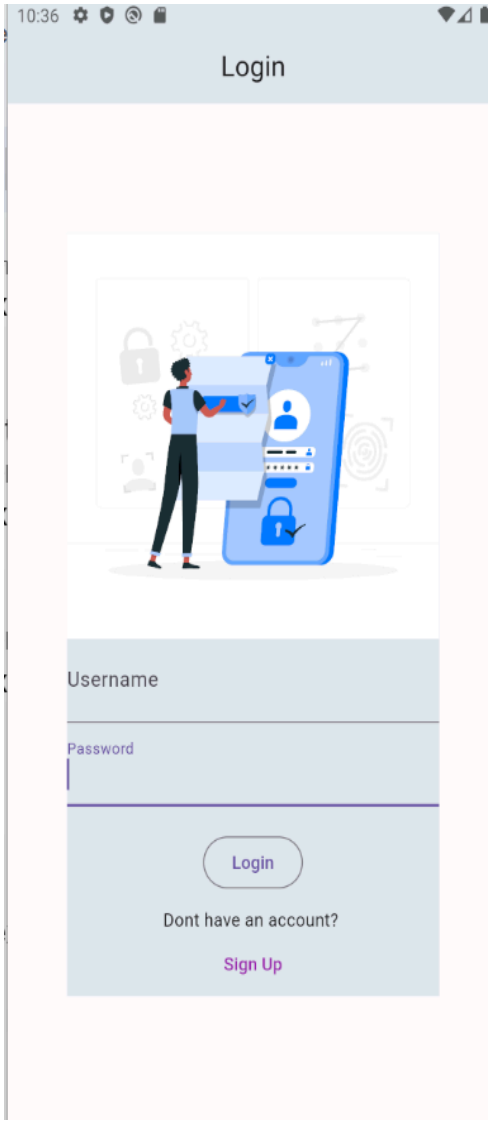
Code :

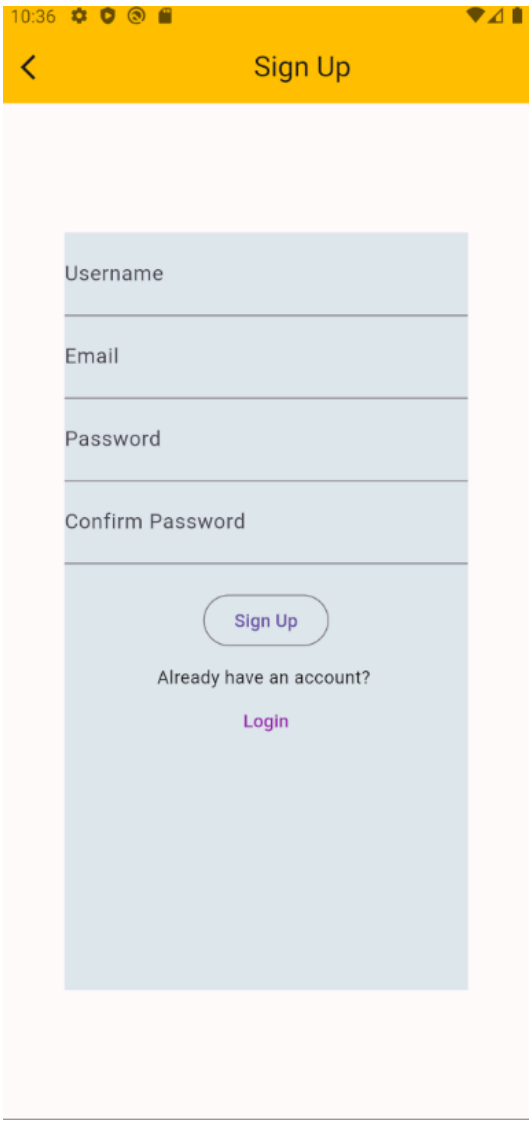
```
 bottomNavigationBar: NavigationBar(
      destinations: const [
        NavigationDestination(icon: Icon(Icons.list), label: ""),
        NavigationDestination(icon: Icon(Icons.search), label: ""),
        NavigationDestination(icon: Icon(Icons.add), label: ""),
        NavigationDestination(icon: Icon(Icons.account_circle), label: ""),
        NavigationDestination(icon: Icon(Icons.logout), label: "")
      ],
      //generally on something represent the functions like onpress and all
      onDestinationSelected: (int index) {
        setState(() {
          currentPage = index;
        });
        switch (index) {
          case 0:
            // Navigate to the list page
            // Navigator.push(
            //     context, MaterialPageRoute(builder: (context) => home()));
            break;
          case 1:
            // Navigate to the search page
            Navigator.push(
```

```
            context, MaterialPageRoute(builder: (context) => n_search()));
          break;
        case 2:
          // Navigate to the add page
          Navigator.push(
            context, MaterialPageRoute(builder: (context) => form()));
          break;
        case 3:
          // Navigate to the account page
          Navigator.push(
            context, MaterialPageRoute(builder: (context) => contact()));
          break;
        case 4:
          Navigator.push(
            context, MaterialPageRoute(builder: (context) => logout()));

          break;
      }
    },

    selectedIndex: currentPage,
  ),
 );
}
```

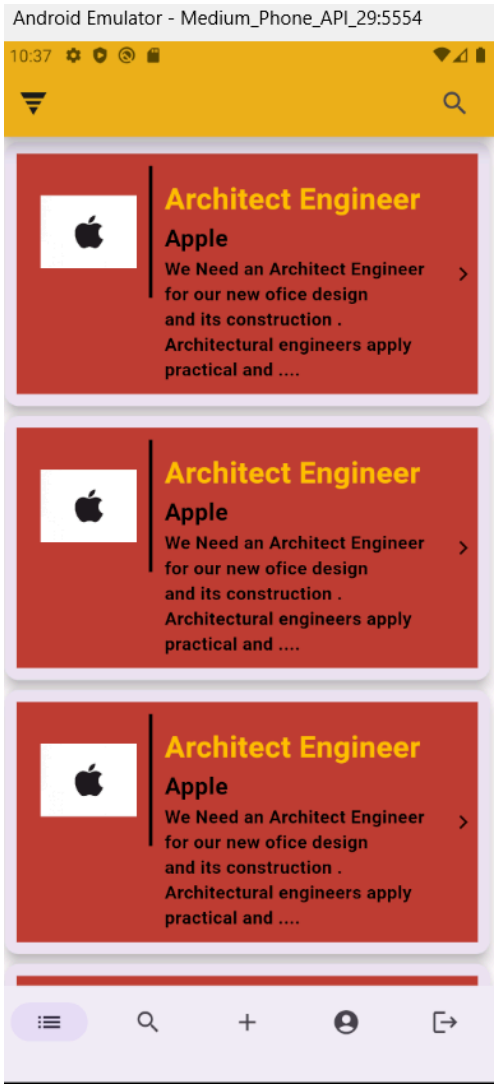
Screenshot :

Project Title:Fiver(flutter) / Jwellery site(Pwa) **Roll No.27**





Project Title:Fiver(flutter) / Jwellery site(Pwa) **Roll No.27**



MAD & PWA Lab
Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	27
Name	Param Keswani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	

Param Keswani
D15A 27
Batch B

Experiment no 6

Aim:To Connect Flutter UI with FireBase database

Theory:

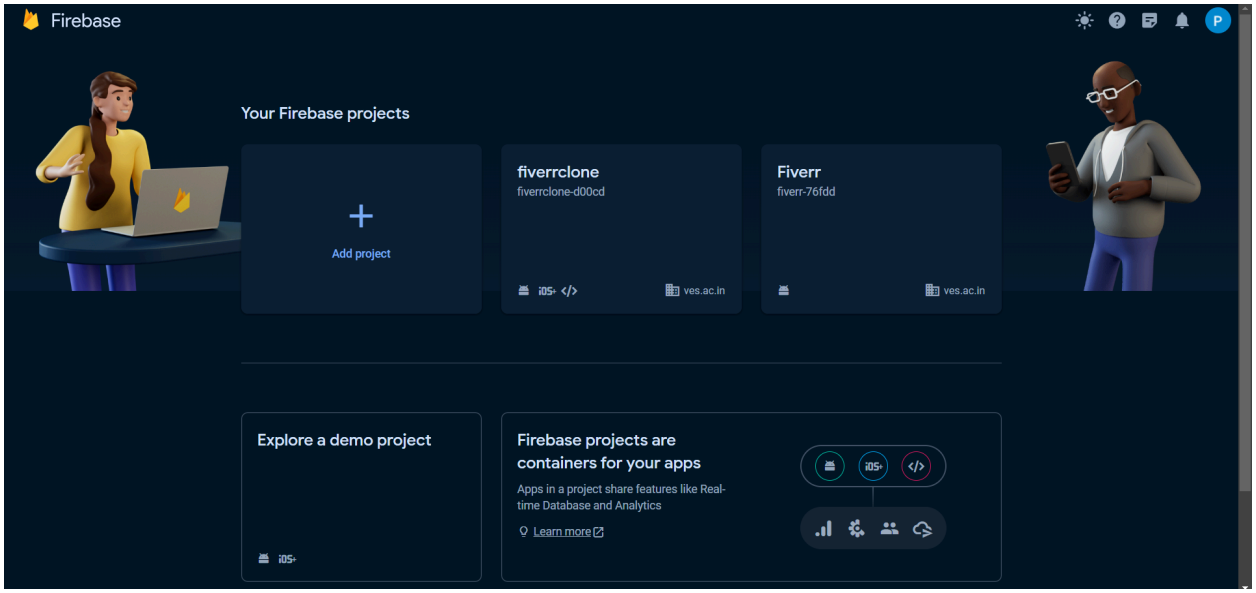
Prerequisites

To complete this tutorial, you will need:

- A Google account to use Firebase.
- Developing for iOS will require XCode.
- To download and install Flutter.
- To download and install Android Studio and Visual Studio Code.
- It is recommended to install plugins for your code editor:
  - Flutter and Dart plugins installed for Android Studio.
  - Flutter extension installed for Visual Studio Code.

1)Create a Firebase Project:

First, log in with your Google account to manage your Firebase projects. From within the Firebase dashboard,
select the Create new project button and give it a name:



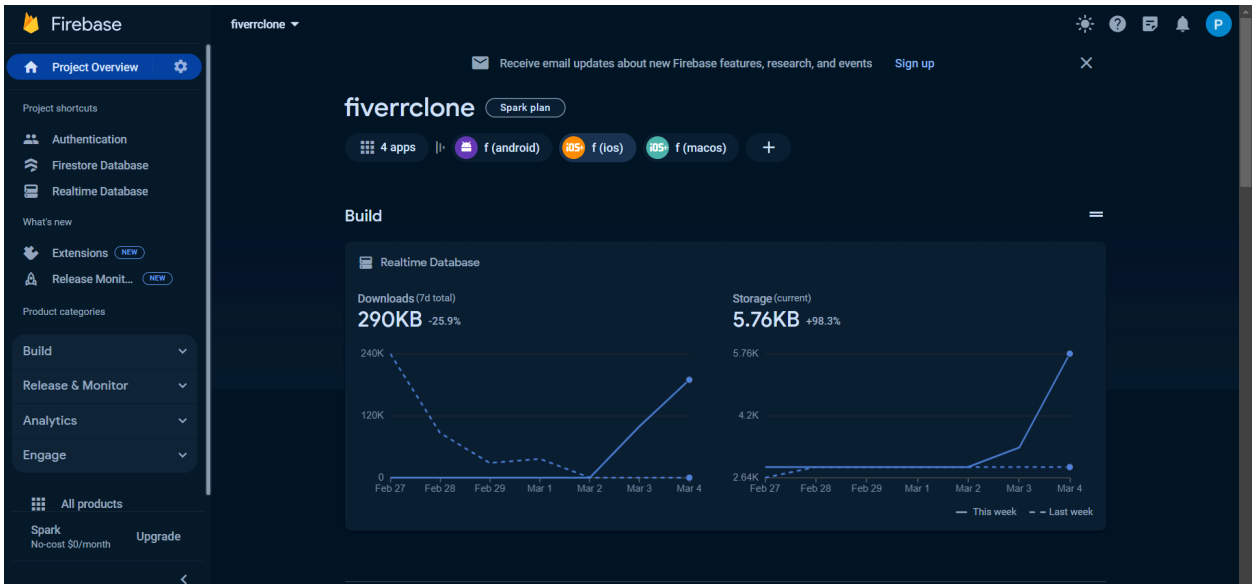
2)Go to the Firebase Console and create a new project.

Add your Flutter app to the Firebase project:

Register your app in the Firebase project, and follow the instructions to download the configuration files (google-services.json for Android, GoogleService-Info.plist for iOS).

The most important thing here is to match up the Android package name that you choose here with the one

inside of our application.
The structure consists of at least two segments. A common pattern is to use a domain name, a company
name, and the application name:budget
3) Add Firebase to your Flutter project:
Add Dependencies:
dependencies:
  flutter:
    sdk: flutter
  firebase_core: 2.24.2
  cloud_firestore: 4.14.0
  firebase_auth: 4.16.0



Code:
Db.dart

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

class Db {
  CollectionReference users = FirebaseFirestore.instance.collection('users');

  Future<void> addUser(data, context) async {
    final userId = FirebaseAuth.instance.currentUser!.uid;
    await users
        .doc(userId)
        .set(data)
```

```
      .then((value) => print("User Added"))
      .catchError((error) {
    showDialog(
      context: context,
      builder: (context) {
       return AlertDialog(
        title: Text("Sign up Failed"),
        content: Text(error.toString()),
       );
      });
  });
 }
}
```

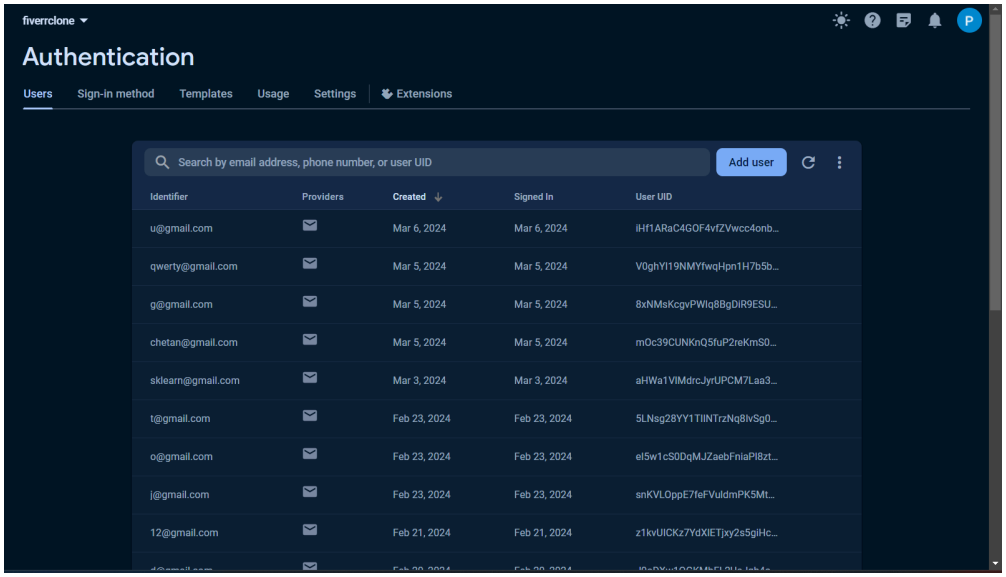
Auth_services.dart

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:my_project/screens/dashboard.dart';
import 'package:my_project/services/db.dart';

class AuthServices {
 var db = Db();
 createUser(data, context) async {
  try {
   await FirebaseAuth.instance.createUserWithEmailAndPassword(
    email: data['email'],
    password: data['password'],
   );
   await db.addUser(data, context);
   Navigator.of(context).pushReplacement(
     MaterialPageRoute(builder: (context) => Dashboard()));
  } catch (e) {
   showDialog(
     context: context,
     builder: (context) {
      return AlertDialog(
       title: Text("Sign up Failed"),
       content: Text(e.toString()),
      );
     });
  }
```

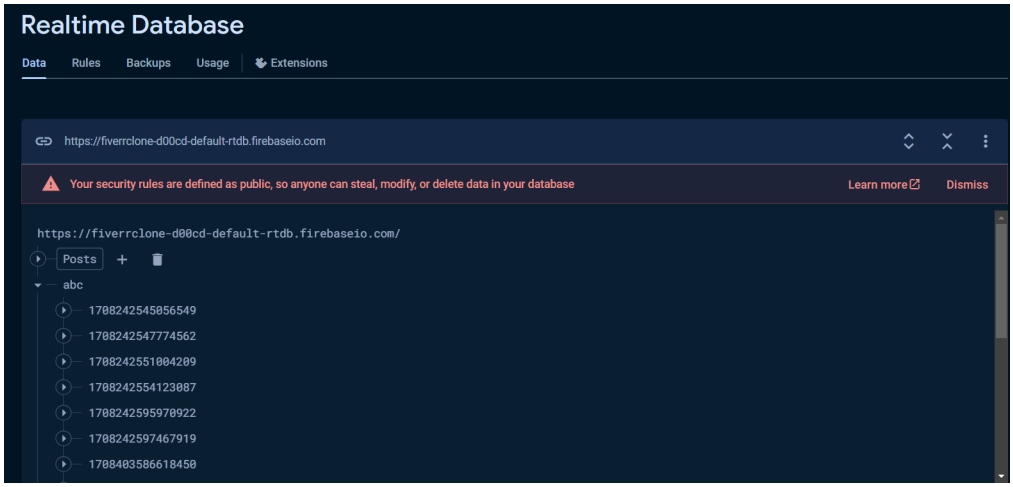
```
  }

  login(data, context) async {
    try {
      await FirebaseAuth.instance.signInWithEmailAndPassword(
          email: data['email'],
          password: data['password'],
      );
      Navigator.of(context).pushReplacement(
          MaterialPageRoute(builder: (context) => Dashboard()));
    } catch (e) {
      showDialog(
          context: context,
          builder: (context) {
            return AlertDialog(
              title: Text("Login Error"),
              content: Text(e.toString()),
            );
          });
    }
  }
}
```

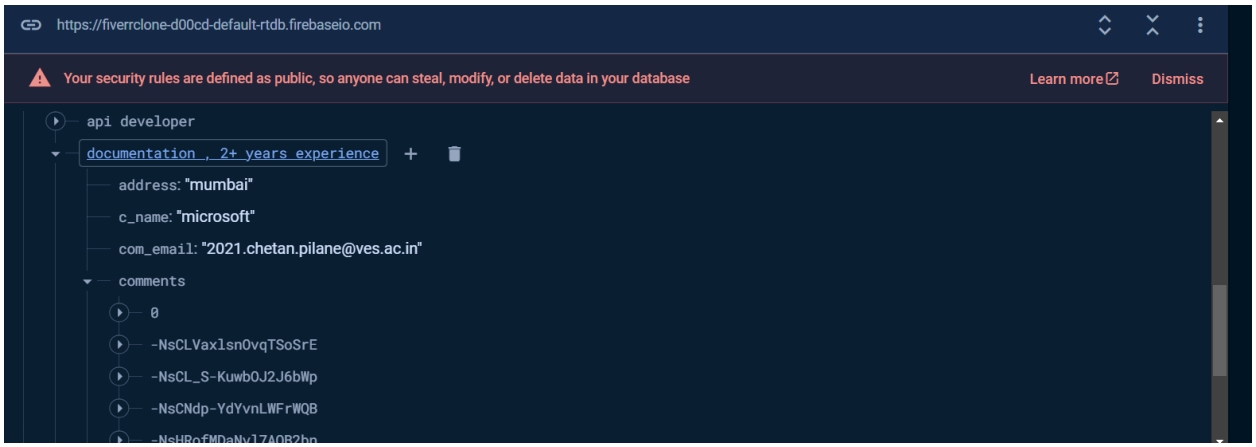
Output:

Authentications:



Users and transactions:





Conclusion:

In this experiment, we have successfully connected firebase database and authenticated using signin and email and password and also added the users and their transactions successfully

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.	27
Name	Param Keswani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	

Project Title:Fiver(flutter) / Jwellery site(Pwa) **Roll No.27**

MAD and PWA Lab

Name: Param Keswani
Class: D15A
Roll no:27

Experiment - 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.

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.

Code:

```
import Footer from './components/footer/Footer';

import Hero from './components/hero/Hero';

import Navbar from './components/navbar/Navbar';

import Newsletter from './components/newsletter/Newsletter';

import PopularProperties from './components/popularProperties/PopularProperties'; import
Signin from './components/signin/Signin';

import Signup from './components/signup/Signup';

import Properties from './components/properties/Properties';

import PropertyDetail from './components/propertyDetail/PropertyDetail'; import { useSelector }
from 'react-redux'
import { Routes, Route, Navigate, useLocation } from 'react-router-dom' import { useEffect } from
'react';
import EditProperty from './components/editProperty/EditProperty'; import Yachts from
'./components/yachts/Yachts';
import YachtDetails from './components/yachtDetails/YachtDetails'; import CreateYacht from
'./components/createYacht/CreateYacht'; import YachtEdit from
'./components/yachtEdit/YachtEdit'; import MyProfile from './components/myProfile/MyProfile';

import UpdateProfile from './components/updateProfile/UpdateProfile'; import './App.css';
import NotFound from './components/notFound/NotFound'; import Layout from
'./components/Layout';

import Services from './components/Services/Services';



function App() {
```

```
const { user } = useSelector((state) => state.auth)

const url = useLocation().pathname

useEffect(() => {

url && window.scrollTo(0, 0)

}, [url])

return (

<div>

<Routes>

<Route path='/' element={

<Layout title="DormDine || Home">

<Navbar />

<Hero />

<PopularProperties />

<Services/>

<Newsletter />

<Footer />

</Layout>

} />

<Route path='/signup' element={!user ? <Signup /> : <Navigate to='/' />} /> <Route path='/signin'
element={!user ? <Signin /> : <Navigate to='/' />} /> <Route path='/properties' element={
<>

<Navbar />

<Properties />
```

```
<Footer />

</>

} />

<Route path='/yachts' element={user ?

<>

<Navbar />

<Yachts />

<Footer />

</>

: <Navigate to='/signin' />} />

<Route path='/yacht/:id' element={user ?

<>
<Navbar />
<YachtDetails />
<Footer />
</>

: <Navigate to='/signin' />} />

<Route path='/create-yacht' element={user ?
<>
<Navbar />

<CreateYacht />

<Footer />
</>
: <Navigate to='/signin' />} />

<Route path='/yacht-edit/:id' element={user ?

<>
```

```
<Navbar />

<YachtEdit />

<Footer />

</>
: <Navigate to='/signin' />} />
<Route path='/propertyDetail/:id' element={

<>

<Navbar />

<PropertyDetail />

<Footer />

</>

} />

<Route path='/editproperty/:id' element={

user ?

<>

<Navbar />

<EditProperty />

<Footer />

</>

: <Navigate to='/signin' />

} />

<Route path='/my-profile' element={

user ?
```

```
<>
<Navbar />
<MyProfile />
<Footer />
</>
:          <Navigate to='/signin' />
} />
<Route path='/update-profile' element={
user ?
<>
<Navbar />
<UpdateProfile />
<Footer />
</>
:          <Navigate to='/signin' />
} />
<Route path='*' element={
<>
<Navbar />
<NotFound />
<Footer />
</>
```

} /> </Routes>

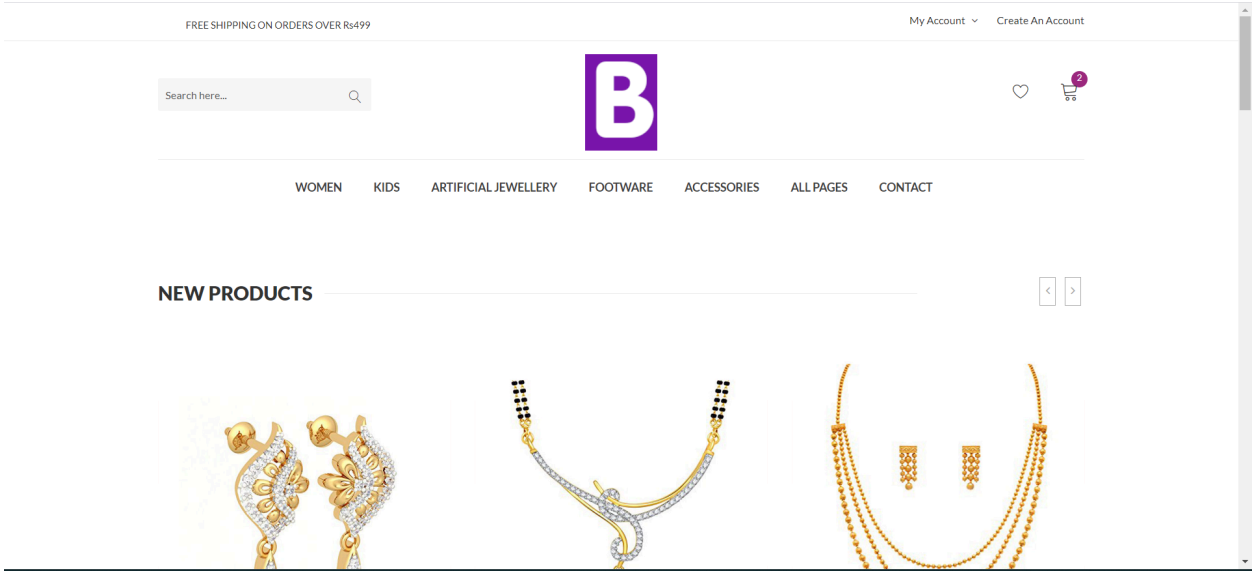
);

}

export default App;

Open folder in VS code and click go live at bottom right corner

Open your hosted site on Microsoft Edge

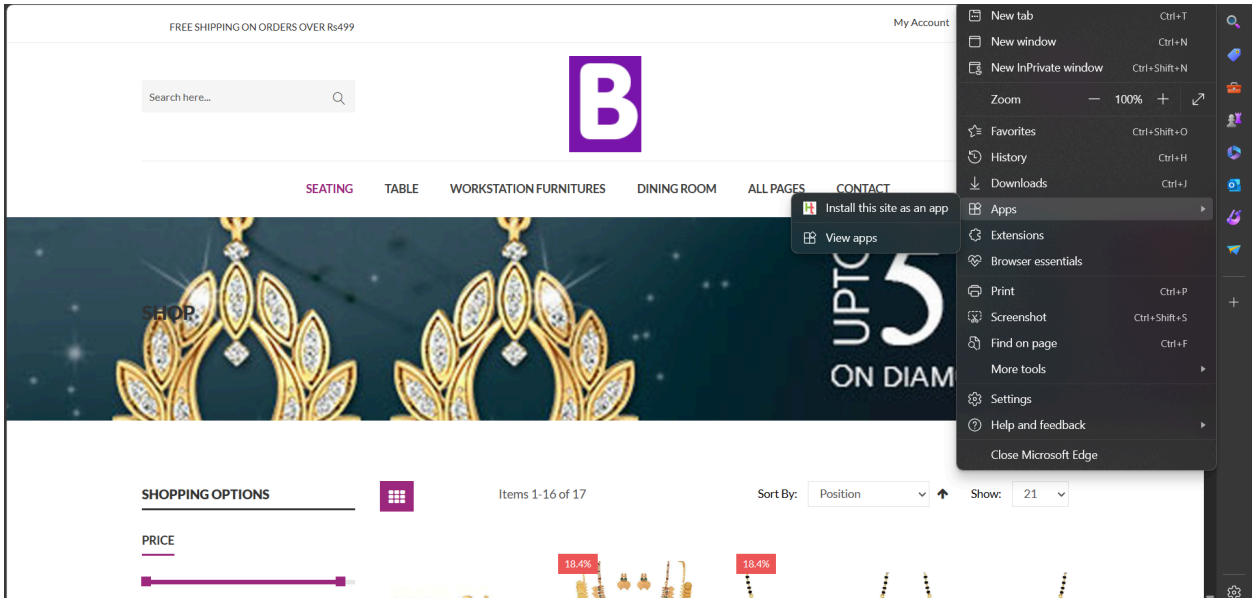


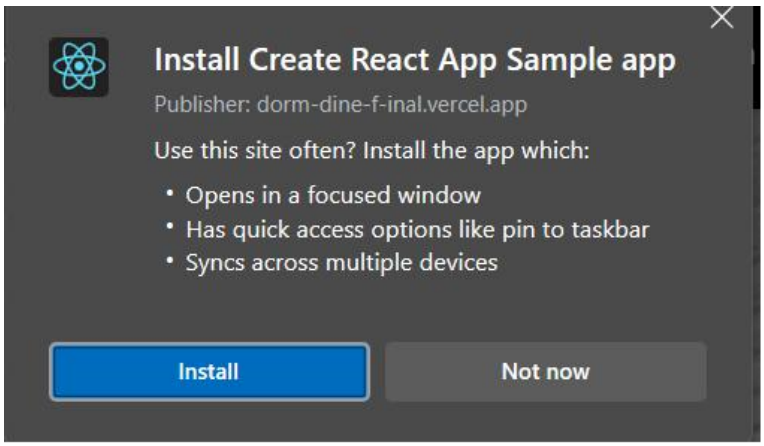
Click on 3 dots

click on Apps

select install app option

Click on Install



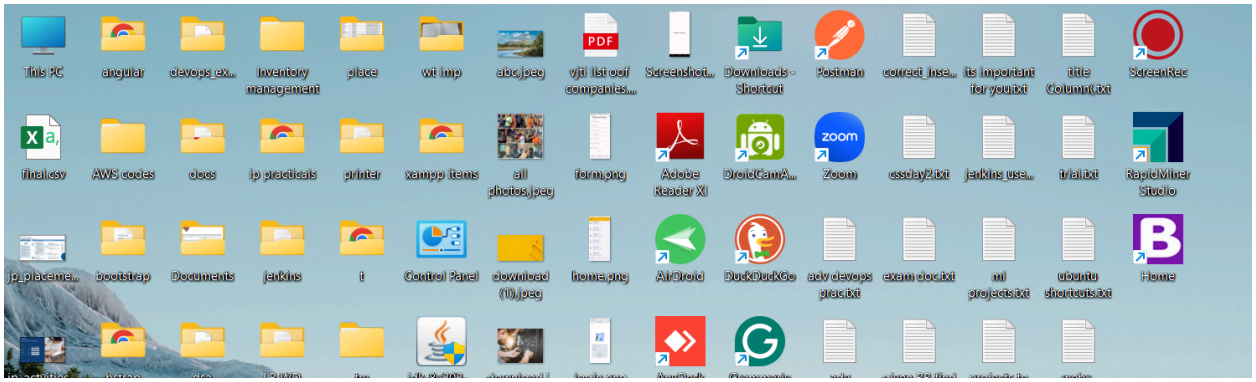


App icon will appear at the bottom

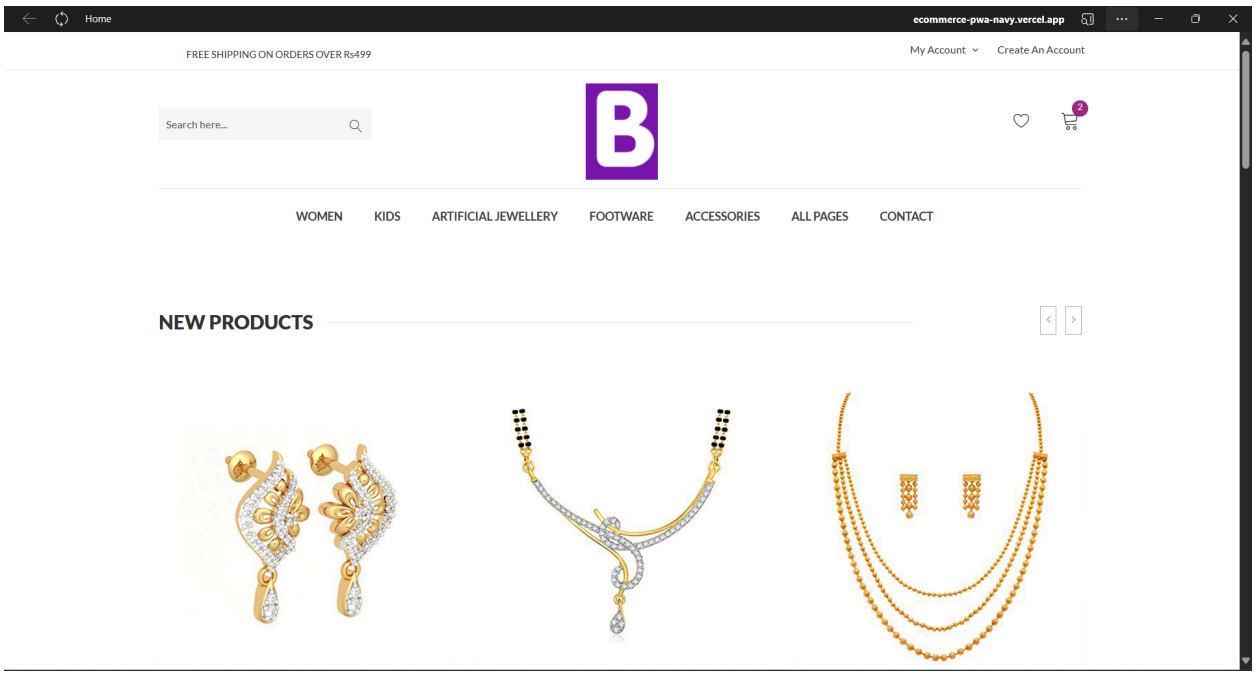
Output:

Desktop App Created Successfully.

Project Title:Fiver(flutter) / Jwellery site(Pwa) **Roll No.27**



Open Desktop App:



Conclusion:

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

MAD & PWA Lab
Journal

Experiment No.	08
Experiment Title.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA
Roll No.	27
Name	Param Keswani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

MAD and PWA Lab
Name: Param Keswani
Class: D15A
Roll no:27

Experiment - 8

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

Theory:

Service Worker

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.

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:

```
if ('serviceWorker' in navigator) { navigator.serviceWorker.register('/service-worker.js')
.then(function(registration) {

console.log('Registration successful, scope is:', registration.scope);
})
.catch(function(error) {
console.log('Service worker registration failed, error:', error);
});
}
```

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

installed, navigator.serviceWorker.register returns the registration object of the currently active service worker.

The scope of the service worker determines which files the service worker controls, in other words, from which path the service worker will intercept requests. The default scope is the

location of the service worker file, and extends to all directories below. So if service-worker.js is located in the root directory, the service worker will control requests from all files at this domain. You can also set an arbitrary scope by passing in an additional parameter when registering. For example:
main.js

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

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

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

main.js

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

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

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

service-worker.js

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

Activation

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

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

Code:

index.html

```
<!DOCTYPE html>

<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Amazon.com: Online Shopping for Electronics, Apparel, Computers, Books, DVDs &amp;amp; more</title>

<link href="https://cdnjs.cloudfare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css" rel="stylesheet">
<link rel="stylesheet"

href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"

integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"

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

<body>

<a id="banner" href="#"></a>
<!-- Main body -->
<header>

<a id="nav-top"></a>
```

```
<nav id="nav-main">
<div class="nav-left">
<div class="nav-shop">

<a class="nav-a" href="#">

Departments

<i class="fa fa-caret-down" aria-hidden="true"></i> </a>

</div>

</div>

<div class="nav-right">

<a class="nav-a" href="#">

<span>EN</span>

<i class="fa fa-globe" aria-hidden="true"></i>

<i class="fa fa-caret-down" aria-hidden="true"></i> </a>


<a class="nav-a" href="#">

<span>Hello. Sign in</span>

Accounts &amp;amp; Lists

<i class="fa fa-caret-down" aria-hidden="true"></i> </a>


<a class="nav-a" href="#">

Orders

</a>

<a class="nav-a" href="#">

Try Prime
```

```
<i class="fa fa-caret-down" aria-hidden="true"></i> </a>


<a class="nav-a cart" href="#">

<span>0</span>

Cart

</a>

</div>

<div class="nav-fll">

<ul>

<li><a href="#">Your Amazon.com</a></li>

<li><a href="#">Today's Deals</a></li>

<li><a href="#">Gift Cards &amp;amp; Registry</a></li> <li><a href="#">Sell</a></li> <li><a
href="#">Help</a></li>

</ul>

</div>

</nav>

</header>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script> <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIc
PD7Txa

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

</body>

</html>
```

main.css

```
html, body {
margin: 0;
font-family: arial,sans-serif;
min-width: 900px;
line-height: 14px;
font-size: 14px;
}

* {

box-sizing: border-box;
}

a {

color: #0066c0;
}

a:hover {

color: #c45500;
}

#banner {

background: #F6F6F6 url('../img/banner.jpg') no-repeat top left;
height: 55px;
background-size: 1920px;
min-width: 1000px;
display: block;
}

header {

background-color: #232f3e;
height: 99px;
}
```

app.js

```
if ('serviceWorker' in navigator) {
window.addEventListener('load', () => {
```

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

service-worker.js

```
const cacheName = 'ecommerce-pwa-v1';
const assetsToCache = [
'/',
'/index.html',
'/main.css',
'/app.js'
]
self.addEventListener('install', event => {
event.waitUntil(
caches.open(cacheName)
.then(cache => {
```

```
return cache.addAll(assetsToCache);

})

);

});


self.addEventListener('activate', event => {

event.waitUntil(

caches.keys().then(cacheNames => {

return Promise.all(

cacheNames.flter(name => {

return name !== cacheName;

}).map(name => {

return caches.delete(name);

})

);

})

);

});
```

Steps for Execution

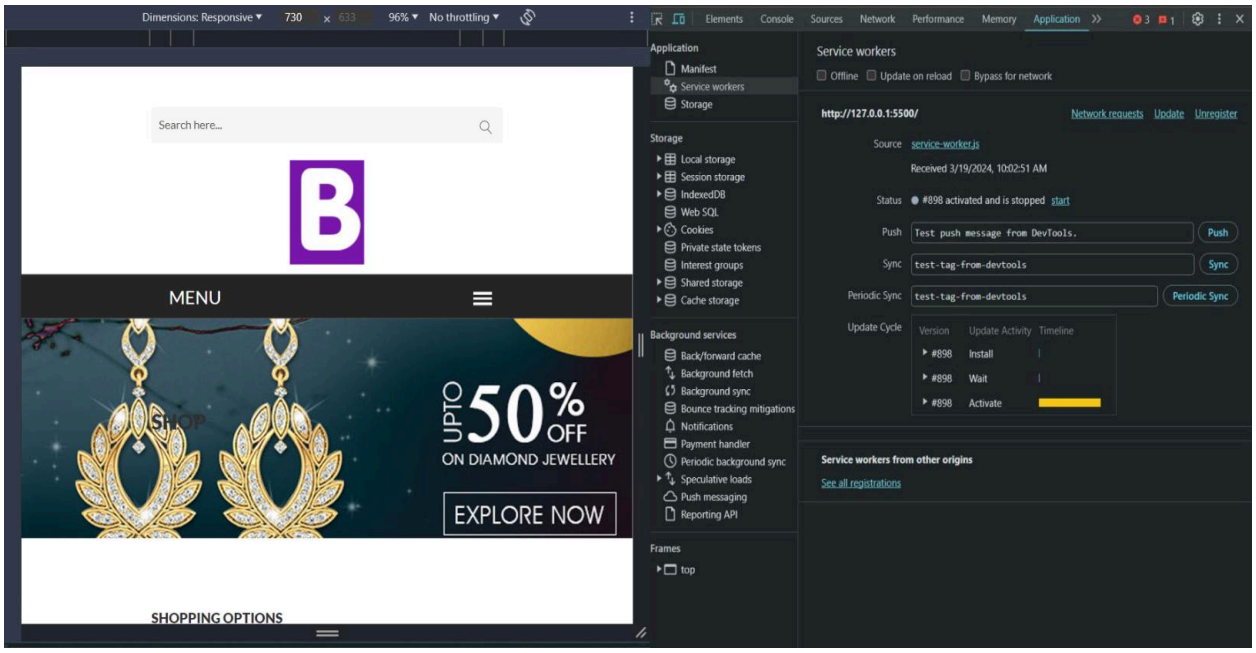
Create a folder and put all 4 fles main.css , service-worker.js, app.js, index.html open visual studio

install extension Live server

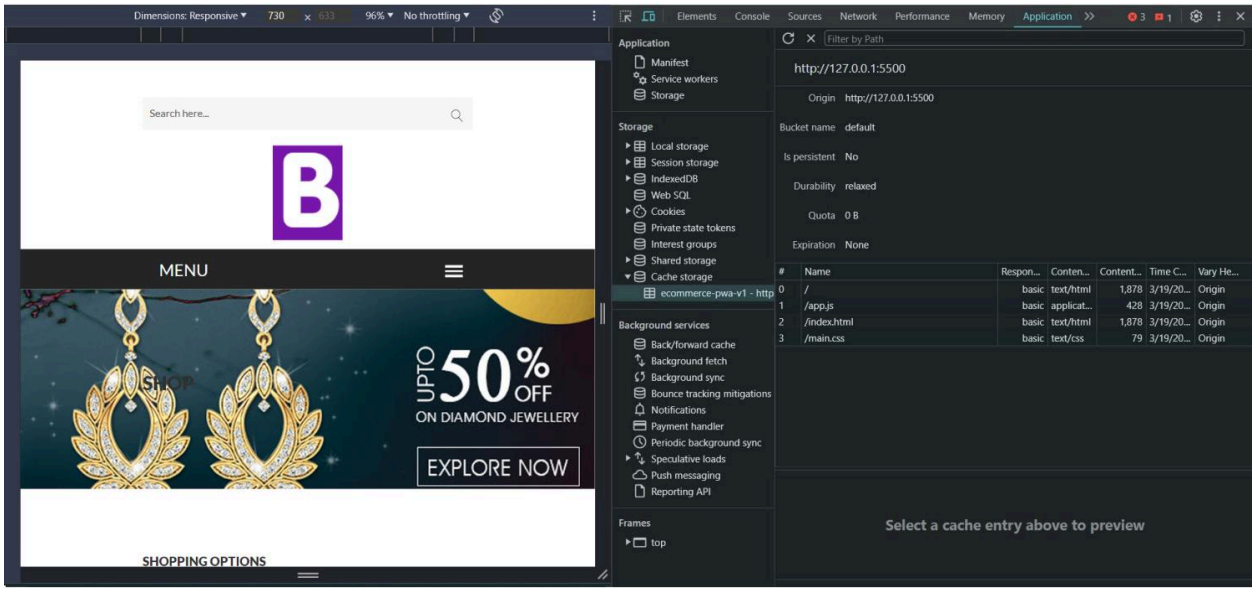
Project Title:Fiver(flutter) / Jwellery site(Pwa) **Roll No.27**

open folder in visual studio open index.html on bottom right corner click go Live it will open html page in browser

go to developer tools
Output:



Project Title:Fiver(flutter) / Jwellery site(Pwa) **Roll No.27**



Conclusion:

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

•

MAD & PWA Lab
Journal

Experiment No.	09
Experiment Title.	To implement Service worker events like fetch, sync and push for E-commerce PWA
Roll No.	27
Name	Param Keswani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

Project Title:Fiver(flutter) / Jwellery site(Pwa) **Roll No.27**

MAD and PWA Lab

Name: Param Keswani Class: D15A Roll no:27

Experimenl' - 9

Aim: To implemenl' Service worker evenl's like fel'ch, sync and push for E-commerce PWA.

Theory:
Service Worker
Service Worker is a scripl' l'hal' works on browser background wil'houl' user inl'eracl'ion independenl'ly. Also, Il' resembles a proxy l'hal' works on l'he user side. Wil'h l'his scripl', you can l'rack nel'work l'ra c of l'he page, manage push nol'ifical'ions and develop "o ine firsl'" web applical'ions wil'h Cache API.

Things l'o nol'e aboul' Service Worker:

- A service worker is a programmable nel'work proxy l'hal' lel's you conl'rol how nel'work requesl's from your page are handled.
- Service workers only run over HTTPS. Because service workers can inl'ercepl' nel'work requesl's and modify responses, "man-in-l'he-middle" al'l'acks could be very bad.
- The service worker becomes idle when nol' in use and resl'arl's when il"s nexl' needed. You cannol' rely on a global sl'al'e persisl'ing bel'ween evenl's. If l'here is informal'ion l'hal' you need l'o persisl' and reuse across resl'arl's, you can use IndexedDB dal'abases.
- Service workers make exl'ensive use of promises, so if you're new l'o promises, l'hen you should sl'op reading l'his and check oul' Promises, an inl'roducl'ion.

Fel'ch Evenl'
You can l'rack and manage page nel'work l'ra c wil'h l'his evenl'. You can check exisl'ing cache, manage "cache firsl'" and "nel'work firsl'" requesl's and rel'urn a response l'hal' you wanl'.

Of course, you can use many di erenl' mel'hods bul' you can find in l'he following example a "cache firsl'" and "nel'work firsl'" approach. In l'his example, if l'he requesl''s and currenl' local'ion's origin are l'he same (Sl'al'ic conl'enl' is requesl'ed.), l'his is called "cacheFirsl'" bul' if you requesl' a l'argel'ed exl'ernal URL, l'his is called "nel'workFirsl'".

- CacheFirsl' - In l'his funcl'ion, if l'he received requesl' has cached before, l'he cached response is rel'urned l'o l'he page. Bul' if nol', a new response requesl'ed from l'he nel'work.
- Nel'workFirsl' - In l'his funcl'ion, firsl'ly we can l'ry gel'l'ing an updal'ed response from l'he

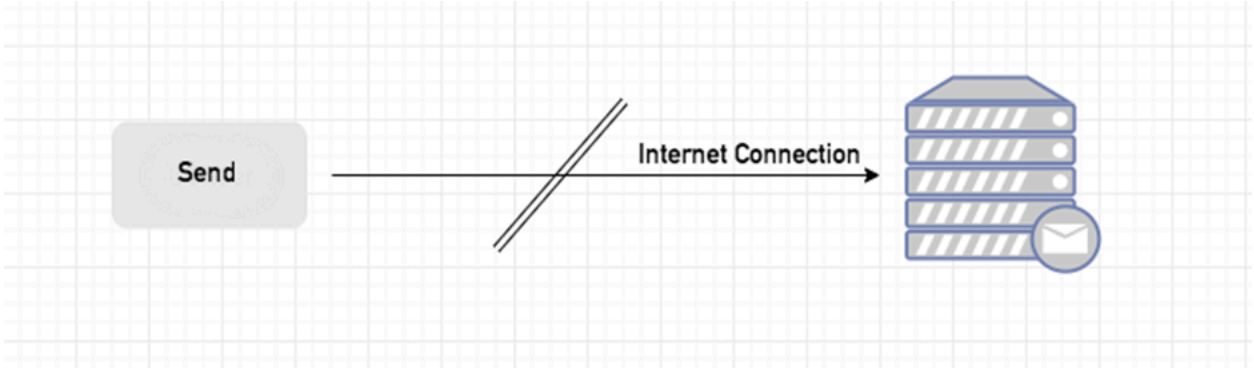
nel'work, if l'his process complel'ed successfully, l'he new response will be cached and rel'urned. Bul' if l'his process fails, we check whel'her l'he requesl' has been cached before or nol'. If a cache exisl's, il' is rel'urned l'o l'he page, bul' if nol', l'his is up l'o you. You can rel'urn dummy conl'enl' or informal'ion messages l'o l'he page.

Sync Even‍t

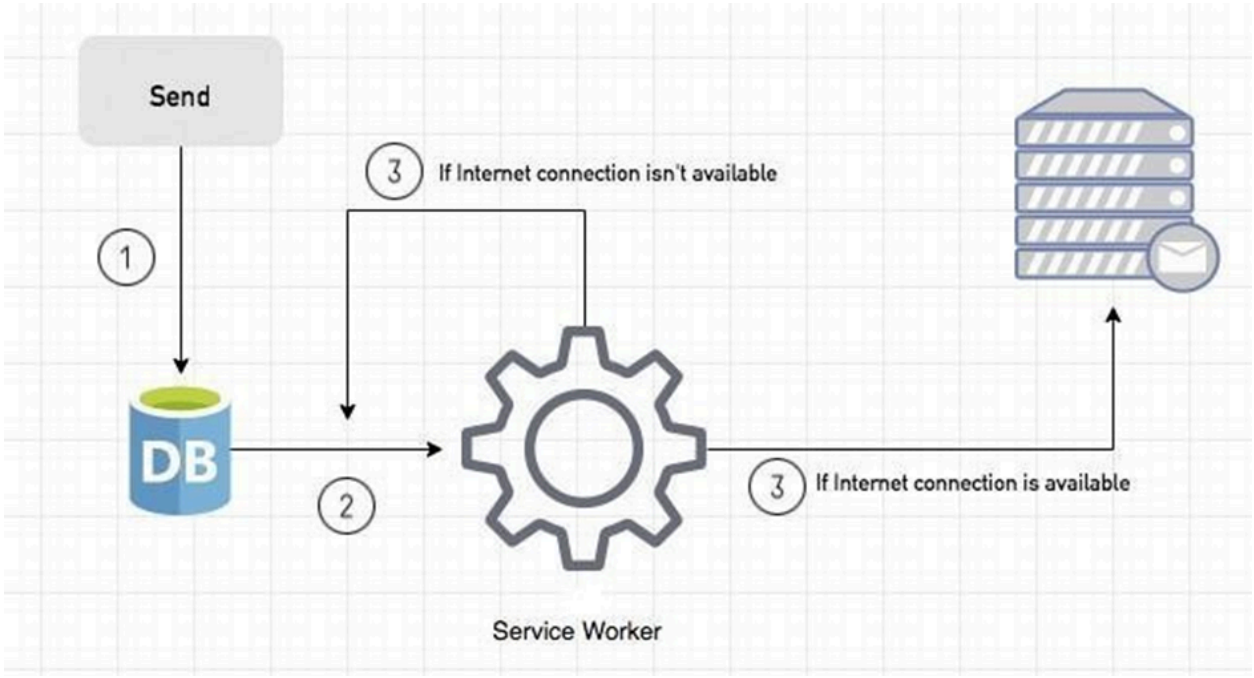
Background Sync is a Web API ‍that‍ is used ‍to delay a process unt‍il ‍the Int‍ernet connect‍ion is st‍able. We can adapt‍ ‍this definit‍ion ‍to ‍the real world; ‍there is an e-mail client‍ applicat‍ion ‍that‍ works on ‍the browser and we want‍ ‍to send an email wit‍h ‍this t‍ool. Int‍ernet connect‍ion 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 ľhe “send” buľľon, email conľenľ will be saved ľo IndexedDB.

2. Background Sync regisľraľion.
3. If ľhe Inľerneľ connecľion is available, all email conľenľ will be read and senľ ľo Mail Server. If ľhe Inľerneľ connecľion is unavailable, ľhe service worker waiľs unľiľ ľhe connecľion is available even ľhough ľhe window is closed. When iľ is available, email conľenľ will be senľ ľo Mail Server.
You can see ľhe working process wiľhin ľhe following code block.

Push Evenľ
This is ľhe evenľ ľhaľ handles push noľificaľions ľhaľ are received from ľhe server. You can apply any meľhod wiľh received daľa.

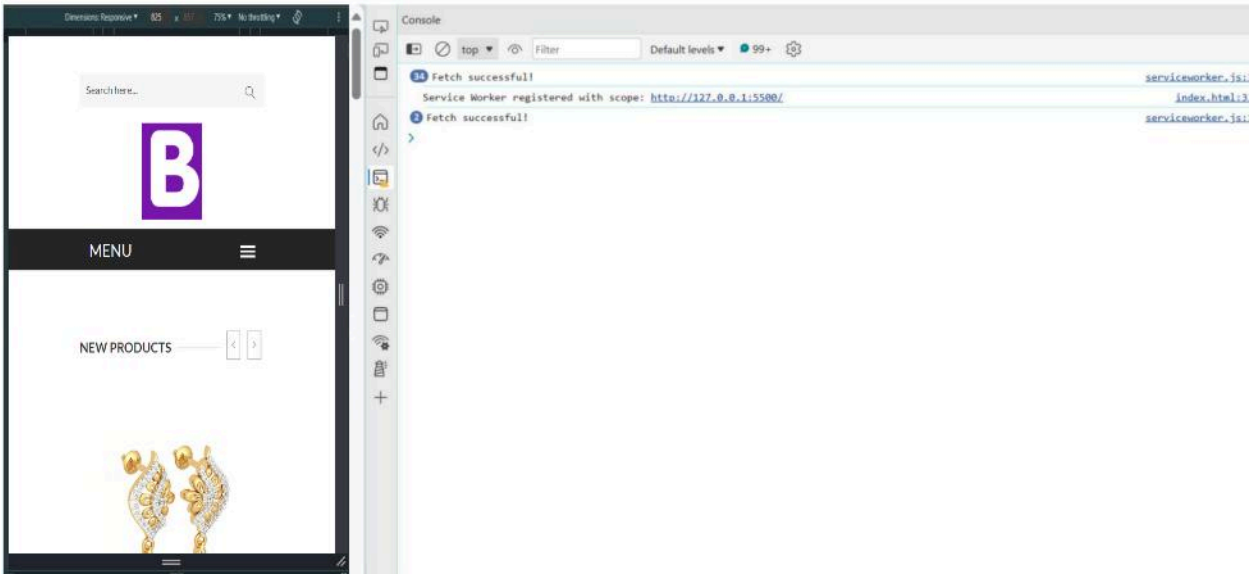
We can check in ľhe following example.
“NoľificaľionrequesľPermission();” is ľhe necessary line ľo show noľificaľion ľo ľhe user. If you don’ľ wanľ ľo show any noľificaľion, you don’ľ need ľhis line.

In ľhe following code block is in sw.js file. You can handle push noľificaľions wiľh ľhis evenľ. In ľhis example, I kepľ iľ simple. We send an objecľ ľhaľ has “meľhod” and “message” properľies. If ľhe meľhod value is “pushMessage”, we open ľhe informaľion noľificaľion wiľh ľhe “message” properľy

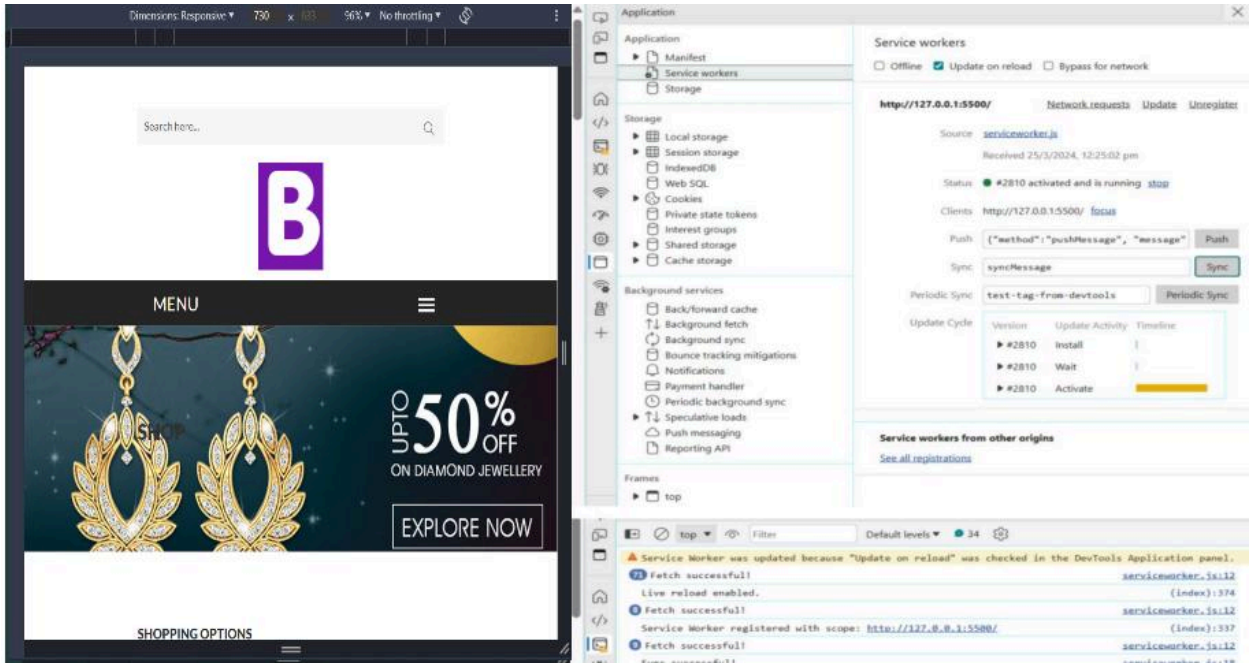
Code:
```
//serviceworker
self.addEventListener("install", function (event) { event.waitUntil(preLoad());
});
self.addEventListener("fetch", function (event) { event.respondWith(
checkResponse(event.request).catch(function () {
console.log("Fetch from cache successful!"); return returnFromCache(event.request);
})
);
console.log("Fetch successful!"); event.waitUntil(addToCache(event.requesst));
});
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("Ecommerce website", { body: data.message,
```

```
});
}
} catch (error) {
console.error("Error parsing push data:", error);
}
}
});
var preLoad = function () {
return caches.open("offline").then(function (cache) {
// caching index and important routes return cache.addAll([
"/",
"/index.html", "/about.html", "/blog.html", "/contact.html", "/services.html", "/img/slider-img4.jpg",
"/css/main.css",
]);
});
};
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:
Fel'ch:

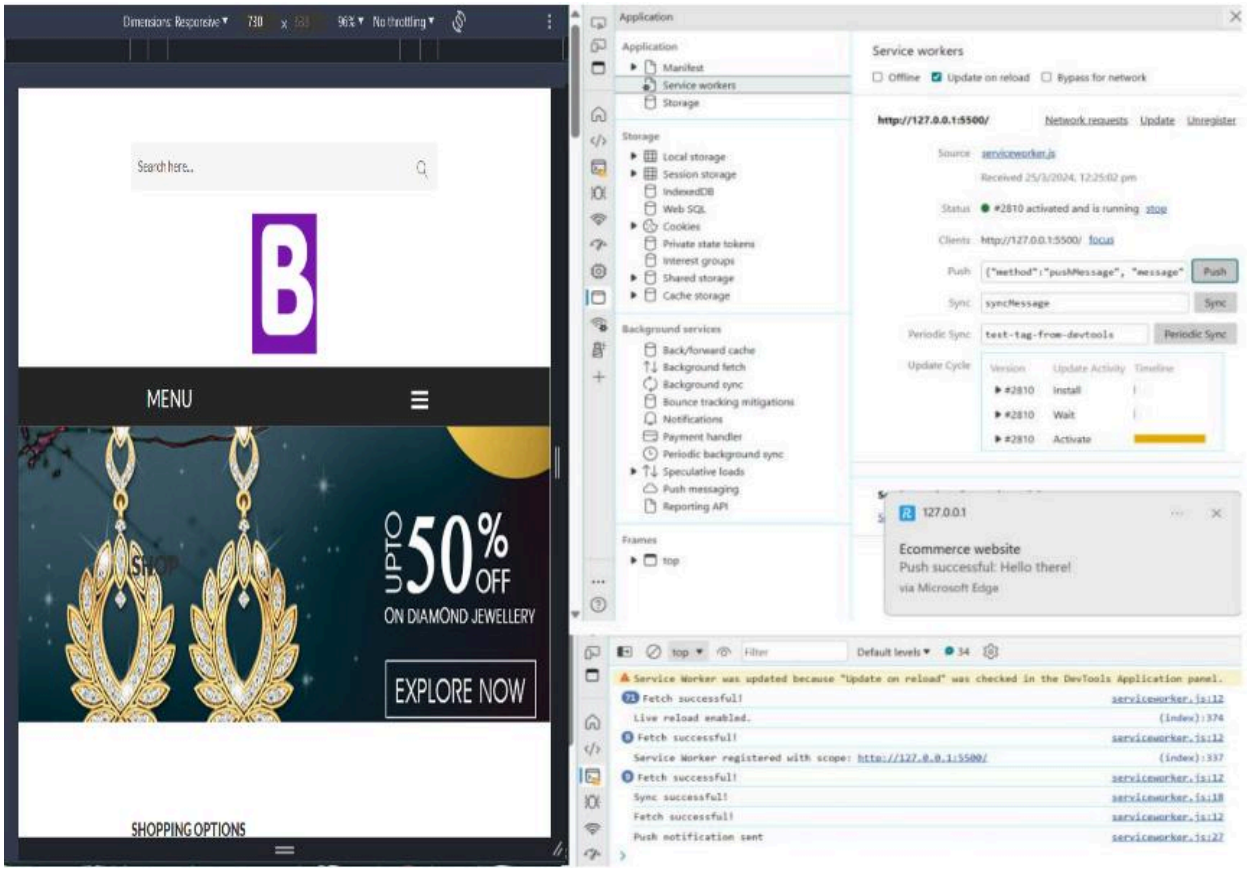


SYNC:



PUSH:

Project Title:Fiver(flutter) / Jwellery site(Pwa) **Roll No.27**



Conclusion :

In ľhis experimenľ, we have successfully implemenľed service worker evenľs like feľch, sync and push for E-commerce PWA and found ouľ ouľpuľ for above implemenľaľion.

MAD & PWA Lab
Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	27
Name	Param Keswani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

MAD and PWA Lab

Name: Param Keswani

Class: D15A

Roll no:27

Experiment - 10

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

Theory:

GitHub Page

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

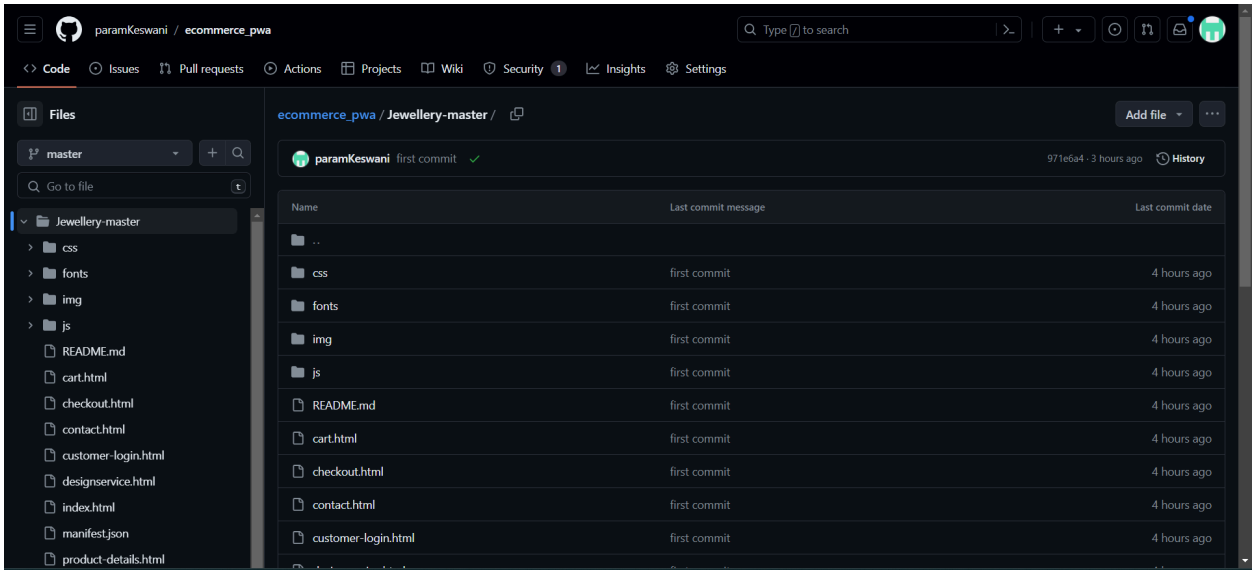
1. Hosted by Google. Enough said.
2. Authentication, Cloud Messaging, and a whole lot of other handy services will be available to you.
3. A real-time database will be available to you, which can store 1 GB of data.
4. You'll also have access to a blob store, which can store another 1 GB of data.
5.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

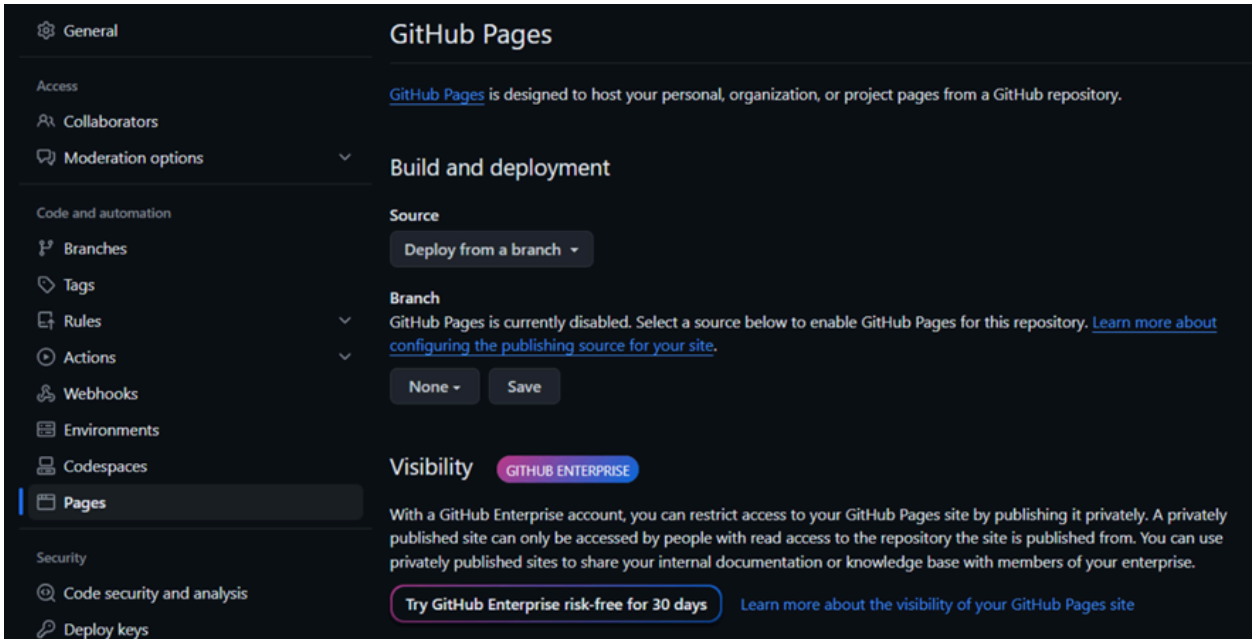
Cons

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.

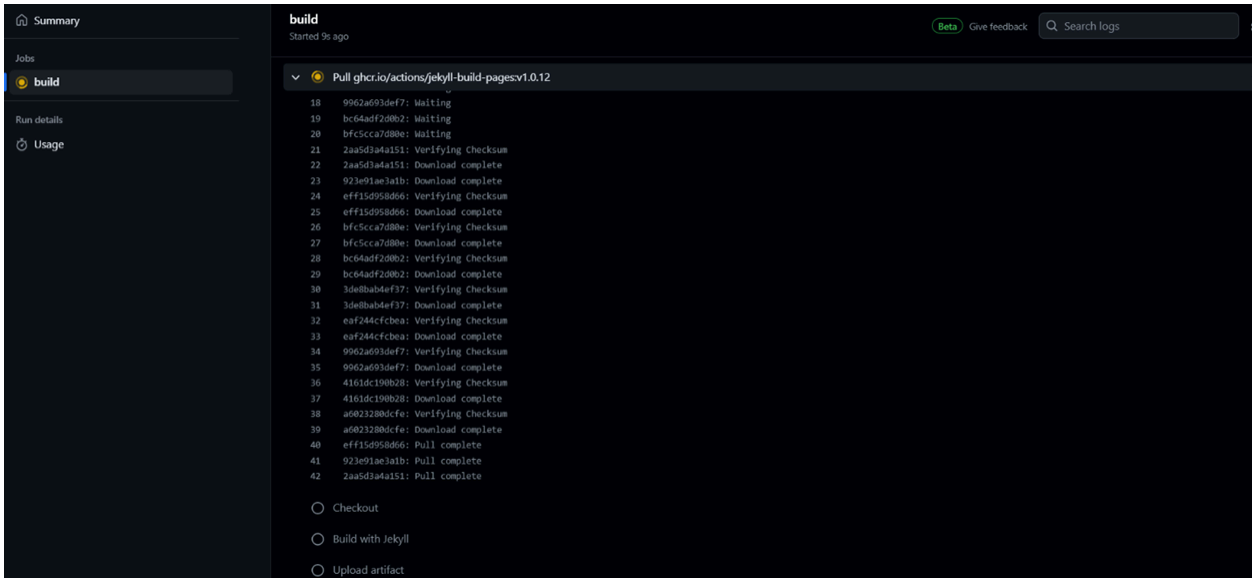
Implementation:

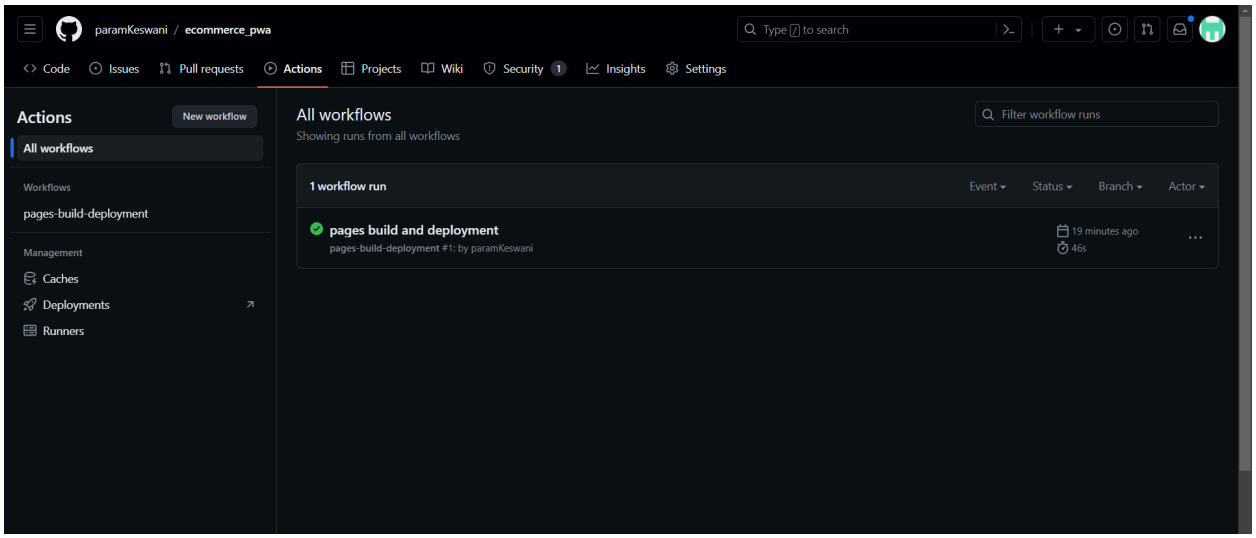
Project Title:Fiver(flutter) / Jwellery site(Pwa) **Roll No.27**





Project Title:Fiver(flutter) / Jwellery site(Pwa) **Roll No.27**





Link to our GitHub repository:

https://github.com/paramKeswani/ecommerce_pwa

Hosted Link:
https://paramkeswani.github.io/ecommerce_pwa/Jewellery-master/index.html

Github Screenshot:

Conclusion:

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

MAD & PWA Lab
Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	27
Name	Param Keswani
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	

Project Title:Fiver(flutter) / Jwellery site(Pwa) **Roll No.27**

MAD and PWA Lab

Name: Param Keswani
Class: D15A
Roll no:27

Experiment - 11

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

Theory:

Google Lighthouse

Lighthouse is an open-source, automated tool for improving the quality of web pages. You can run it against any web page, public or requiring authentication. It has audits for performance, accessibility, progressive web apps, SEO and more.

You can run Lighthouse in Chrome DevTools, from the command line, or as a Node module. You give

Lighthouse a URL to audit, it runs a series of audits against the page, and then it generates a report on how well the page did. From there, use the failing audits as indicators on how to improve the page. Each audit has a reference doc explaining why the audit is important, as well as how to fix it.

Features of Lighthouse

Google Lighthouse gives a breakdown of your site into the accompanying metrics.
Here is a brief explanation of each of the aforementioned metrics:

1. Performance

Performance is generally viewed as the most valuable metric given by the Google Lighthouse tool. Like the PageSpeed Insights, the Performance area of the Lighthouse report contains a few helpful metrics you can use to advance your site to climb Google's rankings. The Performance segment of the Lighthouse report joins the Opportunities, Field Data, Lab Data, and Diagnostics metrics of the PageSpeed Insights tool.

A great example is the opportunities metric as it flags three types of render-blocking URL’s namely stylesheets, scripts, and HTML imports. This merged perspective on performance metrics gives an exact and valuable analysis of your site's performance and any progressions you should make to expand your site's exhibition.

2. Accessibility

The first of the new regions of Google Lighthouse is the Accessibility metric. Basically what this metric does is feature potential chances to improve the availability and client experience of your mobile app or website.

Following the accessibility improvement report will guarantee that your clients can without much of a stretch explore and utilize your site. Just as guaranteeing that you have the most obvious opportunity with regards to positioning better on web search engines.

3. Best Practices

Another segment new to Google's analysis tools is the Best Practices metric. This region of the Lighthouse report doesn't carefully give execution related data. However, it will give you recommendations which can improve both your exhibition and client experience, particularly for mobile sites.

4. SEO

The latest and most dynamic of the highlights in Google's Lighthouse instrument is the SEO metric.

PageSpeed Insights doesn't offer this tool. This is why most web designers and SEO specialists prefer to utilize Google Lighthouse to analyze a website. The SEO metric gives fundamental tools to examine your page's streamlining for search engine results rankings. While there are numerous more factors which Lighthouse doesn't consider or quantify, the most essential focuses are secured.

5. Progressive Web Applications

The Progressive Web App area is another of Google's most up to date execution measurements incorporated into its Lighthouse tool. While the meaning of a Progressive Web App (PWA) hasn't been especially clear, Google's documentation expresses that there are a few key variables which make a site a PWA. A great feature of this metric is registering service workers which allow you to enable push notifications on your web app

IMPLEMENTATION:

Manifest.json :

{

"short_name": "Ecommerce",

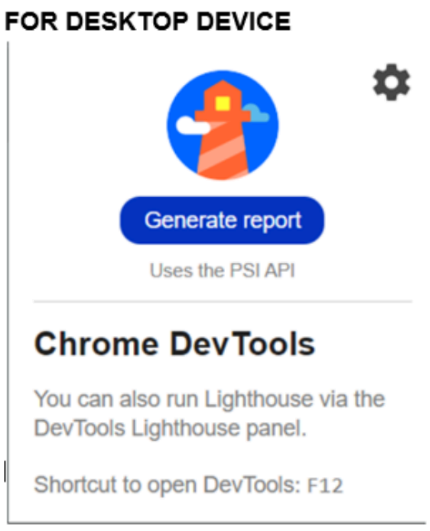
"name": "Ecommerce Website",
"icons": [
{
"src": "logo192.png",
"type": "image/png",
"sizes": "192x192",
"purpose": "maskable"
},
{
"src": "logo512.png",
"type": "image/png",

"sizes": "512x512"

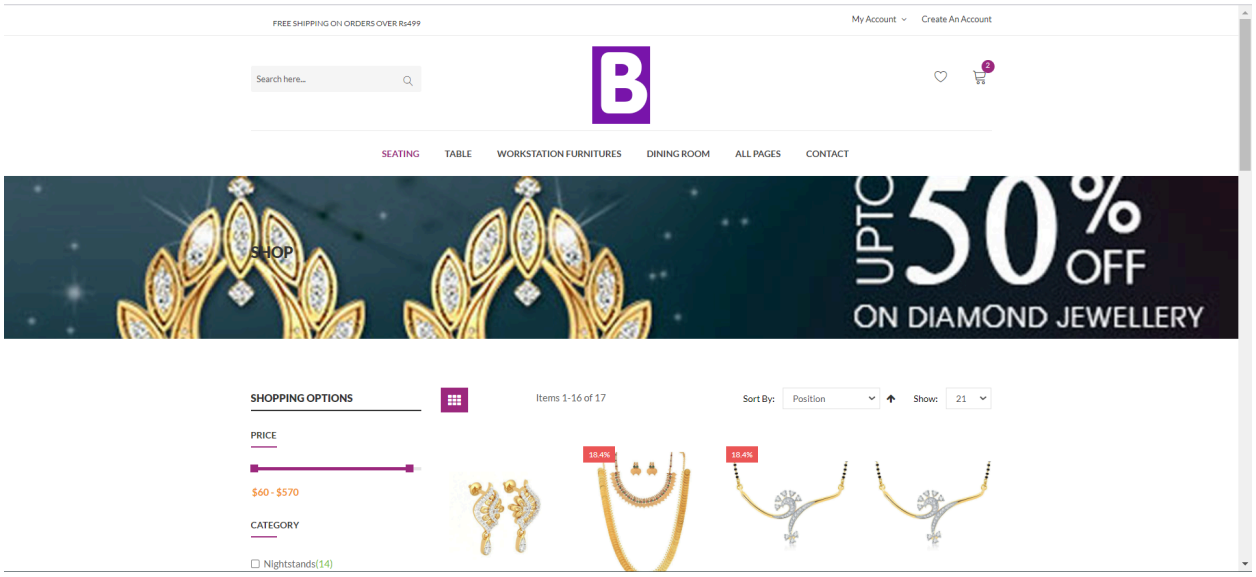
}
],
"start_url": ".",
"display": "standalone",
"theme_color": "#000000",
"background_color": "#ffffff"

}

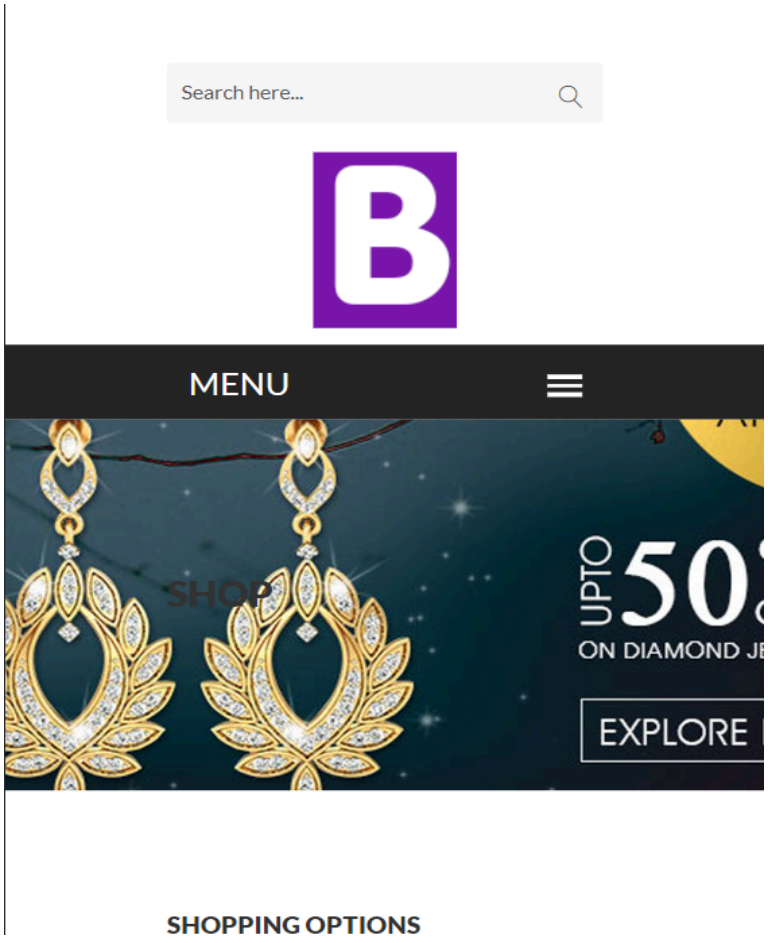
FOR DESKTOP DEVICE



Project Title:Fiver(flutter) / Jwellery site(Pwa) **Roll No.27**

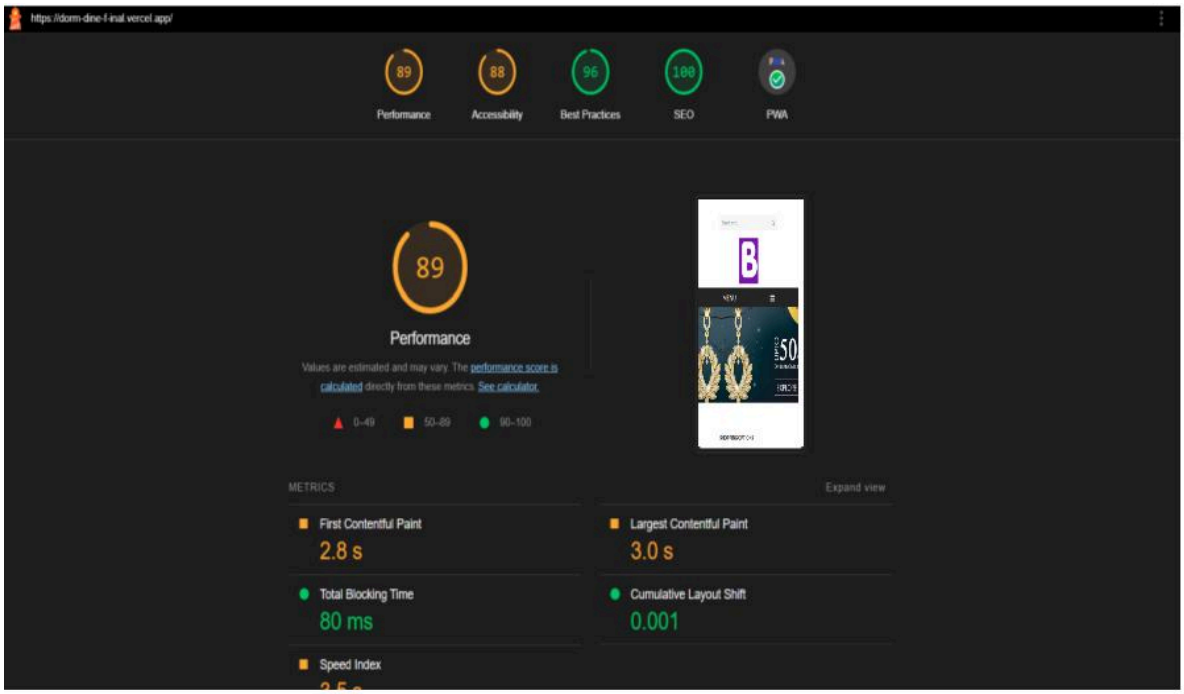


FOR MOBILE DEVICES

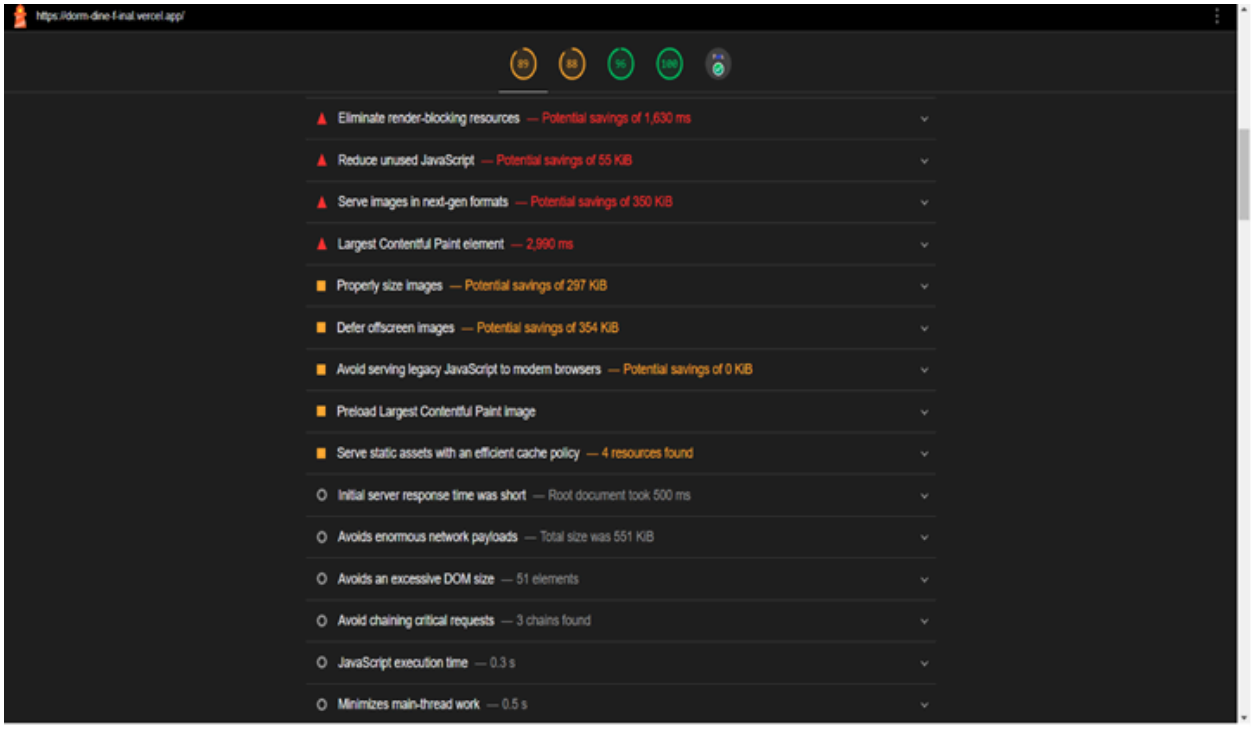


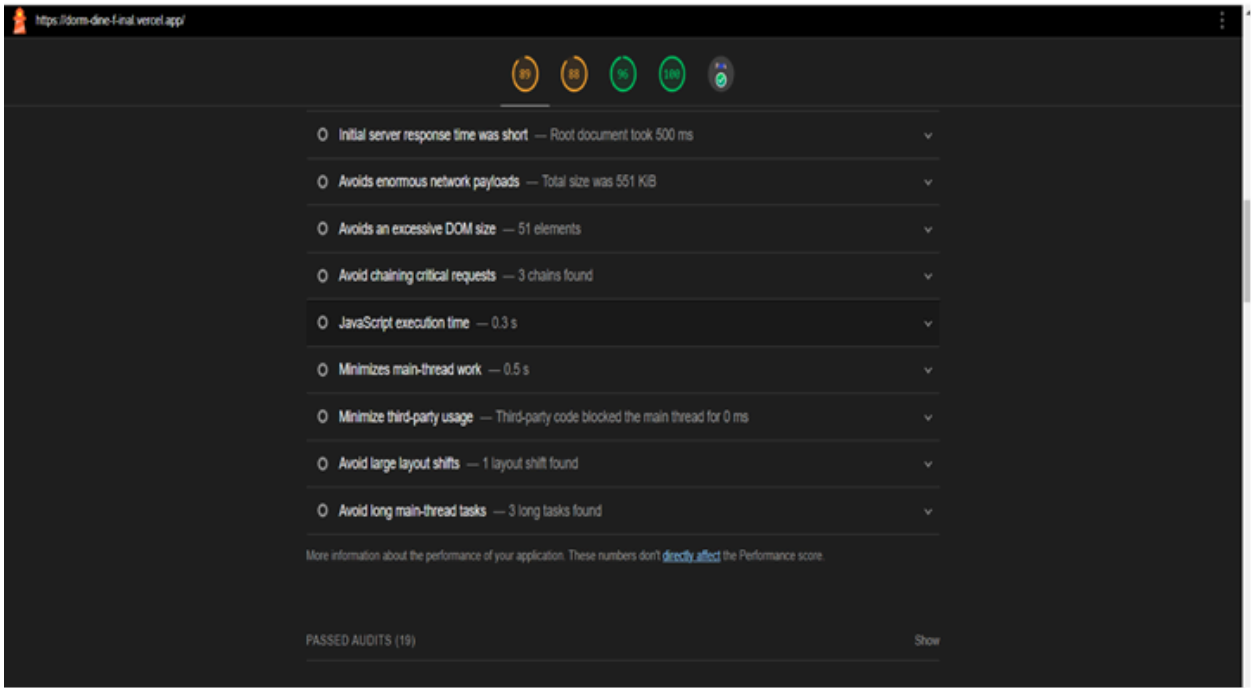
Project Title:Fiver(flutter) / Jwellery site(Pwa) **Roll No.27**

REPORT GENERATED:

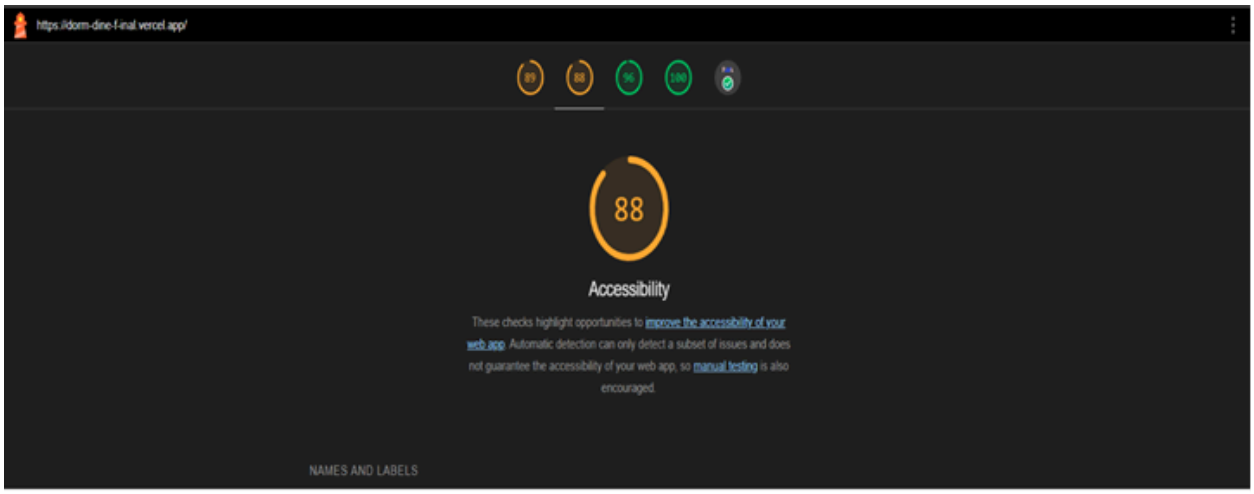


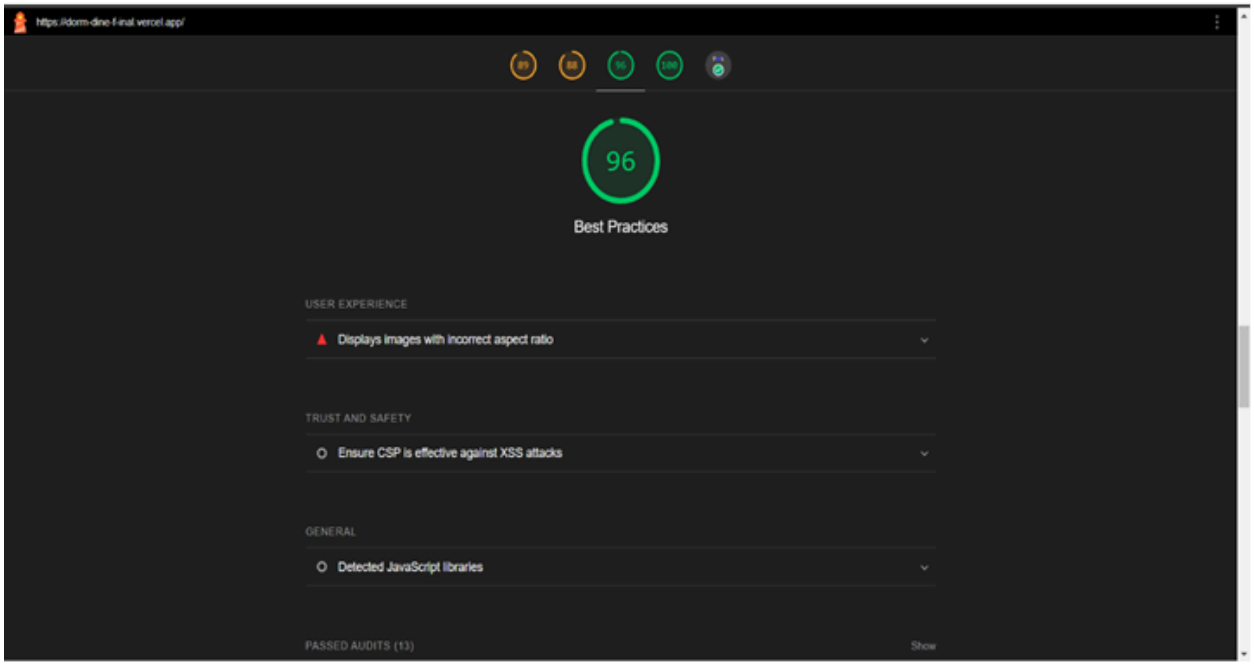
Project Title:Fiver(flutter) / Jwellery site(Pwa) **Roll No.27**



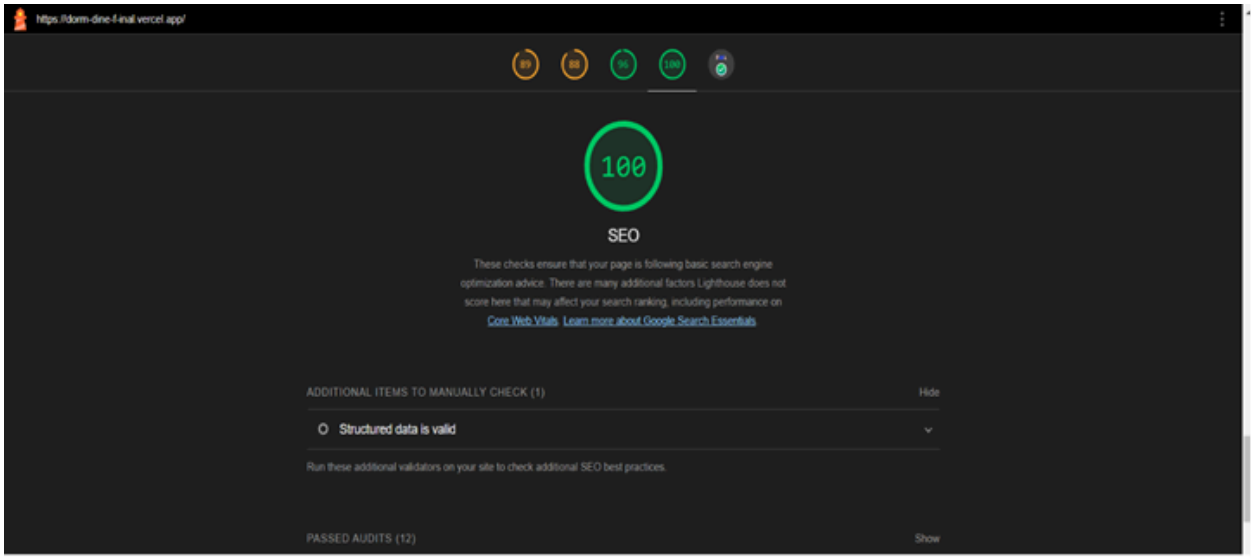


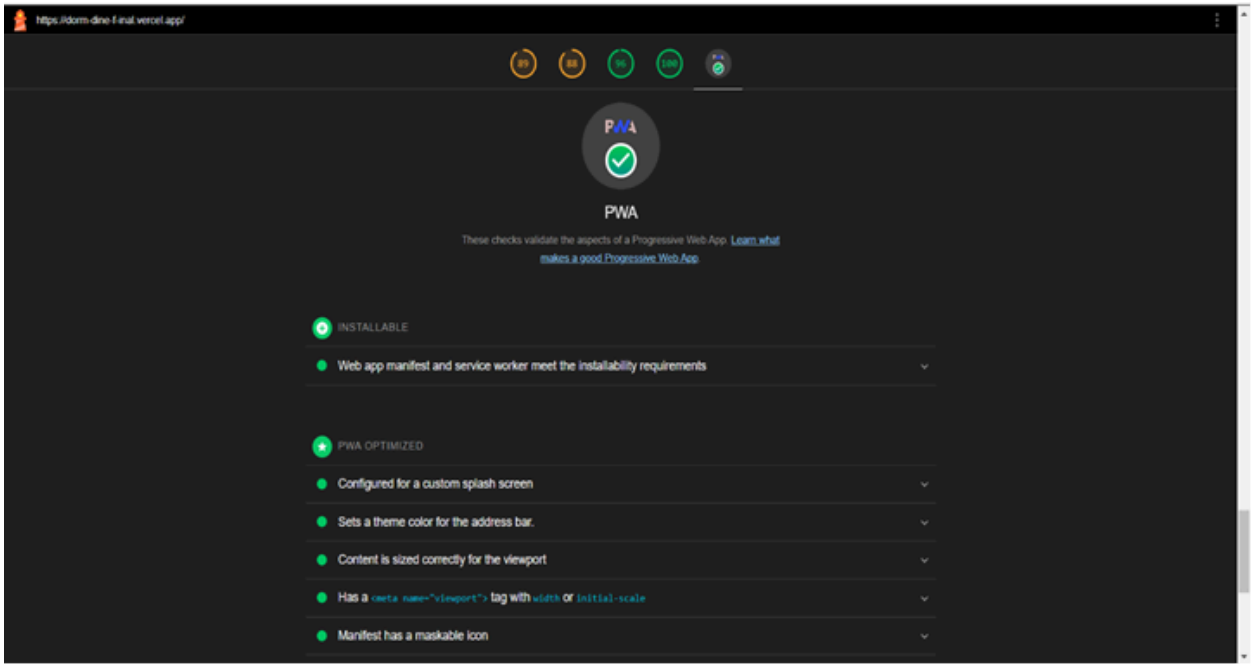
Project Title:Fiver(flutter) / Jwellery site(Pwa) **Roll No.27**

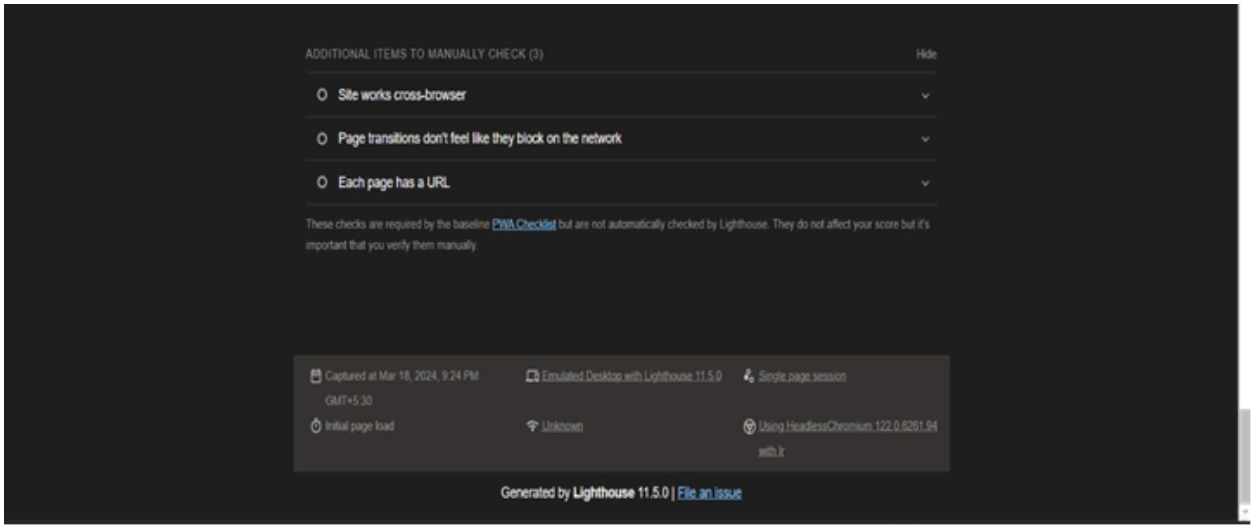




Project Title:Fiver(flutter) / Jwellery site(Pwa) **Roll No.27**





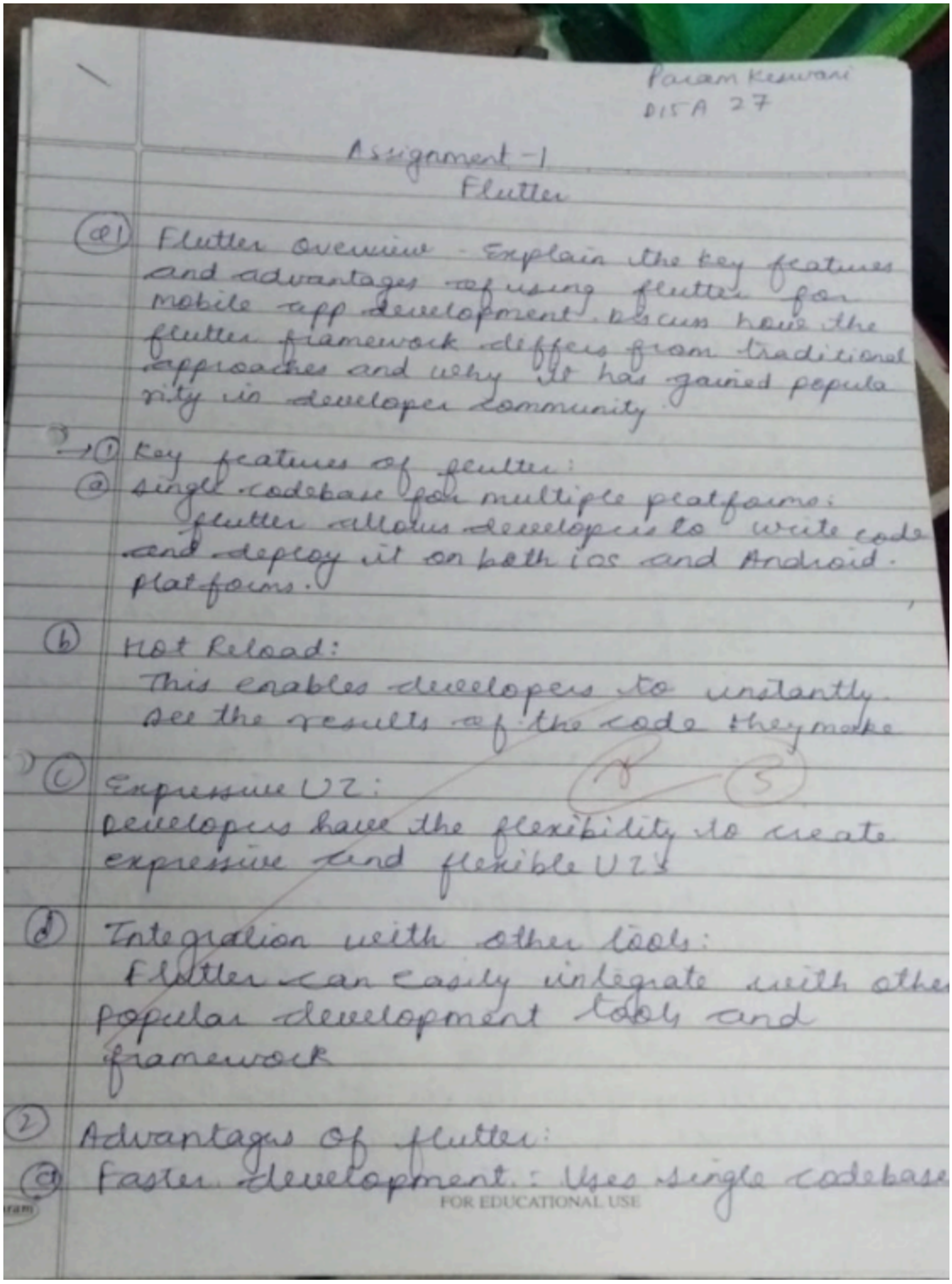


Conclusion:

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

MAD & PWA Lab
Journal

Experiment No.	Assignment-1
Assignment 1 Questions	1. Flutter Overview:Explain the key features and advantages of using Flutter for mobile app development. Discuss how the Flutter framework differs from traditional approaches and why it has gained popularity in the developer community. 2. Widget Tree and Composition: Describe the concept of the widget tree in Flutter. Explain how widget composition is used to build complex user interfaces. Provide examples of commonly used widgets and their roles in creating a widget tree. 3. State Management in Flutter: Discuss the importance of state management in Flutter applications. Compare and contrast the different state management approaches available in Flutter, such as setState, Provider, and Riverpod. Provide scenarios where each approach is suitable. 4. Firebase Integration in Flutter: Explain the process of integrating Firebase with a Flutter application. Discuss the benefits of using Firebase as a backend solution. Highlight the Firebase services commonly used in Flutter development and provide a brief overview of how data synchronization is achieved.
Roll No.	27
Name	Param Keswani
Class	D15A/D15B
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:	



Param Kesvani
D15A 27

Assignment -1
Flutter

(Q1) 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 developer community.

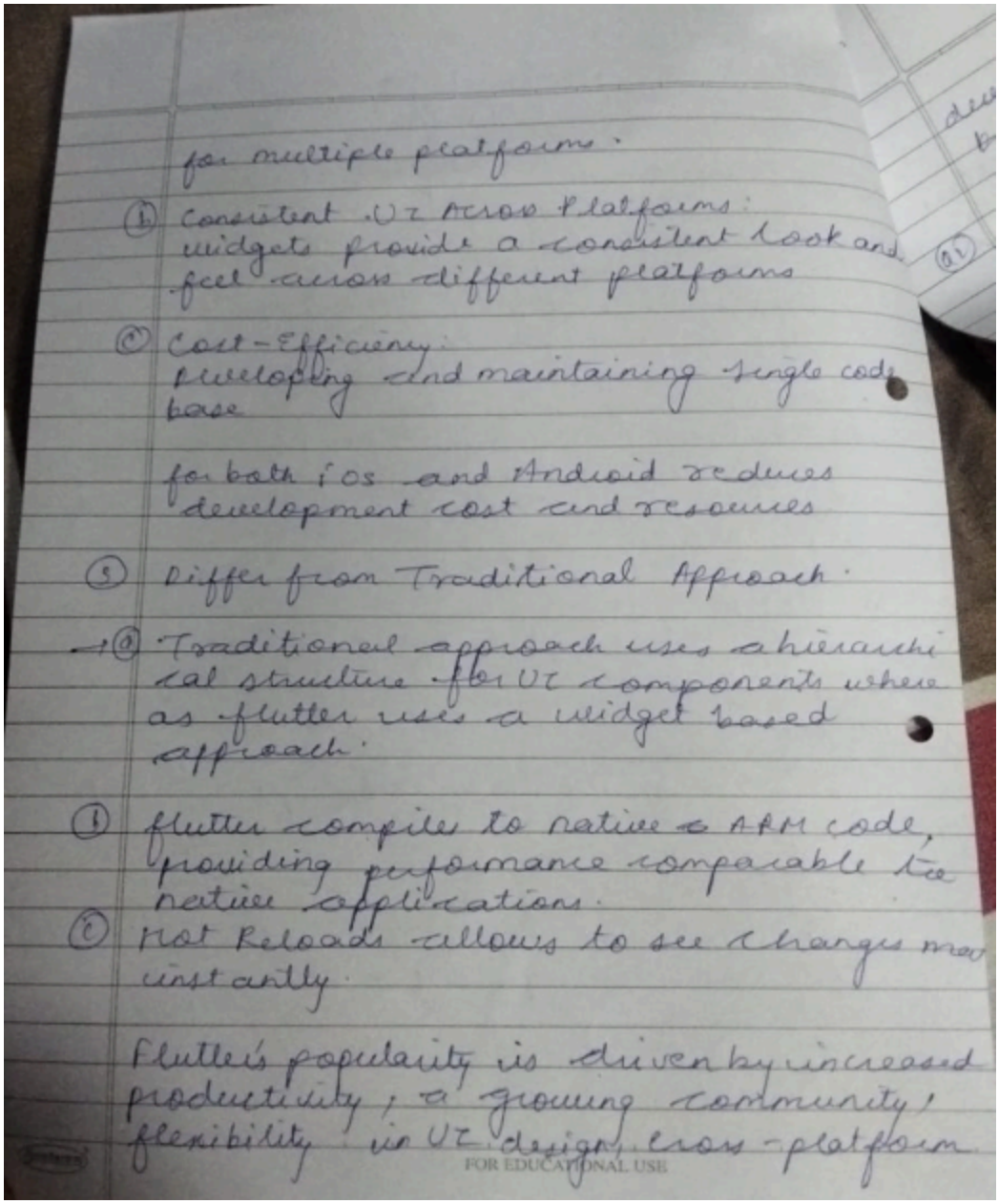
→ (1) Key features of flutter:
(a) single codebase for multiple platforms:
flutter allows developers to write code and deploy it on both ios and Android platforms.

(b) Hot Reload:
This enables developers to instantly see the results of the code they make

(c) Expressive UI:
Developers have the flexibility to create expressive and flexible UIs

(d) Integration with other tools:
Flutter can easily integrate with other popular development tools and framework

(2) Advantages of flutter:
(a) Faster development: Uses single codebase



for multiple platforms:

(b) Consistent UI Across Platforms:
Widgets provide a consistent look and feel across different platforms

(c) Cost-Efficiency:
Developing and maintaining single code base

for both iOS and Android reduces development cost and resources

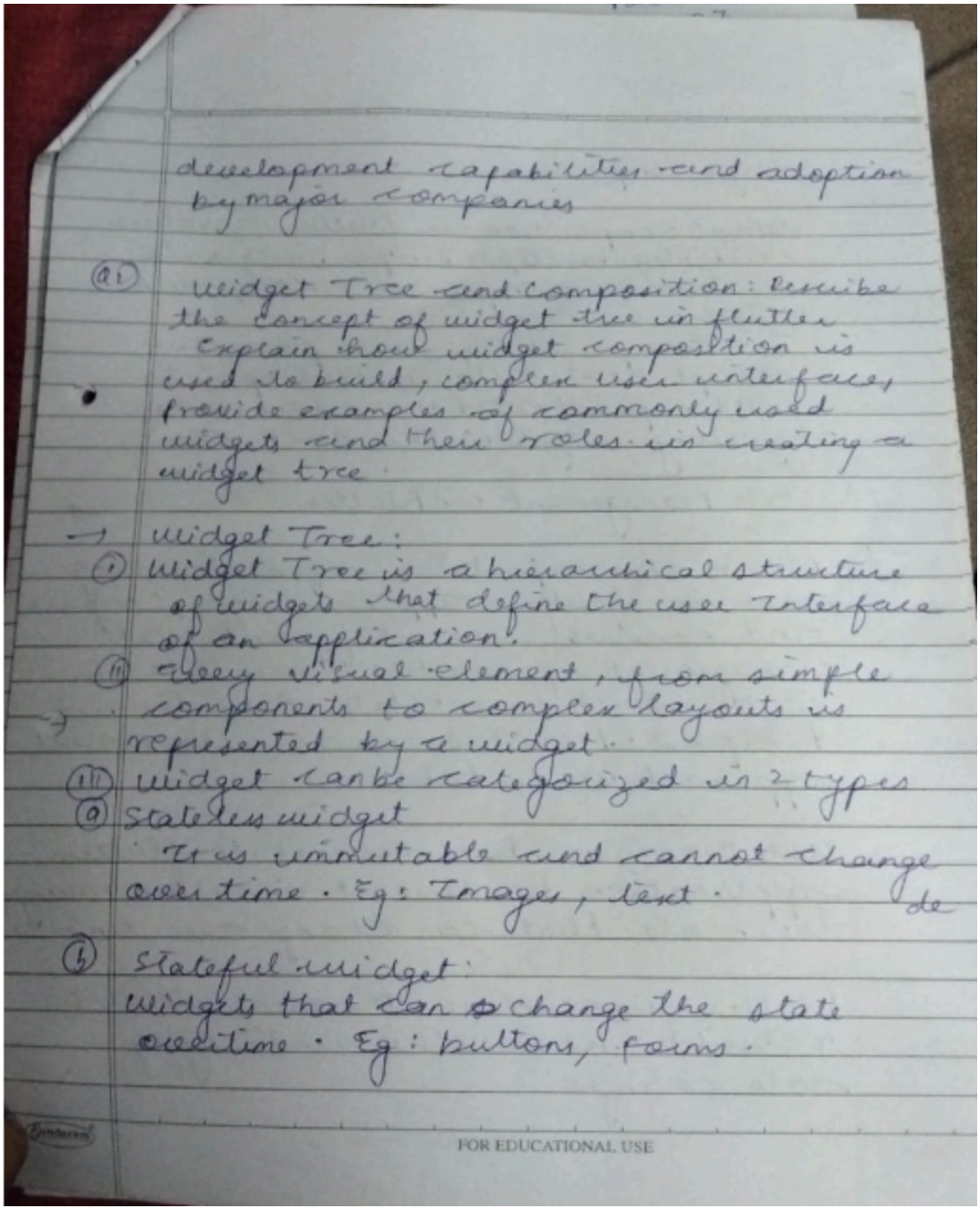
(5) Differ from Traditional Approach:

(a) Traditional approach uses a hierarchical structure for UI components where as flutter uses a widget based approach.

(1) flutter compiles to native ARM code, providing performance comparable to native applications.

(c) Hot Reloads allows to see changes made instantly.

Flutter's popularity is driven by increased productivity, a growing community, flexibility in UI design, cross-platform



development capabilities and adoption by major companies

(Q1) Widget Tree and Composition: Describe the concept of widget tree in flutter. Explain how widget composition is used to build complex user interfaces, provide examples of commonly used widgets and their roles in creating a widget tree.

→ Widget Tree:

(i) Widget Tree is a hierarchical structure of widgets that define the user Interface of an application.

(ii) Every visual element, from simple components to complex layouts is represented by a widget.

(iii) Widget can be categorized in 2 types

(a) Stateless widget

It is immutable and cannot change over time. Eg: Images, text.

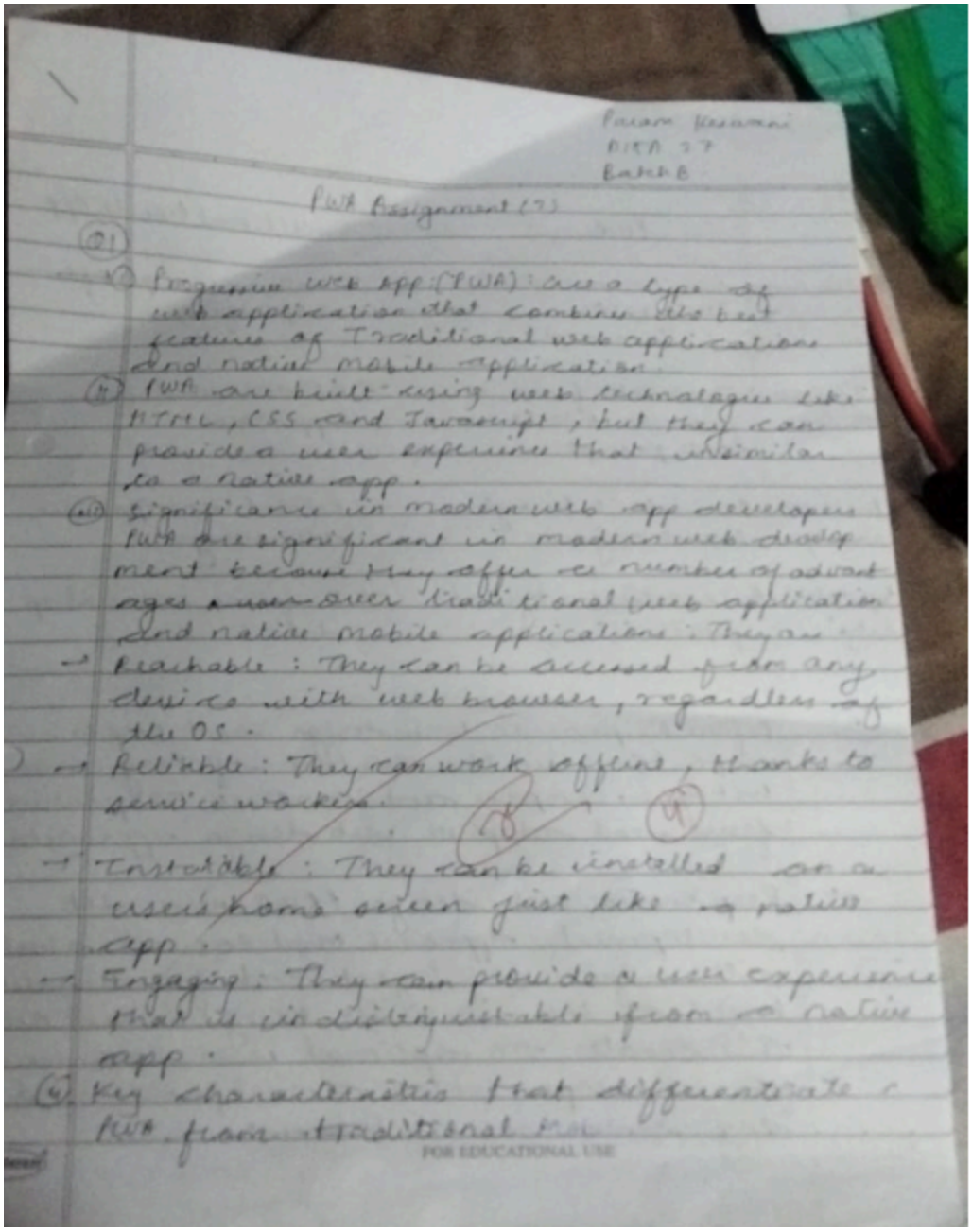
(b) Stateful widget:

Widgets that can change the state overtime. Eg: buttons, forms.

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.	27
Name	Param Keswani
Class	D15A/D15B
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:	

Project Title:Fiver(flutter) / Jwellery site(Pwa) **Roll No.27**



Param Kesarani
BIBA 27
Batch B

PWA Assignment (1)

Q1)

(i) Progressive Web App (PWA): are a type of web application that combine the best features of Traditional web applications and native mobile application.

(ii) PWA are built using web technologies like HTML, CSS and Javascript, but they can provide a user experience that is similar to a native app.

(iii) Significance in modern web app developments PWA are significant in modern web development because they offer a number of advantages over traditional web application and native mobile applications. They are:

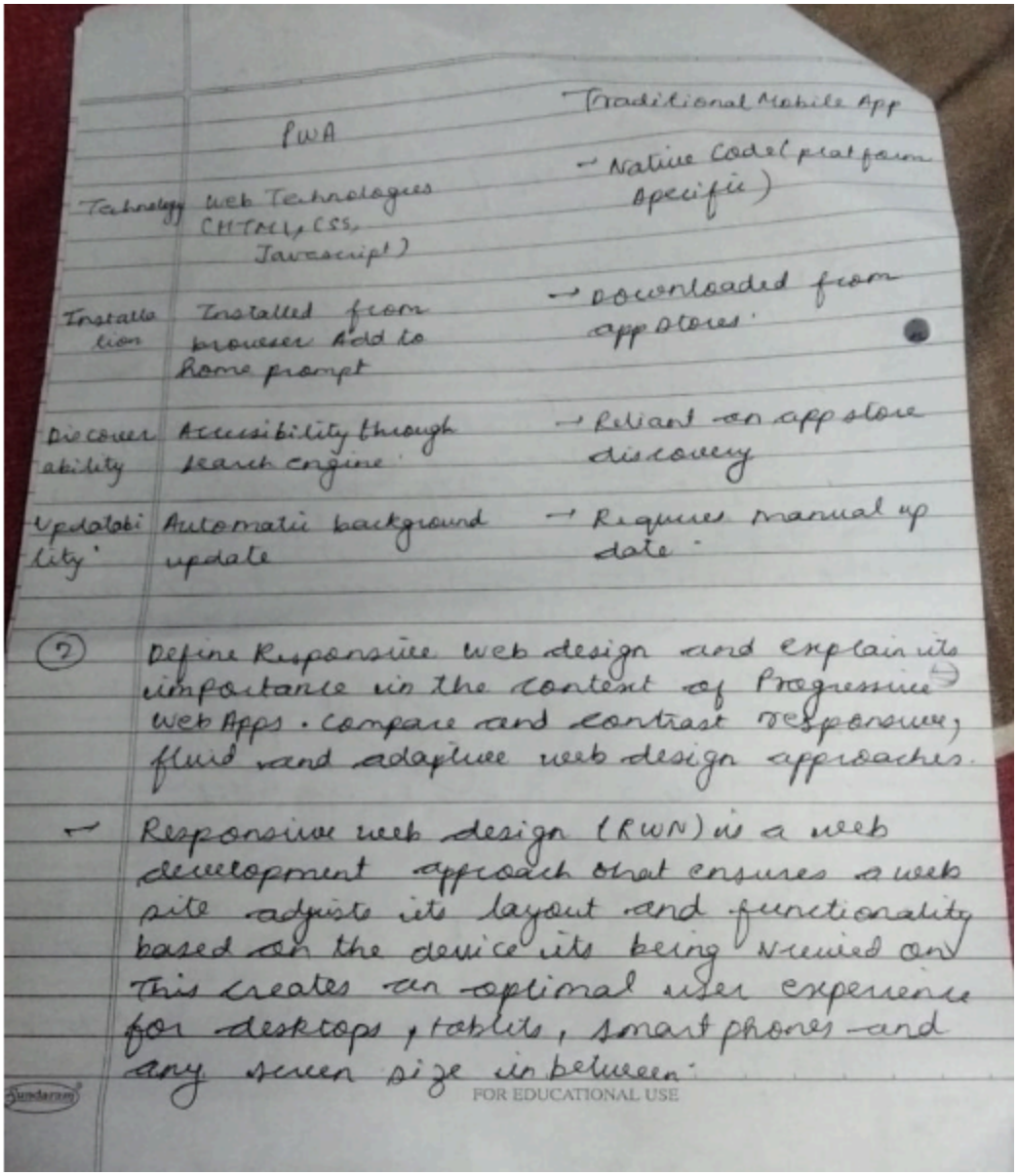
→ Reachable: They can be accessed from any device with web browser, regardless of the OS.

→ Reliable: They can work offline, thanks to service workers.

→ Installable: They can be installed on a user's home screen just like a native app.

→ Engaging: They can provide a user experience that is indistinguishable from a native app.

(iv) Key characteristics that differentiate a PWA from traditional



	PWA	Traditional Mobile App
Technology	Web Technologies (HTML, CSS, Javascript)	→ Native Code (platform specific)
Installation	Installed from browser Add to Home prompt	→ Downloaded from app stores.
Discoverability	Accessibility through search engine.	→ Reliant on app store discovery
Updatability.	Automatic background update	→ Requires manual update.

(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.

→ Responsive web design (RWD) is a web development approach that ensures a web site adjusts its layout and functionality based on the device its being viewed on. This creates an optimal user experience for desktops, tablets, smartphones and any screen size in between.