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 **_Mohd Usmaan Mogal** 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: Reddit **Roll No. 36**

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: Reddit **Roll No. 36**

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

Project Title: Reddit **Roll No. 36**

Index

Sr. No	Experiment Title	LO	DOP	DOS	Grade
1.	To install and configure the Flutter Environment	LO1	17/1/24	24/1/24	15
2.	To design Flutter UI by including common widgets.	LO2	24/1/24	31/1/24	15
3.	To include icons, images, fonts in Flutter app	LO2	31/1/24	7/2/24	15
4.	To create an interactive Form using form widget	LO2	7/2/24	14/2/24	15
5.	To apply navigation, routing and gestures in Flutter App	LO2	14/2/24	21/2/24	15
6.	To Connect Flutter UI with fireBase database	LO3	21/2/24	6/3/24	15
7.	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.	LO4	6/3/24	13/3/24	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	13/3/24	20/3/24	15
9.	To implement Service worker events like fetch, sync and push for E-commerce PWA	LO5	20/3/24	27/3/24	15
10.	To study and implement deployment of Ecommerce PWA to GitHub Pages.	LO5	27/3/24	27/3/24	15
11.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.	LO6	27/3/24	27/3/24	15
12.	Assignment-1	LO1,LO2,LO3	4/2/24	5/2/24	5
13.	Assignment-2	LO4,LO5,LO6	20/3/24	21/3/24	4

MAD & PWA Lab
Journal

Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	36
Name	Mohd Usmaan Mogal
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	15

Project Title: Reddit **Roll No. 36**

Aim: To write a Hello World Program on Flutter

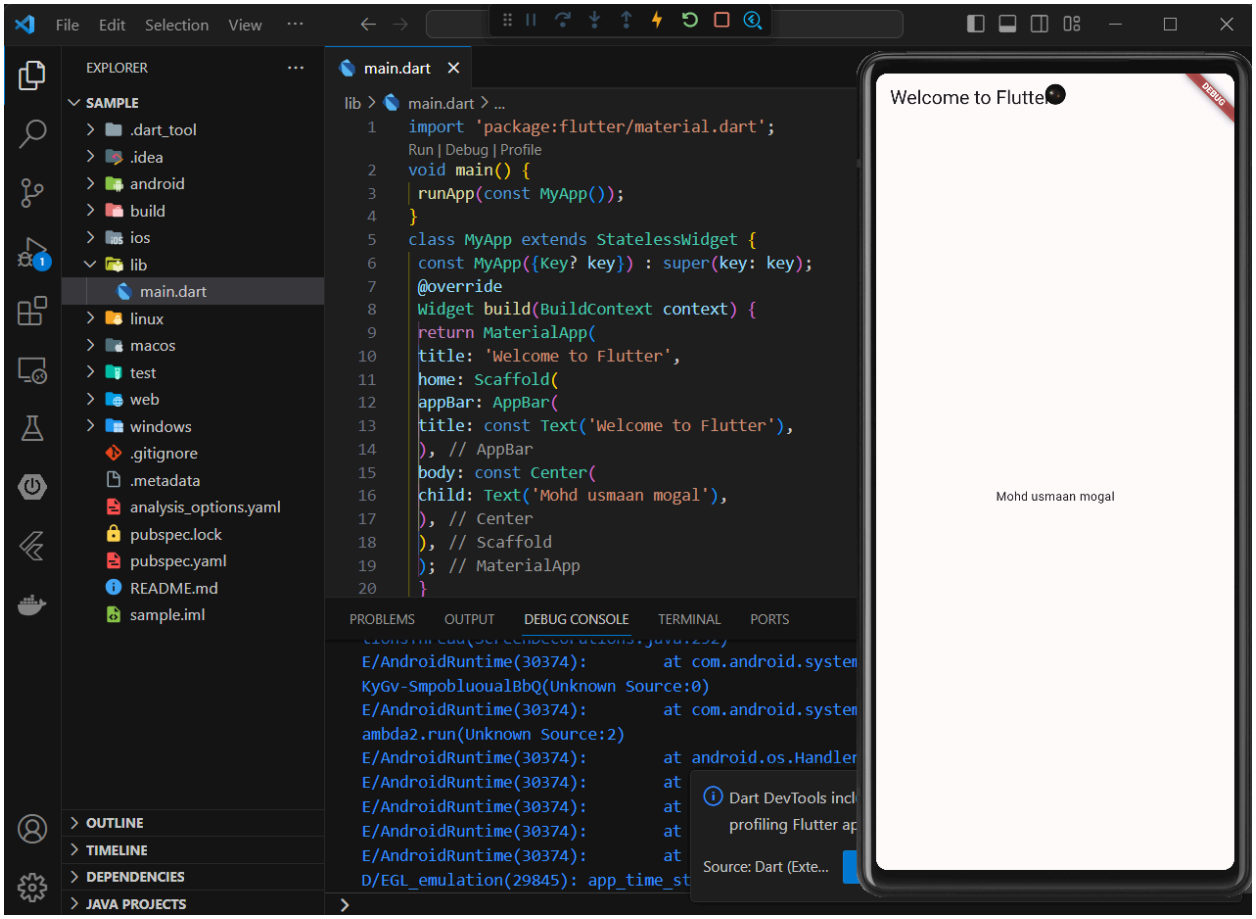
Pre Requisites: Android Studio Hedgehog, Visual Studio Code and Flutter package.I highly recommend watching this video, for Setting up your Flutter Environment and your Virtual Device: https://youtu.be/ZSWfgxrxN0M?feature=shared

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('Vedang Khandagale'),
 ),
 ),
 );
 }
}
```

Project Title: Reddit **Roll No. 36**

Output:



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

MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	36
Name	Mohd Usmaan Mogal
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

Project Title: Reddit **Roll No. 36**

Aim: To design Flutter UI by including common widgets.

Theory:

Image:

The Image widget supports various image formats such as JPEG, PNG, GIF, and more.
It can load images from the network using NetworkImage, from local assets using AssetImage, or from the file system using FileImage.
You can apply transformations, like scaling and cropping, to the displayed image.
Text:

The Text widget is used to display text and can be styled using the style property.
It supports rich formatting, such as different fonts, sizes, colors, and text alignments.
Text can be styled using the TextStyle class.

Icon:

The Icon widget displays a graphical icon. Flutter includes a set of built-in icons (from the Material Icons and Cupertino Icons libraries), and you can also use custom icons.
Icons can be styled using the color, size, and semanticLabel properties.

FlatButton:

The FlatButton widget is a simple button without any elevation.
It responds to touches by filling with color and can trigger actions when pressed.

Container:

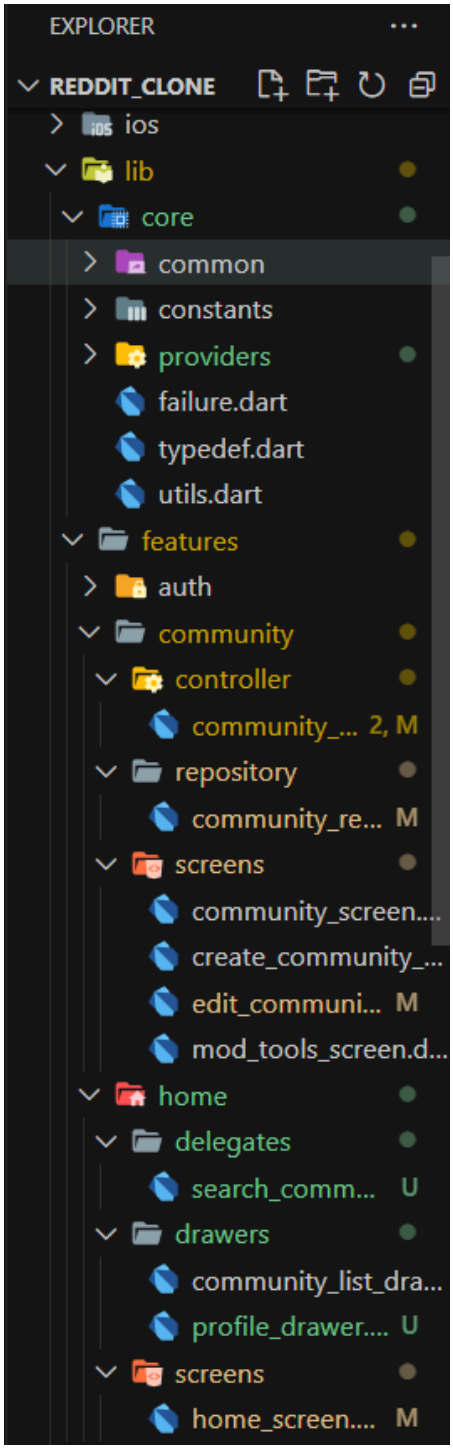
The Container widget is a box model that can contain other widgets.
It can be used for layout purposes, such as setting width, height, padding, margin, and decoration.
The decoration property allows you to apply background color, borders, and more.

Row and Column:

Row and Column widgets are used for arranging child widgets horizontally and vertically, respectively.
They automatically adjust the size of their children based on the available space.

Project Title: Reddit **Roll No. 36**

File structure:



Code:

```
// ignore_for_file: public_member_api_docs, sort_constructors_first
import 'package:flutter/material.dart';
```

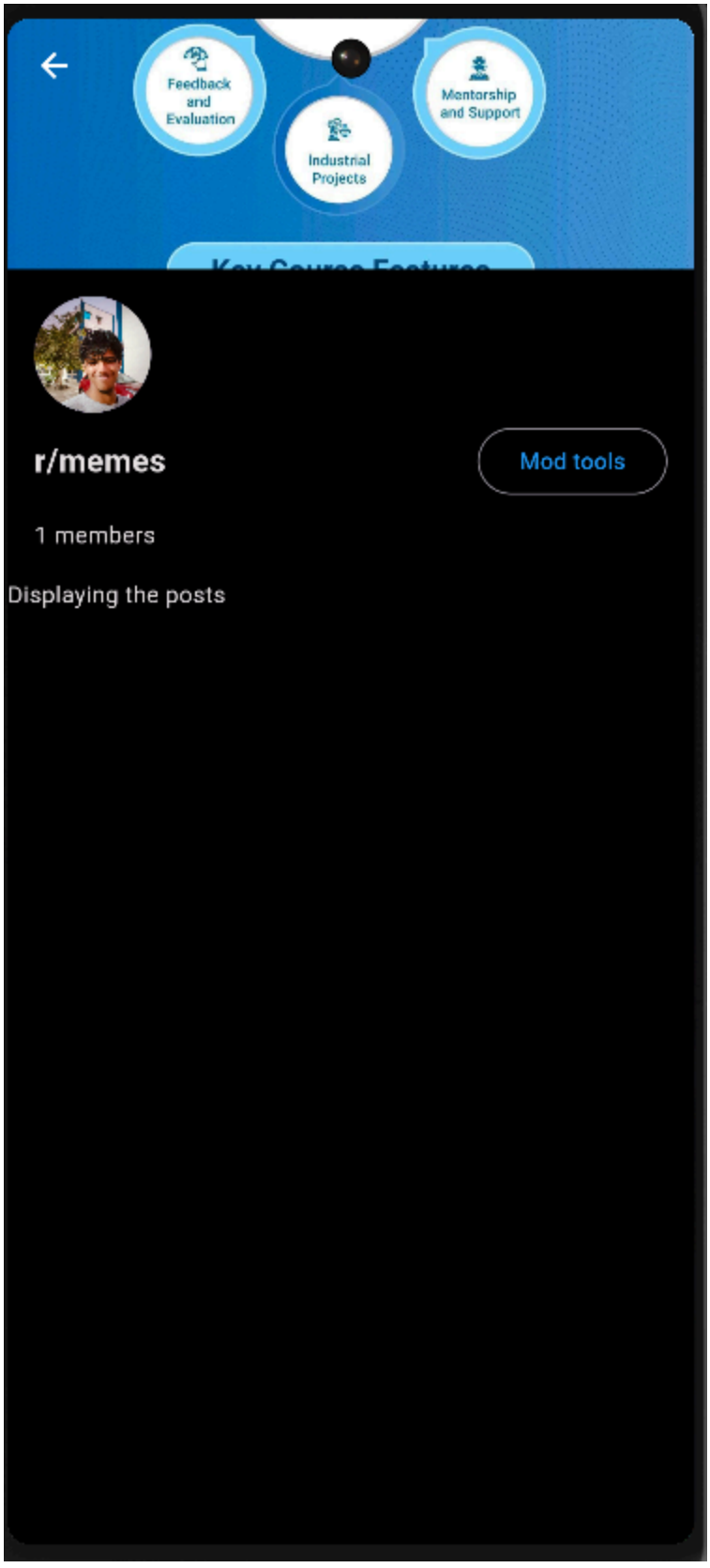
```
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:reddit_clone/core/common/error_text.dart';
import 'package:reddit_clone/core/common/loader.dart';
import 'package:reddit_clone/features/auth/controller/auth_controller.dart';
import 'package:reddit_clone/features/community/controller/community_controller.dart';
import 'package:routemaster/routemaster.dart';

class CommunityScreen extends ConsumerWidget {
  final String name;

  const CommunityScreen({super.key,
    required this.name,
  });

  void navigateToModTools(BuildContext context) {
    Routemaster.of(context).push('/mod-tools/$name');
  }

  @override
  Widget build(BuildContext context, WidgetRef ref) {
    final user = ref.watch(userProvider)!;
    return Scaffold(
      body: ref.watch(getCommunityNameProvider(name)).when(
          data: (community) => NestedScrollView(
              headerSliverBuilder: (context, innerBoxIsScrolled) {
                return [
                  SliverAppBar(
                    expandedHeight: 150,
                    floating: true,
                    snap: true,
                    flexibleSpace: Stack(
                      children: [
                        Positioned.fill(
                          child: Image.network(
                            community.banner,
                            fit: BoxFit.cover,
                          ),
                        ),
                      ],
                    ),
                  ),
                  SliverPadding(
                    padding: const EdgeInsets.all(16),
                    sliver: SliverList(
```

```
  delegate: SliverChildListDelegate([
Align(
  alignment: Alignment.topLeft,
  child: CircleAvatar(
    backgroundImage: NetworkImage(community.avatar),
    radius: 35,
  ),
),
const SizedBox(
  height: 5,
),
Row(
  mainAxisAlignment: MainAxisAlignment.spaceBetween,
  children: [
    Text(
      'r/${community.name}',
      style: const TextStyle(
          fontSize: 19, fontWeight: FontWeight.bold),
    ),
    community.mods.contains(user.uid)
        ? OutlinedButton(
            onPressed: () {
              navigateToModTools(context);
            },
            style: ElevatedButton.styleFrom(
                shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.circular(20),
                ),
                padding: const EdgeInsets.symmetric(
                    horizontal: 25),
                shadowColor: Colors.blue),
            child: const Text(
              'Mod tools',
              style: TextStyle(color: Colors.blue),
            ))
        : OutlinedButton(
            onPressed: () {},
            style: ElevatedButton.styleFrom(
                shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.circular(20),
                ),
                padding: const EdgeInsets.symmetric(
                    horizontal: 25),
                shadowColor: Colors.blue),
```

```
                        child: Text(
                          community.members.contains(user.uid)
                              ? 'Joined'
                              : 'Join',
                          style: const TextStyle(color: Colors.blue),
                        ))
                    ],
                  ),
                  Padding(
                    padding: const EdgeInsets.only(top: 10),
                    child: Text('${community.members.length} members'),
                  )
                ])),
              )
            ];
          },
          body: const Text('Displaying the posts')),
      error: (error, stackTrace) => ErrorText(error: error.toString()),
      loading: () => const Loader()),
    );
  }
}
```

Output:

Project Title: Reddit **Roll No. 36**



MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	36
Name	Mohd Usmaan Mogal
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

Project Title: Reddit **Roll No. 36**

Aim: To include images, fonts and icons in flutter app

Theory:

Image Widget:

The Image widget handles decoding and rendering of various image formats, such as JPEG, PNG, GIF, and more.
It can dynamically adapt to different image sources and resolutions.
Image Caching:

Flutter automatically caches images, improving performance by avoiding redundant network requests or file reads.
This caching mechanism helps in efficiently managing and displaying images.
Image Transformations:

The widget allows for transformations like scaling, cropping, and rotation, providing developers with control over the displayed image's appearance.
Placeholder and Error Handling:

Developers can specify placeholder images to display while the actual image is being loaded.
Error images can be configured to handle cases where the intended image cannot be loaded.
Icon Widget:
Semantic Labels:

The Icon widget allows you to set a semantic label, providing accessibility for users with screen readers.
This enhances the user experience for those who rely on assistive technologies.

Icons:

Developers can use custom icons in addition to the extensive set of built-in icons.
Custom icons can be loaded from various sources, such as font-based icons or image files.
Icon Theming:

Icons can be styled and themed using the color property, allowing for consistent design across the app.
Size Adjustment:

The size of the icon can be adjusted based on application requirements, providing flexibility in the UI design.

NetworkImage:

NetworkImage efficiently manages the retrieval of images from network URLs, optimizing the network requests for performance.
Caching Strategies:

It incorporates caching strategies to minimize redundant network requests, ensuring a smooth user experience even in varying network conditions.
Error Handling:

NetworkImage allows developers to handle errors gracefully, specifying fallback images or other actions in case the network image cannot be loaded.
Progressive Loading:

Progressive loading is supported, where images are displayed gradually as they are being downloaded, improving perceived performance.

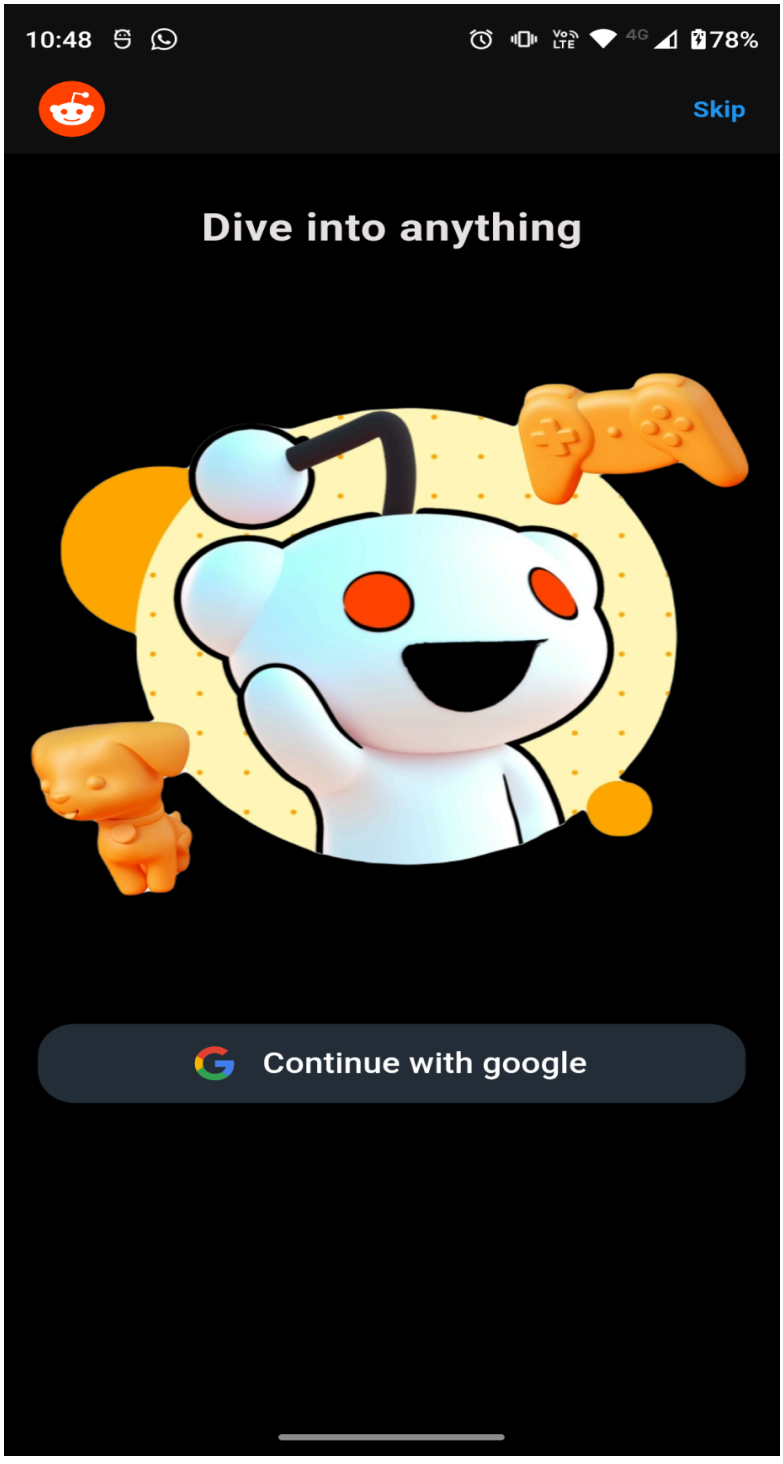
Code:

```
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:reddit_clone/core/common/loader.dart';
import 'package:reddit_clone/core/common/sign_in_button.dart';
import 'package:reddit_clone/core/constants/constants.dart';
import 'package:reddit_clone/features/auth/controller/auth_controller.dart';

class LoginScreen extends ConsumerWidget {
  const LoginScreen({super.key});

  @override
  Widget build(BuildContext context, WidgetRef ref) {
    final isLoading = ref.watch(authControllerProvider);

    return Scaffold(
      appBar: AppBar(
        title: Image.asset(
          Constants.logoPath,
          height: 40,
        ),
        actions: [
          TextButton(
              onPressed: () {},
              child: const Text(
                'Skip',
```

```
            style:
                TextStyle(fontWeight: FontWeight.bold, color: Colors.blue),
            ))
      ],
    ),
    body: isLoading
        ? const Loader()
        : Column(children: [
            const SizedBox(height: 30),
            const Text(
              'Dive into anything',
              style: TextStyle(
                  fontSize: 24,
                  fontWeight: FontWeight.bold,
                  letterSpacing: 0.5),
            ),
            const SizedBox(
              height: 30,
            ),
            Padding(
              padding: const EdgeInsets.all(8.0),
              child: Image.asset(
                Constants.loginEmotePath,
                height: 400,
              ),
            ),
            const SizedBox(
              height: 20,
            ),
            const SignInButton()
          ]),
    );
  }
}
```

Output:

Project Title: Reddit **Roll No. 36**



MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	36
Name	Mohd Usmaan Mogal
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

Project Title: Reddit **Roll No. 36**

AIM: To create an interactive form using form widget

Theory:

Form Fields:

The Form widget serves as a container for various form fields. These form fields could include text input fields for capturing user input, checkboxes for boolean selections, radio buttons for exclusive choices, and more. Each form field is responsible for gathering specific types of data from the user.
Validation:

Validation is a crucial aspect of form handling. The Form widget allows you to define validation logic for each form field. This validation ensures that the data entered by the user meets certain criteria. For example, you can check if an email field contains a valid email address or if a password meets certain complexity requirements.
Global Key:

The global key associated with the Form widget is used to uniquely identify and interact with the form. This key is essential for performing various operations on the form, such as triggering form submission and accessing its state. It provides a way to uniquely reference and manage the form within the widget tree.
Form Submission:

When a user attempts to submit the form, typically by interacting with a submit button, the Form widget comes into play. It checks the validation status of each form field. If all form fields are valid, you can execute a callback function to handle the submitted data. This could involve processing the data, sending it to a server, or navigating to another screen within your app.
State Management:

The Form widget is responsible for managing the state of the form. It keeps track of the values entered by the user, as well as the validation status of each form field. This state management simplifies the process of working with user input, ensuring a seamless and controlled experience for both developers and users.

In essence, the Form widget in Flutter provides a structured and organized way to handle user input through a set of form fields. It encapsulates the complexities of validation, submission, and state management, making it easier for developers to build interactive and user-friendly forms in their applications.

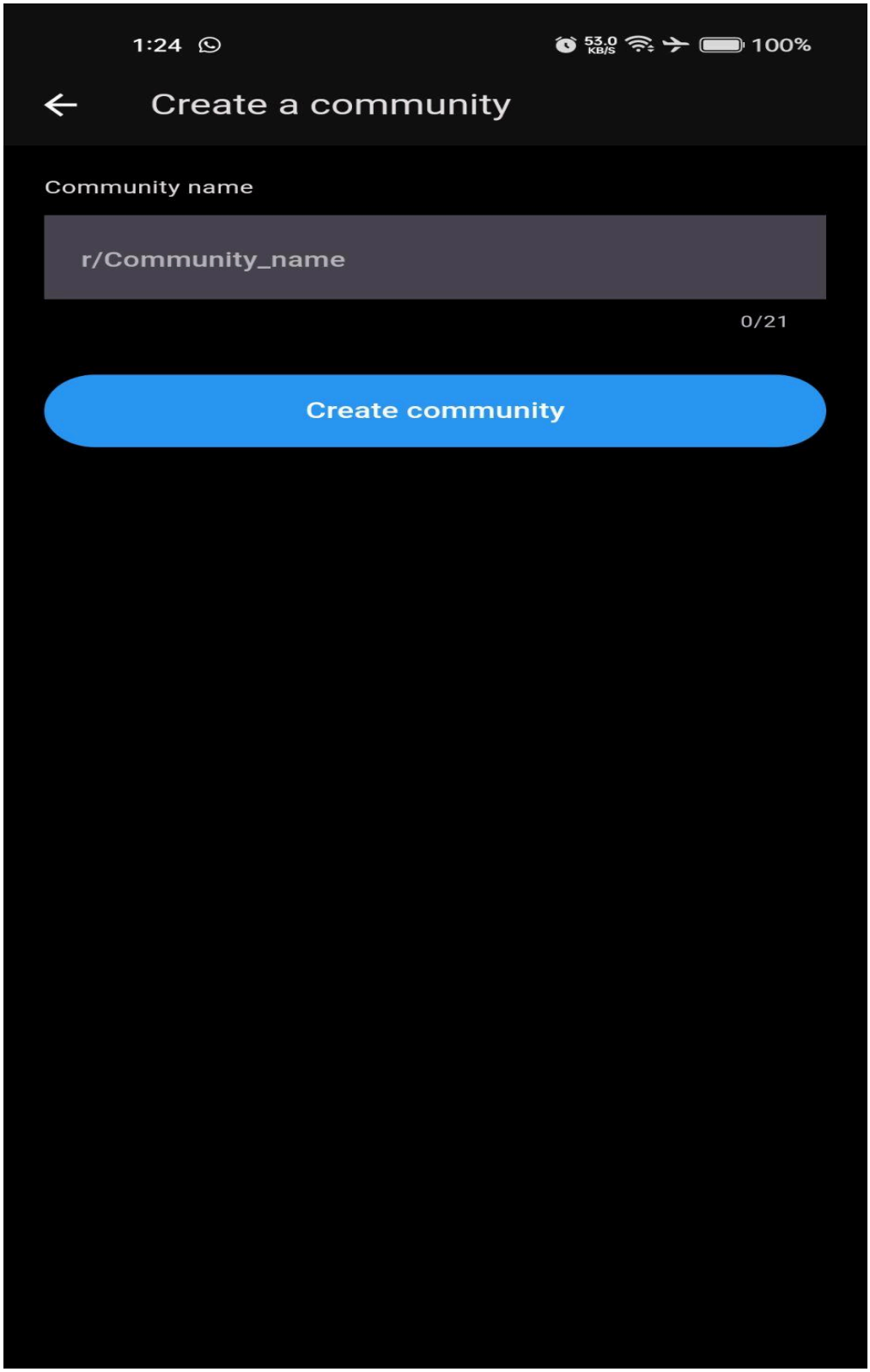
Code:

```
// ignore_for_file: unnecessary_const

import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:reddit_clone/core/common/loader.dart';
import 'package:reddit_clone/features/community/controller/community_controller.dart';

class CreateCommunityScreen extends ConsumerStatefulWidget {
  const CreateCommunityScreen({super.key});

  @override
  ConsumerState<ConsumerStatefulWidget> createState() =>
      _CreateCommunityScreenState();
}

class _CreateCommunityScreenState extends ConsumerState<CreateCommunityScreen> {
  final communityNameController = TextEditingController();
  @override
  void dispose() {
    super.dispose();
    communityNameController.dispose();
  }

  void createCommmunity() {
    ref
        .read(communityControllerProvider.notifier)
        .createCommunity(communityNameController.text.trim(), context);
  }
  @override
  Widget build(BuildContext context) {
    final isLoading = ref.watch(communityControllerProvider);
    return Scaffold(
      appBar: AppBar(title: const Text('Create a community')),
      body: isLoading
          ? const Loader()
          : Padding(
              padding: const EdgeInsets.all(20.0),
              child: Column(
                children: [
                  const Align(
                      alignment: Alignment.topLeft,
                      child: Text("Community name")),
                  const SizedBox(
                    height: 10,
```

```
            ),
            TextField(
              controller: communityNameController,
              decoration: const InputDecoration(
                  hintText: 'r/Community_name',
                  filled: true,
                  border: InputBorder.none,
                  contentPadding: EdgeInsets.all(18)),
              maxLength: 21,
            ),
            const SizedBox(
              height: 30,
            ),
            ElevatedButton(
              onPressed: createCommmunity,
              style: ElevatedButton.styleFrom(
                  minimumSize: const Size(double.infinity, 50),
                  backgroundColor: Colors.blue),
              child: const Text(
                'Create community',
                style: TextStyle(color: Colors.white, fontSize: 17),
              ),
            )
          ],
        ),
      ),
    );
  }
}
```

Output:

Project Title: Reddit **Roll No. 36**



MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	36
Name	Mohd Usmaan Mogal
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

Project Title: Reddit **Roll No. 36**

Aim: To apply navigation routing and gestures in an Flutter App

Theory:

Navigation routing in Flutter refers to the process of defining and managing the routes or screens within your app. Flutter's navigation routing system allows you to move between different screens or pages in a structured and organized way. Here's an overview:

Route Definition: In Flutter, each screen or page in your app is typically represented by a widget. To define routes, you create a mapping between route names and the corresponding widget classes. This mapping is often done in the MaterialApp widget using the routes parameter.

Named Routes: Named routes are a common approach in Flutter for defining routes using unique names. Each route is associated with a name, making it easier to navigate to specific screens by referencing these names. For example, you can define routes like '/home', '/profile', '/settings', etc.

Navigation Stack: Flutter's navigation system maintains a stack of route objects. When you navigate to a new screen, the new route is pushed onto the stack. When you navigate back, the top route is popped off the stack, returning you to the previous screen.

Navigator: The Navigator class in Flutter is responsible for managing the navigation stack and transitioning between routes. You can use methods like Navigator.push() to navigate to a new route, Navigator.pop() to return to the previous route, and Navigator.pushNamed() to navigate to a named route.

MaterialPageRoute and CupertinoPageRoute: Flutter provides two types of page route classes - MaterialPageRoute for Android-style navigation and CupertinoPageRoute for iOS-style navigation. These classes define transitions and behaviors specific to each platform.

Route Parameters: You can pass parameters between routes in Flutter using the arguments parameter of the navigation methods (Navigator.pushNamed() or Navigator.push()). This allows you to send data to the next screen or receive data from the previous screen.

Modal Routes: Modal routes are routes that appear on top of other routes and typically block interaction with the underlying routes until they are dismissed. You can create modal routes using methods like showDialog() or by using Navigator.push() with a PageRouteBuilder and setting the fullscreenDialog parameter to true.

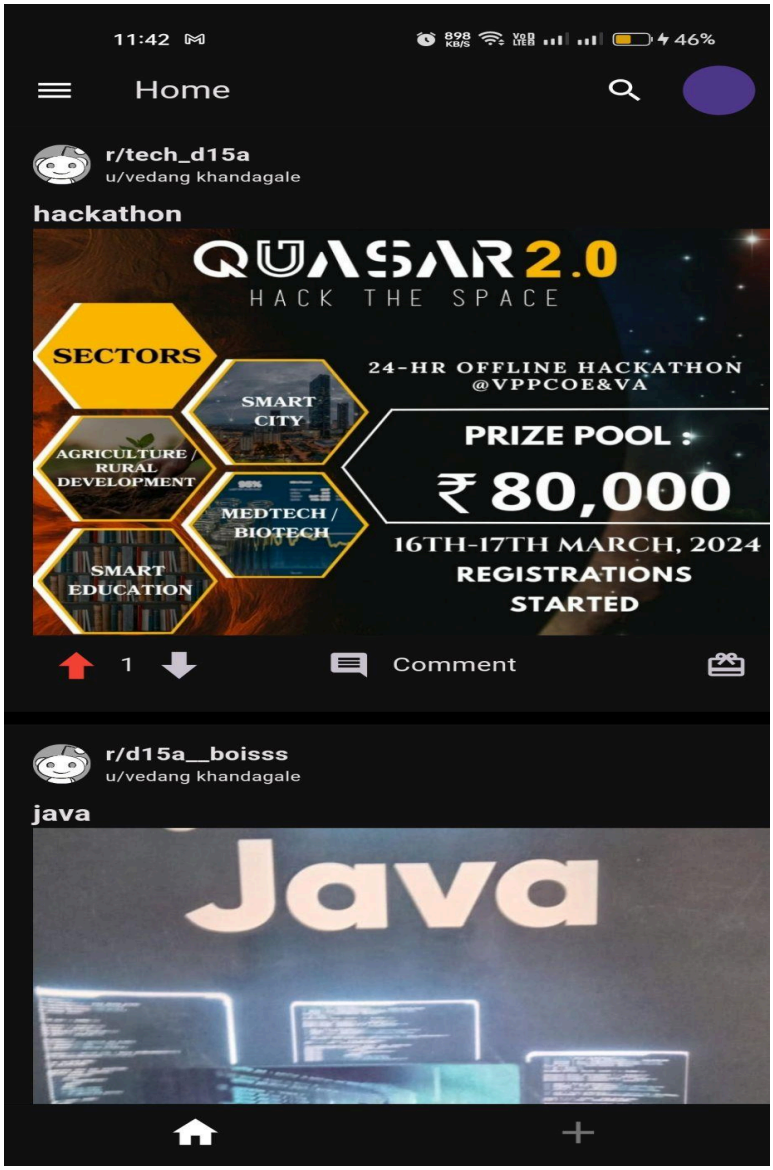
Code:

```
import 'package:flutter/material.dart';
import 'package:reddit_clone/features/auth/screens/login_screen.dart';
import 'package:reddit_clone/features/community/screens/add_mods_screen.dart';
import 'package:reddit_clone/features/community/screens/community_screen.dart';
import 'package:reddit_clone/features/community/screens/create_community_screen.dart';
import 'package:reddit_clone/features/community/screens/edit_community_screen.dart';
import 'package:reddit_clone/features/community/screens/mod_tools_screen.dart';
import 'package:reddit_clone/features/home/screens/home_screen.dart';
import 'package:reddit_clone/features/post/screens/add_post_type_screen.dart';
import 'package:reddit_clone/features/post/screens/comments_screen.dart';
import 'package:reddit_clone/features/user_profile/screens/edit_profile_screen.dart';
import 'package:reddit_clone/features/user_profile/screens/user_profile_screen.dart';
import 'package:routemaster/routemaster.dart';

final loggedOutRoute =
    RouteMap(routes: {'/': (_) => const MaterialPage(child: LoginScreen())});

final loggedInRoute = RouteMap(routes: {
  '/': (_) => const MaterialPage(child: HomeScreen()),
  '/create_community': (_) =>
      const MaterialPage(child: CreateCommunityScreen()),
  '/r/:name': (route) => MaterialPage(
          child: CommunityScreen(
        name: route.pathParameters['name']!,
      )),
  '/mod-tools/:name': (routeData) => MaterialPage(
          child: ModsToolsScreen(
        name: routeData.pathParameters['name']!,
      )),
  '/edit-community/:name': (routeData) => MaterialPage(
          child: EditCommunityScreen(
        name: routeData.pathParameters['name']!,
      )),
  '/add-mods/:name': (routeData) => MaterialPage(
          child: AddModsScreen(
        name: routeData.pathParameters['name']!,
      )),
  '/u/:uid': (routeData) => MaterialPage(
          child: UserProfileScreen(
        uid: routeData.pathParameters['uid']!,
      )),
  '/edit-profile/:uid': (routeData) => MaterialPage(
          child: EditProfileScreen(
```

```
        uid: routeData.pathParameters['uid']!,
      )),
  '/add-post/:type': (routeData) => MaterialPage(
        child: AddPostTypeScreen(
        type: routeData.pathParameters['type']!,
      )),
  '/post/:postId/comments': (route) => MaterialPage(
        child: CommentsScreen(
          postId: route.pathParameters['postId']!,
        ),
      ),
    // '/add-post': (routeData) => const MaterialPage(
    //       child: AddPostScreen(),
    //     ),
});
```

Output:

Project Title: Reddit **Roll No. 36**



MAD & PWA Lab
Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	36
Name	Mohd Usmaan Mogal
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:	15

Project Title: Reddit **Roll No. 36**

Aim: To connect firebase database with flutter ui

Theory:

Firebase is a great backend solution for anyone that wants to use authentication, databases, cloud functions, ads, and countless other features within an app.

In this article, you will create a Firebase project for iOS and Android platforms using Flutter.

Prerequisites

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.

This tutorial was verified with Flutter v2.0.6, Android SDK v31.0.2, and Android Studio v4.1.

Creating a New Flutter Project

This tutorial will require the creation of an example Flutter app.

Once you have your environment set up for Flutter, you can run the following to create a new application:

```
flutter create flutterfirebaseexample
```

1.

Copy

Navigate to the new project directory:

```
cd flutterfirebaseexample
```

```
1.
```

Copy

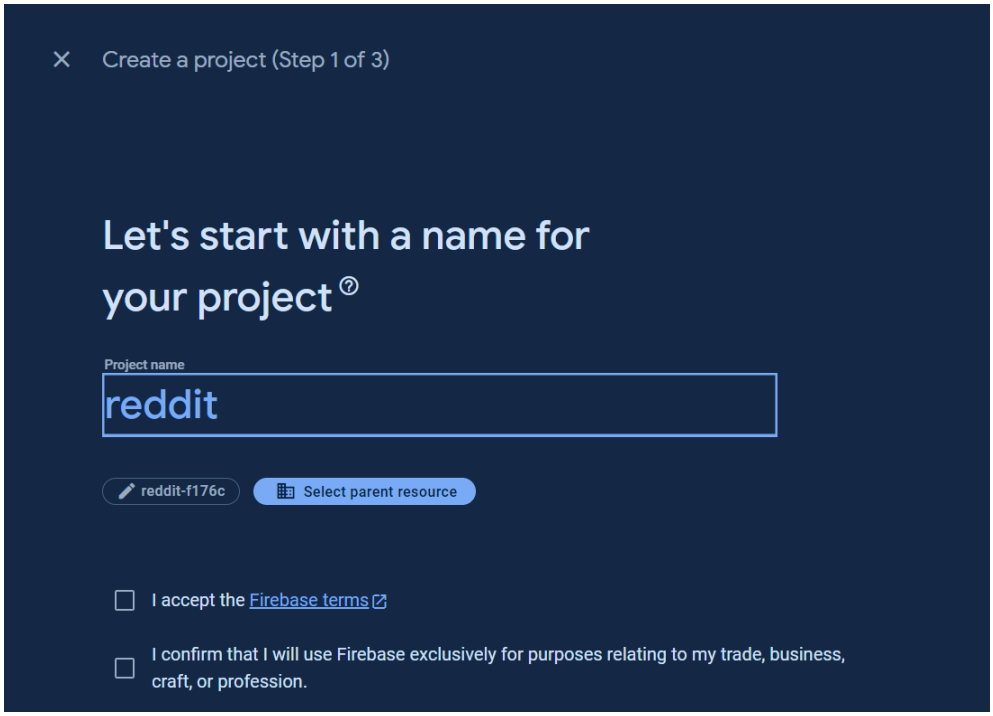
Using `flutter create` will produce a demo application that will display the number of times a button is clicked.

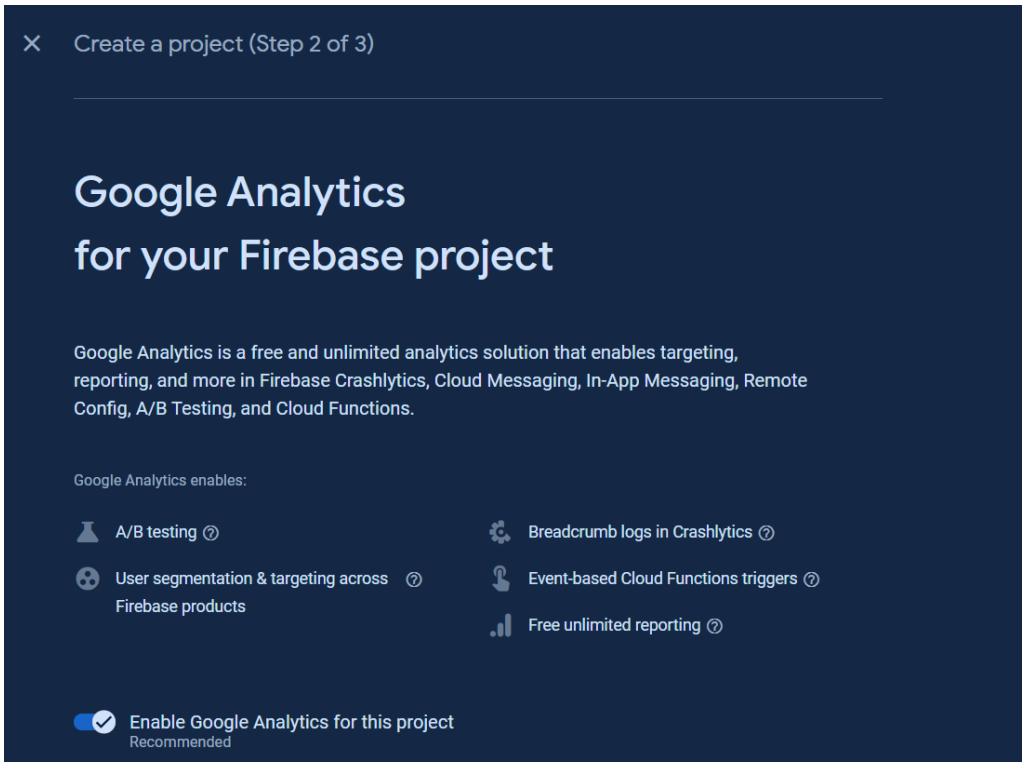
Now that we've got a Flutter project up and running, we can add Firebase.

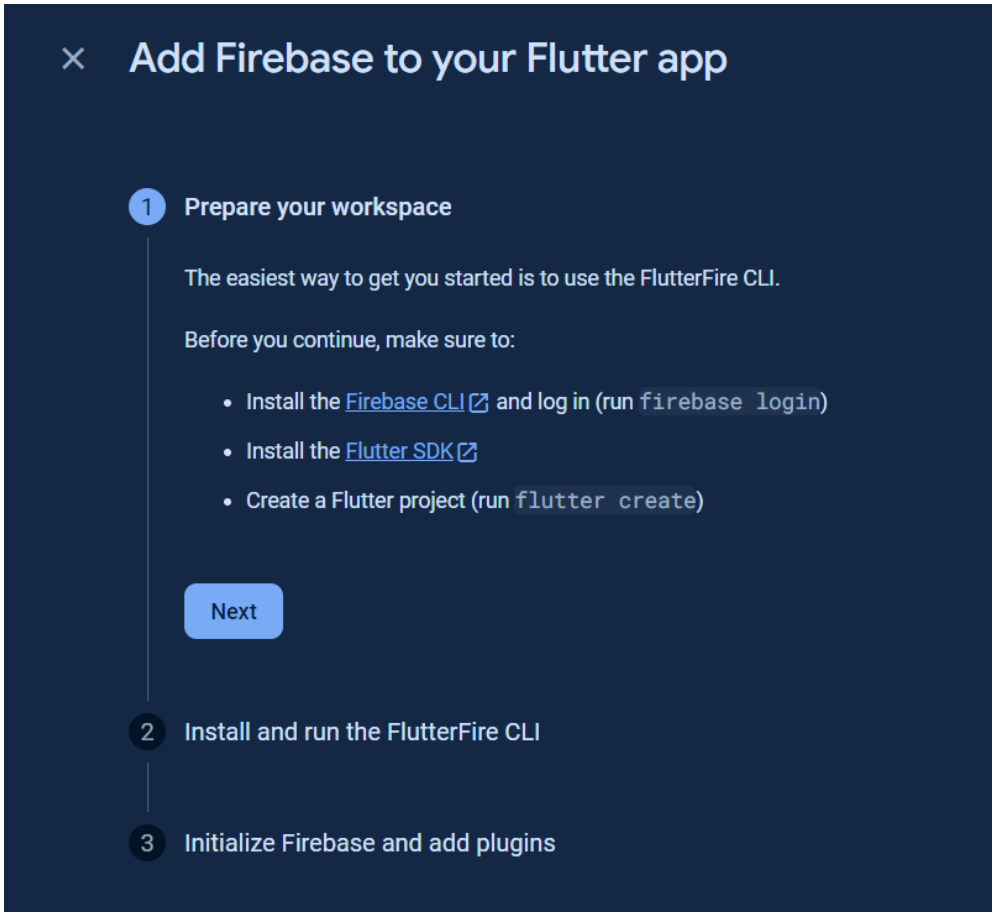
Creating a New Firebase Project

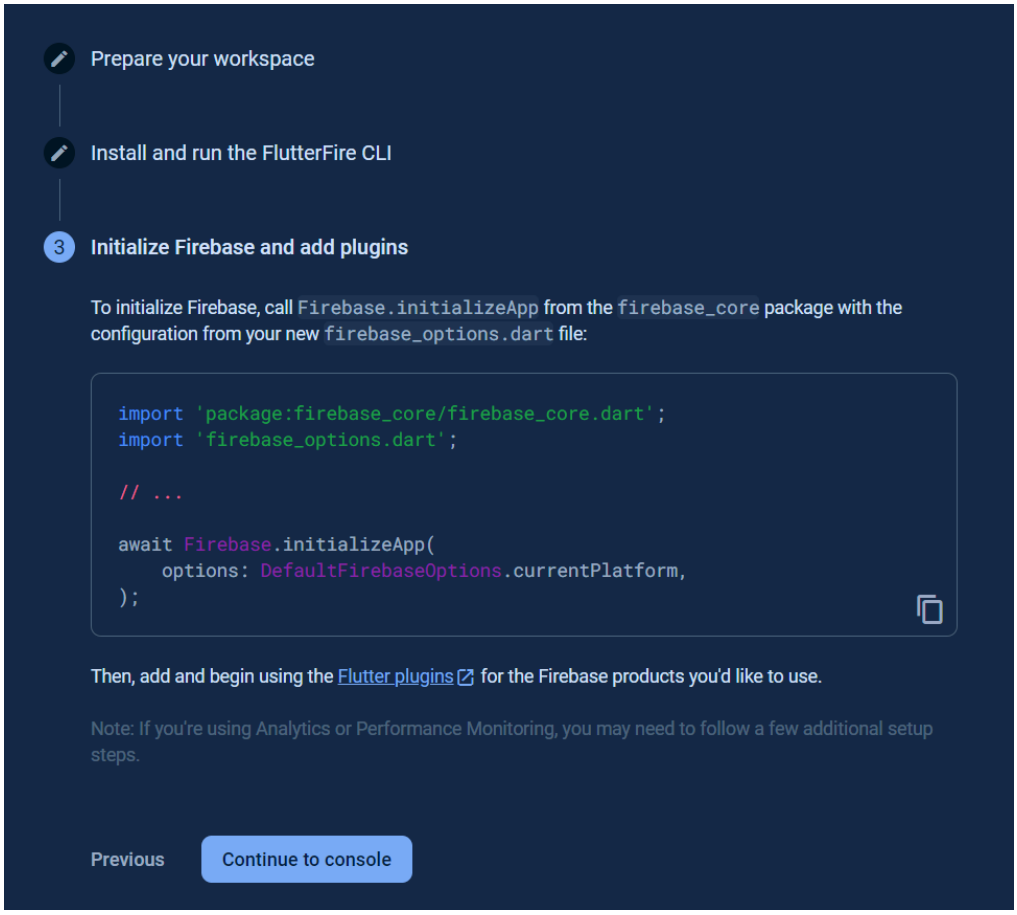
First, log in with your Google account to manage your Firebase projects. From within the Firebase dashboard, select the Create new project button and give it a name:

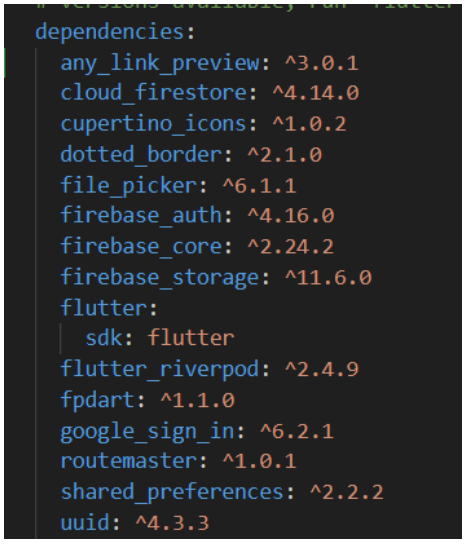
Project Title: Reddit **Roll No. 36**






Add Firebase to your Flutter app
1
Prepare your workspace
The easiest way to get you started is to use the FlutterFire CLI.
Before you continue, make sure to:
Install the Firebase CLI and log in (run firebase login)
Install the Flutter SDK
Create a Flutter project (run flutter create)
Next
2
Install and run the FlutterFire CLI
3
Initialize Firebase and add plugins





Be sure to move this file within Xcode to create the proper file references.

There are additional steps for installing the Firebase SDK and adding initialization code, but they are not necessary for this tutorial.

That's it!

Conclusion

In this article, you learned how to set up and ready our Flutter applications to be used with Firebase.

Flutter has official support for Firebase with the FlutterFire set of libraries.

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.	36
Name	Mohd Usmaan Mogal
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	15

Project Title: Reddit **Roll No. 36**

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

Online Reference:
https://developer.mozilla.org/en-US/docs/Web/Manifest
https://www.geeksforgeeks.org/making-a-simple-pwa-under-5-minutes/
Theory:-

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Linkable — Easily shared via URL without complex installations.

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

Weaknesses refer to:
IOS support from version 11.3 onwards;
Greater use of the device battery;
Not all devices support the full range of PWA features (same speech for iOS and Android operating systems);
It is not possible to establish a strong re-engagement for iOS users (URL scheme, standard web notifications);
Support for offline execution is however limited;
Lack of presence on the stores (there is no possibility to acquire traffic from that channel);
There is no “body” of control (like the stores) and an approval process;
Limited access to some hardware components of the devices;
Little flexibility regarding “special” content for users (eg loyalty programs, loyalty, etc.).

Code:-
manifest.json:-

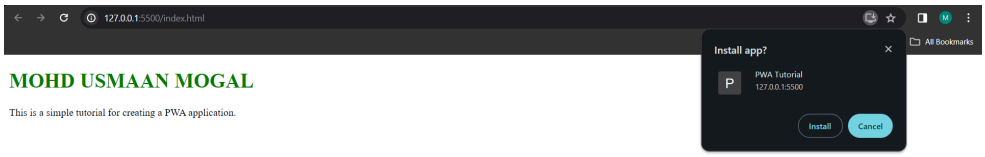
```
{
   "name": "Delice Food App",
   "short_name": "Delice",
   "start_url": "ani.html",
   "scope": "./",
   "icons": [
      {
        "src": "contract.png",
        "sizes": "192x192",
        "type": "image/png"
      },
      {
        "src": "contract.png",
        "sizes": "512x512",
```

```
        "type": "image/png"
      }
    ],
    "theme_color": "#ffd31d",
    "background_color": "#333",
  "display": "standalone"
}
```

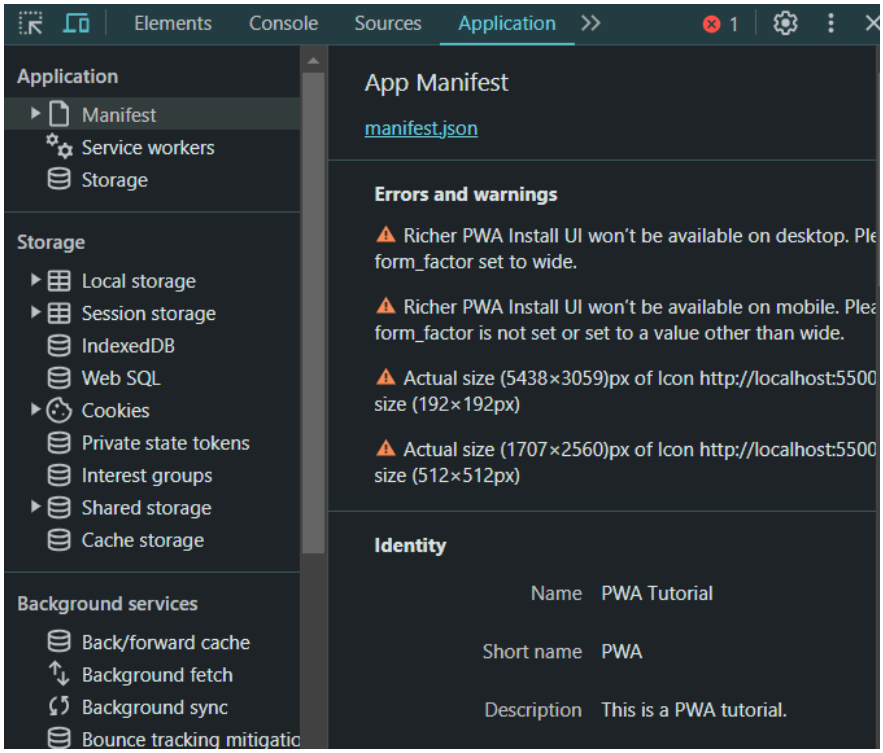
Add the link tag to link to the manifest.json file

```
<html>
  <head>
    <link rel="manifest" href="manifest.json">
    <script src="myscript.js"></script>
    <title>Delice</title>
    <style>
```

Output:-



Project Title: Reddit **Roll No. 36**



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.	36
Name	Mohd Usmaan Mogal
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

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

Theory:

Service Worker

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

Things to note about Service Worker:

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

What can we do with Service Workers?

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

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

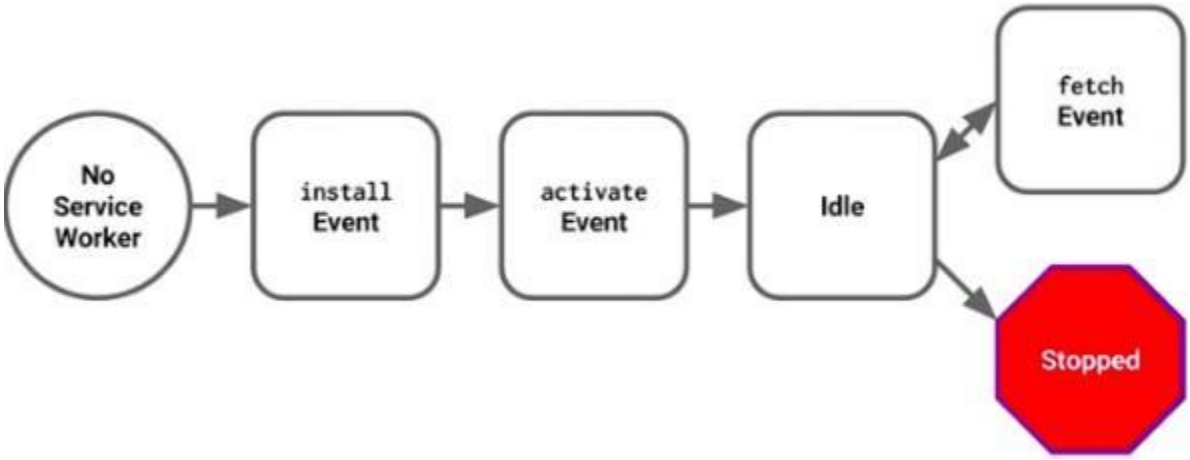
- You can manage **Push Notifications**
  You can manage push notifications with Service Worker and show any information message to the user.
- You can **Continue**
  Although Internet connection is broken, you can start any process with Background Sync of Service Worker.

What can't we do with Service Workers?

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

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

Service Worker Cycle



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

- Registration
- Installation
- Activation

Registration

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

main.js

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

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

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

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

You can also set an arbitrary scope by passing in an additional parameter when

registering. For example: main.js

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

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

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

main.js

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

Installation

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

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

service-worker.js

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

Activation

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

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

service-worker.js

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

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

Code

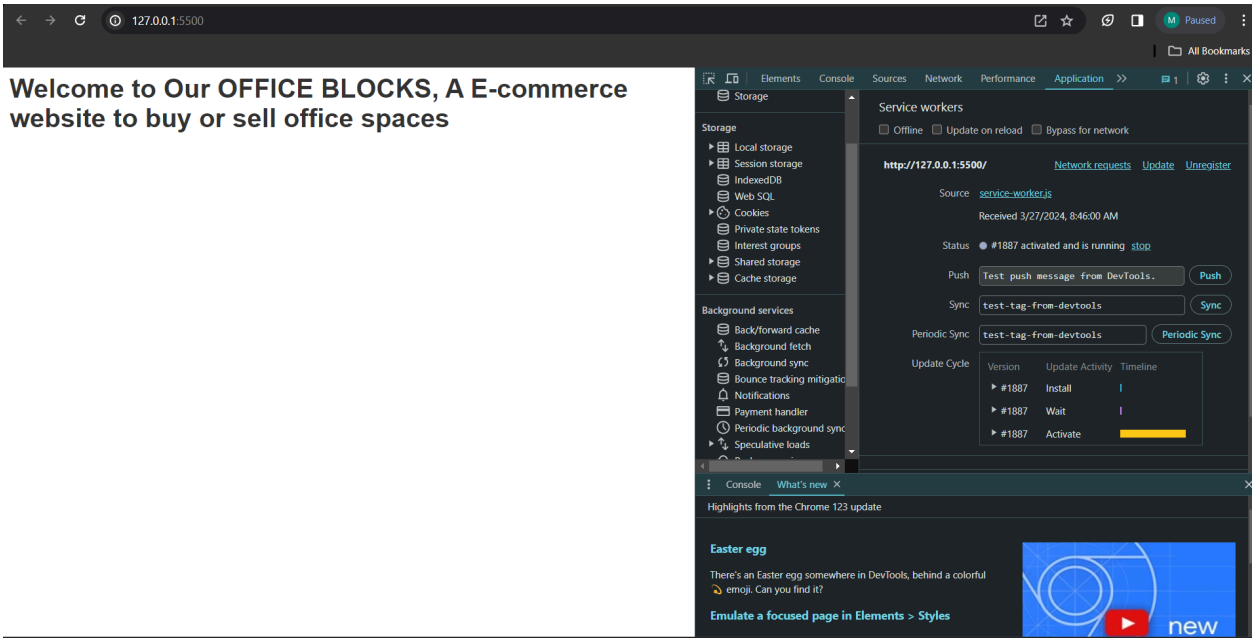
sw.js

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

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

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

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

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

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

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

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

Output:

Project Title: Reddit **Roll No. 36**



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.	36
Name	Mohd Usmaan Mogal
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

Project Title: Reddit **Roll No. 36**

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

Theory:

Service Worker

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

Things to note about Service Worker:

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

Fetch Event

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

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

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

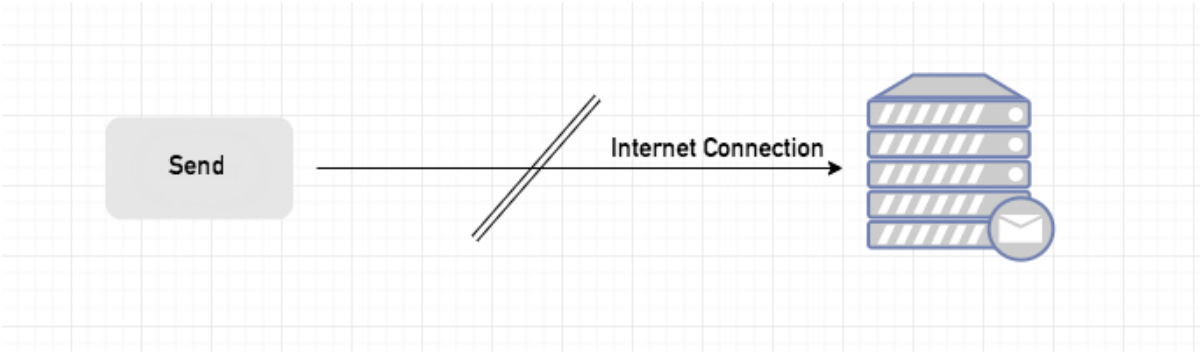
```
self.addEventListener("fetch", function (event) {
    const req = event.request;
    const url = new URL(req.url);

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

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

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

Sync Event

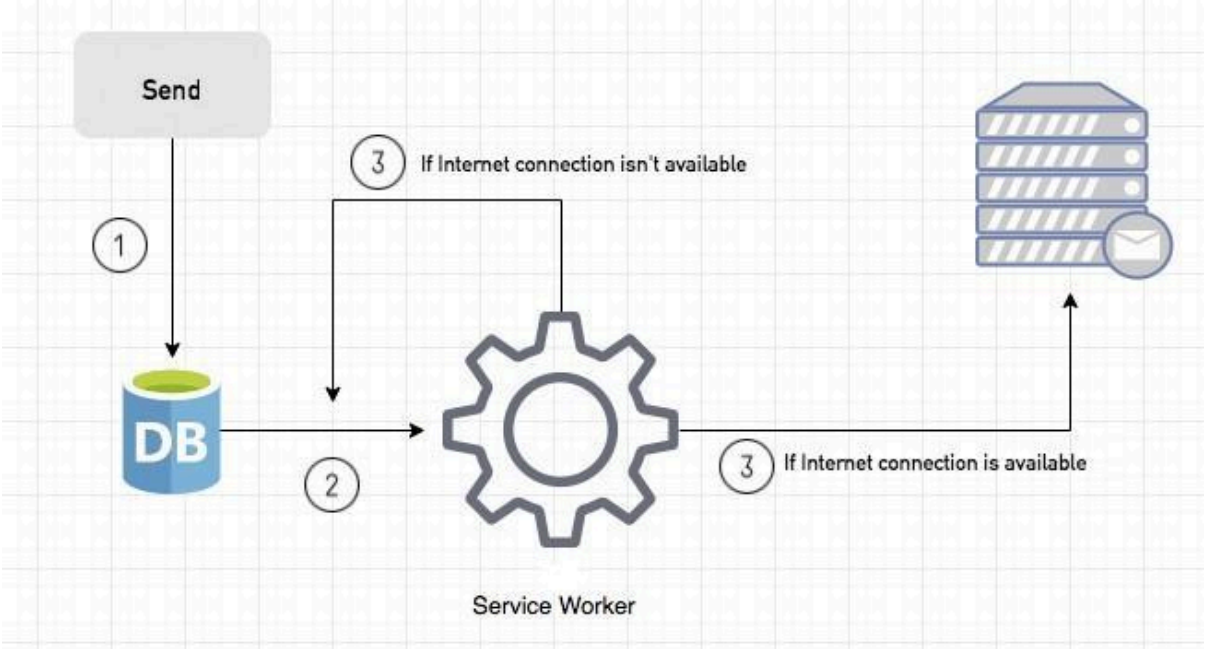
Background Sync is a Web API that is used to delay a process until the Internet connection is stable. We can adapt this definition to the real world; there is an e-mail client application that works on the browser and we want to send an email with this tool. Internet connection is broken while we are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

Here is a job for the Background Sync.

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



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



1. When we click the “send” button, email content will be saved to IndexedDB.
2. Background Sync registration.
3. **If the Internet connection is available**, all email content will be read and sent to Mail Server.
   If the Internet connection is unavailable, the service worker waits until the connection is available even though the window is closed. When it is available, email content will be sent to Mail Server.

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

Event Listener for Background Sync Registration

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

Event Listener for sw.js

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

Push Event

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

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

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

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

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

Code:
sw.js

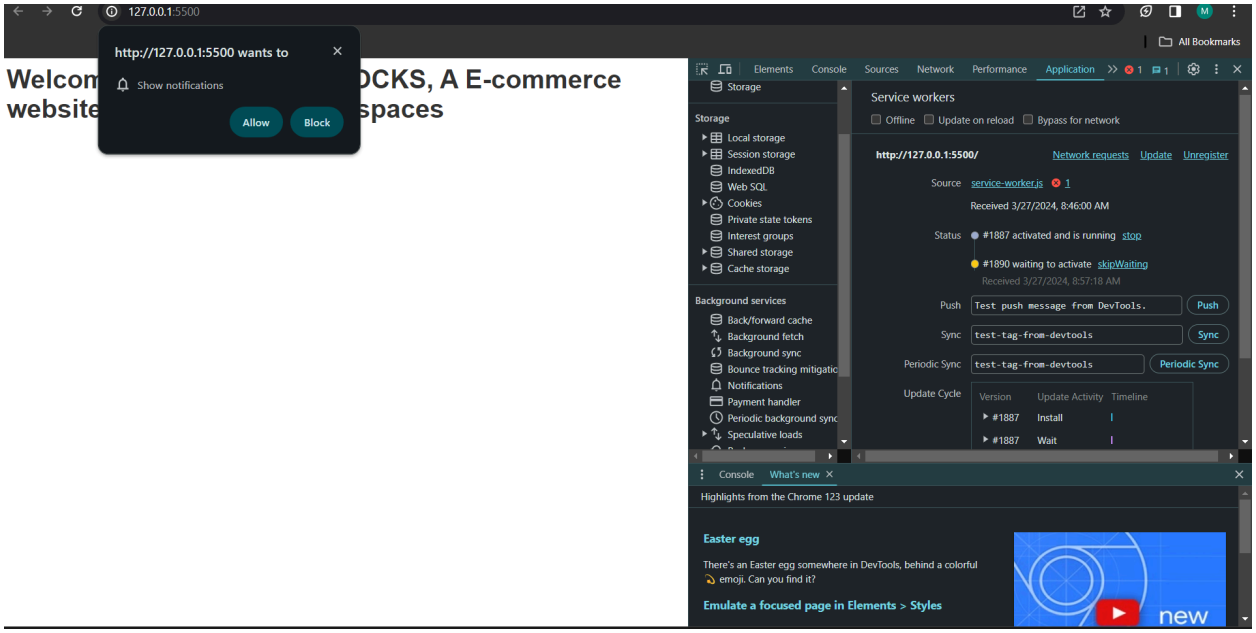
```
self.addEventListener("install", function (event) {
    event.waitUntil(preLoad());
});

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

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

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

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


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

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

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

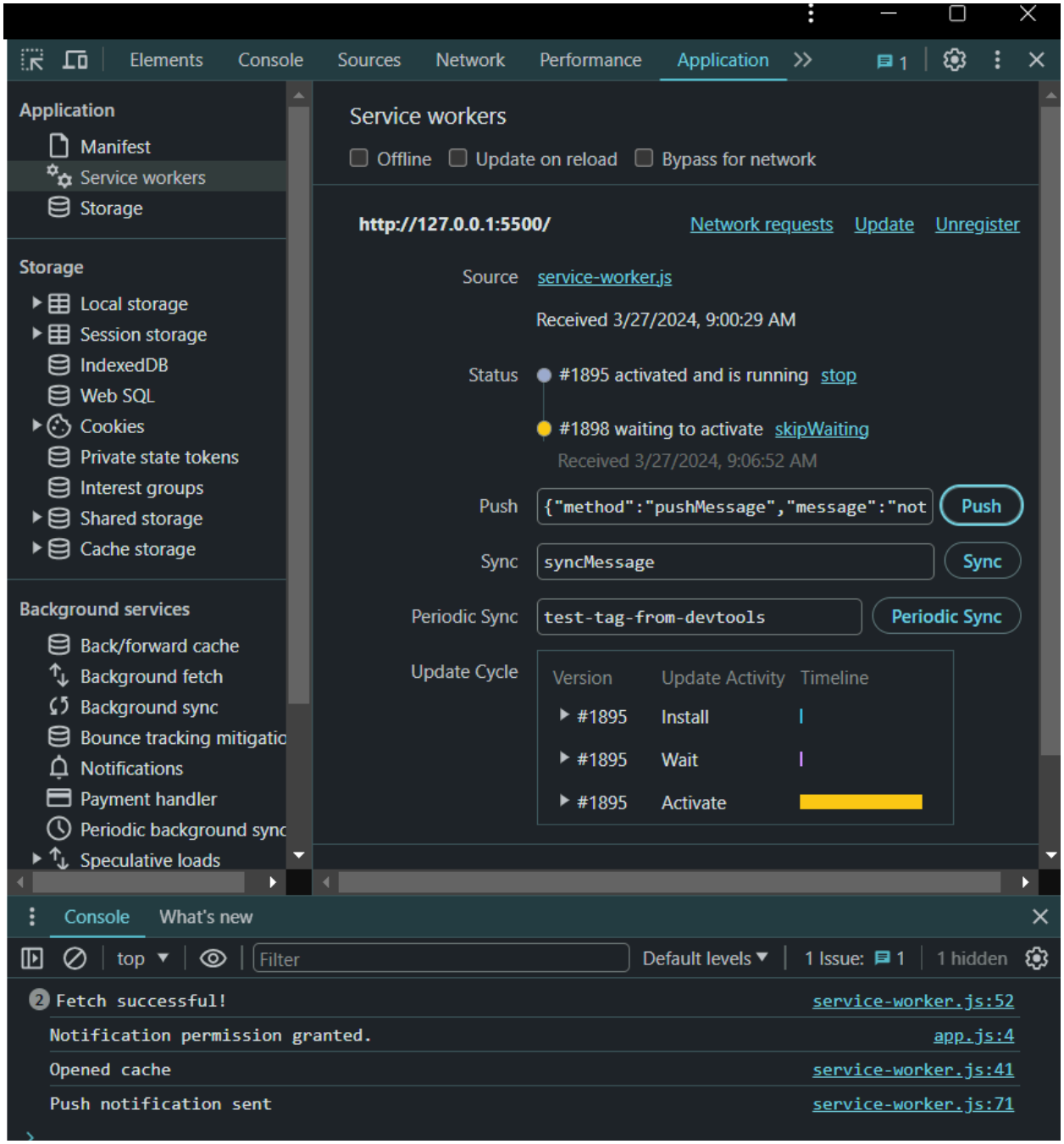
Project Title: Reddit **Roll No. 36**

Output:

Fetch event



Project Title: Reddit **Roll No. 36**



MAD & PWA Lab
Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	36
Name	Mohd Usmaan Mogal
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

Aim:

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

Theory:

GitHub Pages

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

GitHub Pages provides the following key features:

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

Reasons for favoring this over Firebase:

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

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

Pros

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

Cons

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

Firebase

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

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

Some of the features offered by Firebase are:

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

Reasons for favoring over GitHub Pages:

1. Realtime backend made easy
2. Fast and responsive

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

Pros

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

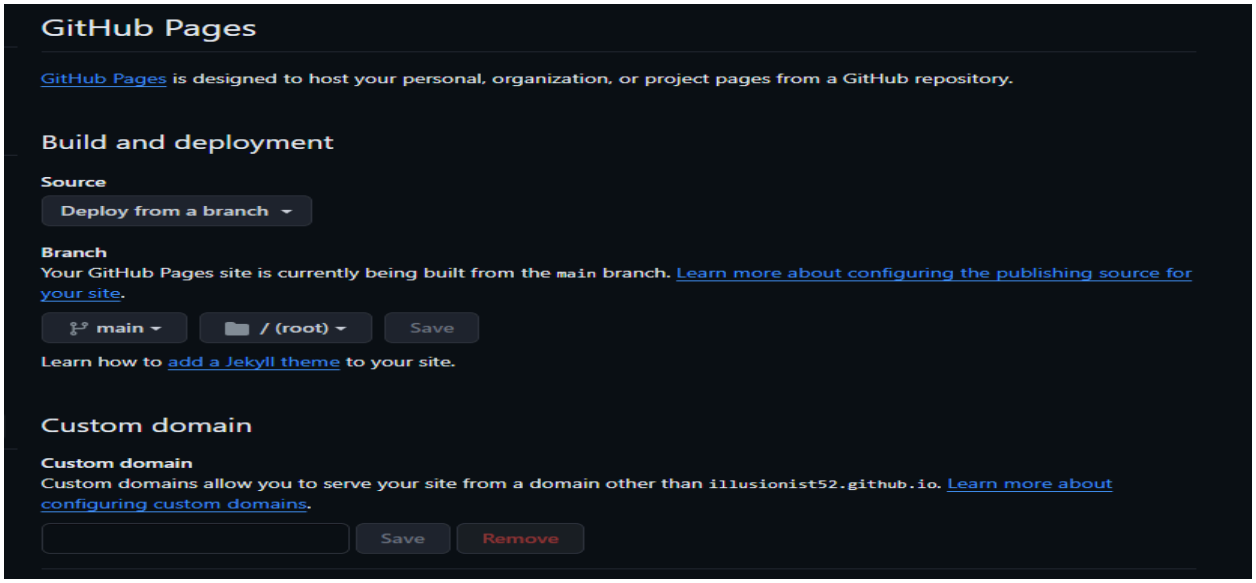
Cons

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

Link to our GitHub repository:

Github Screenshot:

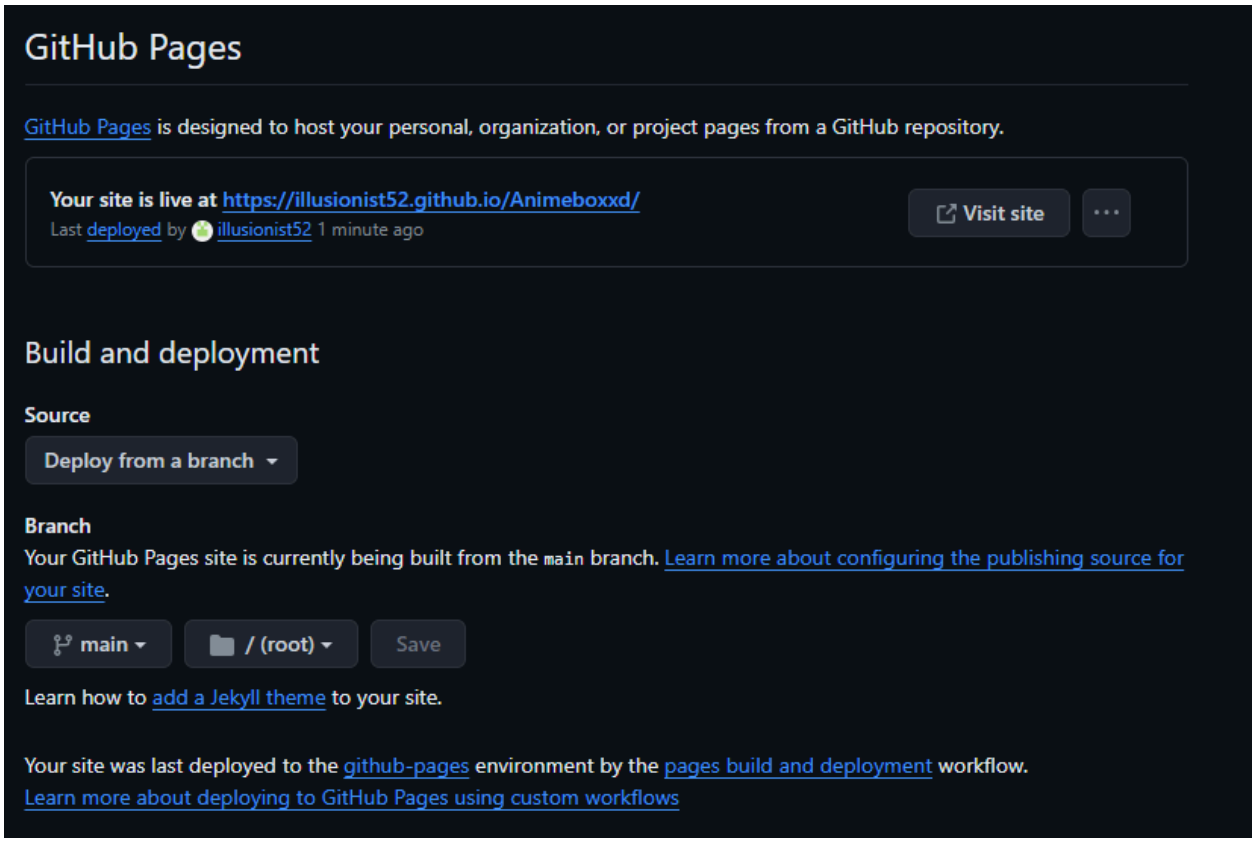
Project Title: Reddit **Roll No. 36**



Animeboxxd

Anime Vault

Project Title: Reddit **Roll No. 36**



\

MAD & PWA Lab
Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	36
Name	Mohd Usmaan Mogal
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	15

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

Theory :

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

Google Lighthouse :

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

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

Key Features and Audit Metrics

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

1. **Performance:** This score is an aggregation of how the page fared in aspects such as (but not limited to) loading speed, time taken for loading for basic frame(s), displaying meaningful content to the user, etc. To a layman, this score is indicative of how decently the site performs, with a score of 100 meaning that you figure in the 98th percentile, 50 meaning that you figure in the 75th percentile and so on.
2. **PWA Score (Mobile):** Thanks to the rise of Service Workers, app manifests, etc., a lot of modern web applications are moving towards

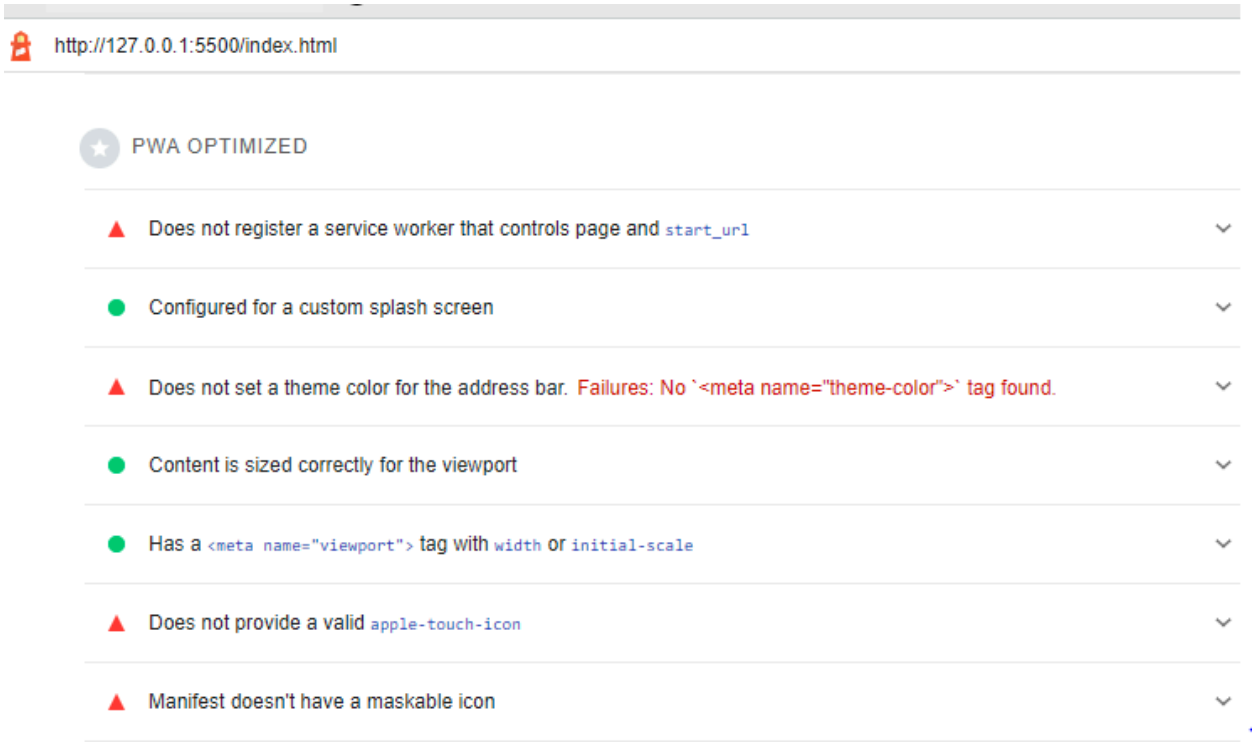
the PWA paradigm, where the objective is to make the application behave as close as possible to native mobile applications. Scoring points are based on the Baseline PWA checklist laid down by Google which includes Service Worker implementation(s), viewport handling, offline functionality, performance in script-disabled environments, etc.

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

   Avoiding the use of deprecated code elements like tags, directives, libraries, etc. Password input with paste-into disabled

   Geo-Location and cookie usage alerts on load, etc.

Changes made to the code :



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

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

For a maskable icon add "purpose": "any maskable" to the icons

in manifest.json file For apple touch icon add the following

meta tag in index.html-

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

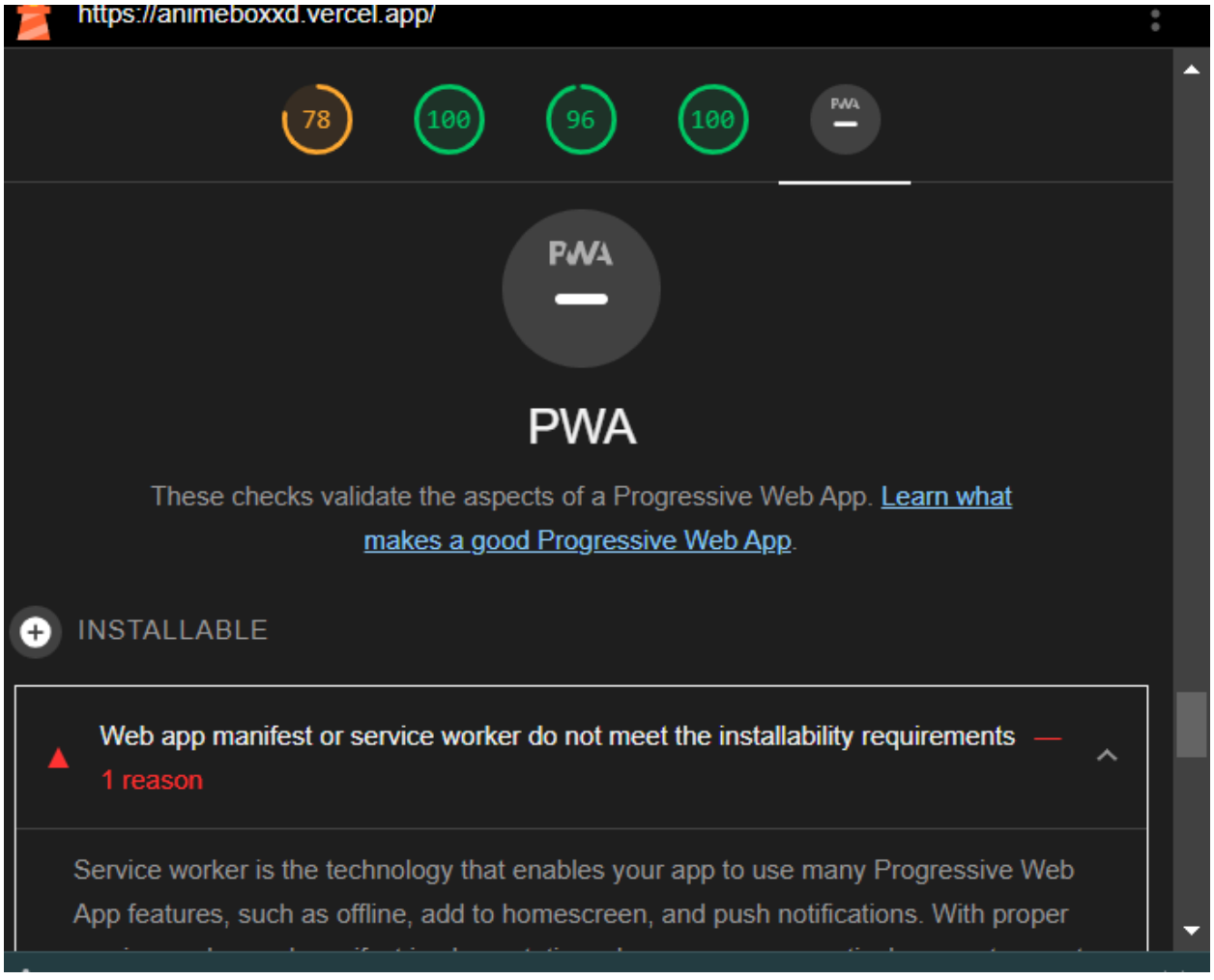
Changes in manifest.json

```
{
  "name":
  "flower
  shop
  website",
  "short_na
```

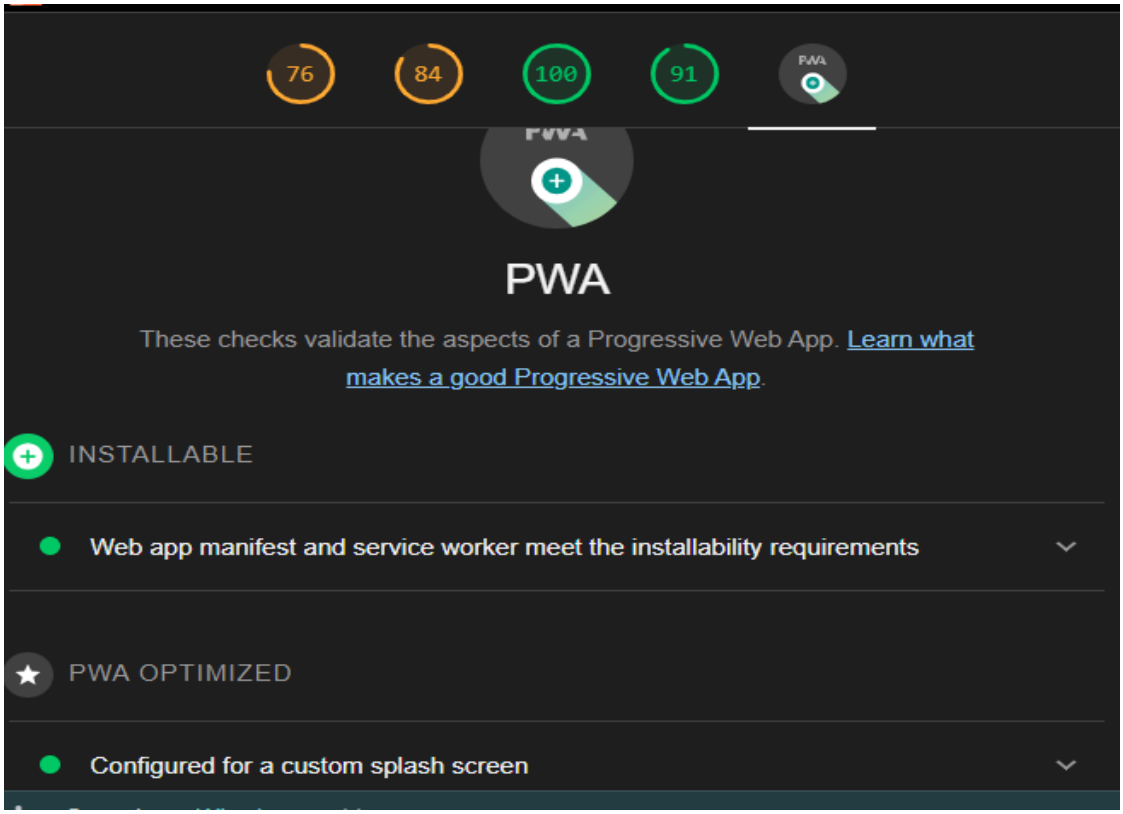
```
me":
"flowers",
"start_url":
"index.htm
l", "scope":
"./",
"theme_
color":
"#ffd31d
",
"backgr
ound_co
lor":
"#333",
"display
":
"standal
one",
"icons":
[
   {
    "src": "icon-1.png",
    "size
    s":
    "192
    x192
    ",
    "typ
    e":
    "ima
    ge/p
    ng"
      “purpose”:"any maskable"

   },
   {
    "src": "icon-2.png",
    "size
    s":
    "512
```

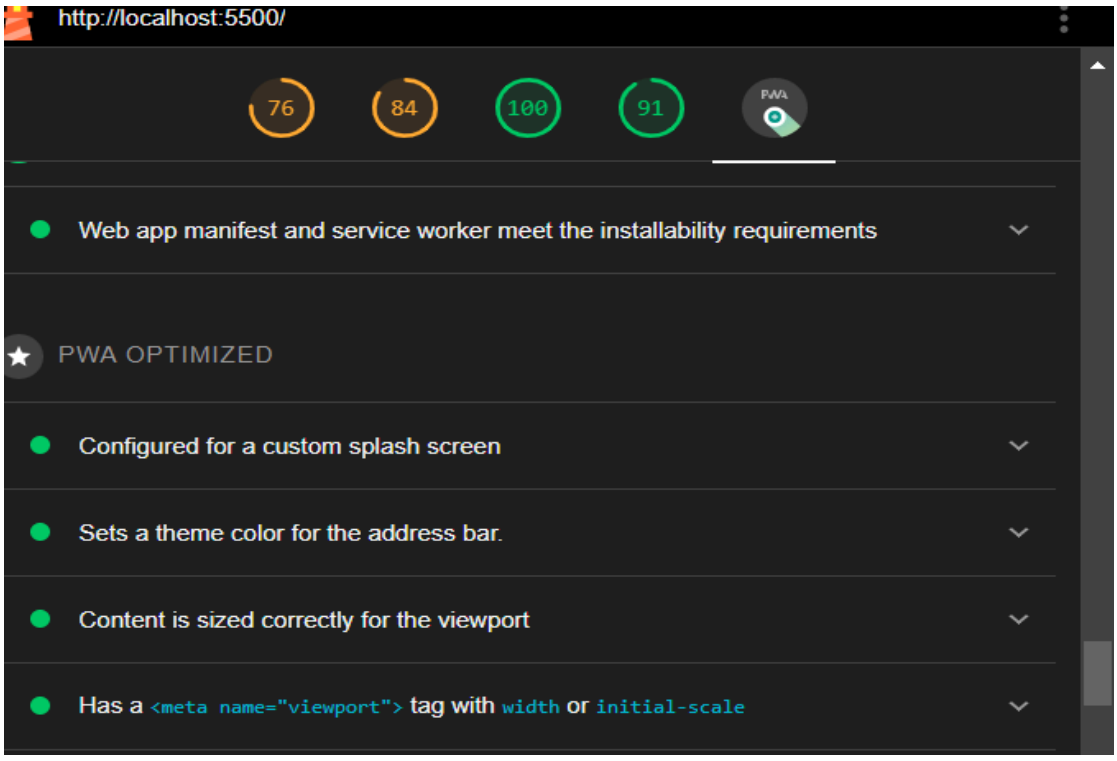
```
                x512
                ",
                "typ
                e":
                "ima
                ge/p
                ng"
                  “purpose”:"any maskable"

              }
            ]
          }
```

Output:



Project Title: Reddit **Roll No. 36**





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.	36
Name	Mohd Usmaan Mogal
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:	5

DATE:

model to ensure that data changes are instantly reflected across all the connected devices. The process involves following key concept

1) Collection and document : Data in firebase organized into collections which can contain multiple documents. Each document is a set of key-value pair

2) Listeners and observable- flutter developers can set up a listener to observe changes in a particular collection or document. These listeners are notified whenever the data in firestore is modified

3) Real time update when the data changes in firestore database the changes are instantly pushed into all connected clients that have set up listeners for that particular data. This real-time update mechanism ensures that the UI reflects the most current data

DATE:

Pros - Simple & easy to implement, offers good performance, well suited for dependency injection
Cons - can be considered overkill for small apps

3) Riverpod - An extension of provider that introduces improvements in terms of readability, scalability and testability
Ideas - offers improved syntax and advanced features compared to provider
Pros - Provides improved readability and state management
- More flexible in terms of state management
Cons - Slightly deeper learning curve compared to provider

Q.17)
→ Integrating firebase with flutter application
1) Create flutter project
2) Register your app of firebase console
3) Add flutter SDK dependencies in your project
4) Initialize firebase in your App
5) Use firebase service in your app

Benefits of using firebase
1) Real-time database
2) Authentication
3) Cloud firestore
4) Cloud function
5) Cloud storage
6) Easy integration
7) Scalability
8) Authentication providers
9) Analytics and crash reporting
10) Firebase hosting

Data segmentation in flutter
Firebase cloud firestore employs a real-time synchronization

DATE:

Importance of state management

1) Reactivity and UI updates : Flutter follows reactive programming model and state management is essential for reflecting changes in the application

2) Performance optimization - Efficient state management optimizes unnecessary widget rebuilds, improving performance by avoiding redundant UI updates

3) Scalability - As application grows in complexity, proper state management becomes crucial for maintaining a manageable and organized database

4) Testing - Proper state management facilitates unit testing and makes it easier to write test cases for different application states

Comparison of state management approaches

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

Use case - Suitable for small to medium sized applications

Pros - Straightforward & easy to use.
- No external dependencies

Cons - Limited to the scope of single widget
- May lead to widget rebuilds higher in widget tree

2) Provider - A popular state management solution that provides global & easily accessible state

Use cases - Good for managing global state, dependency injection

DATE:

There are 2 types of widget tree - 1) Stateless 2) Stateful
Example of commonly used widgets

1) MaterialApp Represents root widget of the application
MaterialApp (home : MyHomepage ());

2) Scaffold - Provides a basic structure for the visual elements such as app bar, body, nav bar
Scaffold (
appBar : AppBar ())
body : MyBody())
)

3) Column/Row - Arranges child widgets vertically or horizontally

4) Container - Basic box model that can contain other widgets and define prop like padding, margin etc

Container (
Margin : EdgeInsets.all (16.0)
Padding : EdgeInsets.symmetric (2°, 16),
child : Text ("Hello")
)

Q3)

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

Traditional approach

1) Developers had to maintain different codebase for iOS and android apps using languages like java, swift, obj-c
2) Change in code used to require rebuilding and restarting the app
3) Achieving a consistent UI across platforms might require using different UI component which was a problem
4) Implementing custom ~~modify~~ animation may be challenging and achieving a highly expressive UI can be time consuming

(Q2)

→ Widget Composition is a powerful concept in flutter that involves combining multiple smaller widgets to create more complex and sophisticated UI. Instead of creating large and monolithic UI component ~~that~~ developers can break it down the UI into smaller reusable widgets and compose them hierarchally to form a ~~com~~ comprehensive UI. Some of its benefits are

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

Widget tree is a fundamental concept that represents the hierarchy of UI elements ~~to~~ in applications. Everything in flutter is a widget from simple elements to more complex ~~stu~~ structures like entire screen or even entire application itself

A key points of widget tree - 1) Hierarchy 2) Composition 3) Reusability
4) Immutability ~~&~~ Not related

Project Title: Reddit **Roll No. 36**

Mohd usmaan mogal
D13A
36

Flutter assignment 1

DATE:

Q1)

→ Flutter is Google's software development kit for building ios and android apps. Flutter allows you to create cross-platform apps that provide native performance. Apps created with flutter feature beautiful and intuitive designs and are able to run animations smoothly. Flutter also increases mobile development speed helping saves costs

Key features and advantages

1) Single Codebase - Code written once can be deployed to ios and android platform This not only reduces development time but also ensures consistency behaviours accross different devices

2) Hot reload - Flutter's hot reload feature allows developers to instantly see the impact of change made to the code without restarting app enabling quicker iteration and debugging

3) Rich widget library - Flutter provides comphrensive set of highly customizable widgets for building UIs

4) Performance - Flutter compiles to native ARM code resulting in high-performance applications. Use of dart language and skia graphics engine contributes smooth animation and fast execution

5) Widget - based architecture - Flutter's architecture revolves around widgets making it modular and easy to organise code. Developers can improve complex UIs by combining smaller, reasonable wid

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.	36
Name	Mohd Usmaan Mogal
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:	4

Project Title: Reddit **Roll No. 36**

Mohd Usmaan Maqsood
BISA
36 MADPWA LAB assignment 2.

1)
→ i) A progressive web app is a type of web application that uses modern web capabilities to provide a user experience similar to that of traditional web app

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

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

4 4

2)
→ i) Responsive web design is an approach to web development that aims to make web pages render well on various devices and window screen sizes

ii) It uses flexible grids and layouts along with media queries to adapt the content and design

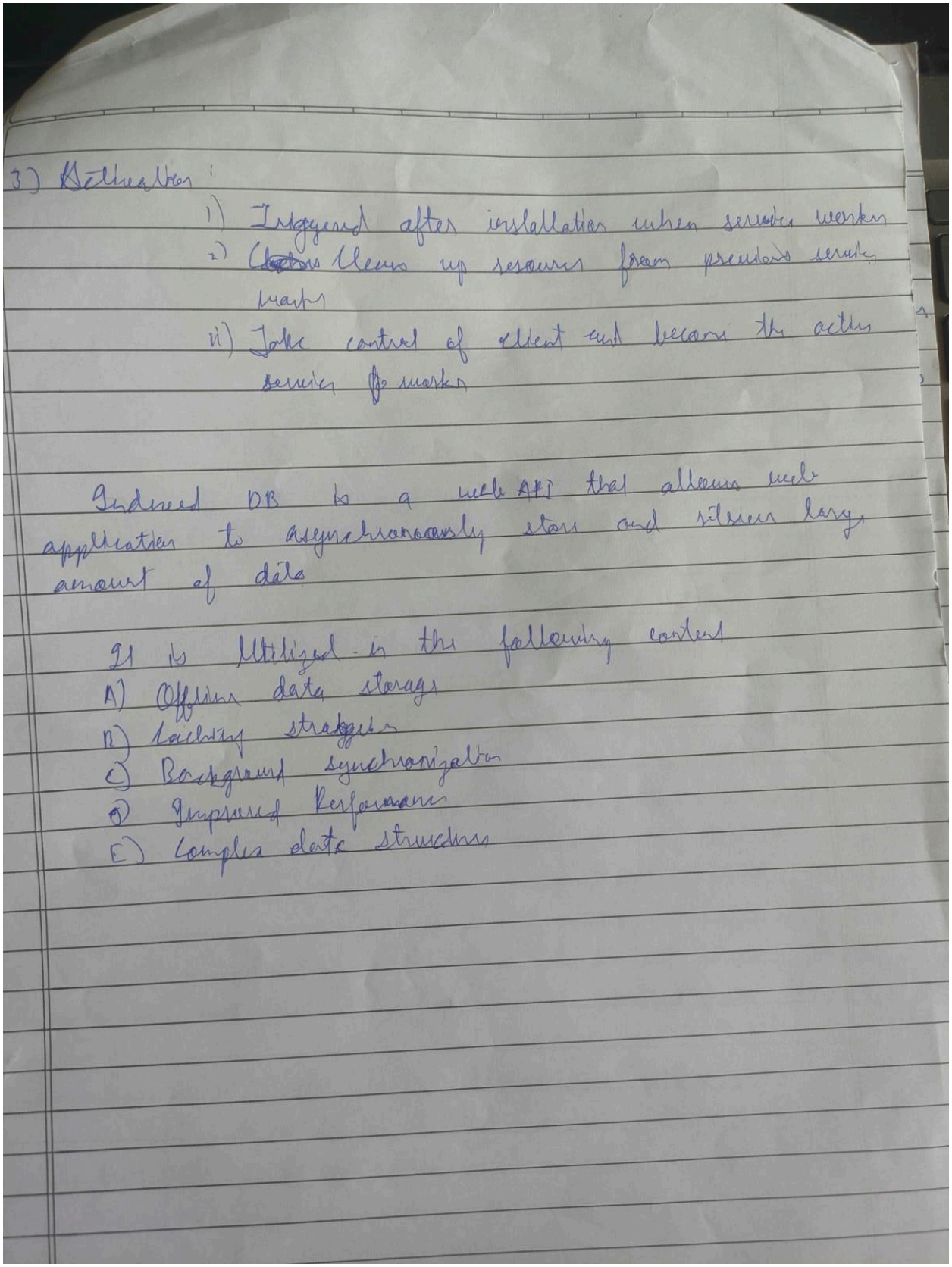
iii) Responsive web design is crucial for PWA's as it ensures that the apps interface adapts seamlessly across different devices maintaining a consistent and user experience

Responsive web design	Fluid web design	Adaptive web design
1) Uses flexible grid and layouts to adapt to any screen	1) Emphasizes on proportional resizing of elements	1) Provides specific layouts for pre-determined breakpoints
2) Adapts through flexible grids and media queries	2) Primarily relies on percentage-based width for fluidity	2) Tailors design specific device categories or screen size
Single codebase with elements adjusting based on screen size	Emphasizes on proportional resizing with fewer media queries	Requires separate layout for different device categories

3) Service workers lifecycle comprises of 3 steps

1) Registration:
   1) Initialization through creation of a JS file
   2) Scope definition to specify controlled pages
   3) Registration using navigator.serviceWorker.register()

2) Installation:
   1) Triggered by a new or updated service worker script
   2) Caches essential resources during installation
   3) Enters the waiting phase when worker is for



3) Activate:

i) Triggered after installation when service worker

ii) Cleans up resources from previous service worker

iii) Take control of client and become the active service worker

Indexed DB is a web API that allows web application to asynchronously store and retrieve large amount of data

It is utilized in the following context

A) Offline data storage

B) Caching strategies

C) Background synchronization

D) Improved Performance

E) Complex data structures