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 **Gargi Angne** 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: Flutter: Youtube Clone / PWA: Travel website **Roll No. 1**

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

Faculty Incharge : Mrs. Kajal Joseph.

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

Email **:** kajal.jewani@ves.ac.in

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

Program specific Outcomes

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

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

Project Title: Flutter: Youtube Clone / PWA: Travel website **Roll No. 1**

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	16/1	23/1	15
2.	To design Flutter UI by including common widgets.	LO2	23/1	30/1	15
3.	To include icons, images, fonts in Flutter app	LO2	30/1	6/2	15
4.	To create an interactive Form using form widget	LO2	6/2	13/2	15
5.	To apply navigation, routing and gestures in Flutter App	LO2	13/2	20/2	15
6.	To Connect Flutter UI with fireBase database	LO3	20/2	5/3	15
7.	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.	LO4	5/3	12/3	15
8.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA	LO5	12/3	19/3	15
9.	To implement Service worker events like fetch, sync and push for E-commerce PWA	LO5	19/3	26/3	15
10.	To study and implement deployment of Ecommerce PWA to GitHub Pages.	LO5	26/3	2/4	15
11.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.	LO6	5/3	12/3	15
12.	Assignment-1	LO1,LO2 ,LO3	2/2	5/2	5
13.	Assignment-2	LO4,LO5 ,LO6	19/3	21/3	4

MAD & PWA Lab
Journal

Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	1
Name	Gargi Angne
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	15

Experiment 1

Aim: To install and configure the Flutter Environment

Theory:

Flutter is an open source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase.
Fast: Flutter code compiles to ARM or Intel machine code as well as JavaScript, for fast performance on any device.
Productive: Build and iterate quickly with Hot Reload. Update code and see changes almost instantly, without losing state.
Flexible: Control every pixel to create customized, adaptive designs that look and feel great on any screen.

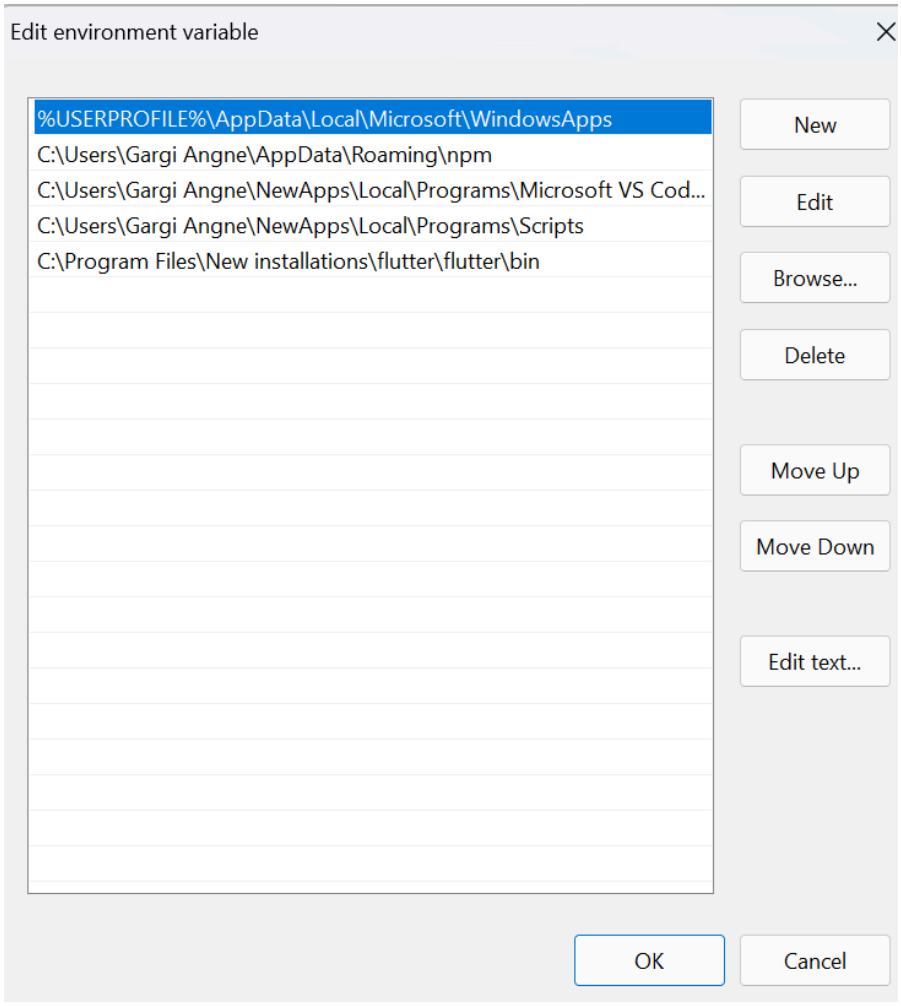
Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for native Android application development.

Flutter is successfully installed

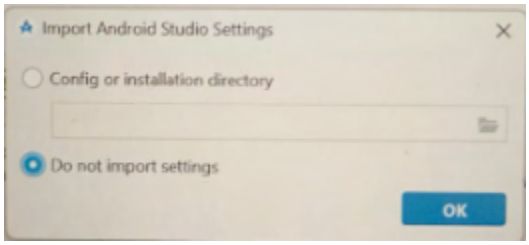
```
PS C:\Users\Student> flutter
Manage your Flutter app development.

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 doctor", shows additional
                            diagnostic information. (Use "-vv" to force verbose logging in those cases.)
-d, --device-id             Target device id or name (prefixes allowed).
    --version               Reports the version of this tool.
    --enable-analytics      Enable telemetry reporting each time a flutter or dart command runs.
    --disable-analytics     Disable telemetry reporting each time a flutter or dart command runs, until it is
                            re-enabled.
    --suppress-analytics    Suppress analytics reporting for the current CLI invocation.

Available commands:

Flutter SDK
```

flutter doctor (command)

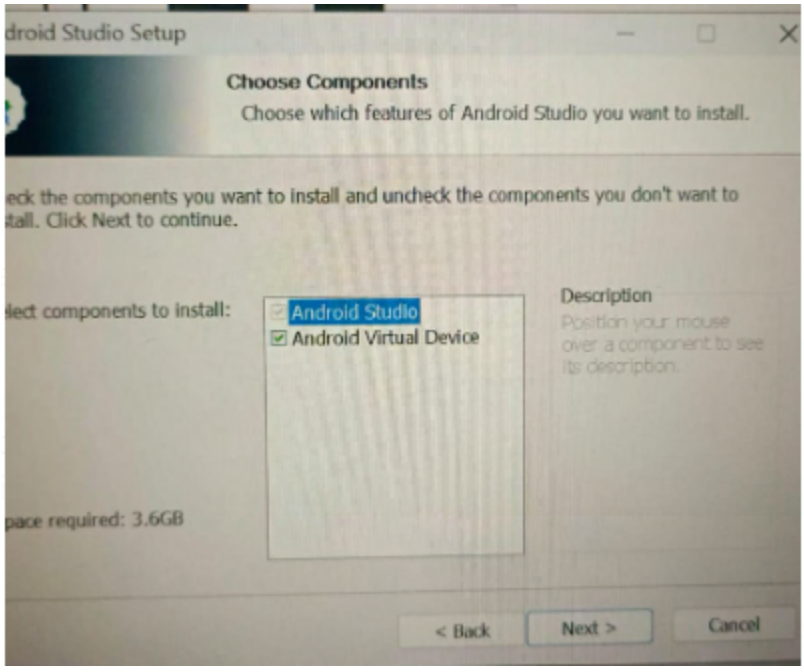
```
Microsoft Windows [Version 10.0.22000.2652]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Student>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.16.7, on Microsoft Windows [Version 10.0.22000.2652], 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)
    X cmdline-tools component is missing
      Run `path/to/sdkmanager --install "cmdline-tools;latest"`
      See https://developer.android.com/studio/command-line for more details.
    X Android license status unknown.
      Run `flutter doctor --android-licenses` to accept the SDK licenses.
      See https://flutter.dev/docs/get-started/install/windows#android-setup for more details.
[√] Chrome - develop for the web
[X] Visual Studio - develop Windows apps
    X Visual Studio not installed; this is necessary to develop Windows apps.
      Download at https://visualstudio.microsoft.com/downloads/.
      Please install the "Desktop development with C++" workload, including all of its default components
[√] Android Studio (version 2023.1)
[√] VS Code (version 1.85.1)
[√] Connected device (3 available)
[!] Network resources
    X A cryptographic error occurred while checking "https://pub.dev/": Connection terminated during handshake
      You may be experiencing a man-in-the-middle attack, your network may be compromised, or you may have malware
      installed on your computer.
    X A cryptographic error occurred while checking "https://maven.google.com/": Connection terminated during handshake
      You may be experiencing a man-in-the-middle attack, your network may be compromised, or you may have malware
      installed on your computer.

! Doctor found issues in 3 categories.
```

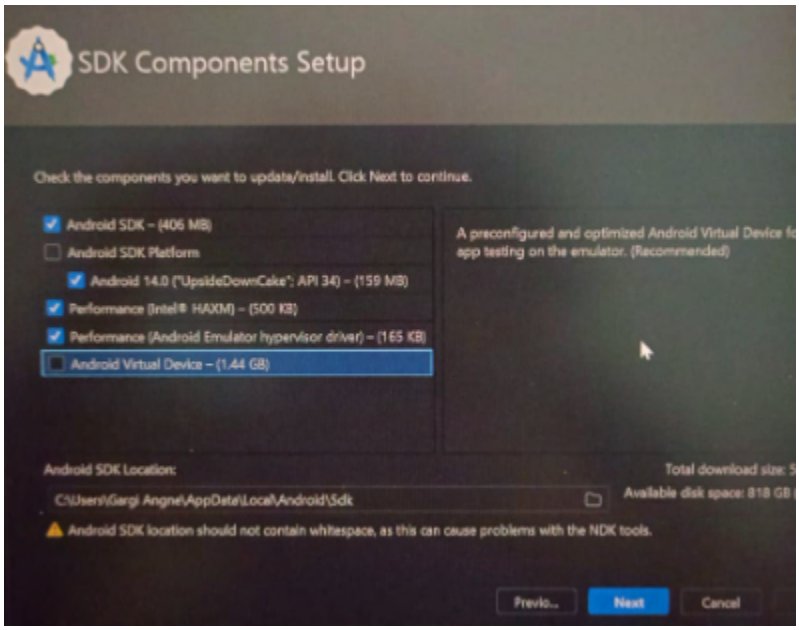


Installing Android Studio:

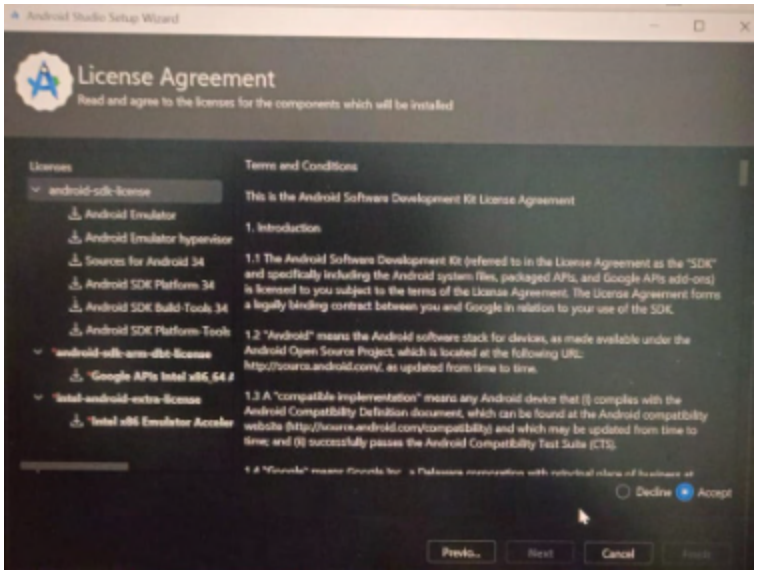


Project Title: Flutter: Youtube Clone / PWA: Travel website **Roll No. 1**

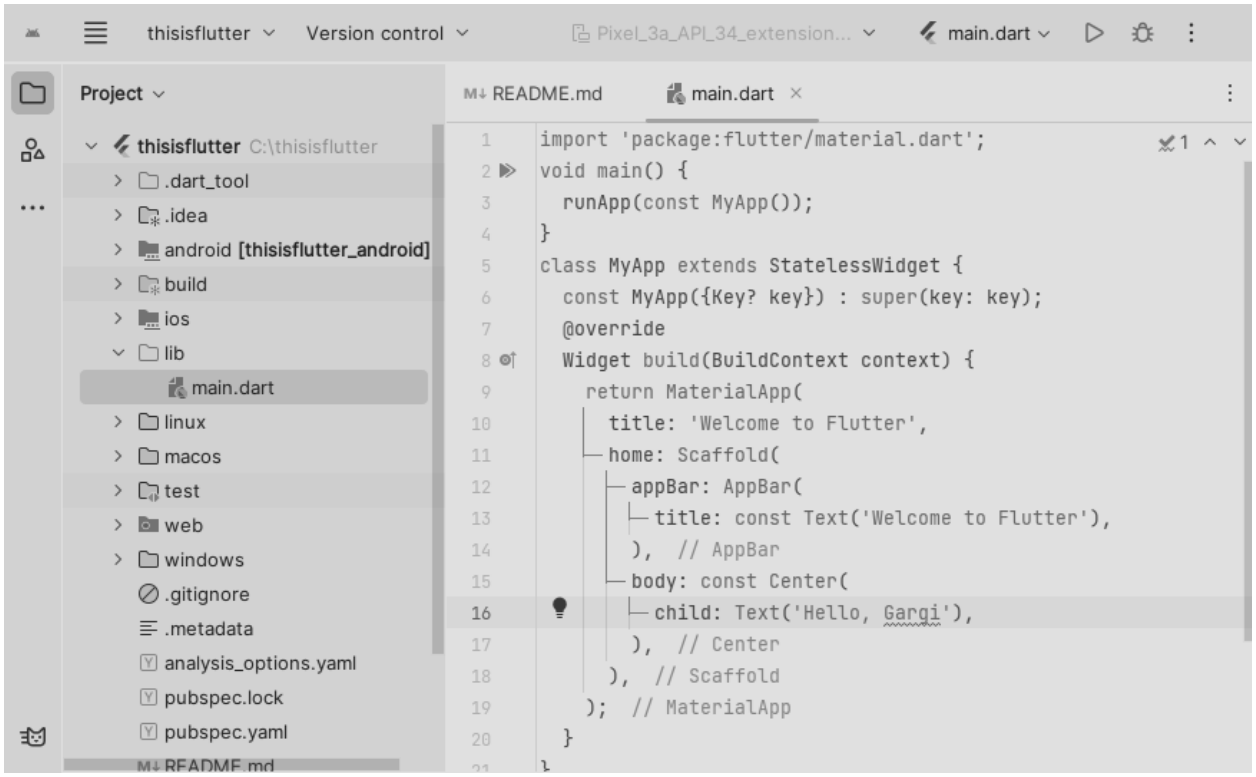




Project Title: Flutter: Youtube Clone / PWA: Travel website **Roll No. 1**



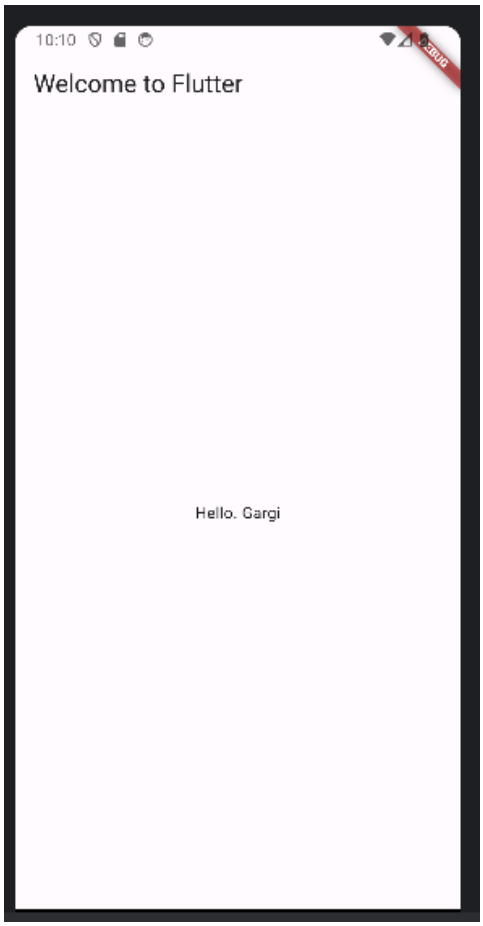
Entering the code in main.dart



import 'package:flutter/material.dart';
void main() {
runApp(const MyApp());
}
class MyApp extends StatelessWidget {
const MyApp({Key? key}) : super(key: key);

```
@override
Widget build(BuildContext context) {
return MaterialApp(
title: 'Welcome to Flutter',
home: Scaffold(
appBar: AppBar(
title: const Text('Welcome to Flutter'),
),
body: const Center(
child: Text('Hello, Gargi’),
),
),
);
}
}
```

OUTPUT :



MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	1
Name	Gargi Angne
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

Experiment 2

Aim: To design Flutter UI by including common widgets.

Theory:

Widgets: Each element on a screen of the Flutter app is a widget. The view of the screen completely depends upon the choice and sequence of the widgets used to build the apps. And the structure of the code of an apps is a tree of widgets.

Category of Widgets:

There are mainly 14 categories in which the flutter widgets are divided. They are mainly segregated on the basis of the functionality they provide in a flutter application.

1. Accessibility: These are the set of widgets that make a flutter app more easily accessible.
2. Animation and Motion: These widgets add animation to other widgets.
3. Assets, Images, and Icons: These widgets take charge of assets such as display images and show icons.
4. Async: These provide async functionality in the flutter application.
5. Basics: These are the bundle of widgets that are absolutely necessary for the development of any flutter application.
6. Cupertino: These are the iOS designed widgets.
7. Input: This set of widgets provides input functionality in a flutter application.
8. Interaction Models: These widgets are here to manage touch events and route users to different views in the application.
9. Layout: This bundle of widgets helps in placing the other widgets on the screen as needed.
10. Material Components: This is a set of widgets that mainly follow material design by Google.
11. Painting and effects: This is the set of widgets that apply visual changes to their child widgets without changing their layout or shape.
12. Scrolling: This provides scrollability of to a set of other widgets that are not scrollable by default.
13. Styling: This deals with the theme, responsiveness, and sizing of the app.
14. Text: This displays text.

Description of few of the widgets are as follows:

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

The code in main.dart:

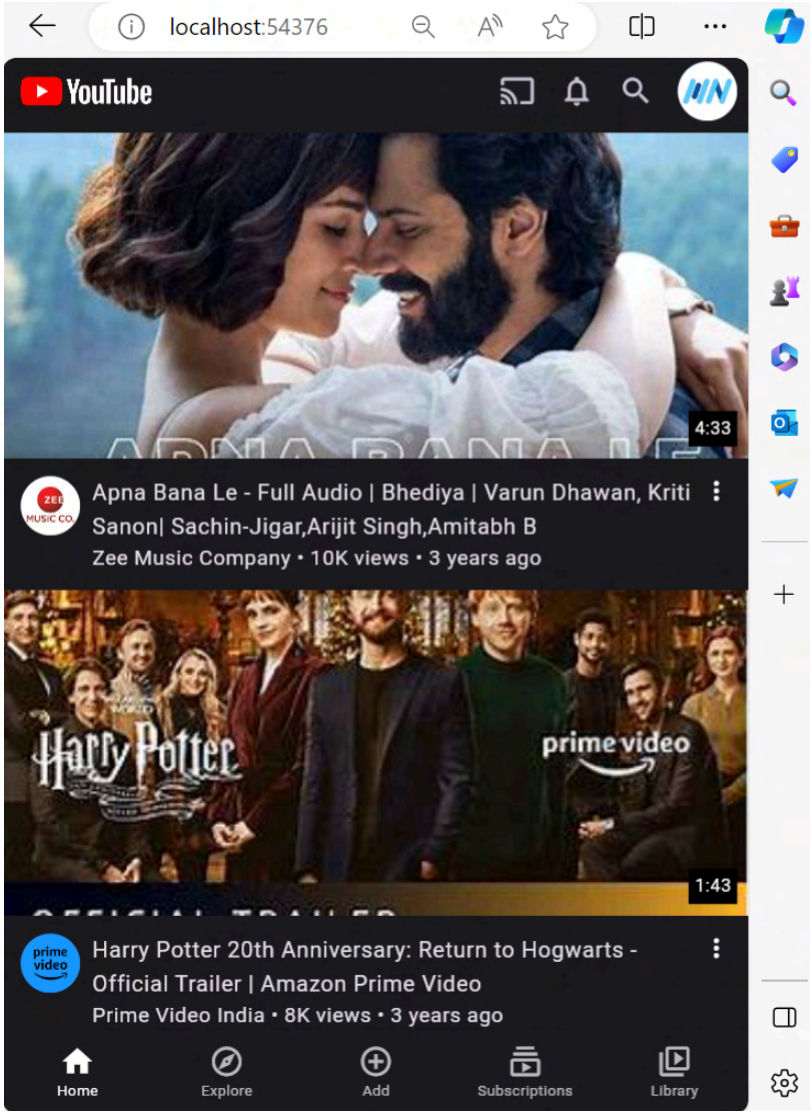
```
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:flutter_youtube_ui/screens/nav_screen.dart';

void main() {
 runApp(ProviderScope(child: MyApp()));
}

class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   SystemChrome.setPreferredOrientations([DeviceOrientation.portraitUp]);
   return MaterialApp(
     title: 'Flutter YouTube UI',
     debugShowCheckedModeBanner: false,
     theme: ThemeData(
       brightness: Brightness.dark,
       bottomNavigationBarTheme:
           const BottomNavigationBarThemeData(selectedItemColor: Colors.white),
     ),
     home: NavScreen(),
   );
 }
}
```

The code in home_screen.dart:

```
import 'package:flutter/material.dart';
import 'package:flutter_youtube_ui/data.dart';
import 'package:flutter_youtube_ui/widgets/widgets.dart';

class HomeScreen extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return Scaffold(
     body: CustomScrollView(
       slivers: [
         CustomSliverAppBar(),
         SliverPadding(
           padding: const EdgeInsets.only(bottom: 60.0),
           sliver: SliverList(
             delegate: SliverChildBuilderDelegate(
               (context, index) {
```

```
              final video = videos[index];
              return VideoCard(video: video);
            },
            childCount: videos.length,
          ),
        ),
      ),
    ],
  ),
 );
 }
}
```

Output:



MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	1
Name	Gargi Angne
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

Experiment 3

Aim: To include icons, images, fonts in Flutter app.

Theory:

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

Steps to Add an Image:

Step 1. Create a new folder

- It should be in the root of your flutter project. You can name it whatever you want, but assets are preferred.
- If you want to add other assets to your app, like fonts, it is preferred to make another subfolder named images.

Step 2. Now you can copy your image to images sub-folder. The path should look like assets/images/yourImage. Before adding images also check the above-mentioned supported image formats.

Step 3. Register the assets folder in pubspec.yaml file and update it.
To add images, write the following code:

```
flutter:
  assets:
      - assets/images/yourFirstImage.jpg
      - assets/image/yourSecondImage.jpg
```

If you want to include all the images of the assets folder then add this:

```
flutter:
  assets:
      - assets/images/
```

Step 4. Insert the image code in the file, where you want to add the image.

Step 5. Now you can save all the files and run the app, you will find the output as shown below.

Icon class in Flutter is used to show specific icons in our app. Instead of creating an image for our icon, we can simply use the Icon class for inserting an icon in our app. For using this class you must ensure that you have set uses-material-design: true in the pubsec.yml file of your object.

Properties:

- color: It is used to set the color of the icon
- size: It is used to resize the Icon
- semanticLable: It comes in play while using the app in accessibility mode (ie, voice-over)
- textDirection: It is used for rendering Icon

Note: The semanticLabel are not visible in the UI.

Code in main.dart:

```
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:flutter_youtube_ui/screens/nav_screen.dart';

void main() {
 runApp(ProviderScope(child: MyApp()));
}

class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   SystemChrome.setPreferredOrientations([DeviceOrientation.portraitUp]);
   return MaterialApp(
     title: 'Flutter YouTube UI',
     debugShowCheckedModeBanner: false,
     theme: ThemeData(
       brightness: Brightness.dark,
       bottomNavigationBarTheme:
           const BottomNavigationBarThemeData(selectedItemColor: Colors.white),
     ),
     home: NavScreen(),
   );
 }
}
```

Code in video_screen.dart:

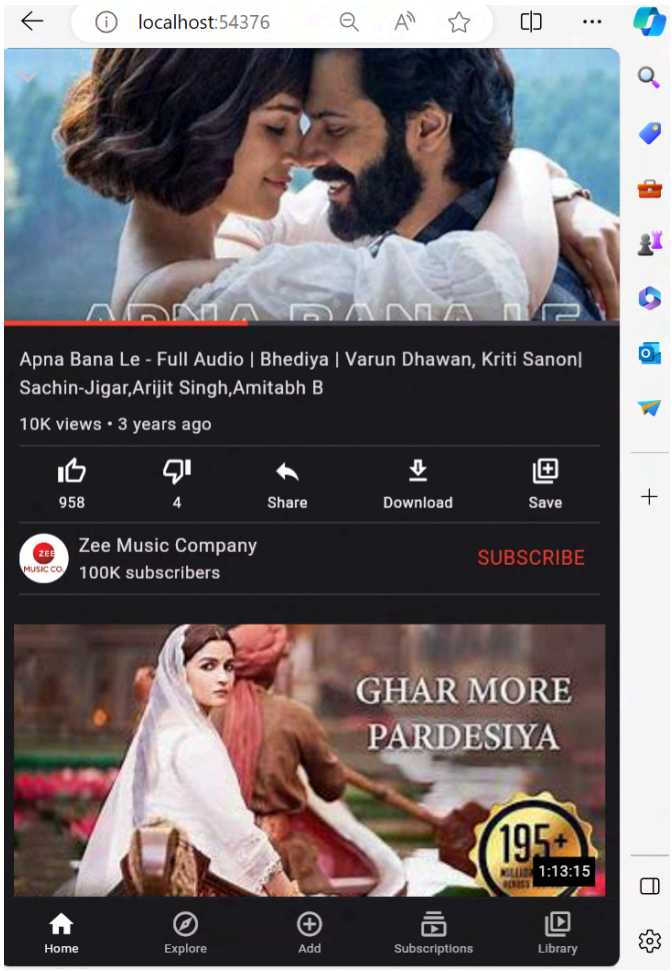
```
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:flutter_youtube_ui/data.dart';
import 'package:flutter_youtube_ui/screens/nav_screen.dart';
import 'package:flutter_youtube_ui/widgets/widgets.dart';
import 'package:miniplayer/miniplayer.dart';

class VideoScreen extends StatefulWidget {
 @override
```

```
 _VideoScreenState createState() => _VideoScreenState();
}

class _VideoScreenState extends State<VideoScreen> {
 ScrollController? _scrollController;

 @override
 void initState() {
  super.initState();
  _scrollController = ScrollController();
 }

 @override
 void dispose() {
  _scrollController?.dispose();
  super.dispose();
 }

 @override
 Widget build(BuildContext context) {
  return GestureDetector(
    onTap: () => context
        .read(miniPlayerControllerProvider)
        .state
        .animateToHeight(state: PanelState.MAX),
    child: Scaffold(
      body: Container(
        color: Theme.of(context).scaffoldBackgroundColor,
        child: CustomScrollView(
          controller: _scrollController,
          shrinkWrap: true,
          slivers: [
            SliverToBoxAdapter(
              child: Consumer(
                builder: (context, watch, _) {
                  final selectedVideo = watch(selectedVideoProvider).state;
                  return SafeArea(
                    child: Column(
                      children: [
                        Stack(
                          children: [
                            Image.network(
                              selectedVideo!.thumbnailUrl,
                              height: 220.0,
```

```
                    width: double.infinity,
                    fit: BoxFit.cover,
                  ),
                  IconButton(
                    iconSize: 30.0,
                    icon: const Icon(Icons.keyboard_arrow_down),
                    onPressed: () => context
                        .read(miniPlayerControllerProvider)
                        .state
                        .animateToHeight(state: PanelState.MIN),
                  ),
                ],
              ),
              const LinearProgressIndicator(
                value: 0.4,
                valueColor: AlwaysStoppedAnimation<Color>(
                  Colors.red,
                ),
              ),
              VideoInfo(video: selectedVideo),
            ],
          ),
        );
      },
    ),
  ),
  SliverList(
    delegate: SliverChildBuilderDelegate(
      (context, index) {
        final video = suggestedVideos[index];
        return VideoCard(
          video: video,
          hasPadding: true,
          onTap: () => _scrollController!.animateTo(
            0,
            duration: const Duration(milliseconds: 200),
            curve: Curves.easeIn,
          ),
        );
      },
      childCount: suggestedVideos.length,
    ),
  ),
],
```

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

Output:


localhost:54376
Apna Bana Le - Full Audio | Bhediya | Varun Dhawan, Kriti Sanon| Sachin-Jigar,Arijit Singh,Amitabh B
10K views • 3 years ago
958
4
Share
Download
Save
Zee Music Company
100K subscribers
SUBSCRIBE
GHAR MORE PARDESIYA
195+
1:13:15
Home
Explore
Add
Subscriptions
Library

MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	1
Name	Gargi Angne
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

Experiment 4

Aim: To create an interactive Form using form widget

Theory:
The Form widget in Flutter is a fundamental widget for building forms. It provides a way to group multiple form fields together, perform validation on those fields, and manage their state.

State Management: Flutter provides various ways to manage the state of interactive forms. You can use StatefulWidget or state management solutions like Provider, Riverpod, or Bloc pattern to handle form data changes efficiently. The choice depends on the complexity and requirements of your application.

Form Widgets: Flutter offers a range of form widgets such as TextField, Checkbox, Radio, DropdownButton, etc., which are used to collect input from users. These widgets can be customized with various parameters to fit the design and functionality of your interactive form.

Validation: Validating user input is essential for ensuring data integrity. Flutter provides a built-in mechanism for form validation using the Form widget along with TextFormField widgets. You can define validation logic for each form field and display error messages accordingly.

Input Handling: Flutter offers various options for handling user input, including keyboard input, gestures, and voice input. You can use event listeners like onChanged, onSubmitted, and onTap to capture user input and update the form state accordingly.

Theming and Styling: Flutter's flexible styling and theming system allow you to customize the appearance of form elements to match your app's design language. You can use themes, colors, fonts, and custom widgets to create visually appealing and consistent interactive forms.

Some Properties of Form Widget

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

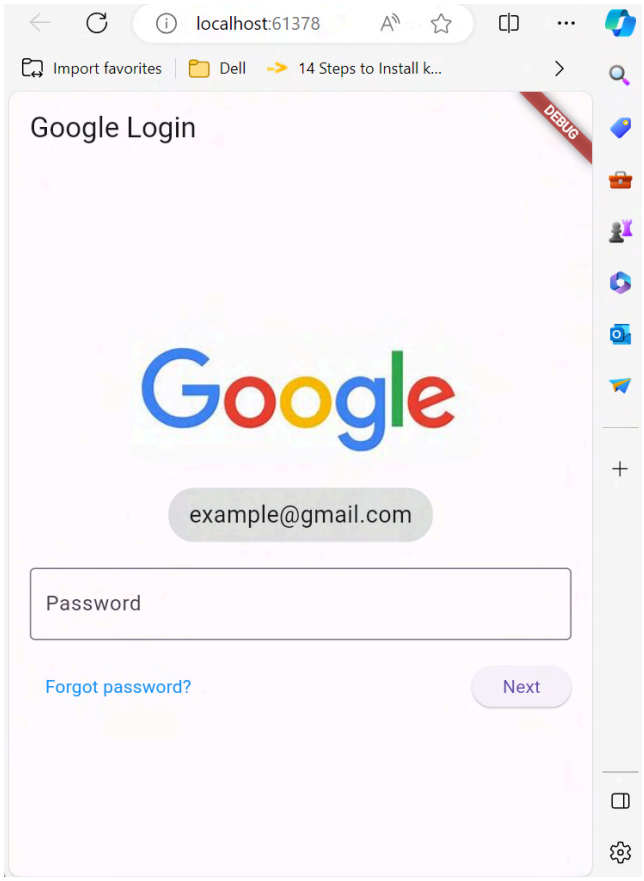
Code in main.dart:

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

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

class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     home: GoogleLoginPage(),
   );
 }
}

class GoogleLoginPage extends StatefulWidget {
 @override
 _GoogleLoginPageState createState() => _GoogleLoginPageState();
}

class _GoogleLoginPageState extends State<GoogleLoginPage> {
 final GlobalKey<FormState> _formKey = GlobalKey<FormState>();

 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       title: Text('Google Login'),
     ),
     body: Padding(
       padding: const EdgeInsets.all(16.0),
       child: Form(
         key: _formKey,
         child: Column(
           mainAxisAlignment: MainAxisAlignment.center,
           children: [
             Image.asset(
               'assets/google_logo.png', // Replace with your Google logo asset
               height: 100,
             ),
             SizedBox(height: 20),
             Container(
               padding: EdgeInsets.symmetric(horizontal: 16, vertical: 8),
```

```
  decoration: BoxDecoration(
    borderRadius: BorderRadius.circular(20),
    color: Colors.grey[300],
  ),
  child: Text(
    'example@gmail.com', // Dummy email address
    style: TextStyle(fontSize: 18),
  ),
),
SizedBox(height: 20),
TextFormField(
  decoration: InputDecoration(
    labelText: 'Password',
    border: OutlineInputBorder(),
  ),
  obscureText: true,
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Password is required';
    }
    return null;
  },
),
SizedBox(height: 20),
Row(
  children: [
    TextButton(
      onPressed: () {
        // Handle forgot password action
        print('Forgot password');
      },
      child: Text(
        'Forgot password?',
        style: TextStyle(color: Colors.blue),
      ),
    ),
  ],
),
SizedBox(height: 20),
ElevatedButton(
  onPressed: () {
    if (_formKey.currentState!.validate()) {
      // Handle form submission
      print('Next');
```

```
                }
              },
              child: Text('Next'),
            ),
          ],
        ),
      ),
    ),
  );
 }
}
```

Output:



MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	1
Name	Gargi Angne
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

Experiment 5

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

Theory:

Flutter offers a comprehensive approach to handle deep links and screen transitions. While the Navigator is suitable for smaller apps with basic deep linking, the Router is recommended for larger apps requiring intricate deep linking. This ensures proper deep link handling across Android, iOS, and web platforms.

Any mobile app's workflow is determined by the user's ability to navigate to various pages; this process is known as routing. The MaterialPageRoute class of Flutter's routing system comes with the two methods that demonstrate how to switch between two routes and they are:

1. Navigator.push() — To navigate to a new route.
2. Navigator.pop() — To return to the previous route.

The Navigator widget manages routes using a stack approach. Routes are stacked based on user actions, and pressing back navigates to the most recent route.

Working of the Navigator:
These following steps show how all the components work together as one piece:

1. The RouteInformationParser converts a new route into a user-defined data type.
2. The RouterDelegate method updates the app state and calls notifyListeners.
3. The Router instructs the RouterDelegate to rebuild via its build() method.
4. The RouterDelegate.build() returns a new Navigator reflecting the updated app state.

Steps to Start Navigation in App:

There are three steps required to start navigation in the app and they are:

1. Create Two Routes: Define the initial and destination routes.
2. Use Navigator.push(): Transition to the new route. To navigate or transition to a new page, route, or screen, the Navigator.push() method is used. The push() method adds a page or route to the stack, and then uses the Navigator to manage it. Once more, we use the MaterialPageRoute class, which enables platform-specific animations for route transitions.
3. Use Navigator.pop(): Navigate back to the initial route. The Navigator controls the stack, and the pop() method enables to remove the current route from it.

The Router and Navigator are designed to work in tandem. By executing imperative methods like push() and pop() on the Navigator, or declarative routing packages like go_router, Router API can ne navigated. Each route on the Navigator is page-backed, which means it was

made from a Page using the pages option on the Navigator constructor, when you navigate using the Router or a declarative routing package. On the other hand, any Route added to the Navigator by calling Navigator.push or showDialog will add a pageless route. Page-backed routes, as opposed to pageless ones, can always be deep-linked when utilising a routing package.

All subsequent pageless routes are also deleted when a page-backed route is deleted from the navigator. For instance, if a deep link navigates by eliminating a page-backed route from the navigator, any pageless _routes that follow up to the following page-backed route are also eliminated.

For a consistent experience when utilising the browser's back and forward buttons, apps that use the Router class integrate with the History API. A History API entry is added to the browser's history stack each time you use the Router to traverse. When the back button is pressed, the Router switches to reverse chronological navigation, which transports the user back to the last

Gestures are used to interact with an application. It is generally used in touch-based devices to physically interact with the application. It can be as simple as a single tap on the screen to a more complex physical interaction like swiping in a specific direction to scrolling down an application. It is heavily used in gaming and more or less every application requires it to function as devices turn more touch-based than ever. In this article, we will discuss them in detail.
Some widely used gestures are mentioned here :

- Tap: Touching the surface of the device with the fingertip for a small duration of time period and finally releasing the fingertip.
- Double Tap: Tapping twice in a short time.
- Drag: Touching the surface of the device with the fingertip and then moving the fingertip in a steadily and finally releasing the fingertip.
- Flick: Similar to dragging, but doing it in a speedier way.
- Pinch: Pinching the surface of the device using two fingers.
- Zoom: Opposite of pinching.
- Panning: Touching the device surface with the fingertip and moving it in the desired direction without releasing the fingertip.

The GestureDetector widget in flutter is used to detect physical interaction with the application on the UI. If a widget is supposed to experience a gesture, it is kept inside the GestureDetector widget. The same widget catches the gesture and returns the appropriate action or response.

Code in main.dart:

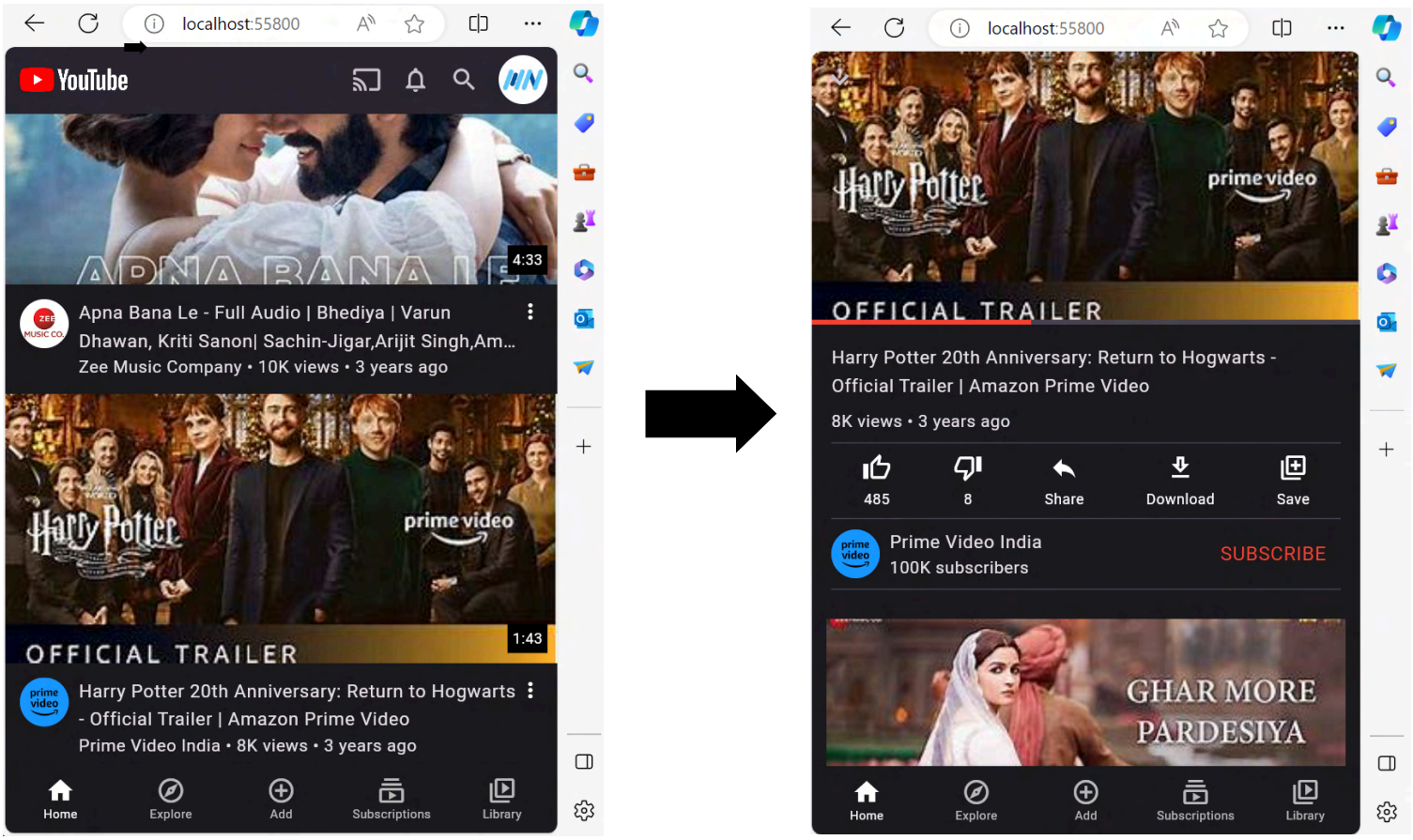
```
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:flutter_youtube_ui/screens/nav_screen.dart';

void main() {
 runApp(ProviderScope(child: MyApp()));
}

class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   SystemChrome.setPreferredOrientations([DeviceOrientation.portraitUp]);
   return MaterialApp(
     title: 'Flutter YouTube UI',
     debugShowCheckedModeBanner: false,
     theme: ThemeData(
       brightness: Brightness.dark,
       bottomNavigationBarTheme:
           const BottomNavigationBarThemeData(selectedItemColor: Colors.white),
     ),
     home: NavScreen(),
   );
 }
}
```

The code in home_screen.dart:

```
import 'package:flutter/material.dart';
import 'package:flutter_youtube_ui/data.dart';
import 'package:flutter_youtube_ui/widgets/widgets.dart';

class HomeScreen extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return Scaffold(
     body: CustomScrollView(
       slivers: [
         CustomSliverAppBar(),
         SliverPadding(
           padding: const EdgeInsets.only(bottom: 60.0),
           sliver: SliverList(
             delegate: SliverChildBuilderDelegate(
               (context, index) {
```

```
              final video = videos[index];
              return VideoCard(video: video);
            },
            childCount: videos.length,
          ),
        ),
      ),
    ],
  ),
 );
 }
}
```

Output:



(Here, clicking on the second video opens up the video player for that video i.e. the second video in this case)

MAD & PWA Lab
Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	1
Name	Gargi Angne
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	15

Experiment 6

Aim: To Connect Flutter UI with fireBase database

Theory:

Firebase is a comprehensive mobile and web application development platform provided by Google. It offers a wide range of features and services that enable developers to build high-quality apps quickly and efficiently.
At its core, Firebase provides tools for various aspects of app development, including authentication, real-time database, cloud storage, hosting, and more. Here's a breakdown of some key Firebase components:

Authentication: Firebase Authentication allows developers to easily integrate secure authentication methods into their apps. It supports various authentication methods, including email/password, phone number, Google, Facebook, Twitter, and more. This feature handles user management, authentication, and security, making it simple to add user sign-up, sign-in, and access control to apps.

Realtime Database: Firebase Realtime Database is a NoSQL cloud database that enables developers to store and sync data in real time. It's particularly useful for applications that require real-time updates, such as chat apps, collaboration tools, and gaming apps. The database is JSON-based, which makes it flexible and easy to use. It automatically synchronizes data across all connected clients and devices, ensuring that changes made by one user are immediately reflected on others' screens.

Cloud Firestore: Cloud Firestore is Firebase's newer, more scalable NoSQL database solution. It offers more powerful querying capabilities, better scalability, and improved performance compared to the Realtime Database. Firestore organizes data into collections and documents, allowing for more structured and efficient data storage. It also supports real-time updates, offline data access, and powerful querying capabilities.

Cloud Storage: Firebase Cloud Storage provides secure and reliable cloud storage for user-generated content, such as images, videos, and documents. It allows developers to easily upload and download files from their apps using simple APIs. Cloud Storage integrates seamlessly with other Firebase services, making it easy to store and serve user-generated content in Firebase-powered apps.

Hosting: Firebase Hosting allows developers to quickly and securely deploy web apps and static content to a global content delivery network (CDN). It provides fast and reliable hosting with SSL encryption, custom domain support, and automatic scaling. With Firebase Hosting, developers can deploy web apps with just a few simple commands, eliminating the need for complex server configurations and maintenance.

Cloud Functions: Firebase Cloud Functions allow developers to run backend code in response to events triggered by Firebase features and HTTPS requests. It enables developers to extend the functionality of their apps without managing servers or infrastructure. Cloud Functions are written in JavaScript or TypeScript and can be deployed with a single command using the Firebase CLI (Command Line Interface).

Analytics: Firebase Analytics provides insights into app usage and user behavior, helping developers understand how users interact with their apps. It tracks key metrics such as active users, retention, engagement, and conversion, allowing developers to make informed decisions about app improvements and optimizations.

Performance Monitoring: Firebase Performance Monitoring helps developers identify and fix performance issues in their apps. It provides detailed insights into app performance, including app startup time, network latency, and UI responsiveness. Performance Monitoring helps developers optimize their apps for better user experiences and higher ratings.

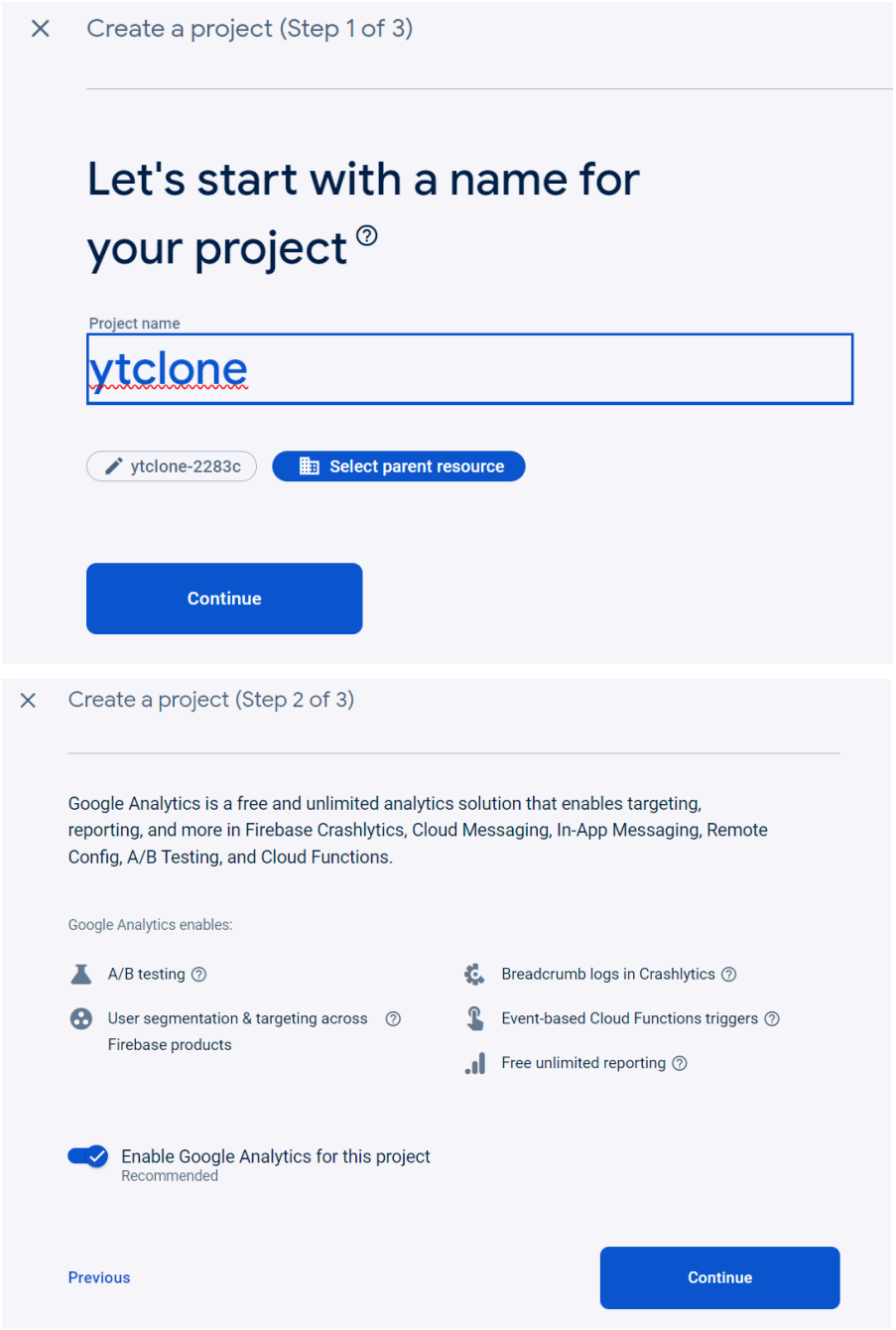
In addition to these core features, Firebase offers many other services, including Cloud Messaging for push notifications, Remote Config for dynamic app configuration, Machine Learning for integrating machine learning models into apps, and more. Overall, Firebase provides a powerful and comprehensive platform for building, managing, and scaling mobile and web applications.

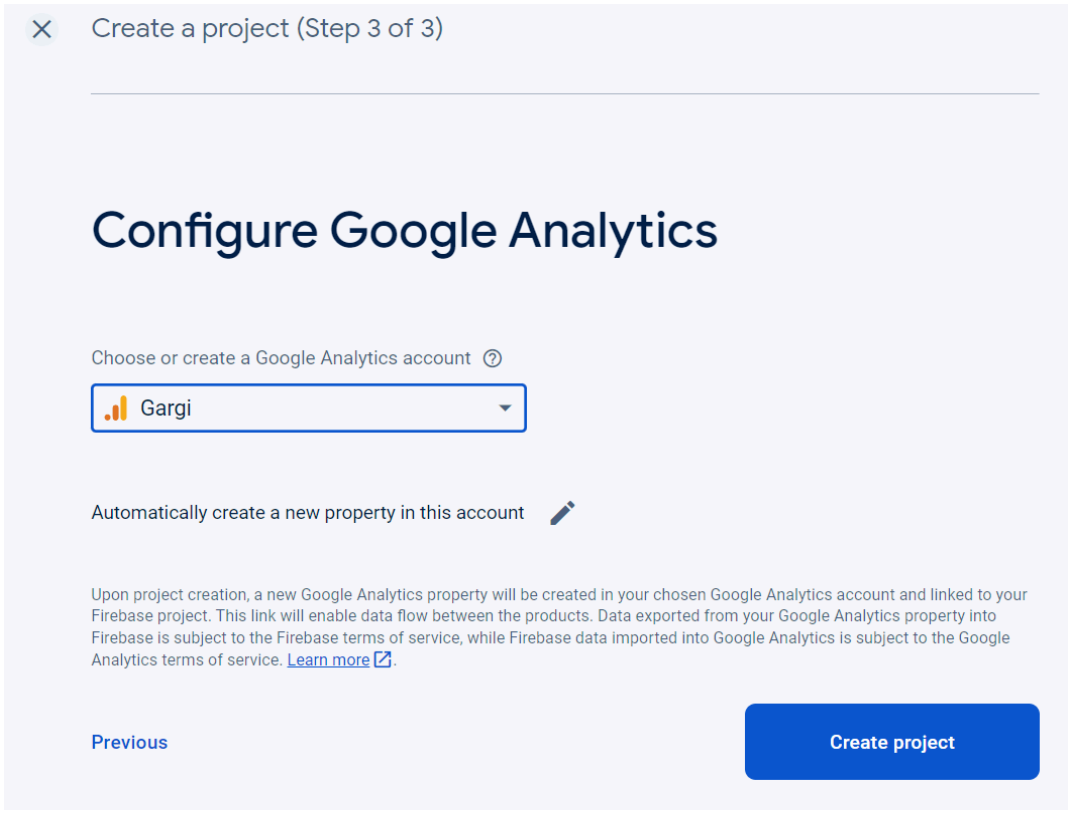
Code in main.dart:

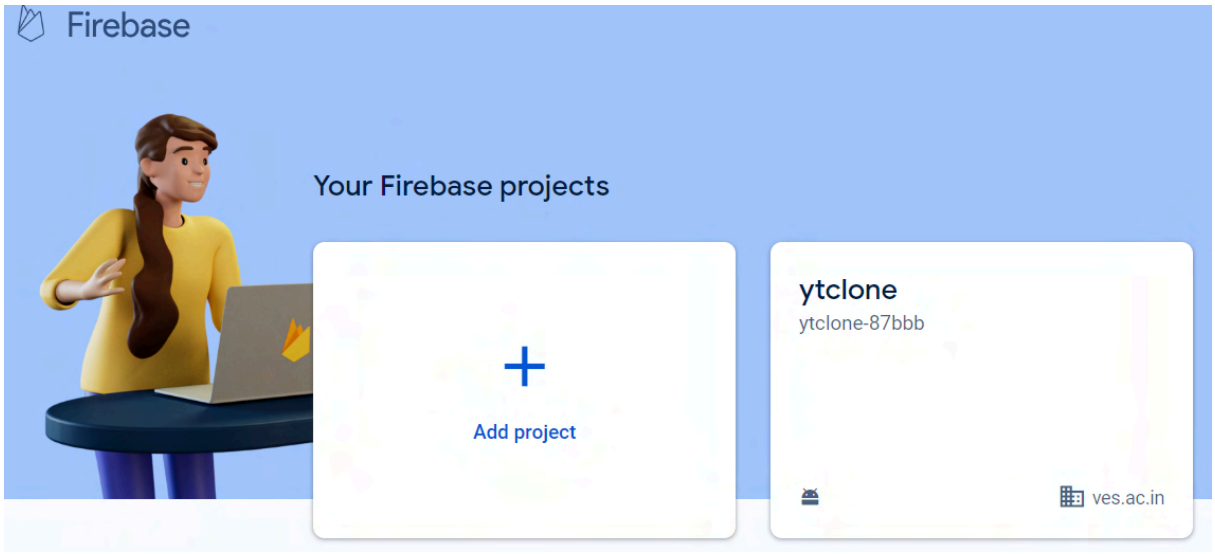
```
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:youtube/login.dart';

void main() async{
 WidgetsFlutterBinding.ensureInitialized();
 try {
      await Firebase.initializeApp(
        options: FirebaseOptions(
          apiKey: "AIzaSyAyMitBa5a5qF9SSVLyeeu3aZUce_f_y3c", // paste your api key here
          appId: "1:702995124628:android:119b3a052e863e5f757f81", //paste your app id here
          messagingSenderId: "702995124628", //paste your messagingSenderId here
          projectId: "ytclone-87bbb", //paste your project id here
        ),
   );
      runApp(ProviderScope(child: MyApp()));
 } catch (e) {
   print('Error initializing Firebase: $e');
   // Handle the error gracefully, possibly show an error dialog or fallback screen
 }
}

class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   SystemChrome.setPreferredOrientations([DeviceOrientation.portraitUp]);
   return MaterialApp(
     title: 'Flutter YouTube UI',
     debugShowCheckedModeBanner: false,
     theme: ThemeData(
       brightness: Brightness.dark,
       bottomNavigationBarTheme:
       const BottomNavigationBarThemeData(selectedItemColor: Colors.white),
     ),
     home: GoogleSignInScreen(),
   );
 }

}
```

Project Title: Flutter: Youtube Clone / PWA: Travel website **Roll No. 1**

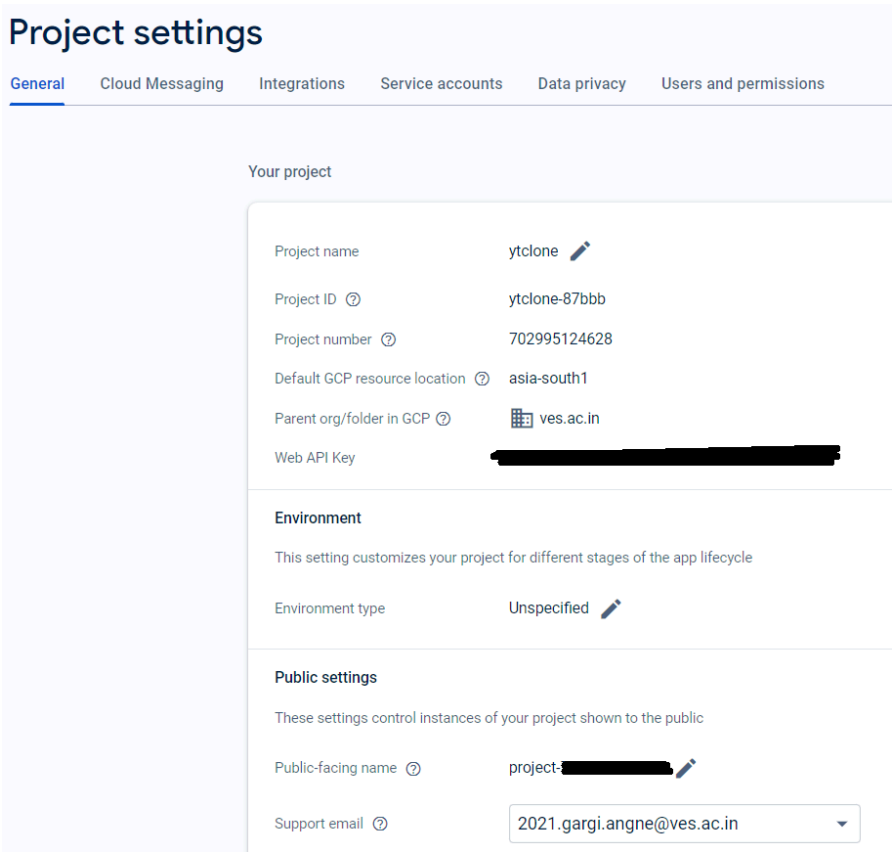
Output:

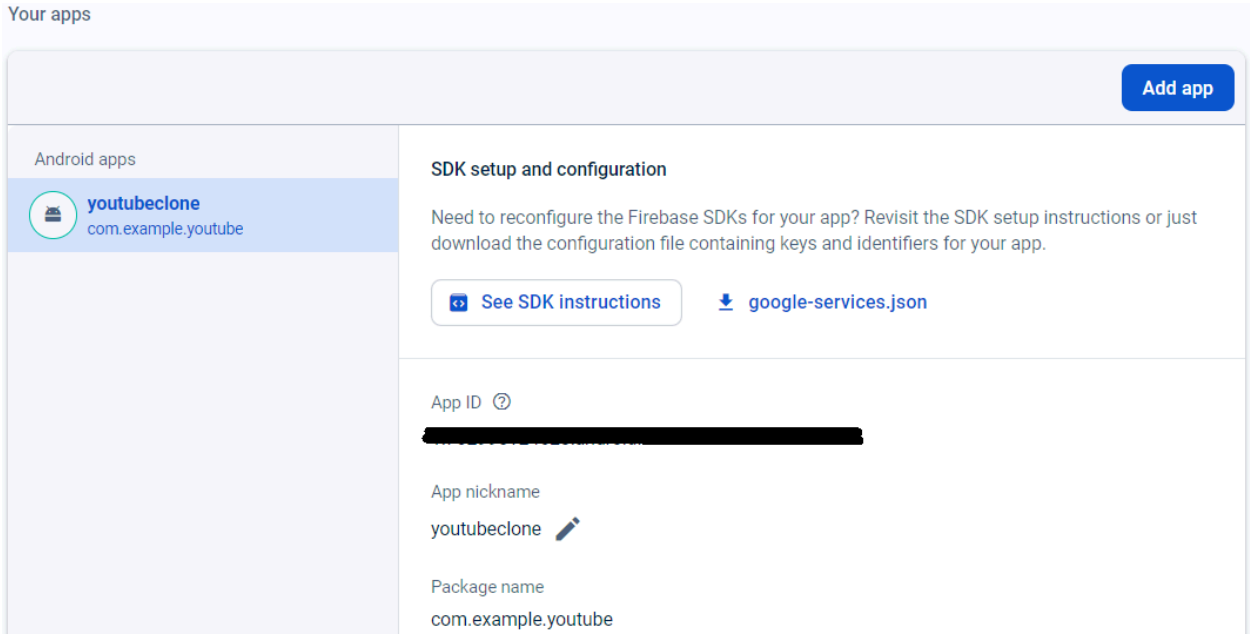



Create a project (Step 3 of 3)
Configure Google Analytics
Choose or create a Google Analytics account
Gargi
Automatically create a new property in this account
Upon project creation, a new Google Analytics property will be created in your chosen Google Analytics account and linked to your Firebase project. This link will enable data flow between the products. Data exported from your Google Analytics property into Firebase is subject to the Firebase terms of service, while Firebase data imported into Google Analytics is subject to the Google Analytics terms of service. Learn more.
Previous
Create project

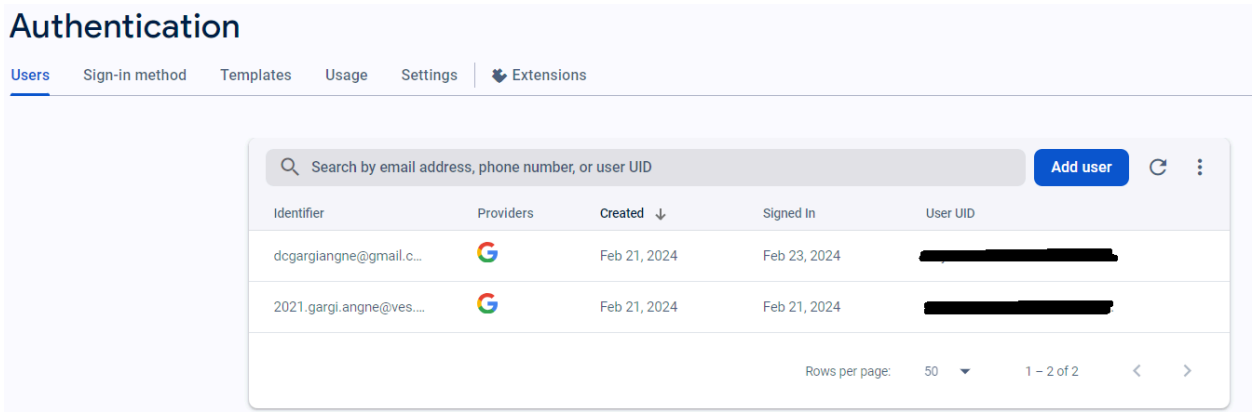

Firebase
Your Firebase projects
Add project
ytclone
ytclone-87bbb
ves.ac.in

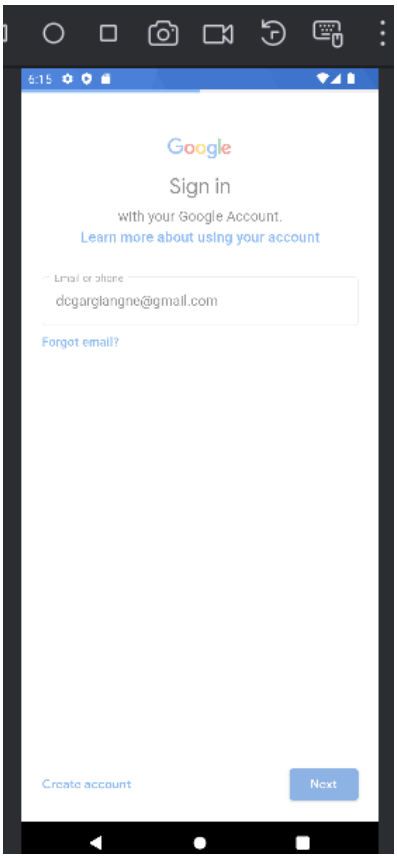
Project Title: Flutter: Youtube Clone / PWA: Travel website **Roll No. 1**

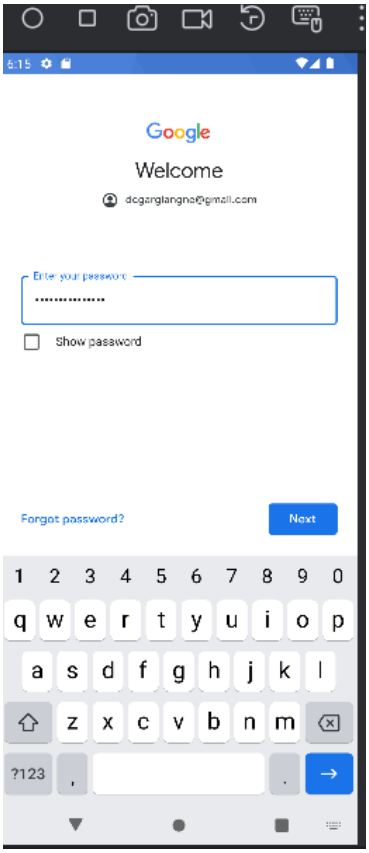




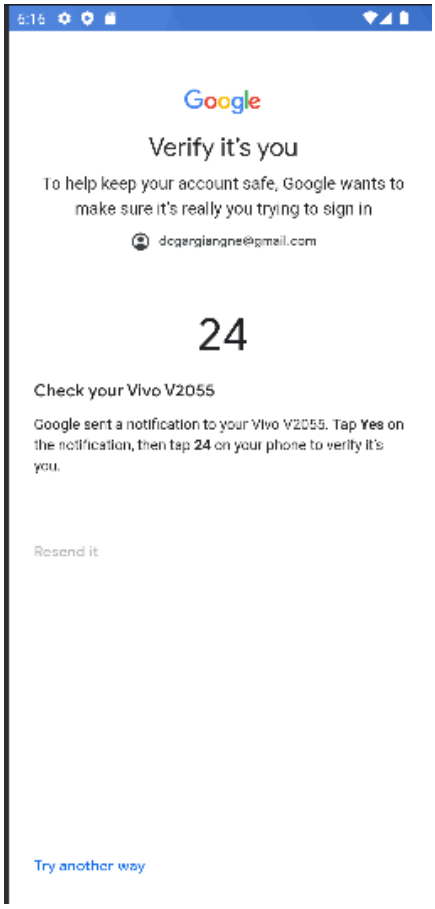
Project Title: Flutter: Youtube Clone / PWA: Travel website **Roll No. 1**

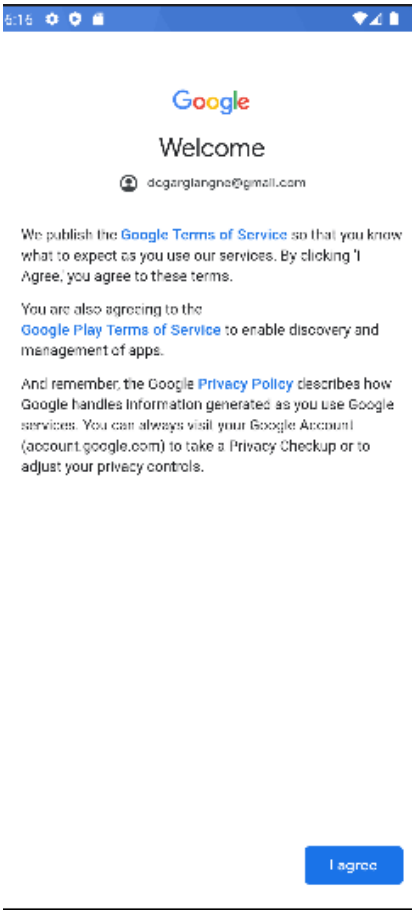




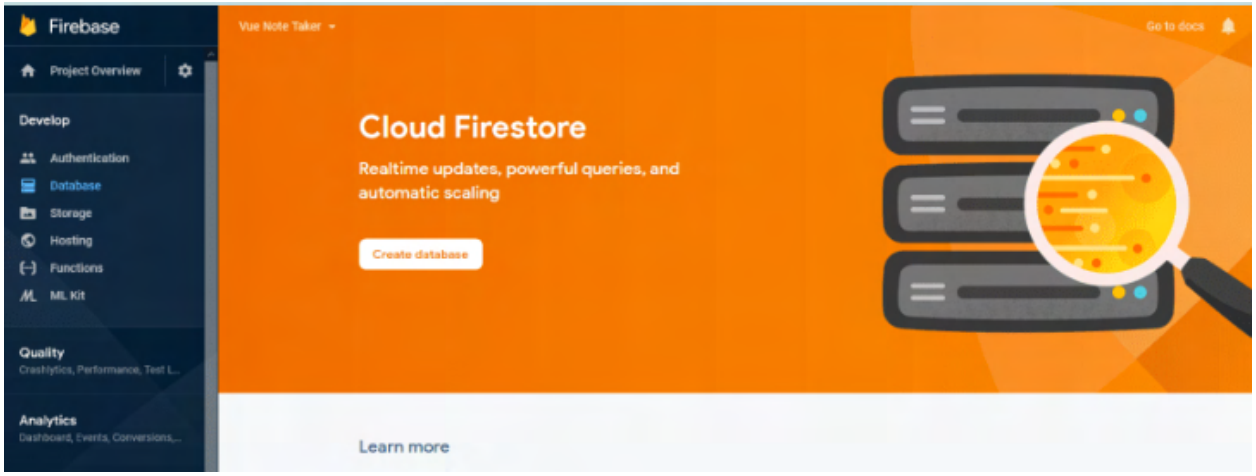


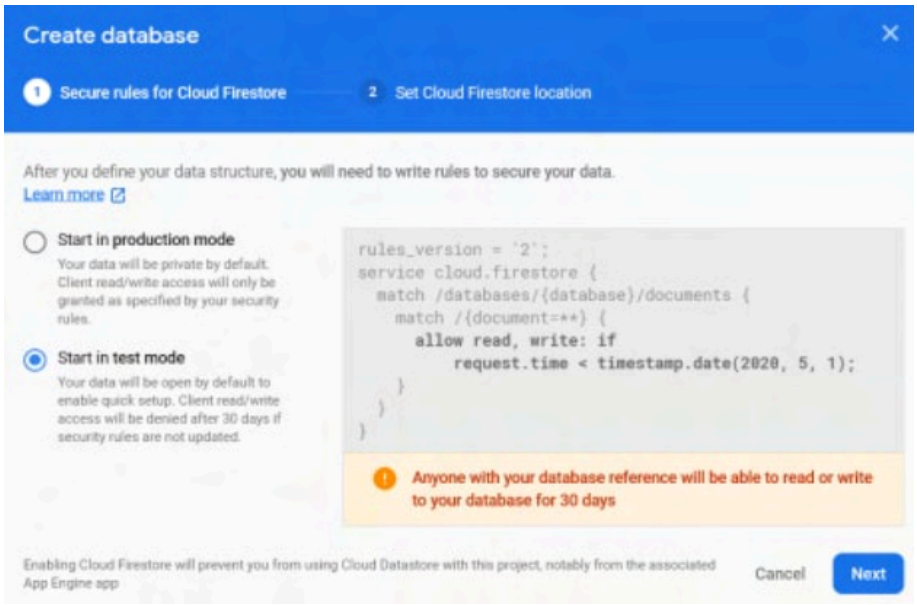
Project Title: Flutter: Youtube Clone / PWA: Travel website **Roll No. 1**

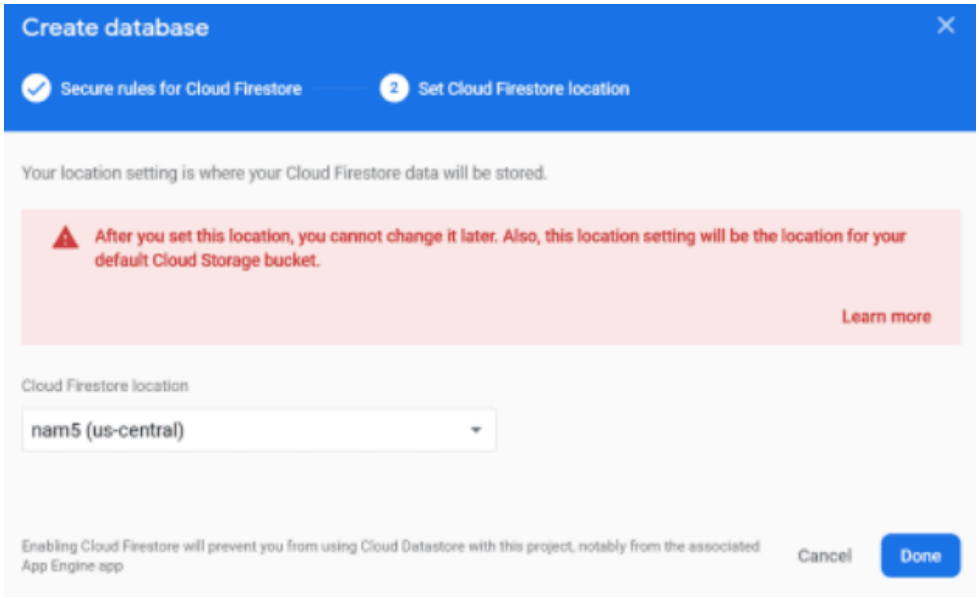




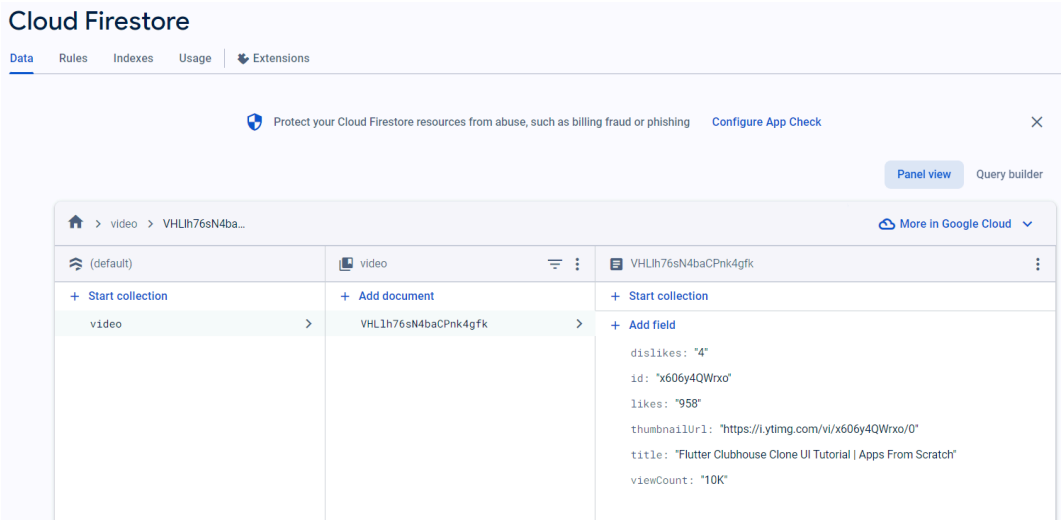
Project Title: Flutter: Youtube Clone / PWA: Travel website **Roll No. 1**







Project Title: Flutter: Youtube Clone / PWA: Travel website **Roll No. 1**



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.	1
Name	Gargi Angne
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	15

Practical 7

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

Theory:

Regular Web App

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

Progressive Web App

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

Difference between PWAs vs. Regular Web Apps:
A Progressive Web is different and better than a Regular Web app with features like:

1. **Native Experience**

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

2. **Ease of Access**

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

3. **Faster Services**

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

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

5. **Updated Real-Time Data Access**
Another plus point of PWAs is that these apps get updated on their own. They do not demand the end-users to go to the App Store or other such platforms to download the update and wait until installed.
In this app type, the web app developers can push the live update from the server, which reaches the apps residing on the user's devices automatically. Therefore, it is easier for the mobile app developer to provide the best of the updated functionalities and services to the end-users without forcing them to update their app.

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

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

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.

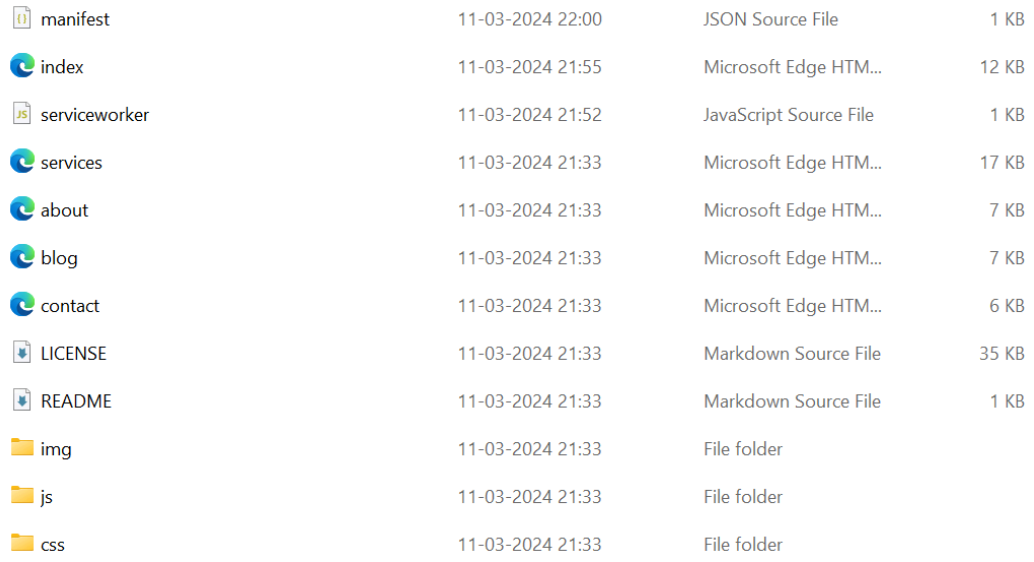
Linkable — Easily shared via URL without complex installations.

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.

Project Title: Flutter: Youtube Clone / PWA: Travel website **Roll No. 1**

Code and output:

Adding the manifest and serviceworker files:



manifest	11-03-2024 22:00	JSON Source File	1 KB
index	11-03-2024 21:55	Microsoft Edge HTM...	12 KB
serviceworker	11-03-2024 21:52	JavaScript Source File	1 KB
services	11-03-2024 21:33	Microsoft Edge HTM...	17 KB
about	11-03-2024 21:33	Microsoft Edge HTM...	7 KB
blog	11-03-2024 21:33	Microsoft Edge HTM...	7 KB
contact	11-03-2024 21:33	Microsoft Edge HTM...	6 KB
LICENSE	11-03-2024 21:33	Markdown Source File	35 KB
README	11-03-2024 21:33	Markdown Source File	1 KB
img	11-03-2024 21:33	File folder	
js	11-03-2024 21:33	File folder	
css	11-03-2024 21:33	File folder	

Inserting the code:

//manifest.json

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

//serviceworker.js

```
var staticCacheName = "pwa";

self.addEventListener("install", function (e) {
e.waitUntil(
   caches.open(staticCacheName).then(function (cache) {
   return cache.addAll(["/"]);
   })
);
});

self.addEventListener("fetch", function (event) {
console.log(event.request.url);
```

```
event.respondWith(
   caches.match(event.request).then(function (response) {
   return response || fetch(event.request);
   })
);
});
```

//index.html

The header includes:

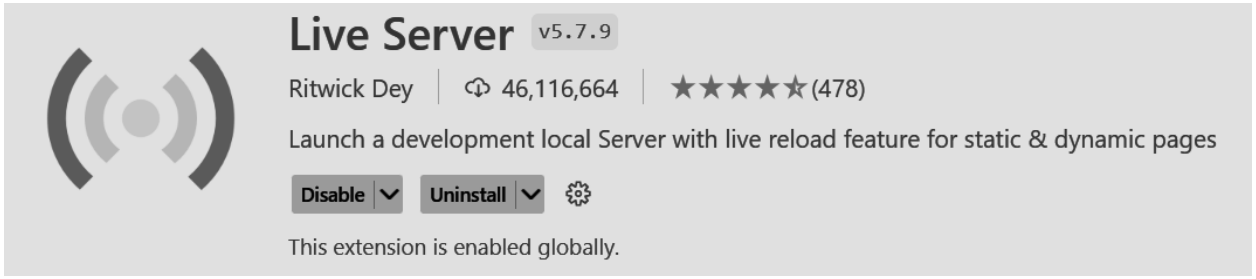
```
<!-- Meta Tags required for
   Progressive Web App -->
<meta name=
"apple-mobile-web-app-status-bar"
      content="#aa7700">
<meta name="theme-color"
      content="black">

      <!-- Manifest File link -->
<link rel="manifest"
href="manifest.json">
```

The body includes:

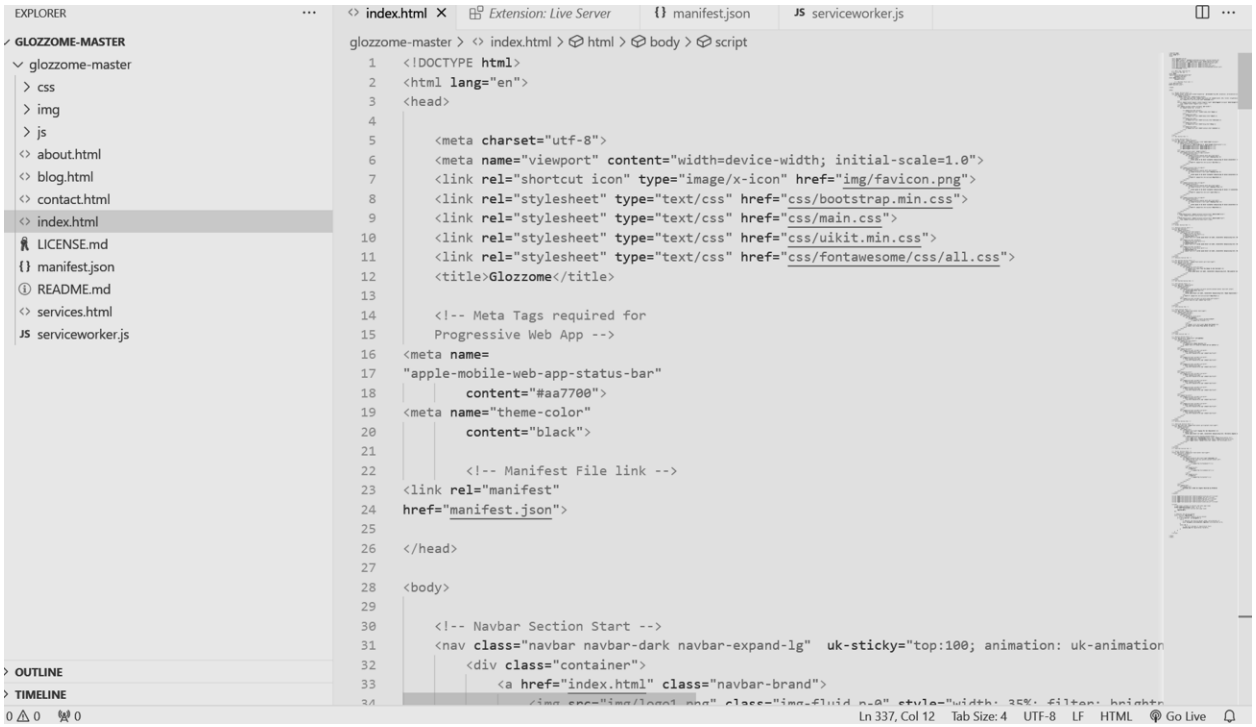
```
<script>
      // Add event listener to execute code when page loads
      window.addEventListener('load', () => {
         // Call registerSW function when page loads
         registerSW();
      });

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

```
</script>
```

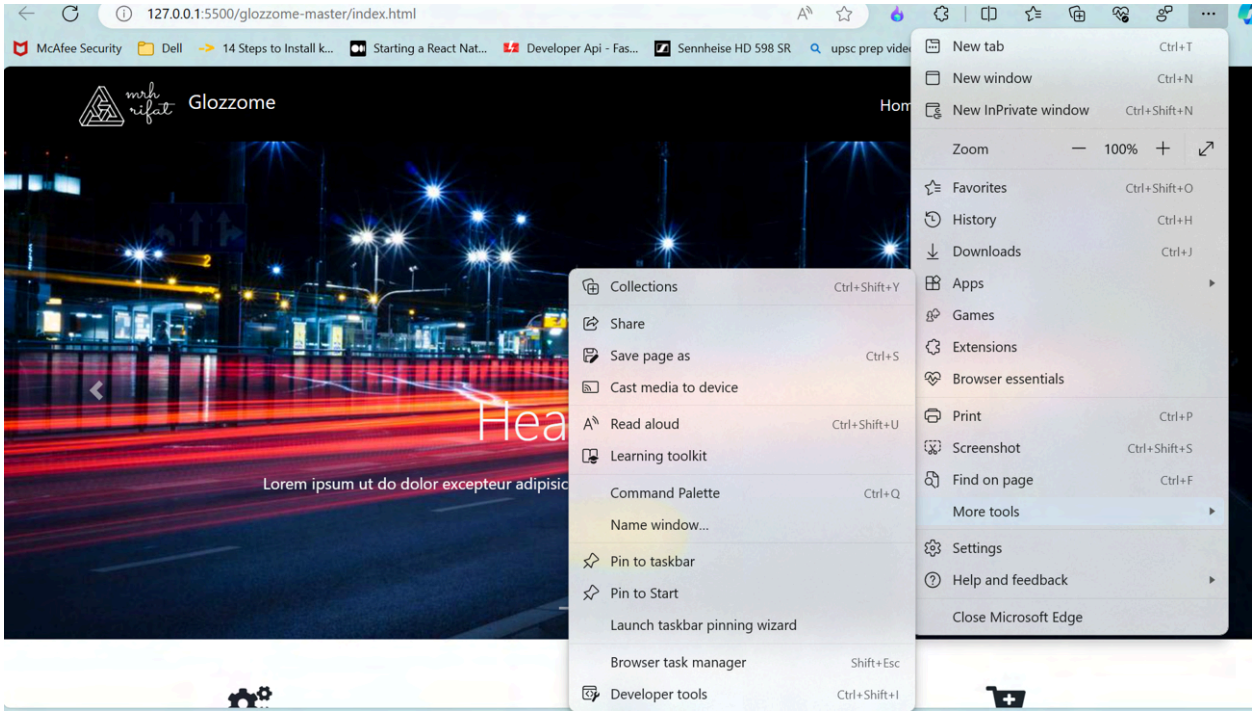
Installing live server:



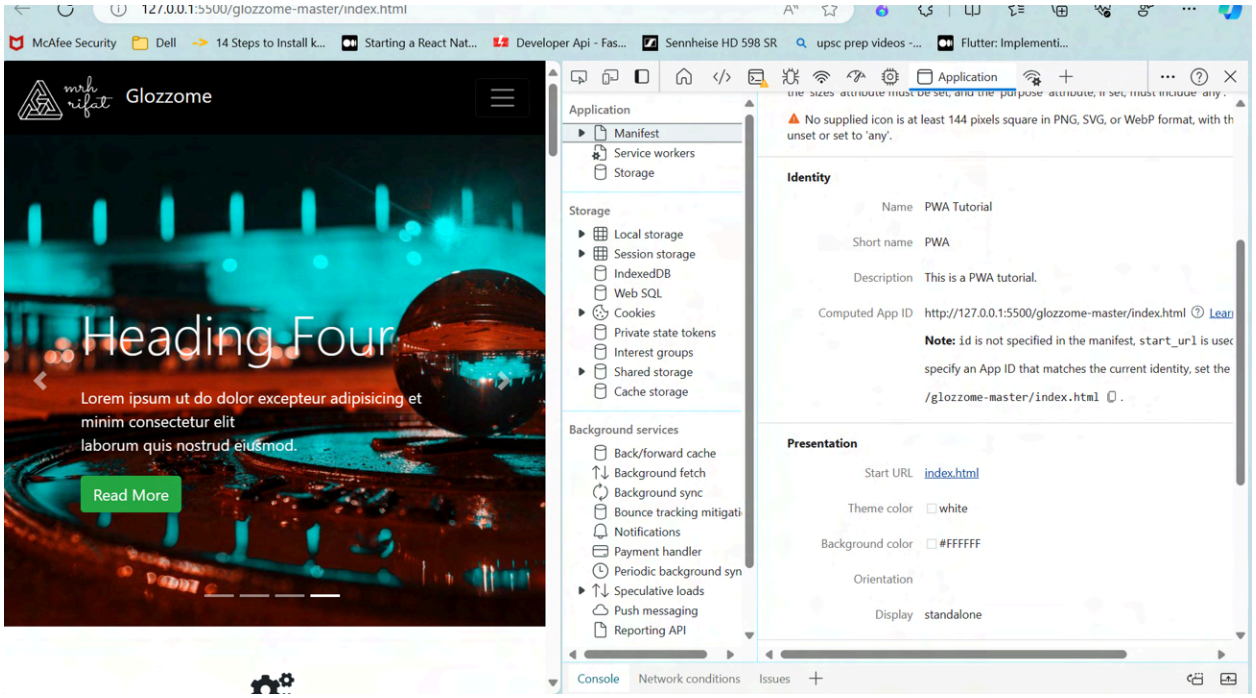
Click on “Go live” in the bottom right corner:



Click on the three dots on the top right corner-> more tools-> developer tools:

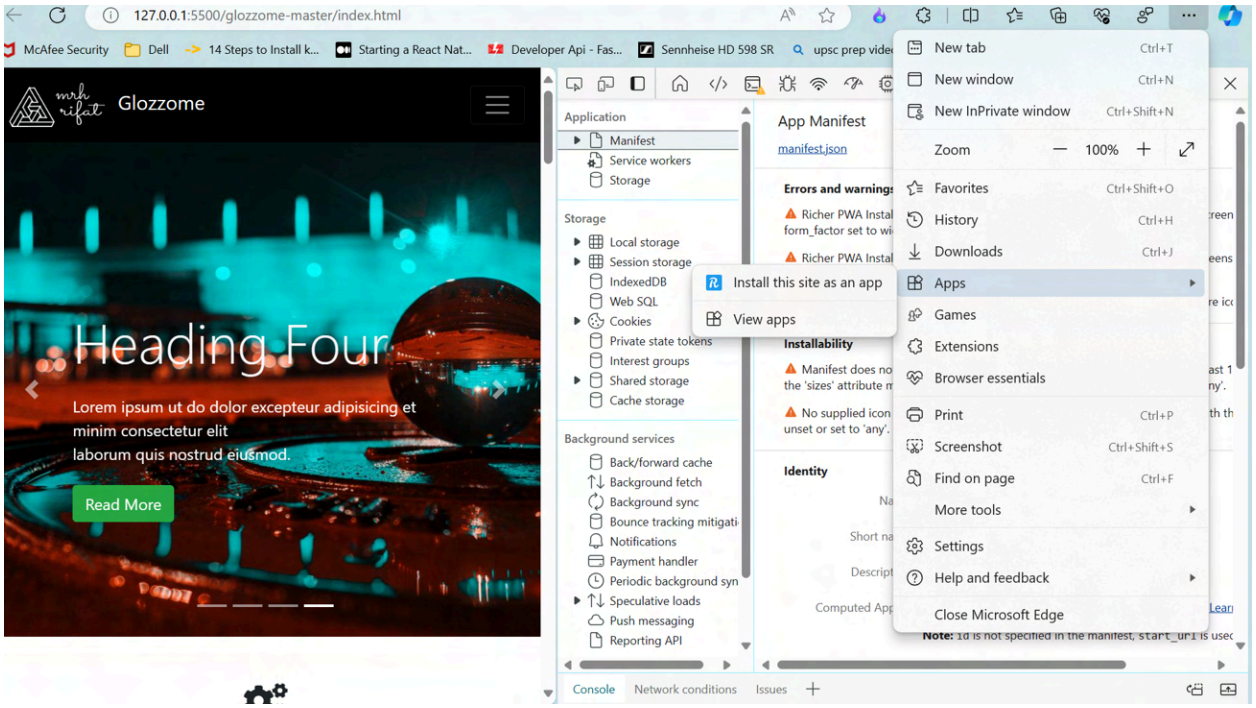


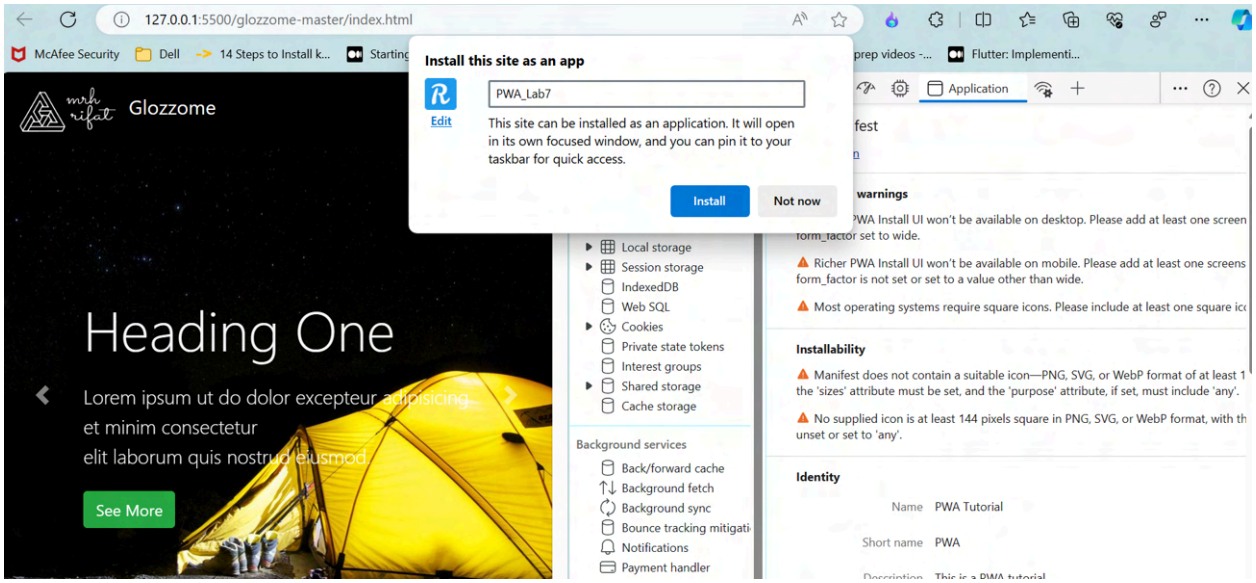
Go to application:



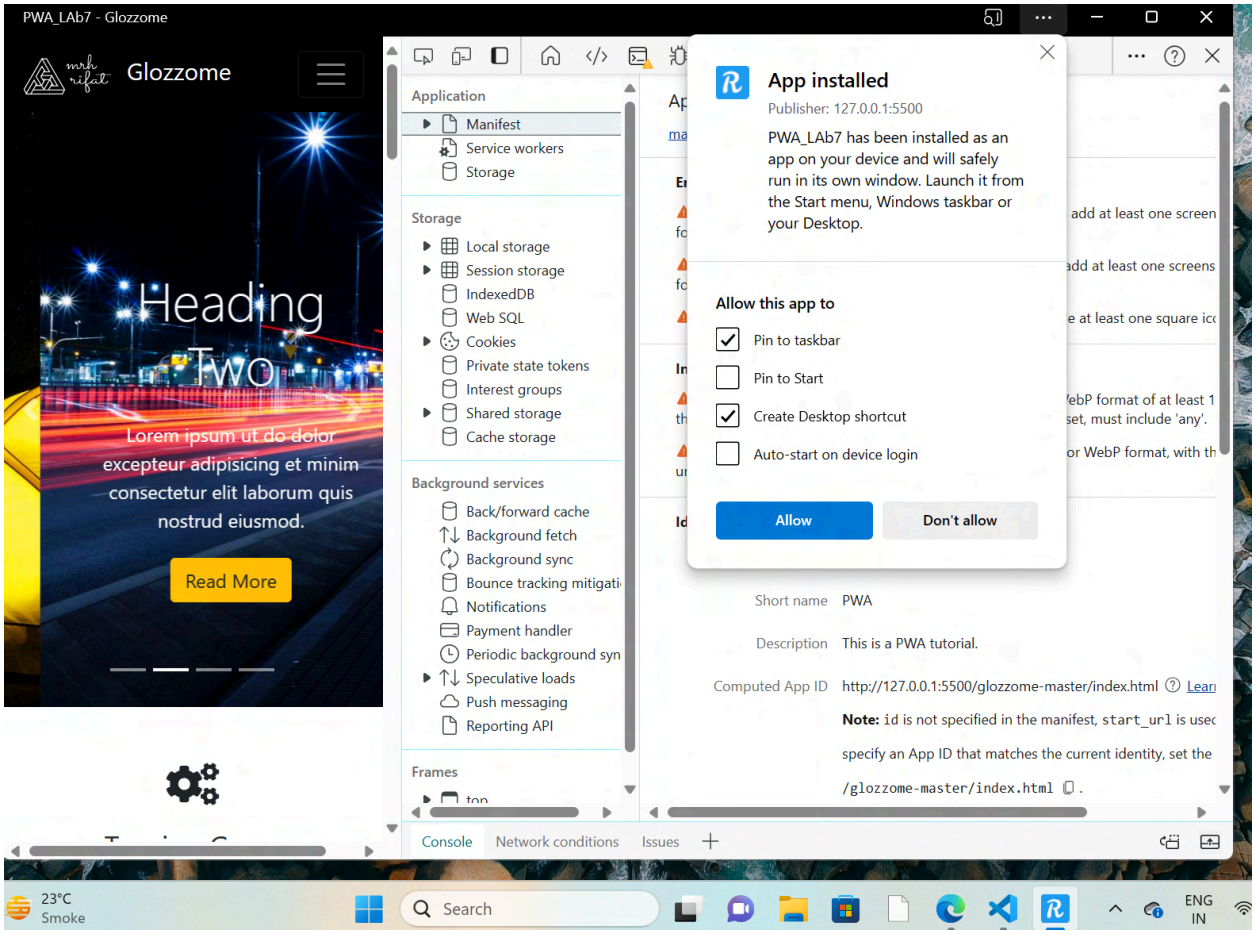
Project Title: Flutter: Youtube Clone / PWA: Travel website **Roll No. 1**

Click on the three dots-> Apps-> Install this site as an app

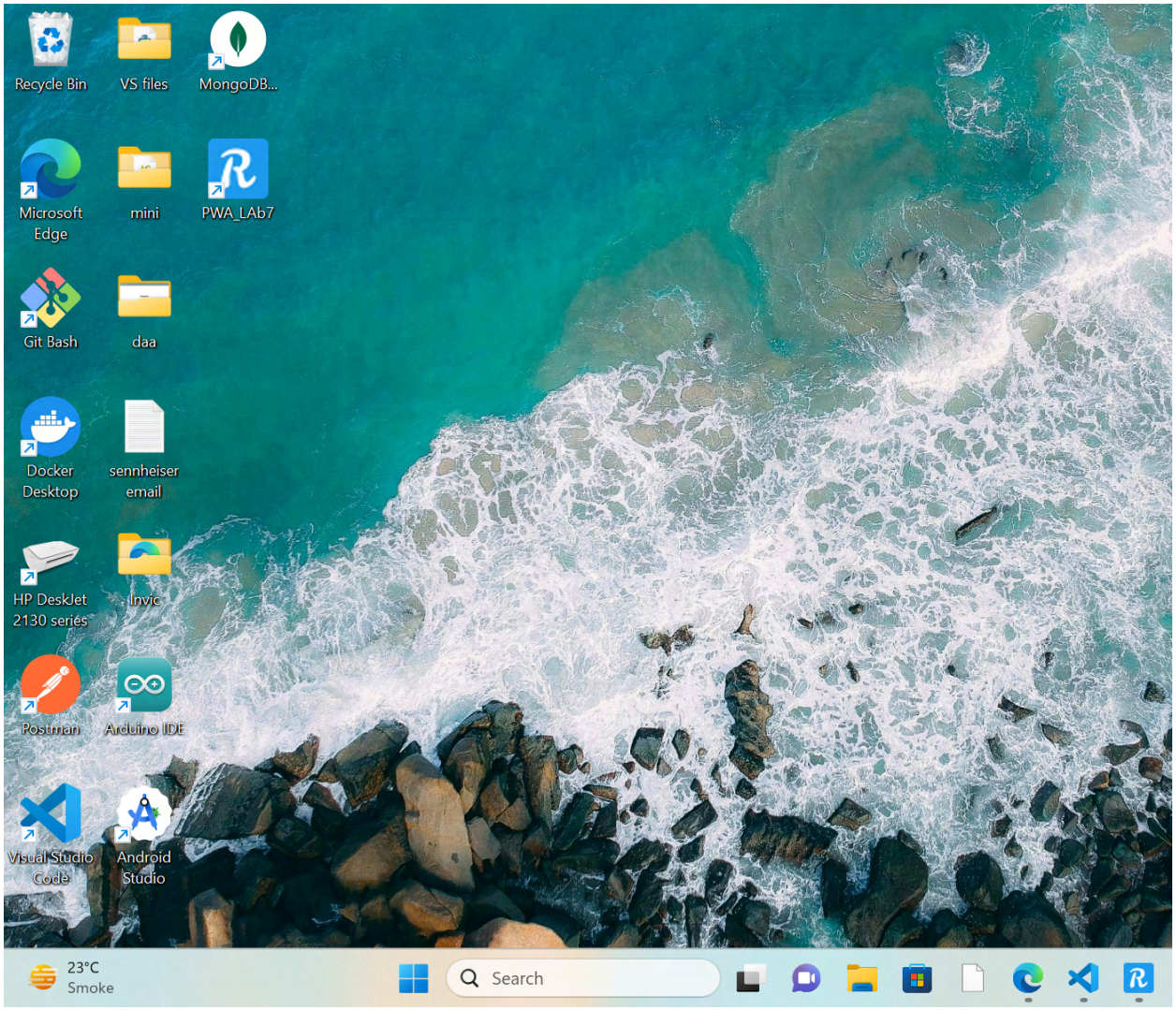




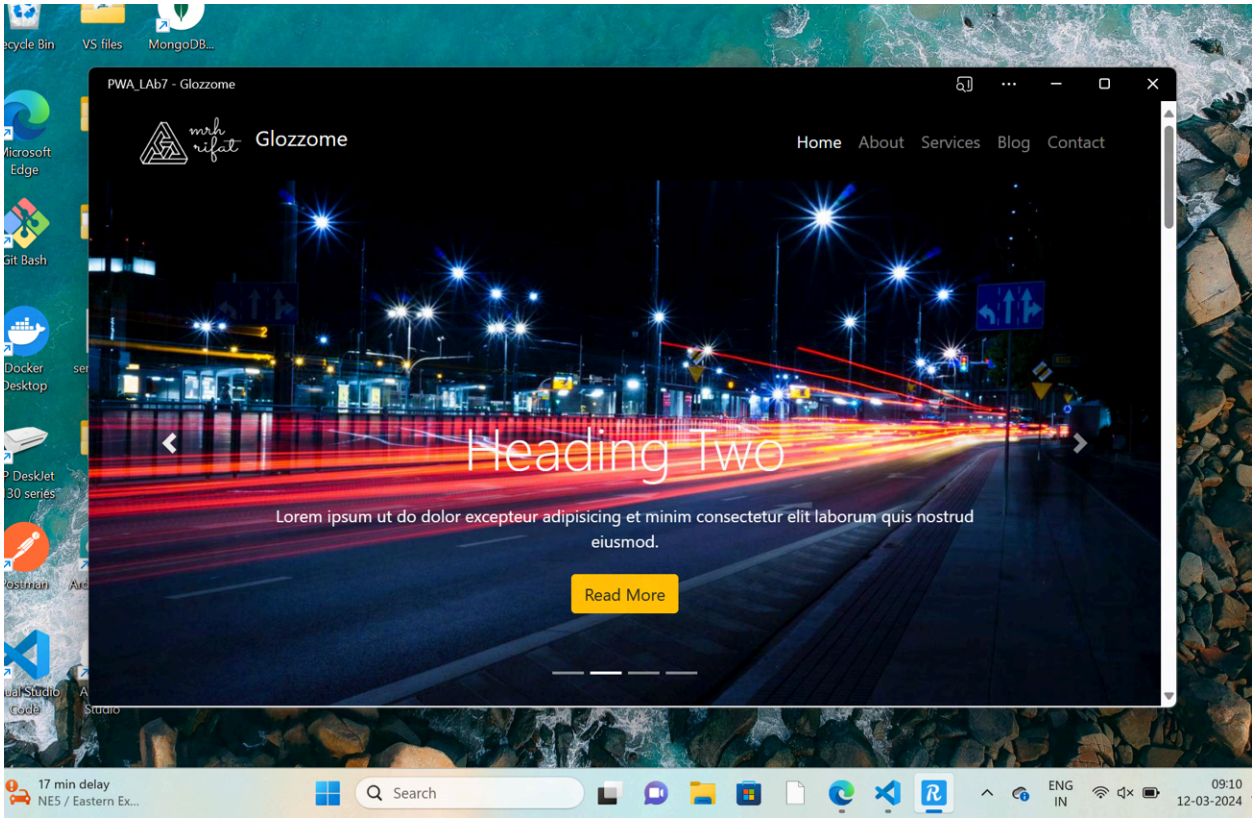
Project Title: Flutter: Youtube Clone / PWA: Travel website **Roll No. 1**



Project Title: Flutter: Youtube Clone / PWA: Travel website **Roll No. 1**



Project Title: Flutter: Youtube Clone / PWA: Travel website **Roll No. 1**



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.	1
Name	Gargi Angne
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

Practical 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.

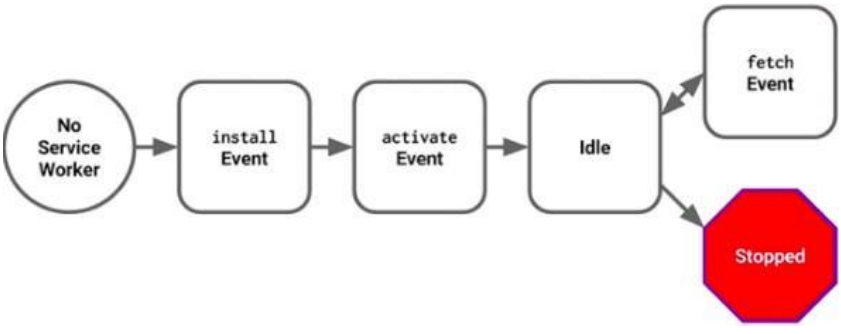
- You can manage Push Notifications

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.

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.

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.

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

Code:

//serviceworker.js

```
var staticCacheName = "pwa";

self.addEventListener("install", function (e) {
e.waitUntil(
  caches.open(staticCacheName).then(function (cache) {
  return cache.addAll(["/", "/index.html", "/about.html", "/blog.html", "/contact.html",
"/services.html","/img/slider-img4.jpg", "/css/main.css"]);
  })
);
});

self.addEventListener("fetch", function (event) {
console.log(event.request.url);

event.respondWith(
  caches.match(event.request).then(function (response) {
  return response || fetch(event.request);
  })
);
});
```

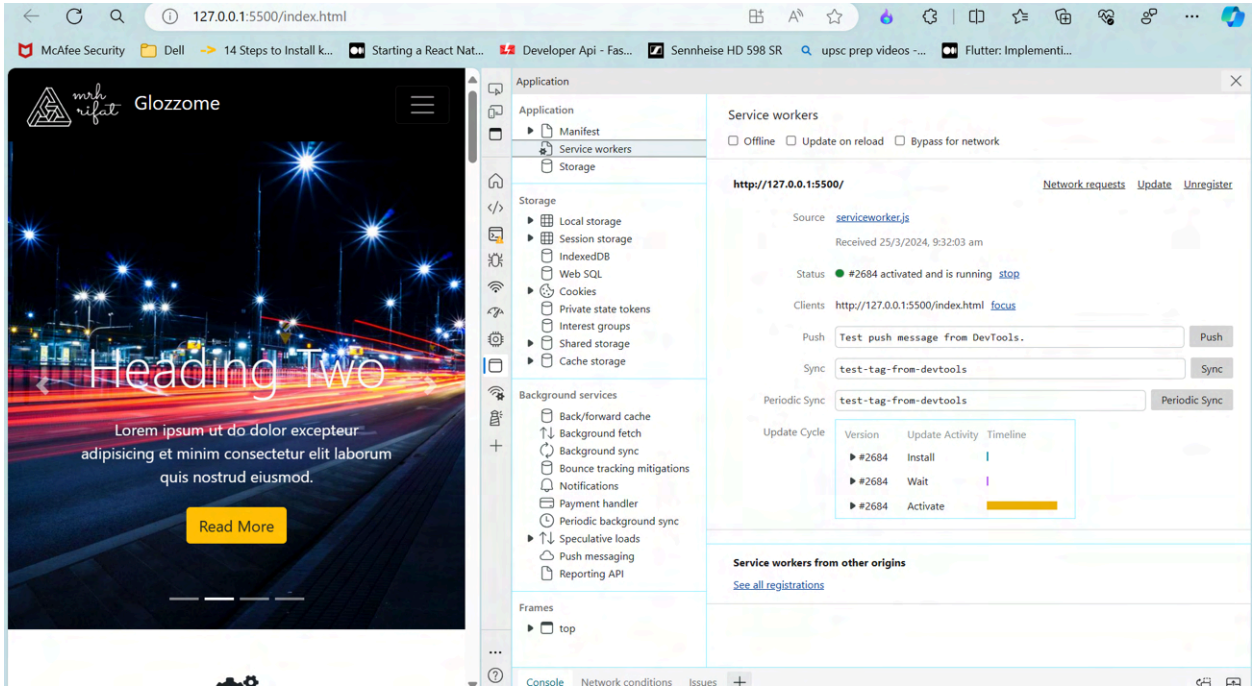
JS in index.html:

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

Project Title: Flutter: Youtube Clone / PWA: Travel website **Roll No. 1**

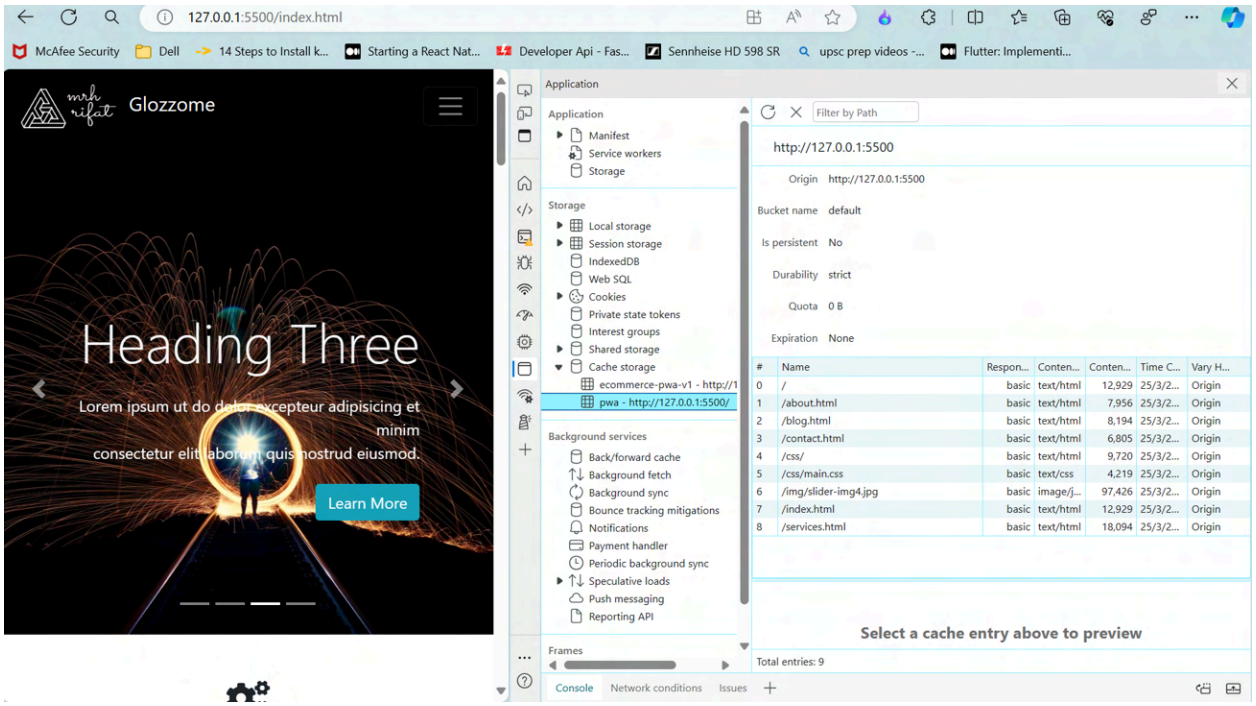
Output:

Click on the three dots-> More tools-> Developer tools-> Application-> Service workers



The service worker is active and running.

Go to Cache storage:



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.	1
Name	Gargi Angne
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

Practical 9

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

Theory:

Service Worker

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

Things to note about Service Worker:

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

Fetch Event

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

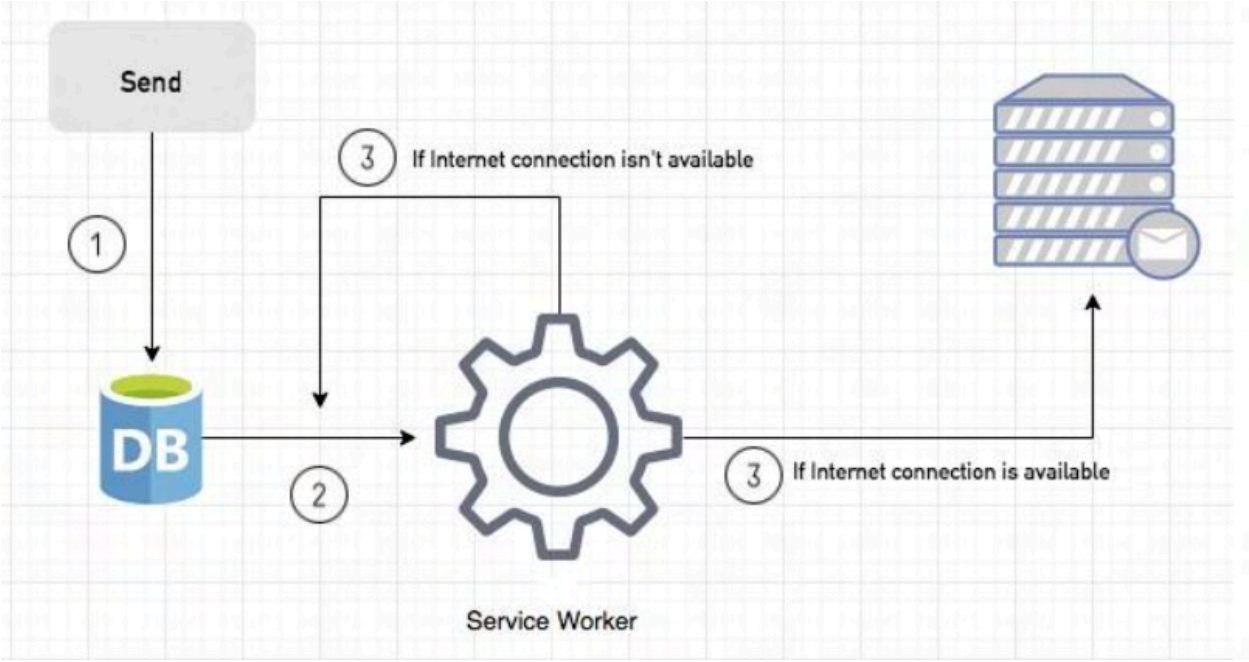
Of course, you can use many different methods but you can find in the following example a “cache first” and “network first” approach. In this example, if the request’s and current location’s origin are the same (Static content is requested.), this is called “cacheFirst” but if you request a targeted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

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

Sync Event

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

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.

Push Event

This is the event that handles push notifications that are received from the server. You can apply any method with received data. "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. You can use Application Tab from Chrome Developer Tools for testing push notification.

Code:

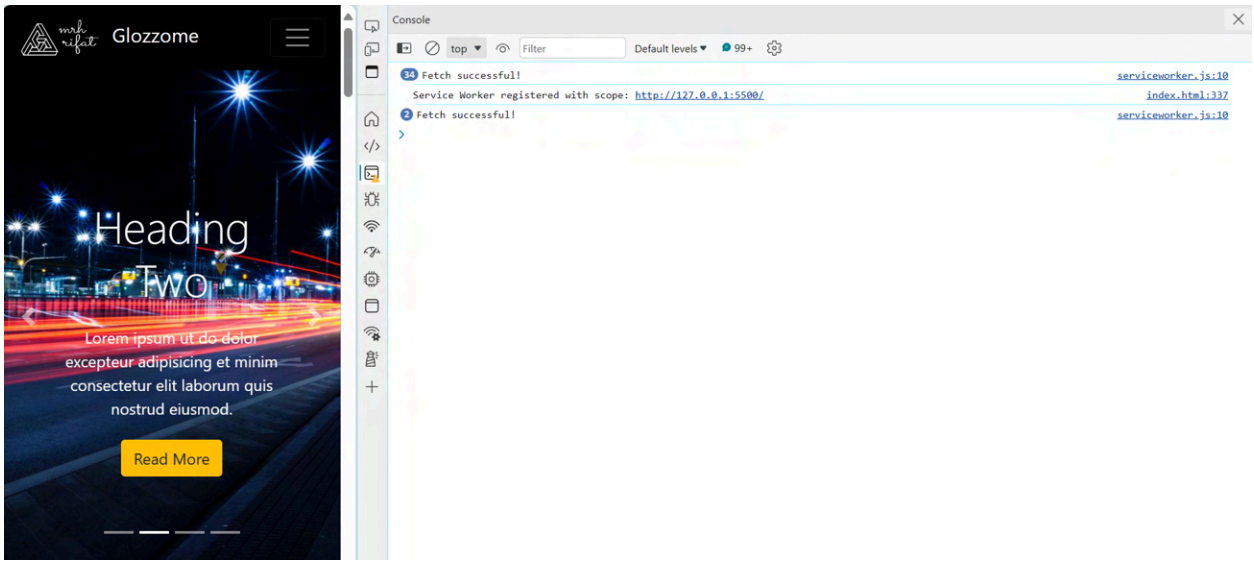
```
//serviceworker
self.addEventListener("install", function(event){
event.waitUntil(preLoad());
});

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

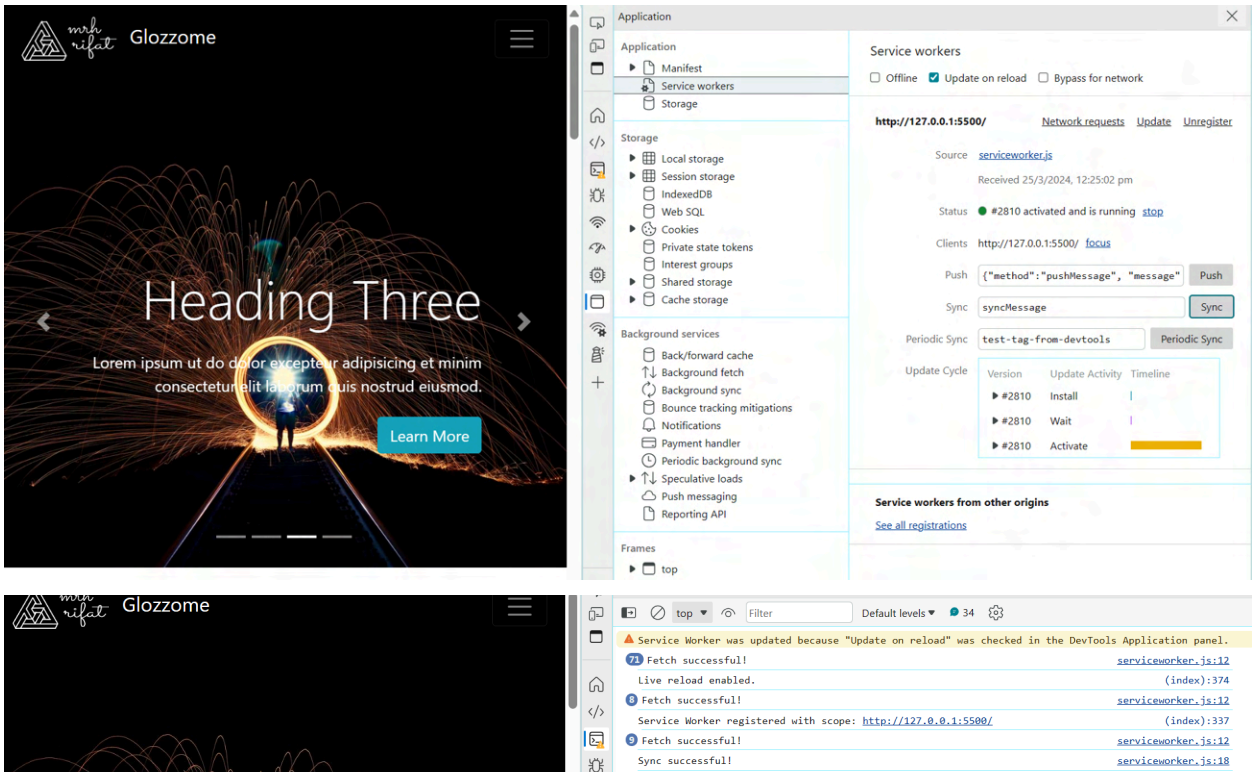
Project Title: Flutter: Youtube Clone / PWA: Travel website **Roll No. 1**

Output:

Fetch:

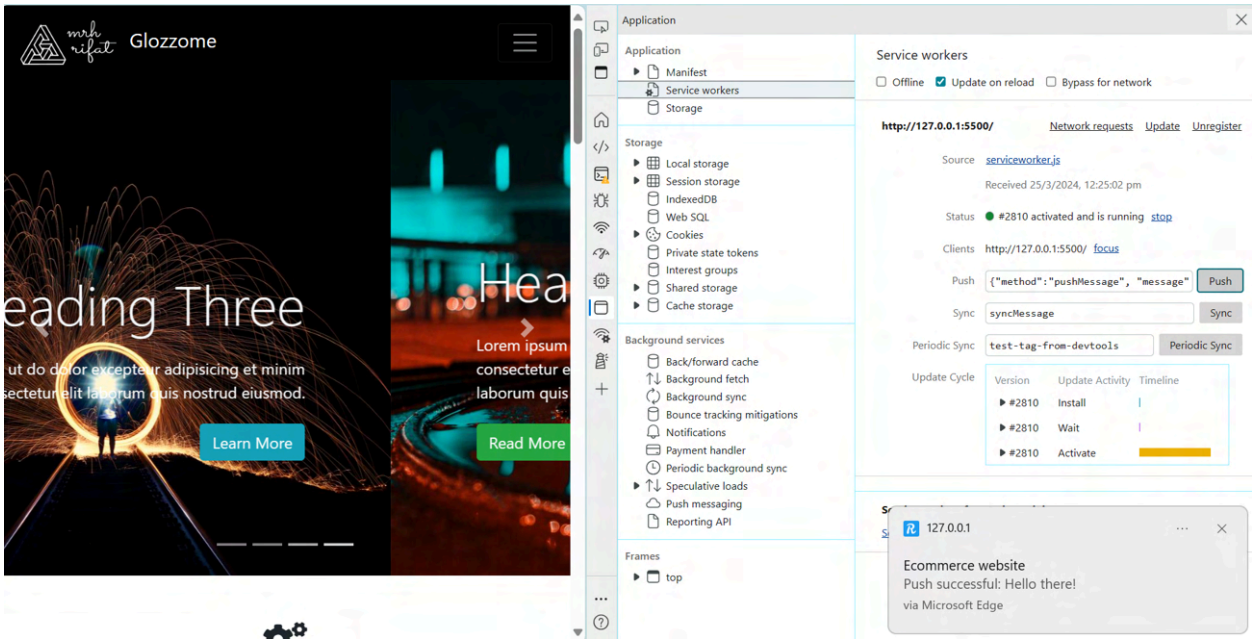


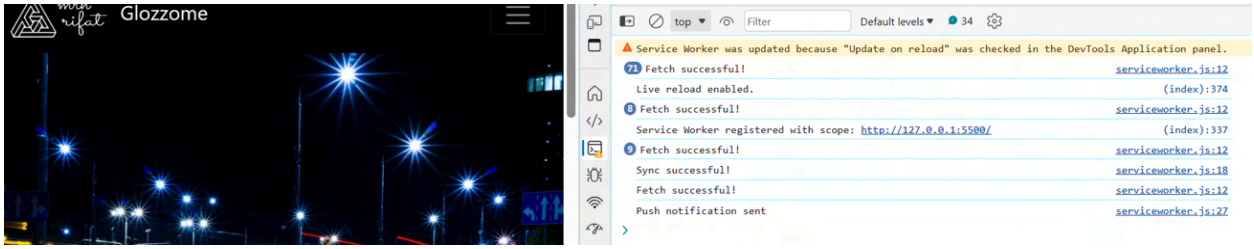
Sync:



Project Title: Flutter: Youtube Clone / PWA: Travel website **Roll No. 1**

Push:





MAD & PWA Lab
Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	1
Name	Gargi Angne
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

Practical
10

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

Theory:

GitHub Pages

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

GitHub Pages provides the following key features:
1. Blogging with Jekyll
2. Custom URL
3. Automatic Page Generator

Reasons for favoring this over Firebase:
1. Free to use
2. Right out of github
3. Quick to set up

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

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

Cons
1. The code of your website will be public, unless you pay for a private repository.
2. Currently, there is no support for HTTPS for custom domains. It's probably coming soon though.

3. Although Jekyll is supported, plug-in support is rather spotty.

Firebase

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

Some of the features offered by Firebase are:

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

Reasons for favoring over GitHub Pages:

1. Realtime backend made easy
2. Fast and responsive

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

Pros

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

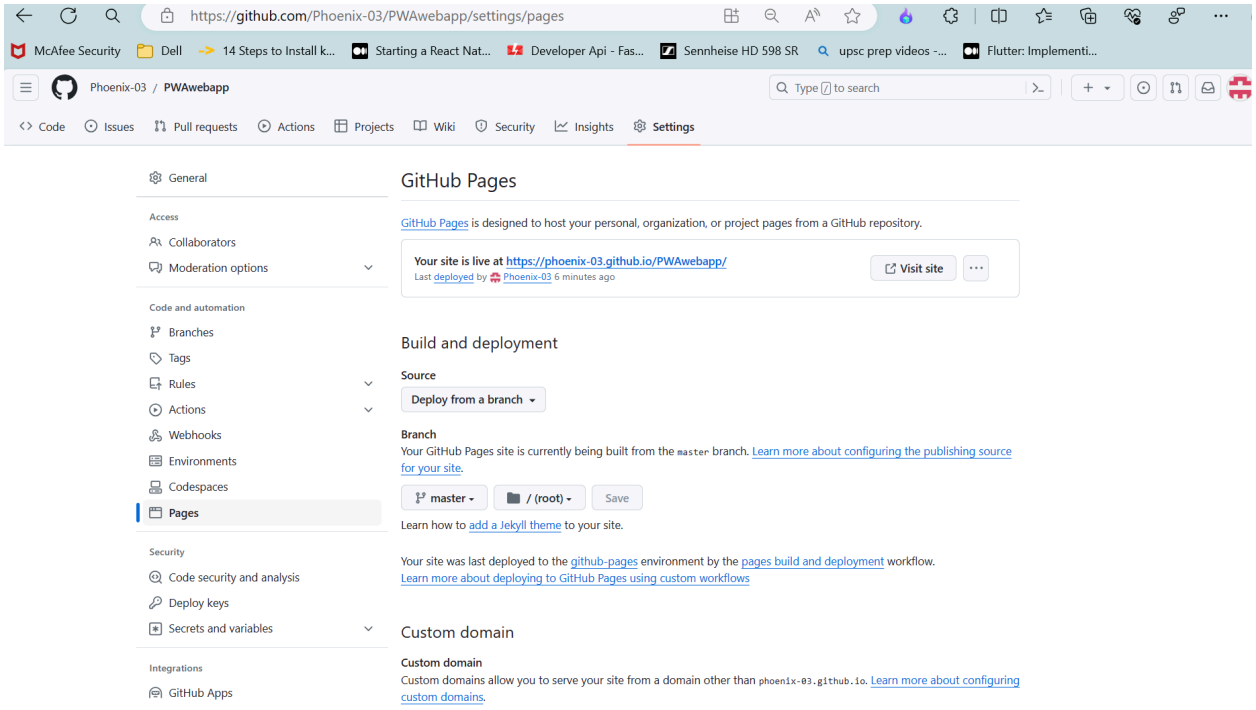
Cons

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

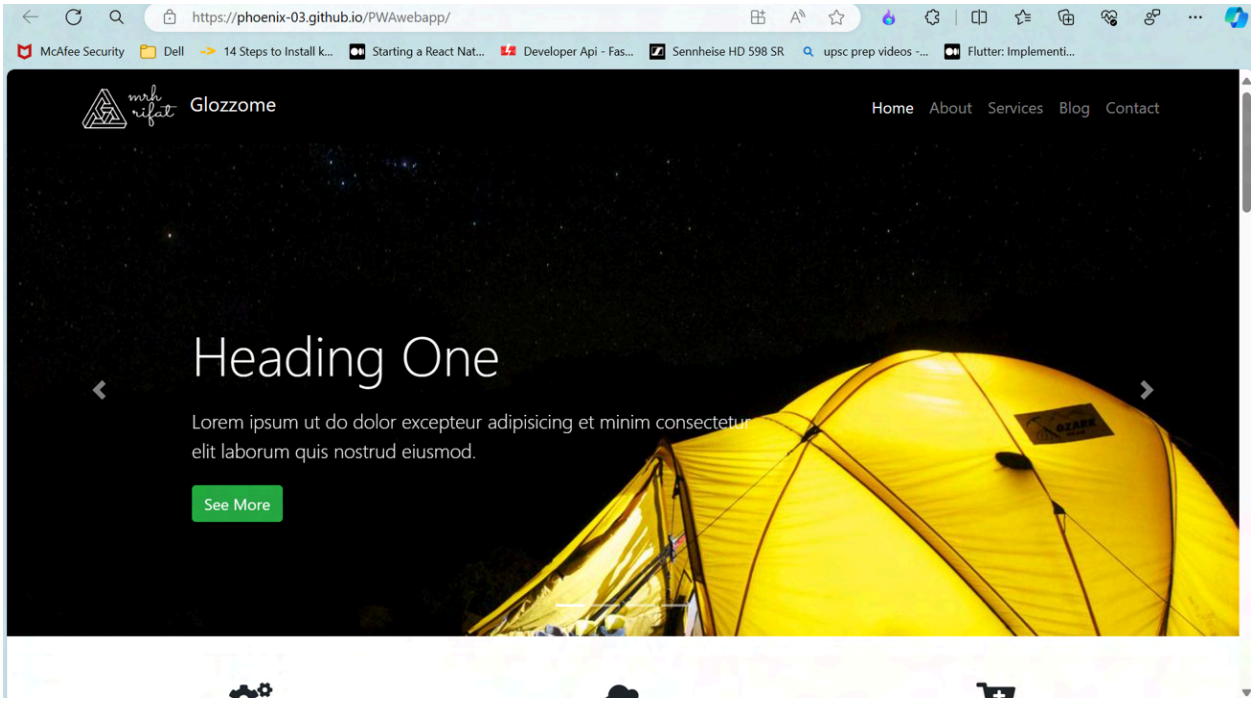
Link to our GitHub repository:

Pages link: [Glozzome (phoenix-03.github.io)]
Github repo: https://github.com/Phoenix-03/PWAwebapp.git

Github Screenshot:



Project Title: Flutter: Youtube Clone / PWA: Travel website **Roll No. 1**



MAD & PWA Lab
Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	1
Name	Gargi Angne
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	15

Practical 11

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

Theory:

Google Lighthouse :

Google Lighthouse is a tool that lets you audit your web application based on a number of parameters including (but not limited to) performance, based on a number of metrics, mobile compatibility, Progressive Web App (PWA) implementations, etc. All you have to do is run it on a page or pass it a URL, sit back for a couple of minutes and get a very elaborate report, not much short of one that a professional auditor would have compiled in about a week.
The best part is that you have to set up almost nothing to get started. Let's begin by looking at some of the top features and audit criteria used by Lighthouse.

Key Features and Audit Metrics

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

1. Performance: This score is an aggregation of how the page fared in aspects such as (but not limited to) loading speed, time taken for loading for basic frame(s), displaying meaningful content to the user, etc. To a layman, this score is indicative of how decently the site performs, with a score of 100 meaning that you figure in the 98th percentile, 50 meaning that you figure in the 75th percentile and so on.

2. PWA Score (Mobile): Thanks to the rise of Service Workers, app manifests, etc., a lot of modern web applications are moving towards the PWA paradigm, where the objective is to make the application behave as close as possible to native mobile applications. Scoring points are based on the Baseline PWA checklist laid down by Google which includes Service Worker implementation(s), viewport handling, offline functionality, performance in script-disabled environments, etc.

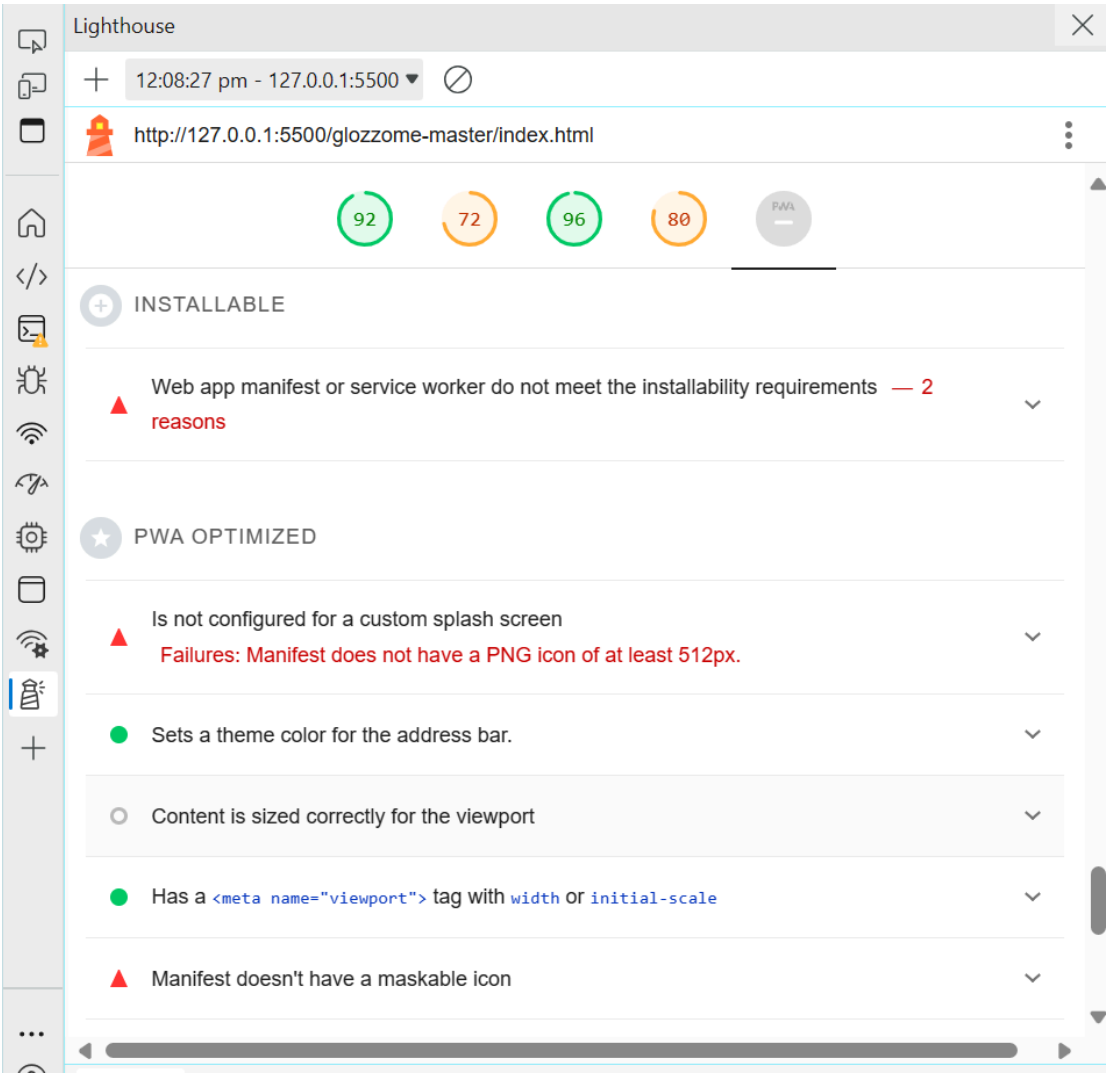
3. Accessibility: As you might have guessed, this metric is a measure of how accessible your website is, across a plethora of accessibility features that can be implemented in your page (such as the 'aria-' attributes like aria-required, audio captions, button names, etc.). Unlike the other metrics though, Accessibility metrics score on a pass/fail basis i.e. if all possible elements of the page are not screen-reader friendly (HTML5 introduced features that would make pages easy to interpret for screen readers used by visually challenged people like tag names, tags such as <section>, <article>, etc.), you get a 0 on that score. The aggregate of these scores is your Accessibility metric score.

4. Best Practices: As any developer would know, there are a number of practices that have been deemed 'best' based on empirical data. This metric is an aggregation of many such points, including but not limited to:Use of HTTPS

Project Title: Flutter: Youtube Clone / PWA: Travel website **Roll No. 1**

Original manifest.js:

```
{
   "name":"PWA Tutorial",
   "short_name":"PWA",
   "start_url":"index.html",
   "display":"standalone",
   "background_color":"#FFFFFF",
   "theme_color":"white",
   "scope": ".",
   "description":"This is a PWA tutorial."
}
```

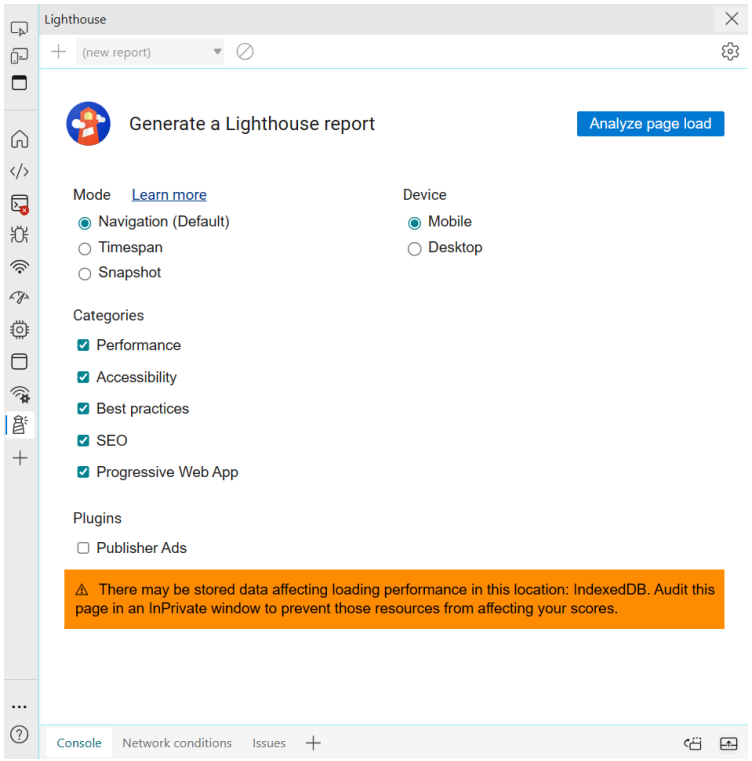


Changes in manifest.js:

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

Add the lighthouse tool to the activity bar by:
Clicking on the three dots on the upper right corner -> more tools -> developer tools -> plus sign -> lighthoue

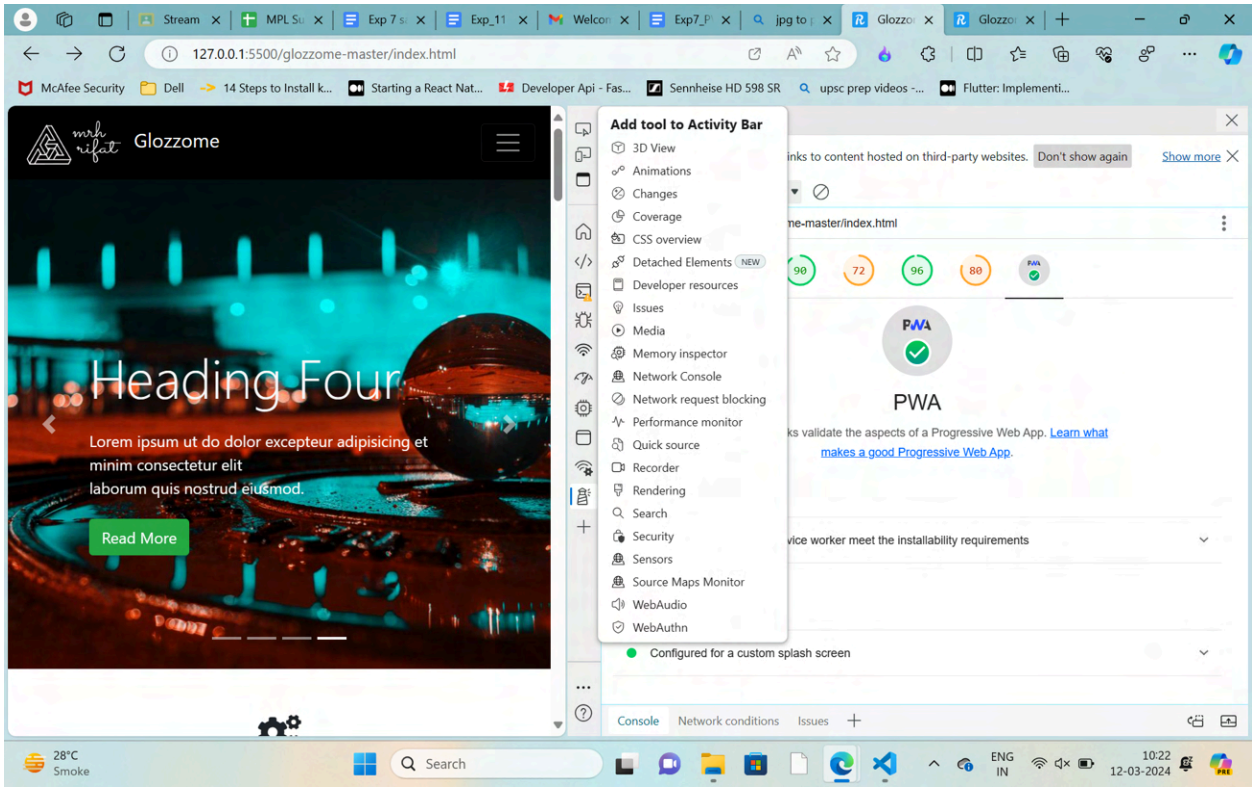
(Note: These steps are applicable for the Microsoft Edge browser)



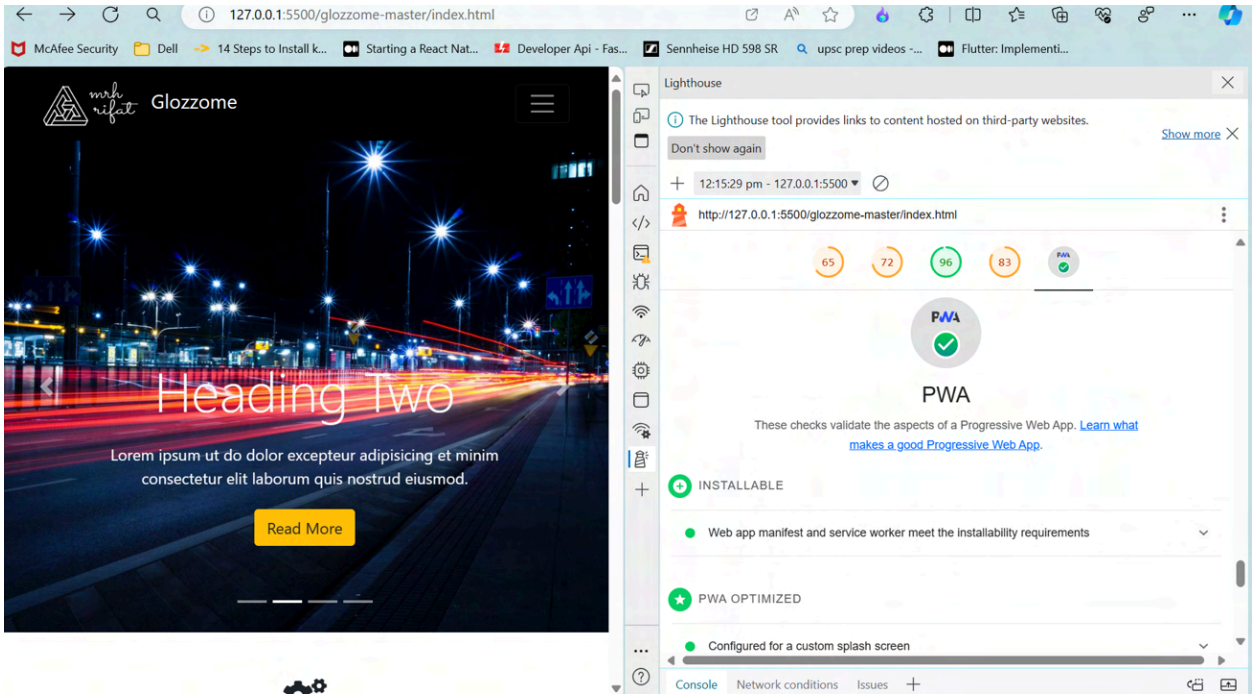
Project Title: Flutter: Youtube Clone / PWA: Travel website **Roll No. 1**

Output:

Desktop-



Mobile-



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.	1
Name	Gargi Angne
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	5

Gargi Angne
DLSA

MPL Assignment 1

1) Flutter overview:

→ a) Key features:

i) Cross-platform development - Flutter allows developers to write a single codebase for both android and ios and it reduces the time and costs of development.

ii) Fast development - It has a hot reload feature that allows developers to see the changes immediately that occur in code.

iii) Attractive UI - Flutter provides a rich set of customizable widgets that allows developers to create visually attractive and responsive user interfaces.

iv) Performance - Flutter uses the Dart programming language and its efficient rendering engine, Skia, ensures high performance fast app startup times and smooth animations.

v) Large Community - Flutter has a growing and supportive community which provides flutter developers with vast documentation, resources and third party packages which makes the development easy.

vi) Open Source - Flutter is free and open-source framework for developing mobile applications.

Advantages:

- Flutter comes with attractive customizable widgets.

- Dart has a big repository of software packages that let you develop the capabilities of your application.
- Because of its single code base, it is sufficient if we write automated tests once for both android and IOS platforms.
- Simplicity.
- Full control over widgets and their layout.

b) • Flutter stands out with its single codebase for multiple platforms, hot reload feature for rapid iteration, consistent UI aeross devices, high performance and strong community support.
• Its versatality, adoption by major companies and growing ecosystem further contribute to its popularity among developers.

2) Widget Tree and Composition

→ a) Describe widget tree:

- A widget describes what the view of the app would look like in the given state.
- As the user interface contains several widgets, those several widgets are combined to form a widget tree.
- The widget tree can be divided into 2 small trees depending on what widget the user has chosen for their code

i) Stateless widget tree:

It is used to create a non dynamic application.

ii) Stateful widget tree
Used to create dynamic application.

b) Building complex UI:
- Widget composition is the process of combining smaller, reusable UI components called widgets to create more complex user interfaces.
- By nesting widgets within each other and leveraging composition patterns, developers can create custom layouts and functionalities effectively.
- This approach promotes code reusability, maintainability, and flexibility in building sophisticated UIs.

c) Examples:
i) Container: A versatile widget that can be used to create boxes with background, padding, margins and more. It's often used as a parent or wrapper for other widgets.
ii) Row and ~~container~~ Column: These widgets are used to create horizontal and vertical layouts respectively. You can nest them to create complex layouts.
iii) Text: The text widget is used to display text on screen
iv) ListView/GridView: ListView and GridView are used to display a list of widgets either vertically or in a grid layout.
v) Stack: The stack widget is used to overlay widgets on top of each other. It allows you to position widgets relative to the edge or corners of the stack.

v.) AppBar: It typically used as the top app bar in a flutter application. It provided a built-in layout for displaying a title, actions.
vi.) Scaffold: It is a structural widget that provides the basic layout structure for a flutter app.

3) State Management

→ a) Importance:

i) Efficient UI Updates: Updating only that necessary parts of the UI in response to changes.
ii) Data Synchronization: Keeping the UI in sync with data from various sources.
iii) Code organisation: Structuring code for clarity, reusability and maintainability.
iv) Performance optimization: Avoiding unnecessary UI updates for better app performance.
v) State persistence: Saving and restoring state across app sessions.
vi) Cross-Platform Consistency: Ensuring a consistent user experience across different platforms
vii) Concurrency and Asynchrony: Managing asynchronous operations for a responsive UI.

b) Different approaches:

Comparing popular approaches of: setState
Provider
Riverpod.

	setState	Provider	RiverPod
What	Built in method for updating widget state	Simple state management solution uses inherited widget mechanism	Advance state management library.
Complexity	Simple	Moderate	Advanced
Scalability	Limited	moderate to high	High
Dependency management	None	Basic	Advanced.
Testability	Limited	Moderate	High
Suitable scenario	- Simple applications - Small subtree of widgets - Rapid prototyping	- Medium to large scale apps. - state sharing across widgets. - Dependency injection	- large scale apps - Fine-grained control over dependencies - Testability & separation of concerns

4) Firebase integration

→ a) Step 1 : Setup firebase project:

- Visit console.firebase.google.com and log in
- Click on 'Add project' to create a new firebase project.
- Provide a name for the project and choose your country/region.

Step 2 : Add Flutter to Project.

- Open flutter project on your IDE
- Open pubsec.yaml file
- Add the firebase core and other firebase libraries you plan to use, under the dependencies section

- Save the file and run flutter pub get in your terminal to fetch new dependencies.

Step 3: Connect App to firebase
- Go back to firebase console
- In the Firebase project overview, click on 'Add app' button
- Choose the flutter icon to register your flutter app
- Enter a package name
- Click on 'Register App' and download google-services json file.
- Move the downloaded ~~and download~~ google file into the android/app directory of your flutter project

Step 4: Configure Android app
- Add classpath in the android/build gradle file
- open android/app/build gradle file and add plug in

Step 5: Initialize firebase
- Open your apps main.Dart file.
- Import firebase_core package:
  import 'package:firebase_core/firebase_core.dart';
- In main function initialize Firebase:

```
void main() async
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(MyApp());
}
```

b) Benefits:

1) Real-time database helps to store and synchronize data
2) Google analytic integration enabling tracking of user journey.
3) Crash reporting
4) Fast and secure web hosting.
5) Authentication.
6) Offers easy storage for content
7) Provides ready to use APIs for ML.
8) Dynamic links to share app recommendations.
9) App indexing helps engage users.

c) Services & data synchronisation.

→ Services

i) Authentication - User authentication and authorisation.
ii) Cloud firebase - Flexible scalable NoSQL cloud DB for storing and syncing app data in real time; supports offline data access.
iii) Firebase cloud messaging - Allows sending push notifications to flutter apps across platforms.
iv) Firebase storage - Storage of various content
v) Analytics - measure user behaviour and gain insights.

Synchronisation is achieved by real-time sync mechanisms like cloud Firestore:

1) When data is updated in FS DB, FB automatically synchronizes these changes in real time.

2) Changes can be listened to in FS using stream base APIs, allowing the reactively update in UI.
3) Supports offline data persistence; automatic sync when online.
4) Security roles ensure that only authenticate users can access, or modify data.

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.	1
Name	Gargi Angne
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4:Understand various PWA frameworks and their requirements LO5: Design and Develop a responsive User Interface by applying PWA Design techniques LO6:Develop and Analyze PWA Features and deploy it over app hosting solutions
Grade:	4

Gargi Angne
DLSA

PWA

Assignment 2

1) Progressive Web App (PWA)

→ • A progressive web app (PWA) is a set of mobile web application development techniques that entails building apps that feel and look like native ones.

• Using a web stack, PWA combine a rich functionality and smooth user experience associated with native apps.

• Significance:

- Cross platform compatibility (can run on any device)
- Improved user experience (similar to native apps, with fast loading times, smooth animations and offline functionality)
- Offline functionality (allows users to access when offline)
- Push notifications (It can send push notification to users, keeping them engaged and informed even when the app is not actively in use, similar to native apps)
- Installation free (PWAs don't need downloaded and installed. These can be simply added to the home screen)

• Key characteristics that differentiate PWA:

- Cross-Platform compatibility
- No installation required
- Offline functionality
- Automatic updates
- Low device storage usage.

• PWAs offer a modern approach to web development by combining the best features of web and mobile applications, providing a cross-platform, user-friendly, and efficient solution for delivering engaging experiences to users.

2) Responsive web design

→ • Responsive web design refers to a web design approach that enables your website content to "respond" to and adapt to

the screen and window sizes of its accessed devices.

- Importance:
  - Consistent user experience (regardless of screen size or resolution. Crucial for usability and engagement across various platforms.) maintaining
  - Accessibility (Due to responsiveness, accesible to a broader range of users)
  - Improved SEO (tend to perform better in search engine rankings because search engines prioritize mobile-friendly websites)
  - Cost efficiency (Create a single codebase that adapts to various orientations. This approach is cost-effective and easier to maintain)

Aspect	Responsive web design	Fluid web design	Adaptive web design
layout adjustment	Based on viewport size	Elements are sized proportionally	Switches between predefined layouts
Control mechanism	Used media queries for layout changes	Relies on proportional sizing	Used predefined breakpoints.
Flexibility	Continuous adaption	Continuous adaption	Control at specific breakpoints
Development complexity	Moderate	Moderate	Needs more effort
user experience across devices)	consistent	Consistent	May vary (at predefined breakpoints)
SEO impact	Positive (mobile-friendly ranking boost)	Positive	May vary (depending on implementation)
Cost-effectiveness	Yes due to single codebase	Yes due to proportional sizing	No, may require more resources initially.

Sundaram

8) Lifecycle of service-workers

→ A serviceworker is a script that runs independently in the browser background.

i) Registration -

- Involves the initial setup of the service worker.
- Developers define a js file that acts as a service worker for web application.
- It is registered using 'navigator.serviceworker.register()' method.
- During registration, the scope is specified which determines which URLs to control; the scope restricts the service worker's access to resources and defines the context in which it operates.
- Once registered, sw is installed in the background (by browser)

ii) Installation -

- The browser then downloads the sw script.
- Browser fires 'install' event which triggers the execution of sw's 'install' event handler.
- This can be used to cache static assets and other resources need for offline functionality/performance optimization.
- Provides an opportunity to set up the initial state of sw and prepare it for activation.

iii) Activation -

- Browser fires 'activate' event.
- During activation, developers can perform tasks like cleaning up outdated caches, managing versioning or initialising service-specific logic.
- Here, sw takes control of clients within its scope, intercepting network requests and handling events.
- Remains active until explicitly unregistered or replaced by newer version

4) IndexDB in SW

→ • IndexDB is a NoSQL DB which is a key-value database that stores the data in the user's browser.
• It helps store large data, that is structured, locally.
• Use within SW:
- Caching resources: By storing cache resources locally, the application can load and serve them more quickly, especially when the user is offline/on slow network connection.
- Offline support: SW can intercept network requests and serve cached data from IndexDB when the network is unavailable, thus allowing offline access.
- Dynamic data storage: Provides a flexible and efficient way to store dynamic data generated by the web app. SW can use IndexedDB to store user preferences, session data, application state, etc. that needs to persists across sessions/browser restarts.
- Background synchronization: Conjunction of IndexedDB with background synchronisation techniques to queue and store data while offline and synchronise it later on, so as to maintain data consistency.
- Efficient data management: Offers advanced features like indexes, transactions and cursors which allow SW to efficiently query, update, retrieve and delete stored data.

Project Title: Flutter: Youtube Clone / PWA: Travel website **Roll No. 1**