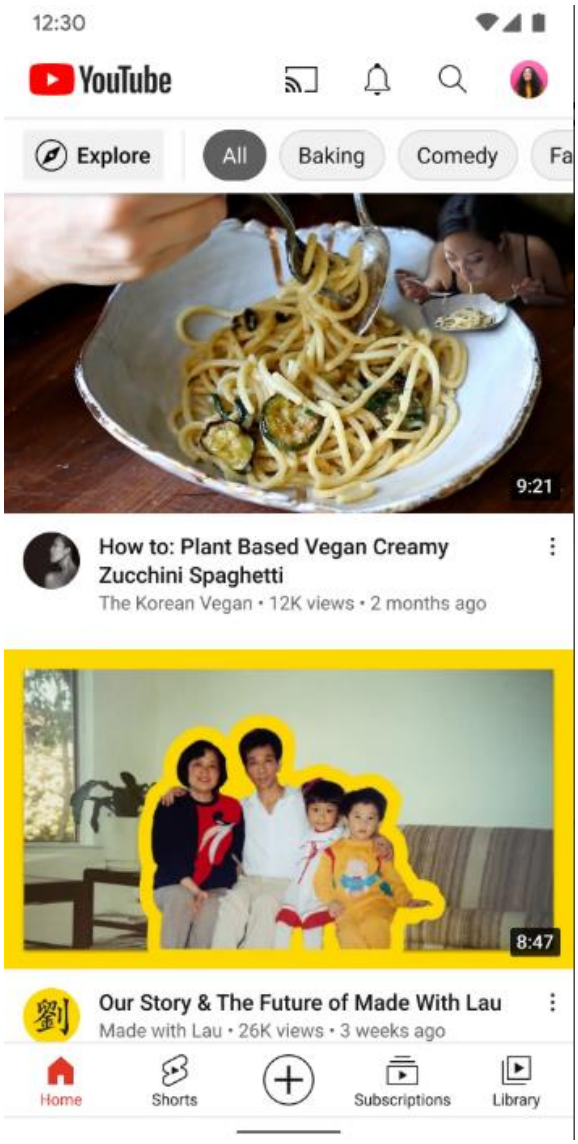
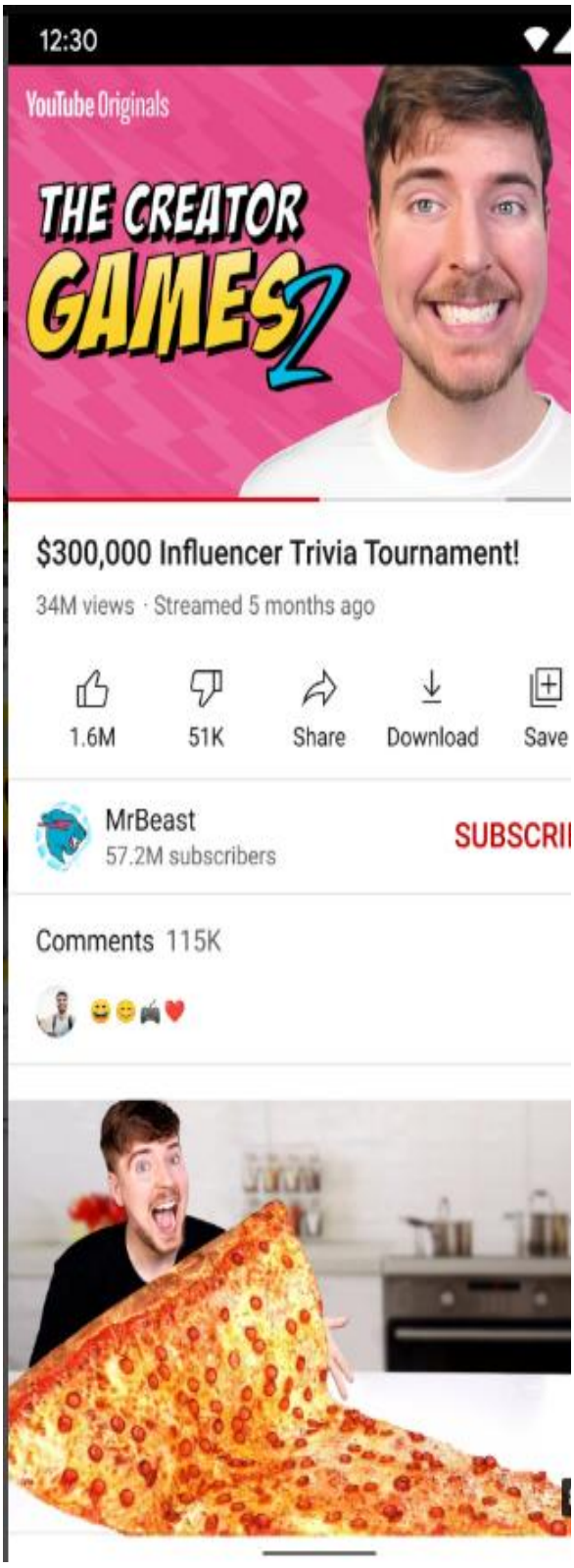
MAD and PWA LAB

Name: Gunjan Chandnani **Class:** D15A **Roll no:** 06

Aim: Selecting features for application development, the features should comprise of:

1. Common widgets
2. Should include icons, images, charts etc.
3. Should have an interactive Form
4. Should apply navigation, routing and gestures
5. Should connect with FireBase database

Screen Shot	Features
	**Landing Page :** 1. Sign in using email address OR using with Google 2. Firebase for authentication 3. **FireBase:** User : Username, email and password.

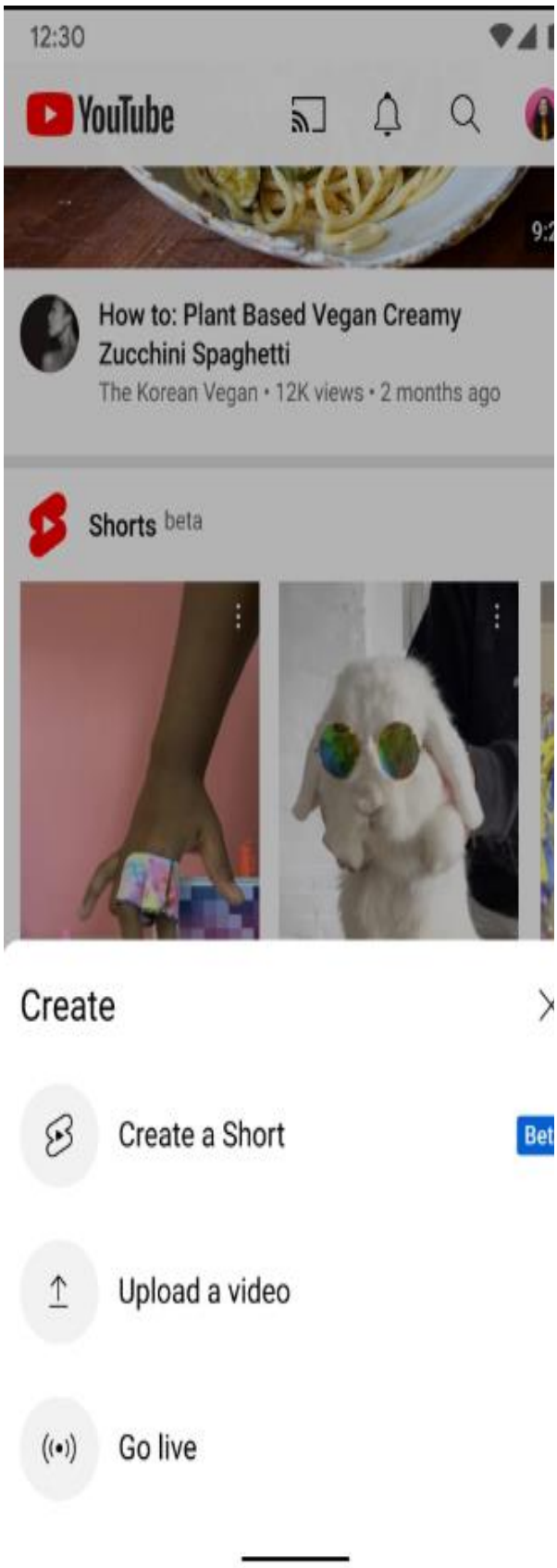


After Clicking video:

1. Navigation Button
2. Icons for - like, unlike, share, download and share
3. Subscribe Button
4. Suggestion Video

After Clicking Reel Icon:

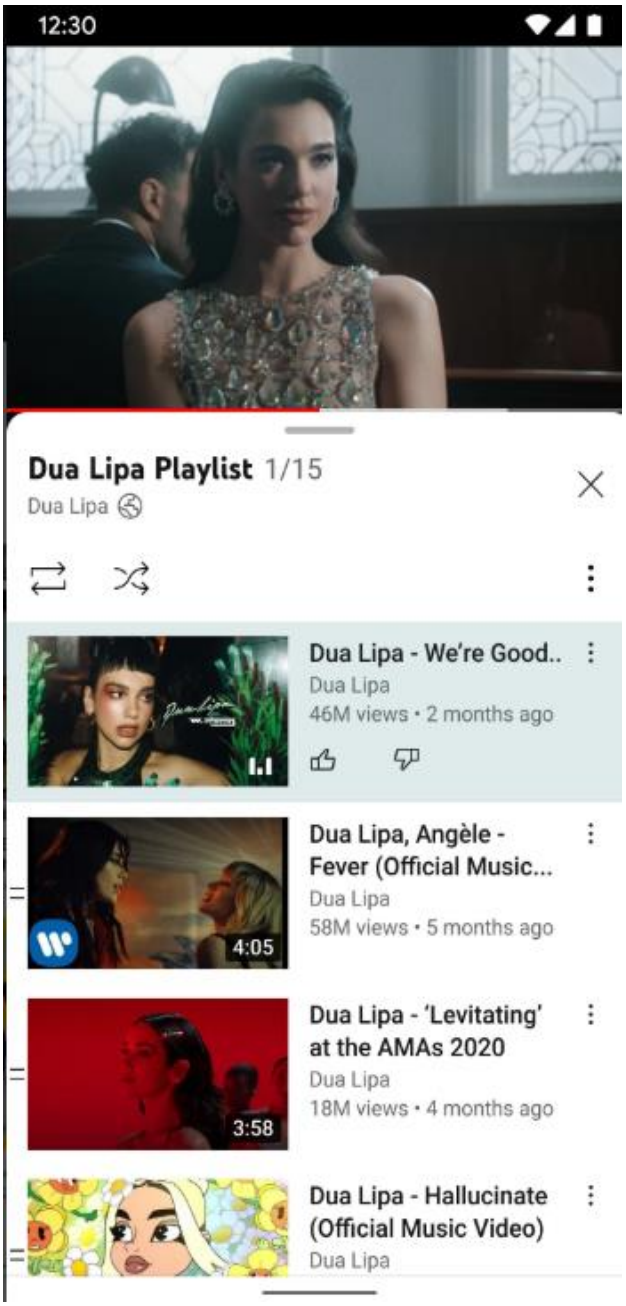
1. There will be a Suggestive Reel
2. Icons for Like, Unlike, Comment and Share



In Home Page:

After Clicking Plus Icon:

1. Create a short option
2. Upload a video option
3. Go a live option



Playlist Page:

1. List of dedicated channel playlist

Name: Gunjan Chandnani Roll No: 06 Div: D15A

Experiment 02

Aim:

To design Flutter UI by including common widgets.

Theory:

Flutter provides a wide range of pre-built widgets that help developers create visually appealing and responsive user interfaces. Widgets in Flutter can be classified into two main types: Stateless Widgets, which remain unchanged once built, and Stateful Widgets, which can dynamically update based on user interactions.

To structure a UI layout, Container is commonly used as a fundamental widget that holds other widgets while allowing customization with padding, margins, and colors. Row and Column widgets help arrange elements horizontally and vertically, respectively, ensuring a flexible and responsive design. For overlapping elements, Stack is used, allowing widgets to be layered on top of each other.

Flutter also provides interactive widgets such as TextField for user input, Button widgets like ElevatedButton and FloatingActionButton for user interactions, and ListView for displaying scrollable lists. Additionally, Card is a useful widget for creating material design-like structured components with shadows and elevation effects.

For navigation, Flutter includes Navigator and Drawer, which help in managing multiple screens and creating side menus. Styling and customization can be done using ThemeData, which enables consistent design throughout the app. With these common widgets, developers can build intuitive, responsive, and visually rich applications efficiently.

Types of widgets:

1. Based on State Management

1.1 Stateless Widgets

Stateless widgets are immutable, meaning their properties do not change once they are built. They are used for displaying static content that does not depend on user interaction or external data. Examples include `Text`, `Icon`, and `Container`.

1.2 Stateful Widgets

Stateful widgets maintain an internal state that can change dynamically based on user interactions or data updates. They require a `State` object to manage changes. Examples include `TextField`, `Checkbox`, and `ListView`.

---

2. Based on Functionality

2.1 Layout Widgets

Layout widgets help structure the UI by arranging elements in a specific order.

- Container: A flexible box that holds child widgets with padding, margins, and styling.
- Row & Column: Arrange widgets horizontally (`Row`) or vertically (`Column`).
- Stack: Overlays widgets on top of each other for layered UI designs.
- Expanded & Flexible: Adjust child widget sizes dynamically based on available space.

2.2 Interactive Widgets

These widgets respond to user actions like clicks, typing, or gestures.

- TextField: Allows user input.
- ElevatedButton & TextButton: Buttons for user interaction.
- FloatingActionButton (FAB): A circular button used for quick actions.
- GestureDetector: Detects taps, swipes, and other gestures.

2.3 Scrolling Widgets

These widgets help display content in a scrollable format.

- ListView: Displays a scrollable list of items.
- GridView: Shows items in a grid format.
- SingleChildScrollView: Enables scrolling for a single child widget.

2.4 Styling & Theming Widgets

These widgets help in customizing the UI.

- Padding & Margin: Add spacing around widgets.
- DecoratedBox: Provides styling like background color, border, and shadows.
- Theme & ThemeData: Defines app-wide themes and styles.

Code:

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


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


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

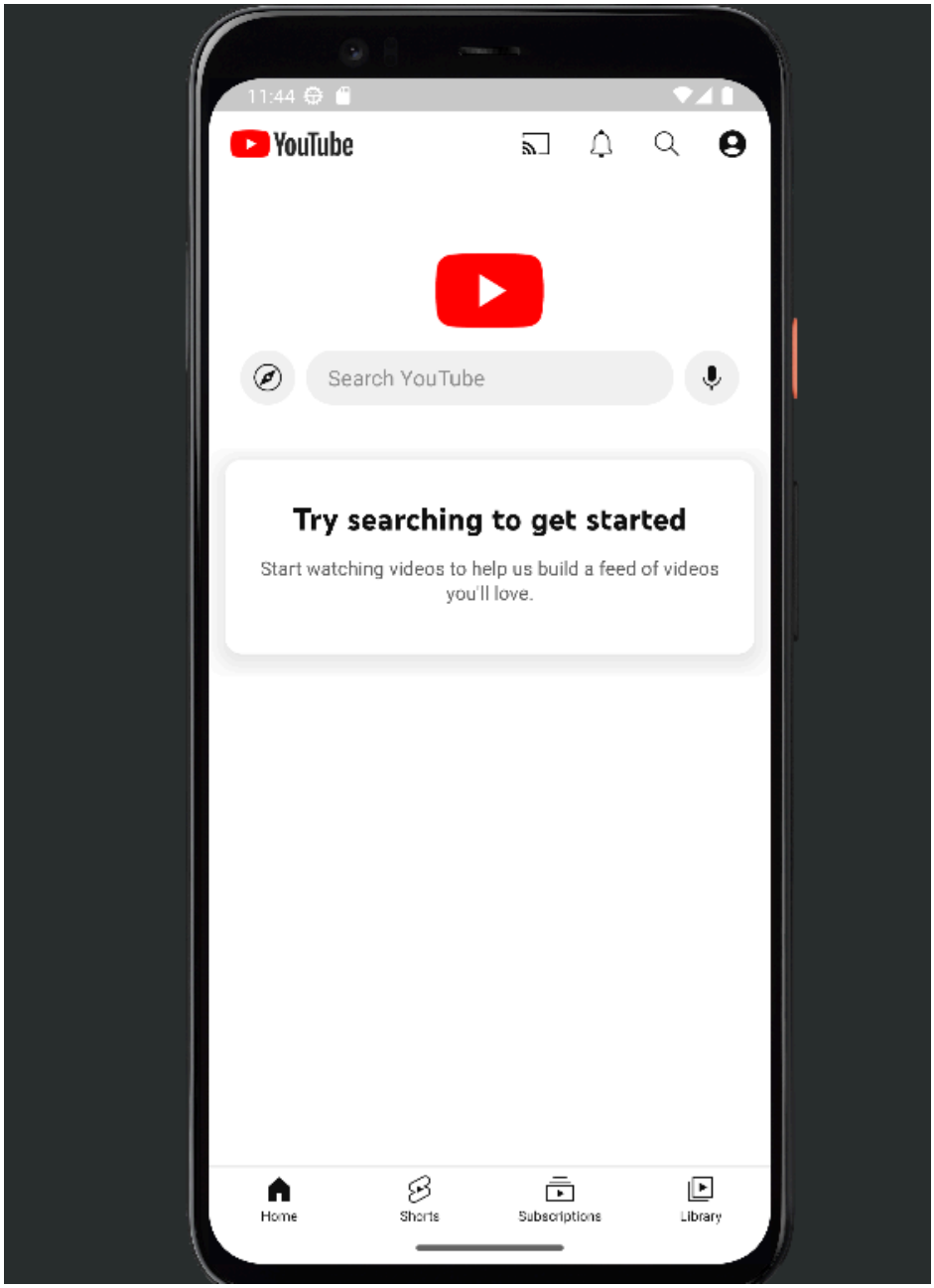
```
class YouTubeHomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.white,
        elevation: 0,
        title: Image.asset('assets/youtube_logo.png', height: 40),
        actions: [
          IconButton(icon: Icon(Icons.cast, color: Colors.black), onPressed: () {}),
          IconButton(icon: Icon(Icons.notifications_none, color: Colors.black),
onPressed: () {}),
          IconButton(icon: Icon(Icons.search, color: Colors.black), onPressed: () {}),
          CircleAvatar(backgroundColor: Colors.grey, radius: 15),
          SizedBox(width: 10),
        ],
      ),
      body: ListView.builder(
        itemCount: 10,
        itemBuilder: (context, index) {
          return VideoCard();
        },
      ),
```

```
      bottomNavigationBar: BottomNavigationBar(
        type: BottomNavigationBarType.fixed,
        selectedItemColor: Colors.red,
        unselectedItemColor: Colors.grey,
        items: [
          BottomNavigationBarItem(icon: Icon(Icons.home), label: 'Home'),
          BottomNavigationBarItem(icon: Icon(Icons.play_arrow), label: 'Shorts'),
          BottomNavigationBarItem(icon: Icon(Icons.subscriptions), label:
'Subscriptions'),
          BottomNavigationBarItem(icon: Icon(Icons.video_library), label: 'Library'),
        ],
      ),
      floatingActionButton: FloatingActionButton(
        backgroundColor: Colors.red,
        child: Icon(Icons.add),
        onPressed: () {},
      ),
    );
  }
}

class VideoCard extends StatelessWidget {
  @override
```

```
Widget build(BuildContext context) {
  return Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      Image.network('https://via.placeholder.com/400x200'),
      Padding(
        padding: const EdgeInsets.symmetric(vertical: 8.0, horizontal: 12.0),
        child: Row(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            CircleAvatar(backgroundColor: Colors.grey),
            SizedBox(width: 10),
            Expanded(
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text(
                    'Video Title',
                    style: TextStyle(fontWeight: FontWeight.bold, fontSize: 16),
                  ),
                  SizedBox(height: 5),
                  Text(
                    'Channel Name • 1M views • 2 days ago',
```

```
                    style: TextStyle(color: Colors.grey[600], fontSize: 14),
                  ),
                ],
              ),
            ),
            Icon(Icons.more_vert),
          ],
        ),
      ),
    ],
  );
 }
}
```

OutPut:


11:44
YouTube
Search YouTube
Try searching to get started
Start watching videos to help us build a feed of videos you'll love.
Home
Shorts
Subscriptions
Library

EXPERIMENT NO: - 03

Name:- Gunjan Chandnani Class:- D15A
Roll:No: - 06

AIM: - To include icons, images, fonts in Flutter app.

Theory: -

Flutter is a versatile open-source UI framework , which allows developers to build natively compiled applications for mobile, web, and desktop platforms from a single codebase.
One of the key strengths of Flutter is its flexibility in creating highly customizable UIs. This practical focuses on incorporating essential visual elements — icons, images, and custom fonts—into a Flutter application.
These elements enhance the visual appeal and usability of the app, providing an engaging experience for users.
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.

Visual elements play a significant role in app development.

• Enhanced User Experience: Images and icons make your app visually appealing and userfriendly.

• Information Conveyance: They convey information quickly and intuitively. A wellchosen icon can replace lengthy text.

• Branding: Custom icons and images reinforce your app's branding, making it memorable.

➢ Adding Icons in Flutter Flutter provides built-in material design icons through the Icons class. Custom icons can also be added using third-party packages such as flutter_launcher_icons and font_awesome_flutter. Icon(Icons.home, size: 40,)

➢ **Adding Images in Flutter**

Flutter supports images from three sources:

1. Assets (Stored locally in the project)

• Place the image inside the assets/images folder in the project.

• Declare the image in pubspec.yaml flutter: assets: - assets/images/sample.png • Display the image in the app

Image.asset('assets/images/sample.png');

2. Network (Fetched from the internet)

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

Image.network('https://example.com/sample.jpg');

3. Memory or File (Stored on the device)

➢ Adding Custom Fonts in Flutter By default, Flutter uses the Roboto font, but custom fonts can be added for a unique UI.

• Download the font and place it in the assets/fonts/ folder.

• Declare the font in pubspec.yaml

• Use the font in the app Text('Custom Font Example',

style: TextStyle(fontFamily: 'CustomFont', fontSize: 24),);

Code:

Main.dart

```dart
// ignore_for_file: unnecessary_null_comparison, prefer_const_constructors

import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:youtube_clone/cores/screens/loader.dart';
import 'package:youtube_clone/features/auth/pages/login_page.dart';
import 'package:youtube_clone/features/auth/pages/username_page.dart';
import
'package:youtube_clone/features/upload/long_video/video_details_page.dart';
import 'package:youtube_clone/firebase_options.dart';
import 'package:youtube_clone/home_page.dart';

void main() async {
 WidgetsFlutterBinding.ensureInitialized();
 await Firebase.initializeApp(
   options: DefaultFirebaseOptions.currentPlatform,
 );
 runApp(ProviderScope(child: MyApp()));
}

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

 @override
 Widget build(BuildContext context, WidgetRef ref) {
   return MaterialApp(
     debugShowCheckedModeBanner: false,
     // we check wthether we are signed in or not
     home: StreamBuilder(
       stream: FirebaseAuth.instance.authStateChanges(),
       builder: (context, snapshot) {
         if (!snapshot.hasData) {
           return LoginPage();
         } else if (snapshot.connectionState == ConnectionState.waiting) {
           return Loader();
         }
         return StreamBuilder(
           stream: FirebaseFirestore.instance
               .collection("users")
               .doc(FirebaseAuth.instance.currentUser!.uid)
               .snapshots(),
           builder: (context, snapshot) {
             final user = FirebaseAuth.instance.currentUser;
             if (!snapshot.hasData || !snapshot.data!.exists) {
```

```
              return UsernamePage(
                displayName: user!.displayName!,
                profilePic: user.photoURL!,
                email: user.email!,
              );
            } else if (snapshot.connectionState == ConnectionState.waiting) {
              return Loader();
            }
            return HomePage();
          },
        );
      },
    ),
  );
 }
}
```

upload_bottom_sheet.dart:

```
// ignore_for_file: use_build_context_synchronously

import 'package:flutter/material.dart';
import 'package:youtube_clone/cores/methods.dart';
import 'package:youtube_clone/cores/widgets/image_item.dart';

class CreateBottomSheet extends StatelessWidget {
 const CreateBottomSheet({super.key});

 @override
 Widget build(BuildContext context) {
   return Container(
     color: const Color(0xffFFFFFF),
     child: Padding(
       padding: const EdgeInsets.only(left: 7, top: 12),
       child: SizedBox(
         height: 270,
         child: Column(
           crossAxisAlignment: CrossAxisAlignment.start,
           children: [
             const Padding(
               padding: EdgeInsets.only(left: 10),
               child: Text(
                 "Create",
                 style: TextStyle(
                   fontSize: 20,
                   fontWeight: FontWeight.w500,
                 ),
               ),
             ),
             const SizedBox(height: 28),
```

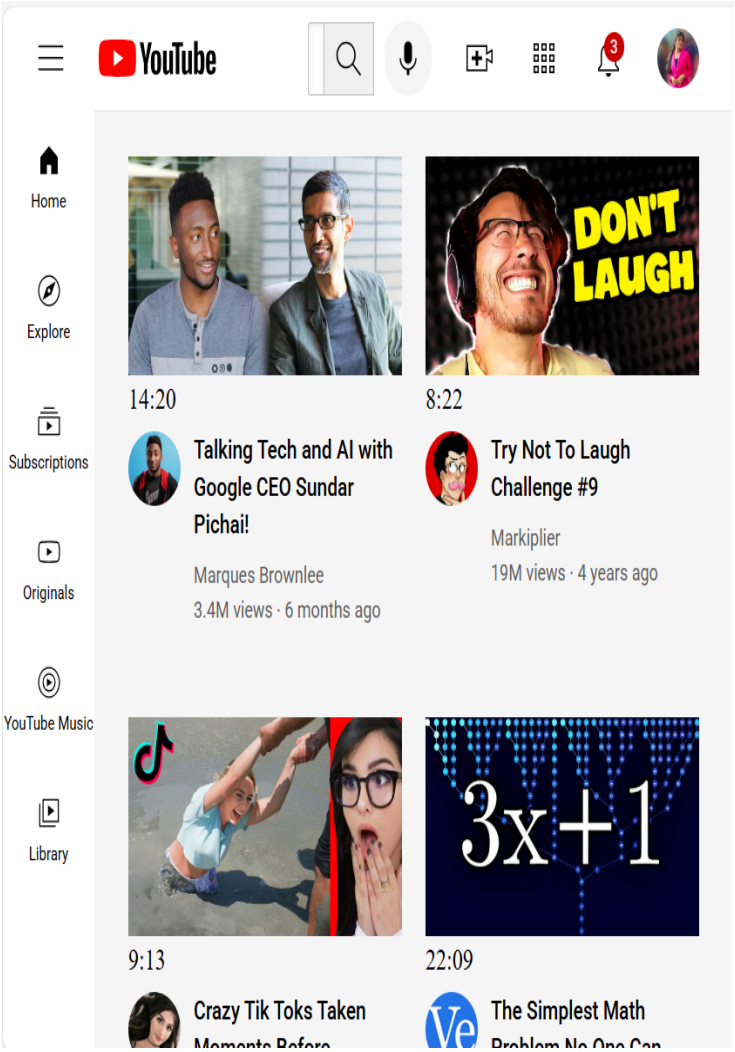
```
            SizedBox(
              height: 38,
              child: ImageItem(
                itemText: "Create a Short",
                itemClicked: () async {
                  await pickShortVideo(context);
                },
                imageName: "short-video.png",
                haveColor: true,
              ),
            ),
            const SizedBox(height: 12),
            SizedBox(
              height: 38,
              child: ImageItem(
                itemText: "Upload a Video",
                itemClicked: () async {
                  await pickVideo(context);
                },
                imageName: "upload.png",
                haveColor: true,
              ),
            ),
            const SizedBox(height: 12),
            SizedBox(
              height: 38,
              child: ImageItem(
                itemText: "Go Live",
                itemClicked: () {},
                imageName: "go-live.png",
                haveColor: true,
              ),
            ),
            const SizedBox(height: 12),
            SizedBox(
              height: 38,
              child: ImageItem(
                itemText: "Create a post",
                itemClicked: () {},
                imageName: "create-post.png",
                haveColor: true,
              ),
            ),
          ],
        ),
      ),
    ),
  );
}
```

video_model.dart:

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

class VideoModel {
 final String videoUrl;
 final String thumbnail;
 final String title;
 final DateTime datePublished;
 final int views;
 final String videoId;
 final String userId;
 final List likes;
 final String type;
 VideoModel({
   required this.videoUrl,
   required this.thumbnail,
   required this.title,
   required this.datePublished,
   required this.views,
   required this.videoId,
   required this.userId,
   required this.likes,
   required this.type,
 });

 Map<String, dynamic> toMap() {
   return <String, dynamic>{
     'videoUrl': videoUrl,
     'thumbnail': thumbnail,
     'title': title,
     'datePublished': datePublished,
     'views': views,
     'videoId': videoId,
     'userId': userId,
     'likes': likes,
     'type': type,
   };
 }

 factory VideoModel.fromMap(Map<String, dynamic> map) {
   return VideoModel(
     videoUrl: map['videoUrl'] as String,
     thumbnail: map['thumbnail'] as String,
     title: map['title'] as String,
     datePublished: map["datePublished"] is Timestamp
         ? (map["datePublished"] as Timestamp).toDate()
         : DateTime.fromMillisecondsSinceEpoch(
             map["datepublished"] as int,
```

```
            ),
      views: map['views'] as int,
      videoId: map['videoId'] as String,
      userId: map['userId'] as String,
      likes: List.from(
        (map['likes'] as List),
      ),
      type: map['type'] as String,
    );
  }
}
```

OutPut:



Create

Create a Short

Upload a Video

Go Live

Create a post

Enter the title

T this project by Gunjan - 06[D15A]

Enter the Description

My first yt vdo

SELECT THUMBNAIL

Name: Gunjan Chandnani Div: D15A Roll No: 06

Exp 4

Aim:

To create an interactive Form using form widget.

Theory:

A form in Flutter is a structured container that collects user input through various fields like text fields, dropdowns, checkboxes, and buttons. It plays a crucial role in applications that require user data entry, such as login pages, registration forms, and feedback submissions.

Flutter provides the Form widget, which works alongside TextFormField and other input elements to manage validation, state handling, and error messages efficiently. By using form validation techniques, developers can ensure data accuracy and enhance user experience. When you create a form, it is necessary to provide the GlobalKey. This key uniquely identifies the form and allows you to do any validation in the form fields. The form widget uses child widget TextFormField to provide the users to enter the text field. This widget renders a material design text field and also allows us to display validation errors when they occur.

Creation of a Form

➢ While creating a form in Flutter, the Form widget is essential as it acts as a container for grouping multiple form fields and managing validation.

➢ A GlobalKey is required to uniquely identify the form and enable validation or data retrieval from the form fields.

➢ The TextFormField widget is used to provide input fields where users can enter data such as names, phone numbers, or email addresses.

➢ To enhance the appearance and usability of input fields, InputDecoration is used, allowing customization of labels, icons, borders, and hint text.

➢ Validation plays a crucial role in forms, and the validator property within TextFormField ensures user input meets specific criteria before submission.

Some Properties of Form Widget

• **key:** A GlobalKey that uniquely identifies the Form. You can use this key to interact with the form, such as validating, resetting, or saving its state.

• **child:** The child widget that contains the form fields. Typically, this is a Column, ListView, or another widget that allows you to arrange the form fields vertically.

• **autovalidateMode:** An enum that specifies when the form should automatically validate its fields. Some Methods of Form Widget

• **validate():** This method is used to trigger the validation of all the form fields within the Form. It returns true if all fields are valid, otherwise false. You can use it to check the overall validity of the form before submitting it.

• **save():** This method is used to save the current values of all form fields. It invokes the onSaved callback for each field. Typically, this method is called after validation succeeds.

• **reset():** Resets the form to its initial state, clearing any user-entered data.

• **currentState:** A getter that returns the current FormState associated with the Form

Code:

Home_page.dart

```
// ignore_for_file: invalid_use_of_visible_for_testing_member

import 'package:cached_network_image/cached_network_image.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:path/path.dart';
import 'package:youtube_clone/cores/screens/error_page.dart';
import 'package:youtube_clone/cores/screens/loader.dart';
import 'package:youtube_clone/cores/widgets/image_button.dart';
import 'package:youtube_clone/features/account/account_page.dart';
import 'package:youtube_clone/features/auth/provider/user_provider.dart';
import 'package:youtube_clone/features/content/bottom_navigation.dart';
```

```dart
import 'package:youtube_clone/features/upload/upload_bottom_sheet.dart';
import 'package:youtube_clone/pages_list.dart';

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

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  int currentIndex = 0;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: const Color(0xffFFFFFF),
      body: SafeArea(
        child: Column(
          children: [
            Row(
              children: [
                Image.asset(
                  "assets/images/youtube.jpg",
                  height: 36,
                ),
                const SizedBox(width: 4),
                const Spacer(),
                Padding(
                  padding: const EdgeInsets.only(right: 12),
                  child: SizedBox(
                    height: 42,
                    child: ImageButton(
                      image: "cast.png",
                      onPressed: () {},
                      haveColor: false,
```

```
      ),
    ),
  ),
  SizedBox(
    height: 38,
    child: ImageButton(
      image: "notification.png",
      onPressed: () {},
      haveColor: false,
    ),
  ),
  Padding(
    padding: const EdgeInsets.only(left: 12, right: 15),
    child: SizedBox(
      height: 41.5,
      child: ImageButton(
        image: "search.png",
        onPressed: () {},
        haveColor: false,
      ),
    ),
  ),
  Consumer(
    builder: (context, ref, child) {
      return ref.watch(currentUserProvider).when(
            data: (currentUser) => Padding(
              padding: const EdgeInsets.only(right: 12),
              child: GestureDetector(
                onTap: () {
                  Navigator.push(
                    context,
                    MaterialPageRoute(
                      builder: (context) => AccountPage(
                        user: currentUser,
                      ),
```

```
                                  ),
                              );
                            },
                            child: CircleAvatar(
                              radius: 14,
                              backgroundColor: Colors.grey,
                              backgroundImage:
CachedNetworkImageProvider(
                                currentUser.profilePic,
                              ),
                            ),
                          ),
                        ),
                        error: (error, stackTrace) => const ErrorPage(),
                        loading: () => const Loader(),
                      );
                },
              ),
            ],
          ),
          Expanded(
            child: pages[currentIndex],
          ),
        ],
      ),
    ),
    bottomNavigationBar: BottomNavigation(
      onPressed: (index) {
        if (index != 2) {
          currentIndex = index;
          setState(() {});
        } else {
          showModalBottomSheet(
            context: context,
            builder: (context) => const CreateBottomSheet(),
          );
```

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

Main.dart

```
// ignore_for_file: unnecessary_null_comparison, prefer_const_constructors

import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:youtube_clone/cores/screens/loader.dart';
import 'package:youtube_clone/features/auth/pages/login_page.dart';
import 'package:youtube_clone/features/auth/pages/username_page.dart';
import
'package:youtube_clone/features/upload/long_video/video_details_page.dart'
;
import 'package:youtube_clone/firebase_options.dart';
import 'package:youtube_clone/home_page.dart';

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

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

  @override
  Widget build(BuildContext context, WidgetRef ref) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      // we check wthether we are signed in or not
      home: StreamBuilder(
        stream: FirebaseAuth.instance.authStateChanges(),
        builder: (context, snapshot) {
          if (!snapshot.hasData) {
            return LoginPage();
          } else if (snapshot.connectionState == ConnectionState.waiting)
{
            return Loader();
          }
          return StreamBuilder(
            stream: FirebaseFirestore.instance
                .collection("users")
                .doc(FirebaseAuth.instance.currentUser!.uid)
                .snapshots(),
            builder: (context, snapshot) {
              final user = FirebaseAuth.instance.currentUser;
              if (!snapshot.hasData || !snapshot.data!.exists) {
                return UsernamePage(
                  displayName: user!.displayName!,
                  profilePic: user.photoURL!,
                  email: user.email!,
                );
              } else if (snapshot.connectionState ==
ConnectionState.waiting) {
                return Loader();
              }
              return HomePage();
```

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

In folder:
Long_video
|_video_detail.dart

```
// ignore_for_file: public_member_api_docs, sort_constructors_first
import 'dart:io';

import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:uuid/uuid.dart';

import 'package:youtube_clone/cores/methods.dart';
import
'package:youtube_clone/features/upload/long_video/video_repository.dart';

class VideoDetailsPage extends ConsumerStatefulWidget {
  final File? video;
  const VideoDetailsPage({
    super.key,
    this.video,
  });

  @override
  ConsumerState<VideoDetailsPage> createState() =>
_VideoDetailsPageState();
}
```

```dart
class _VideoDetailsPageState extends ConsumerState<VideoDetailsPage> {
  final titleController = TextEditingController();
  final descriptionController = TextEditingController();
  File? image;
  bool isThumbnailSelected = false;
  String randomNumber = const Uuid().v4();
  String videoId = const Uuid().v4();
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
        child: Padding(
          padding: const EdgeInsets.only(
            top: 20,
            left: 10,
            right: 10,
          ),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.stretch,
            children: [
              const Text(
                "Enter the title",
                style: TextStyle(
                  fontSize: 15,
                  color: Colors.grey,
                ),
              ),
              const SizedBox(height: 5),
              TextField(
                controller: titleController,
                decoration: const InputDecoration(
                  hintText: "Enter the title",
                  prefixIcon: Icon(Icons.title),
                  border: OutlineInputBorder(
                    borderSide: BorderSide(
```

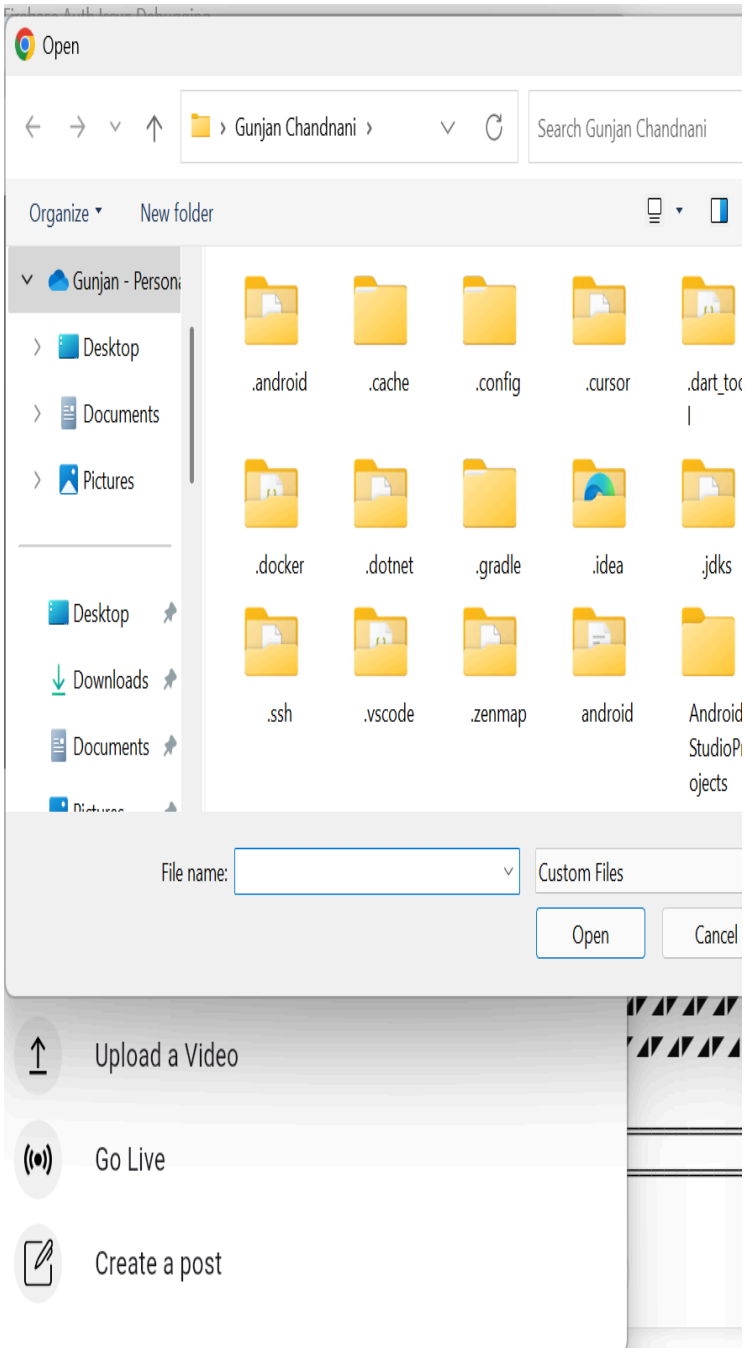
```
            color: Colors.blue,
          ),
        ),
      ),
    ),
    const SizedBox(height: 30),
    const Text(
      "Enter the Description",
      style: TextStyle(
        fontSize: 15,
        color: Colors.grey,
      ),
    ),
    const SizedBox(height: 5),
    TextField(
      controller: descriptionController,
      maxLines: 5,
      decoration: const InputDecoration(
        hintText: "Enter the Description",
        border: OutlineInputBorder(
          borderSide: BorderSide(
            color: Colors.blue,
          ),
        ),
      ),
    ),

    // select thumnbnail

    Padding(
      padding: const EdgeInsets.only(top: 12),
      child: Container(
        decoration: const BoxDecoration(
          color: Colors.blue,
          borderRadius: BorderRadius.all(
```

```
                Radius.circular(11),
              ),
            ),
            child: TextButton(
              onPressed: () async {
                // pick image
                image = await pickImage();
                isThumbnailSelected = true;
                setState(() {});
              },
              child: const Text(
                "SELECT THUMBNAIL",
                style: TextStyle(
                  color: Colors.white,
                ),
              ),
            ),
          ),
        ),
        isThumbnailSelected
            ? Padding(
                padding: const EdgeInsets.only(top: 12, bottom: 12),
                child: Image.file(
                  image!,
                  cacheHeight: 160,
                  cacheWidth: 400,
                ),
              )
            : const SizedBox(),

        isThumbnailSelected
            ? Padding(
                padding: const EdgeInsets.only(top: 12),
                child: Container(
                  decoration: const BoxDecoration(
```

```
                          color: Colors.green,
                          borderRadius: BorderRadius.all(
                            Radius.circular(11),
                          ),
                        ),
                        child: TextButton(
                          onPressed: () async {
                            // publish video

                            String thumbnail = await putFileInStorage(
                                image, randomNumber, "image");

                            String videoUrl = await putFileInStorage(
                                widget.video, randomNumber, "video");


ref.watch(longVideoProvider).uploadVideoToFirestore(
                                  videoUrl: videoUrl,
                                  thumbnail: thumbnail,
                                  title: titleController.text,
                                  videoId: videoId,
                                  datePublished: DateTime.now(),
                                  userId:

FirebaseAuth.instance.currentUser!.uid,
                                );
                          },
                          child: const Text(
                            "Publish",
                            style: TextStyle(
                              color: Colors.white,
                            ),
                          ),
                        ),
                      ),
                    ),
                  )
```

```
                    : const SizedBox(),
              ],
            ),
          ),
        ),
      );
  }
}
```

OutPut:


Open
Gunjan Chandnani
Search Gunjan Chandnani
Organize
New folder
Gunjan - Persona
Desktop
Documents
Pictures
Desktop
Downloads
Documents
.android
.cache
.config
.cursor
.dart_tool
.docker
.dotnet
.gradle
.idea
.jdks
.ssh
.vscode
.zenmap
android
AndroidStudioProjects
File name:
Custom Files
Open
Cancel
Upload a Video
Go Live
Create a post

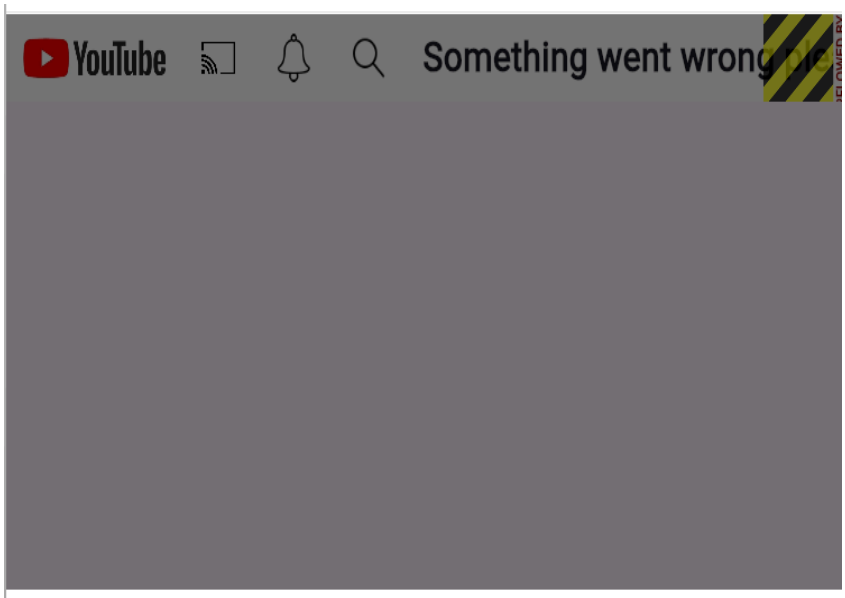
Enter the title

T this project by Gunjan - 06[D15A]

Enter the Description

My first yt vdo

SELECT THUMBNAIL



Create

Create a Short

Upload a Video

Go Live

Create a post

EXPERIMENT NO: - 05

Name:- Gunjan Chandnani **Class:- D15A** **Roll:No: - 06**

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

Theory: -
In Flutter, the screens and pages are known as routes, and these routes are just a widget.
In Android, a route is similar to an Activity. In any mobile app, navigating to different pages defines the workflow of the application, and the way to handle the navigation is known as routing.

Flutter provides a basic routing class MaterialPageRoute and two methods Navigator.push() and Navigator.pop() that shows how to navigate between two routes.

The following steps are required to start navigation in your application. Gestures enable the app to respond to user interactions, making the application more dynamic and responsive.

➢ Navigation and Routing in Flutter Navigation is the process of moving between different screens or pages in an app.
Flutter provides a simple and effective way to handle this through the use of the Navigator widget and routes.

1. Using Navigator Widget The Navigator widget manages a stack of routes, allowing for pushing and popping routes on the stack.

• **Pushing a Route:** To navigate to a new screen,
use Navigator.push().

• **Popping a Route:** To go back to the previous screen,
use Navigator.pop().

ElevatedButton(onPressed: () { Navigator.push(Gunjan Chandnani– D15A / 06 context, MaterialPageRoute(builder: (context) => SecondScreen()),);},); 2. Named Routes Flutter also allows the use of named routes to navigate, which can make the routing process cleaner, especially in larger applications.

MaterialApp(initialRoute: '/', routes: { '/': (context) => HomeScreen(), '/second': (context) => SecondScreen(), },);

Navigate to the route using Navigator.pushNamed() Navigator.pushNamed(context, '/second');
Handling Gestures in Flutter Gestures refer to user interactions with the app, such as taps, swipes, pinches, and drags.

Flutter provides several widgets and gesture detectors to handle these interactions. Tap Gestures The most common gesture is the tap, which can be handled using the GestureDetector widget or specific buttons like InkWell or ElevatedButton.
Long Press Gesture For long press gestures, Flutter provides the onLongPress callback in GestureDetector or InkWell.

Swipe and Drag Gestures Flutter also provides swipe and drag gesture handling. The onHorizontalDragUpdate and onVerticalDragUpdate callbacks are used for dragging gestures

2. Named Routes:
Flutter also allows the use of named routes to navigate, which can make the routing process cleaner, especially in larger applications.

MaterialApp(initialRoute: '/', routes: { '/': (context) => HomeScreen(), '/second': (context) => SecondScreen(), },);
Navigate to the route using Navigator.pushNamed() Navigator.pushNamed(context, '/second');

Handling Gestures in Flutter
Gestures refer to user interactions with the app, such as taps, swipes, pinches, and drags.
Flutter provides several widgets and gesture detectors to handle these interactions.

Tap Gestures
The most common gesture is the tap, which can be handled using the GestureDetector widget or specific buttons like InkWell or ElevatedButton.

Long Press Gesture For long press gestures, Flutter provides the onLongPress callback in GestureDetector or InkWell.

Code:

main.dart:

```
// ignore_for_file: unnecessary_null_comparison, prefer_const_constructors

import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:youtube_clone/cores/screens/loader.dart';
import 'package:youtube_clone/features/auth/pages/login_page.dart';
import 'package:youtube_clone/features/auth/pages/username_page.dart';
import
'package:youtube_clone/features/upload/long_video/video_details_page.dart';
import 'package:youtube_clone/firebase_options.dart';
import 'package:youtube_clone/home_page.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );
  runApp(ProviderScope(child: MyApp()));
}

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

  @override
  Widget build(BuildContext context, WidgetRef ref) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      // we check wthether we are signed in or not
      home: StreamBuilder(
        stream: FirebaseAuth.instance.authStateChanges(),
        builder: (context, snapshot) {
          if (!snapshot.hasData) {
```

```
          return LoginPage();
        } else if (snapshot.connectionState == ConnectionState.waiting) {
          return Loader();
        }
        return StreamBuilder(
          stream: FirebaseFirestore.instance
              .collection("users")
              .doc(FirebaseAuth.instance.currentUser!.uid)
              .snapshots(),
          builder: (context, snapshot) {
            final user = FirebaseAuth.instance.currentUser;
            if (!snapshot.hasData || !snapshot.data!.exists) {
              return UsernamePage(
                displayName: user!.displayName!,
                profilePic: user.photoURL!,
                email: user.email!,
              );
            } else if (snapshot.connectionState == ConnectionState.waiting) {
              return Loader();
            }
            return HomePage();
          },
        );
      },
    ),
  );
 }
}
```

login.dart:

```dart
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:youtube_clone/features/auth/repository/auth_service.dart';

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

 @override
 Widget build(BuildContext context, WidgetRef ref) {
   return Scaffold(
```

```
    body: SafeArea(
      child: Center(
        child: Column(
          children: [
            Padding(
              padding: const EdgeInsets.only(
                top: 20,
                bottom: 25,
              ),
              child: Image.asset(
                "assets/images/youtube-signin.jpg",
                height: 150,
              ),
            ),
            const Text(
              "Welcome To Youtube",
              style: TextStyle(
                fontSize: 30,
                fontWeight: FontWeight.bold,
                color: Colors.blueGrey,
              ),
            ),
            const Spacer(),
            Padding(
              padding: const EdgeInsets.only(bottom: 55),
              child: GestureDetector(
                onTap: () async {
                  await ref.read(authServiceProvider).signInWithGoogle();
                },
                child: Image.asset(
                  "assets/images/signinwithgoogle.png",
                  height: 60,
                ),
              ),
            ),
          ],
        ),
      ),
    ),
  );
 }
}
```

home_page.dart:

```
// ignore_for_file: invalid_use_of_visible_for_testing_member

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

```dart
import 'package:path/path.dart';
import 'package:youtube_clone/cores/screens/error_page.dart';
import 'package:youtube_clone/cores/screens/loader.dart';
import 'package:youtube_clone/cores/widgets/image_button.dart';
import 'package:youtube_clone/features/account/account_page.dart';
import 'package:youtube_clone/features/auth/provider/user_provider.dart';
import 'package:youtube_clone/features/content/bottom_navigation.dart';
import 'package:youtube_clone/features/upload/upload_bottom_sheet.dart';
import 'package:youtube_clone/pages_list.dart';

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

 @override
 State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
 int currentIndex = 0;
 @override
 Widget build(BuildContext context) {
   return Scaffold(
     backgroundColor: const Color(0xffFFFFFF),
     body: SafeArea(
       child: Column(
         children: [
           Row(
             children: [
               Image.asset(
                 "assets/images/youtube.jpg",
                 height: 36,
               ),
               const SizedBox(width: 4),
               const Spacer(),
               Padding(
                 padding: const EdgeInsets.only(right: 12),
                 child: SizedBox(
                   height: 42,
                   child: ImageButton(
                     image: "cast.png",
                     onPressed: () {},
                     haveColor: false,
                   ),
                 ),
               ),
               SizedBox(
                 height: 38,
                 child: ImageButton(
                   image: "notification.png",
```

```
                onPressed: () {},
                haveColor: false,
              ),
            ),
            Padding(
              padding: const EdgeInsets.only(left: 12, right: 15),
              child: SizedBox(
                height: 41.5,
                child: ImageButton(
                  image: "search.png",
                  onPressed: () {},
                  haveColor: false,
                ),
              ),
            ),
            Consumer(
              builder: (context, ref, child) {
                return ref.watch(currentUserProvider).when(
                      data: (currentUser) => Padding(
                        padding: const EdgeInsets.only(right: 12),
                        child: GestureDetector(
                          onTap: () {
                            Navigator.push(
                              context,
                              MaterialPageRoute(
                                builder: (context) => AccountPage(
                                  user: currentUser,
                                ),
                              ),
                            );
                          },
                          child: CircleAvatar(
                            radius: 14,
                            backgroundColor: Colors.grey,
                            backgroundImage: CachedNetworkImageProvider(
                              currentUser.profilePic,
                            ),
                          ),
                        ),
                      ),
                      error: (error, stackTrace) => const ErrorPage(),
                      loading: () => const Loader(),
                    );
              },
            ),
          ],
        ),
        Expanded(
          child: pages[currentIndex],
```

```
          ),
        ],
      ),
    ),
    bottomNavigationBar: BottomNavigation(
      onPressed: (index) {
        if (index != 2) {
          currentIndex = index;
          setState(() {});
        } else {
          showModalBottomSheet(
            context: context,
            builder: (context) => const CreateBottomSheet(),
          );
        }
      },
    ),
  );
 }
}
```

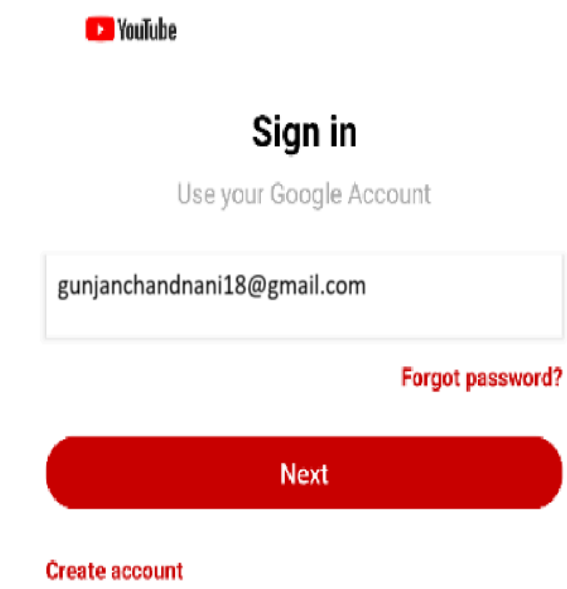
upload_bottom_sheet.dart:

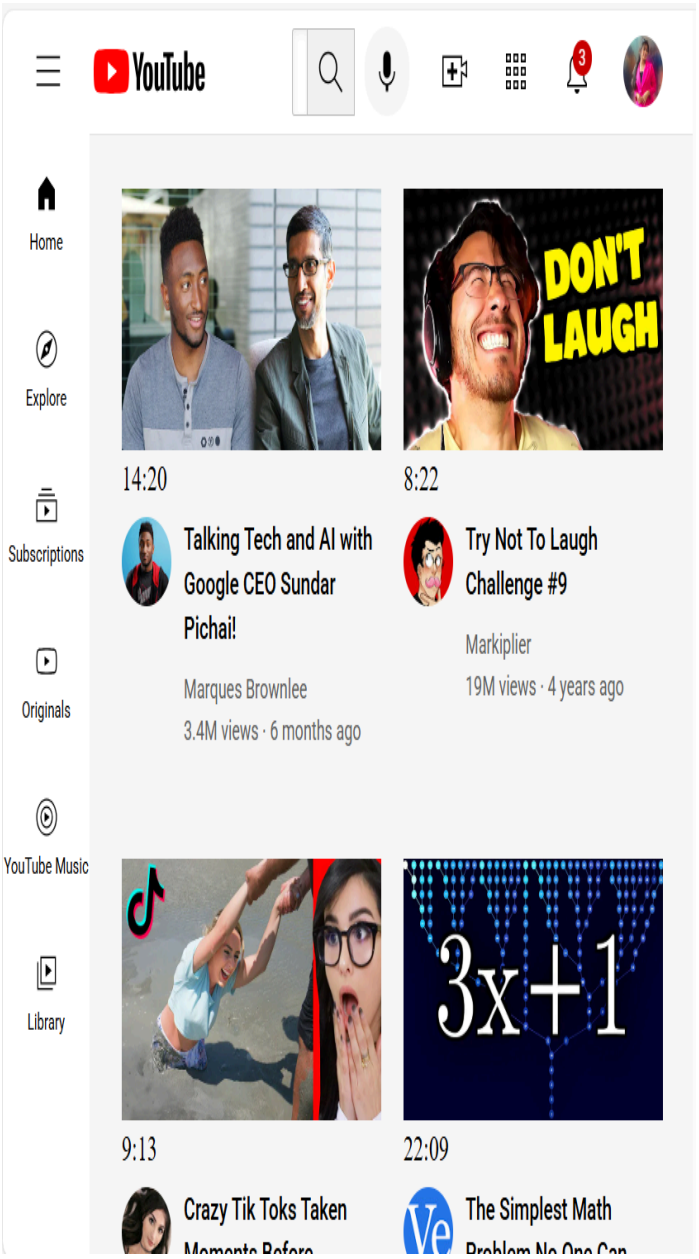
```
// ignore_for_file: use_build_context_synchronously

import 'package:flutter/material.dart';
import 'package:youtube_clone/cores/methods.dart';
import 'package:youtube_clone/cores/widgets/image_item.dart';

class CreateBottomSheet extends StatelessWidget {
 const CreateBottomSheet({super.key});

 @override
 Widget build(BuildContext context) {
   return Container(
     color: const Color(0xffFFFFFF),
     child: Padding(
       padding: const EdgeInsets.only(left: 7, top: 12),
       child: SizedBox(
         height: 270,
         child: Column(
           crossAxisAlignment: CrossAxisAlignment.start,
           children: [
             const Padding(
               padding: EdgeInsets.only(left: 10),
               child: Text(
                 "Create",
                 style: TextStyle(
                   fontSize: 20,
                   fontWeight: FontWeight.w500,
```

```
            ),
          ),
        ),
        const SizedBox(height: 28),
        SizedBox(
          height: 38,
          child: ImageItem(
            itemText: "Create a Short",
            itemClicked: () async {
              await pickShortVideo(context);
            },
            imageName: "short-video.png",
            haveColor: true,
          ),
        ),
        const SizedBox(height: 12),
        SizedBox(
          height: 38,
          child: ImageItem(
            itemText: "Upload a Video",
            itemClicked: () async {
              await pickVideo(context);
            },
            imageName: "upload.png",
            haveColor: true,
          ),
        ),
        const SizedBox(height: 12),
        SizedBox(
          height: 38,
          child: ImageItem(
            itemText: "Go Live",
            itemClicked: () {},
            imageName: "go-live.png",
            haveColor: true,
          ),
        ),
        const SizedBox(height: 12),
        SizedBox(
          height: 38,
          child: ImageItem(
            itemText: "Create a post",
            itemClicked: () {},
            imageName: "create-post.png",
            haveColor: true,
          ),
        ),
      ],
    ),
```

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

OutPut:


YouTube
Sign in
Use your Google Account
gunjanchandnani18@gmail.com
Forgot password?
Next
Create account


YouTube
3
Home
Explore
Subscriptions
Originals
YouTube Music
Library
14:20
Talking Tech and AI with Google CEO Sundar Pichai!
Marques Brownlee
3.4M views · 6 months ago
8:22
DON'T LAUGH
Try Not To Laugh Challenge #9
Markiplier
19M views · 4 years ago
9:13
Crazy Tik Toks Taken
22:09
3x+1
The Simplest Math

Create

Create a Short

Upload a Video

Go Live

Create a post

EXPERIMENT NO: - 06

Name:- Gunjan Chandnani Class:- D15A Roll:No: - 06

AIM: - To connect Flutter UI with Firebase database.

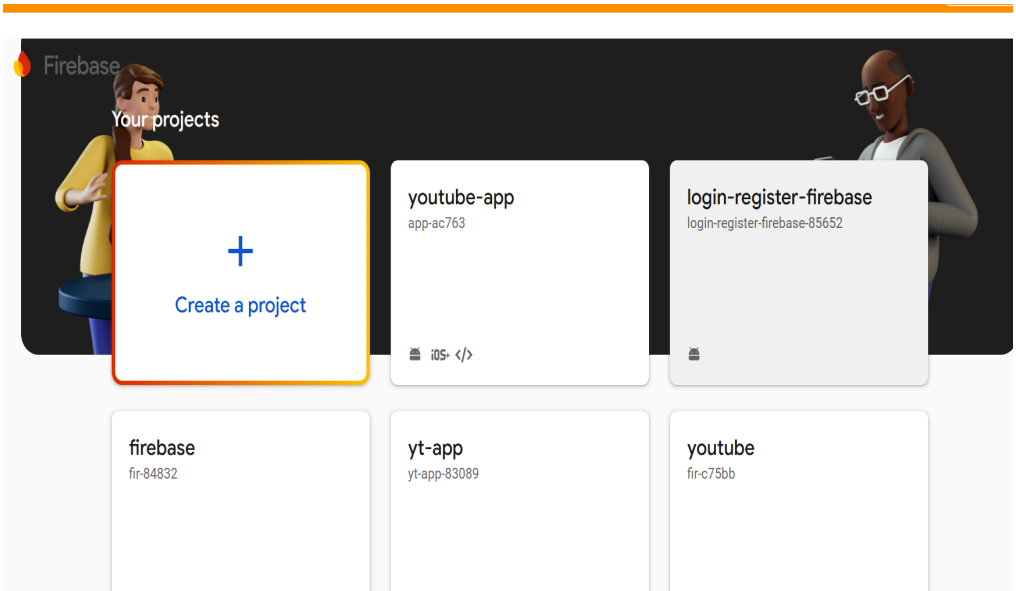
Theory: -
Flutter is an open-source UI toolkit developed by Google for building natively compiled applications for mobile, web, and desktop from a single codebase.

Firebase, a Backend-as-aService (BaaS) platform, provides real-time database, authentication, and cloud storage services, making it a powerful backend solution for Flutter applications.
By integrating Firebase with Flutter, developers can store and retrieve data in real time, authenticate users, and manage cloud-based data efficiently.

This is particularly useful for applications requiring dynamic content updates and user interactions.
➢ **Steps to Connect Flutter UI with Firebase Database**

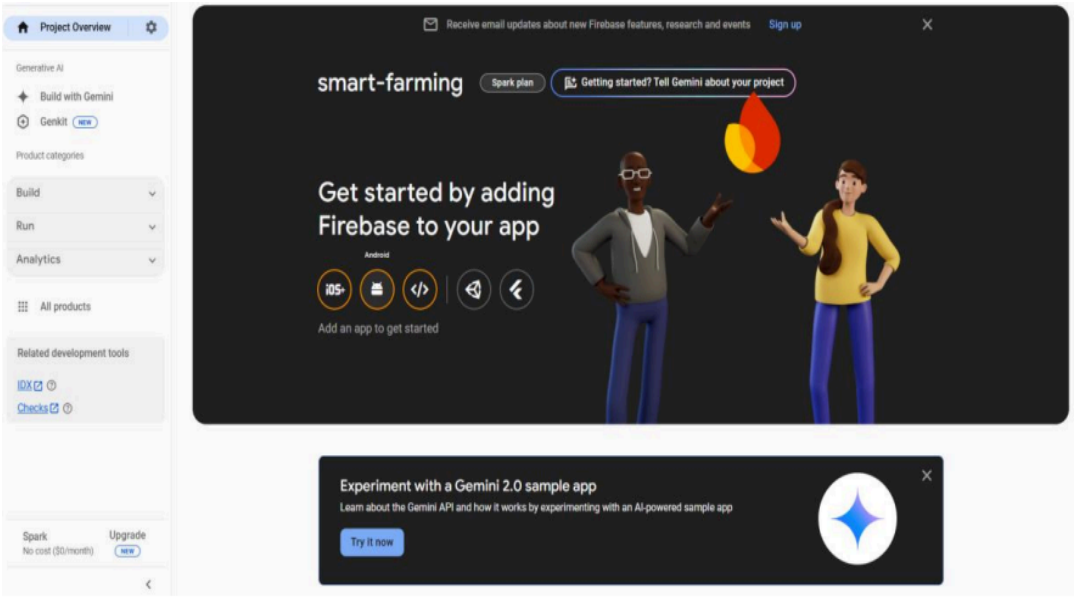
Step 1:

1.1) Go to Firebase Console and Create a Firebase Project

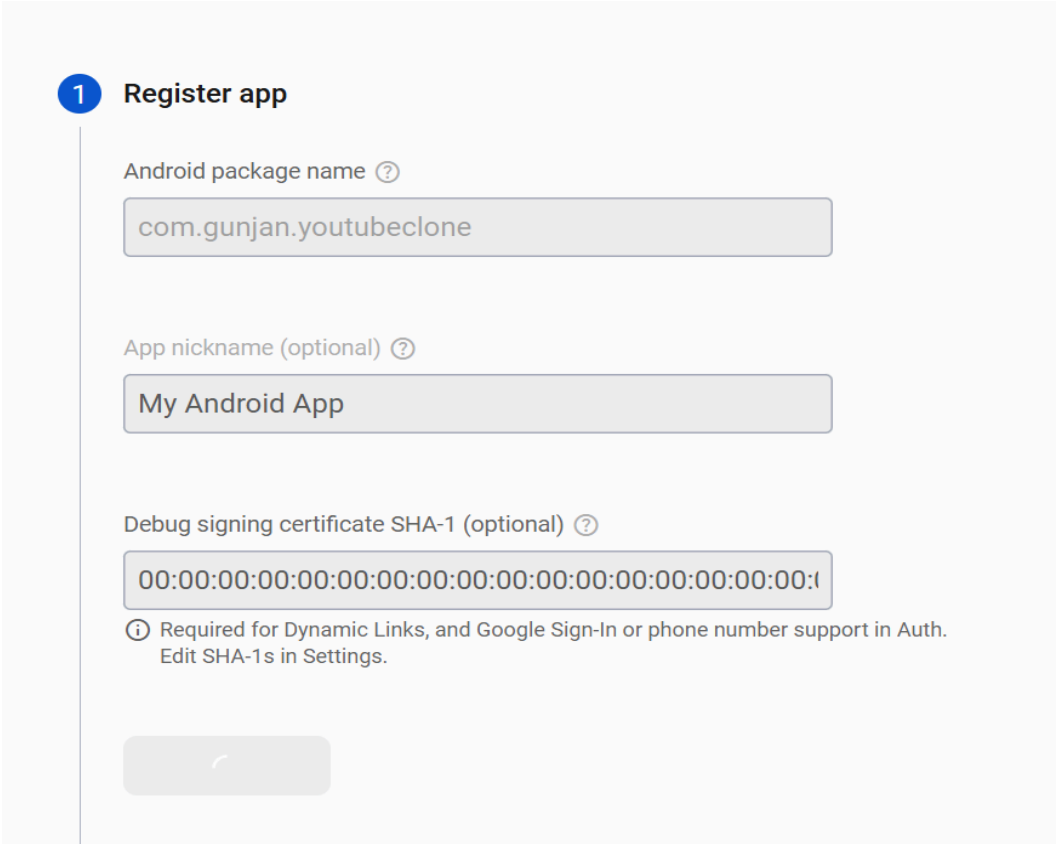


Step 2:- Add Firebase to Your Flutter App

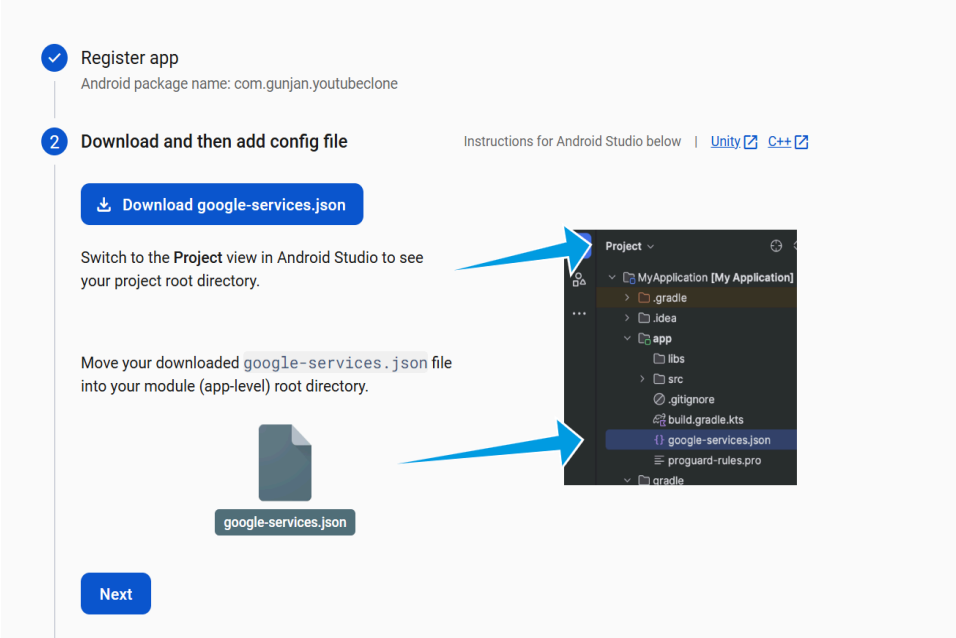
2.1) Click on Android/iOS/Web based on your Flutter application



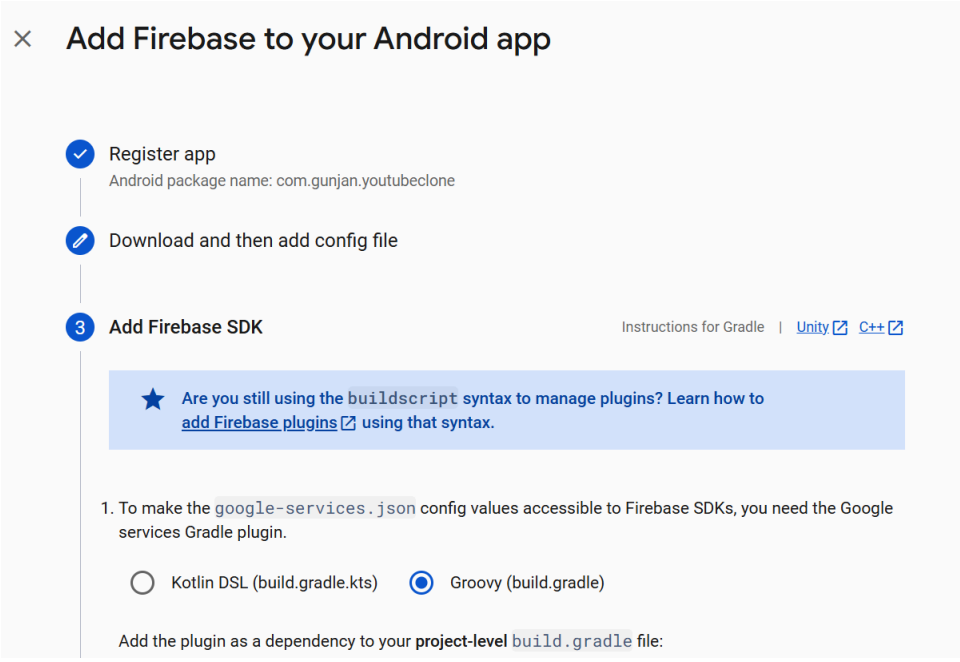
2.2) Register your app with a unique package name (found in android/app/build.gradle for Android)

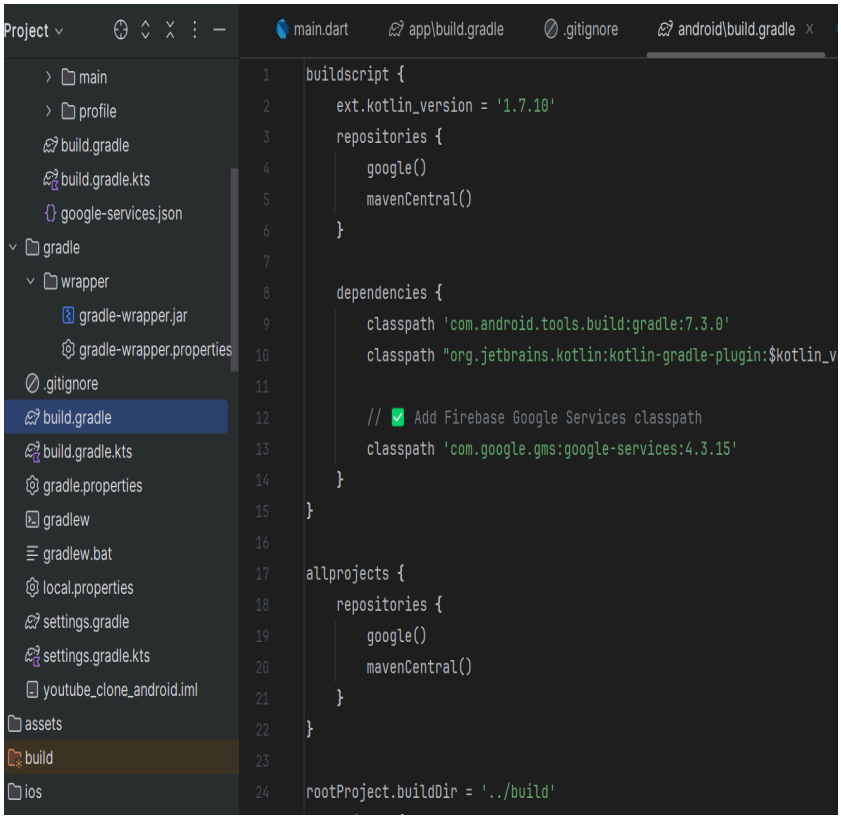


2.3) Download the google-services.json (for Android) & place the JSON file inside android/app/ directory.



2.4) Add Firebase SDK dependencies to android/build.gradle



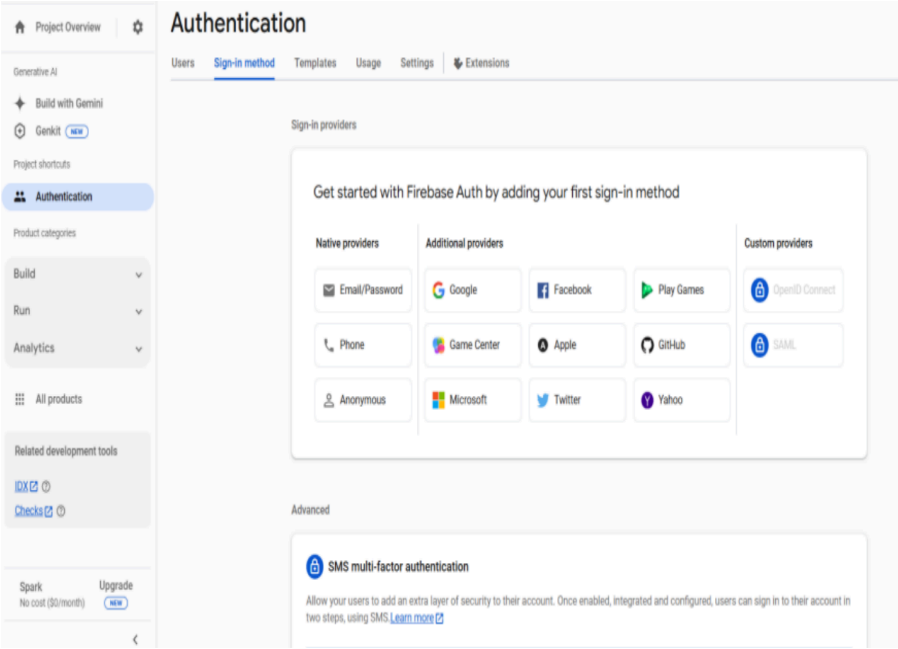


Step 3: - Add Firebase Authentication to Your App

3.1) Add Firebase Authentication Dependencies

```yaml
dependencies:
  flutter:
    sdk: flutter
  firebase_core: ^3.11.0
  firebase_auth: ^5.4.2  # For authentication
  cloud_firestore: ^5.6.3  # For Firestore, if you need it
  firebase_messaging: ^15.2.2
  http: ^0.13.3
  image_picker: ^1.0.4
  tflite_flutter: ^0.11.0
  image: ^3.2.0
  url_launcher: ^6.1.14
```

3.2) Enable Authentication in **Firebase Console**
Go to Firebase Console → **Authentication.**
Click on **Sign-in method** and **enable Email/Password** (or any other method like Google).
Click Save



3.3) Implement Authentication in **Flutter Modify main.dart**

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

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

Step 4: -Configure Firebase Realtime Database

4.1) Go to Firebase Console → Realtime Database.
4.2) Click Create Database → Choose location → Set rules (for development, set read/write to true).
4.3) Click Publish

Code:

login_page.dart

```dart
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:youtube_clone/features/auth/repository/auth_service.dart';

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

 @override
 Widget build(BuildContext context, WidgetRef ref) {
   return Scaffold(
     body: SafeArea(
       child: Center(
         child: Column(
           children: [
             Padding(
               padding: const EdgeInsets.only(
                 top: 20,
                 bottom: 25,
               ),
               child: Image.asset(
                 "assets/images/youtube-signin.jpg",
                 height: 150,
               ),
             ),
             const Text(
               "Welcome To Youtube",
               style: TextStyle(
                 fontSize: 30,
                 fontWeight: FontWeight.bold,
                 color: Colors.blueGrey,
               ),
             ),
             const Spacer(),
             Padding(
               padding: const EdgeInsets.only(bottom: 55),
               child: GestureDetector(
                 onTap: () async {
                   await ref.read(authServiceProvider).signInWithGoogle();
                 },
                 child: Image.asset(
                   "assets/images/signinwithgoogle.png",
                   height: 60,
                 ),
               ),
             ),
           ],
         ),
```

firebase_option.dart:

```dart
// File generated by FlutterFire CLI.
// ignore_for_file: lines_longer_than_80_chars,
avoid_classes_with_only_static_members
import 'package:firebase_core/firebase_core.dart' show FirebaseOptions;
import 'package:flutter/foundation.dart'
    show defaultTargetPlatform, kIsWeb, TargetPlatform;

/// Default [FirebaseOptions] for use with your Firebase apps.
///
/// Example:
/// ```dart
/// import 'firebase_options.dart';
/// // ...
/// await Firebase.initializeApp(
///   options: DefaultFirebaseOptions.currentPlatform,
/// );
/// ```
class DefaultFirebaseOptions {
 static FirebaseOptions get currentPlatform {
   if (kIsWeb) {
     return web;
   }
   switch (defaultTargetPlatform) {
     case TargetPlatform.android:
       return android;
     case TargetPlatform.iOS:
       return ios;
     case TargetPlatform.macOS:
       return macos;
     case TargetPlatform.windows:
       return windows;
     case TargetPlatform.linux:
       throw UnsupportedError(
         'DefaultFirebaseOptions have not been configured for linux - '
         'you can reconfigure this by running the FlutterFire CLI again.',
       );
     default:
       throw UnsupportedError(
         'DefaultFirebaseOptions are not supported for this platform.',
       );
   }
 }

 static const FirebaseOptions android = FirebaseOptions(
   apiKey: 'AIzaSyDc-JOUuH78RpTHXObsa0RDVmia8cRBsjQ',
   appId: '1:188407165111:android:b564cebc78ad2e861edd32',
   messagingSenderId: '188407165111',
   projectId: 'app-ac763',
```

```
    storageBucket: 'app-ac763.firebasestorage.app',
  );

  static const FirebaseOptions ios = FirebaseOptions(
    apiKey: 'AIzaSyB_FXM7S3pqeGvJE0e6DTmAL-Uyfd16agc',
    appId: '1:188407165111:ios:404e3cf1890c2b481edd32',
    messagingSenderId: '188407165111',
    projectId: 'app-ac763',
    storageBucket: 'app-ac763.firebasestorage.app',
    iosBundleId: 'com.example.youtubeClone',
  );

  static const FirebaseOptions web = FirebaseOptions(
    apiKey: 'AIzaSyBSVAuYp50tr_S8-O60E7o5dKNKW-1Ejew',
    appId: '1:188407165111:web:4daa739182e168b11edd32',
    messagingSenderId: '188407165111',
    projectId: 'app-ac763',
    authDomain: 'app-ac763.firebaseapp.com',
    storageBucket: 'app-ac763.firebasestorage.app',
    measurementId: 'G-LVMQ4SZ270',
  );

  static const FirebaseOptions macos = FirebaseOptions(
    apiKey: 'AIzaSyB_FXM7S3pqeGvJE0e6DTmAL-Uyfd16agc',
    appId: '1:188407165111:ios:404e3cf1890c2b481edd32',
    messagingSenderId: '188407165111',
    projectId: 'app-ac763',
    storageBucket: 'app-ac763.firebasestorage.app',
    iosBundleId: 'com.example.youtubeClone',
  );

  static const FirebaseOptions windows = FirebaseOptions(
    apiKey: 'AIzaSyBSVAuYp50tr_S8-O60E7o5dKNKW-1Ejew',
    appId: '1:188407165111:web:0e4eb752f4c3873e1edd32',
    messagingSenderId: '188407165111',
    projectId: 'app-ac763',
    authDomain: 'app-ac763.firebaseapp.com',
    storageBucket: 'app-ac763.firebasestorage.app',
    measurementId: 'G-86VN7VQZJX',
  );

}
```

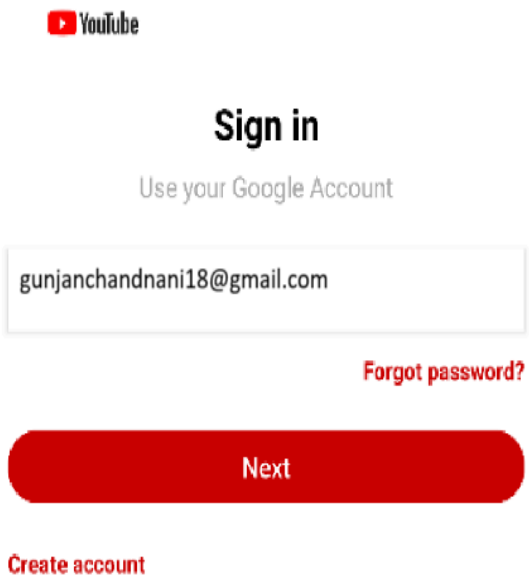
home_page.dart:

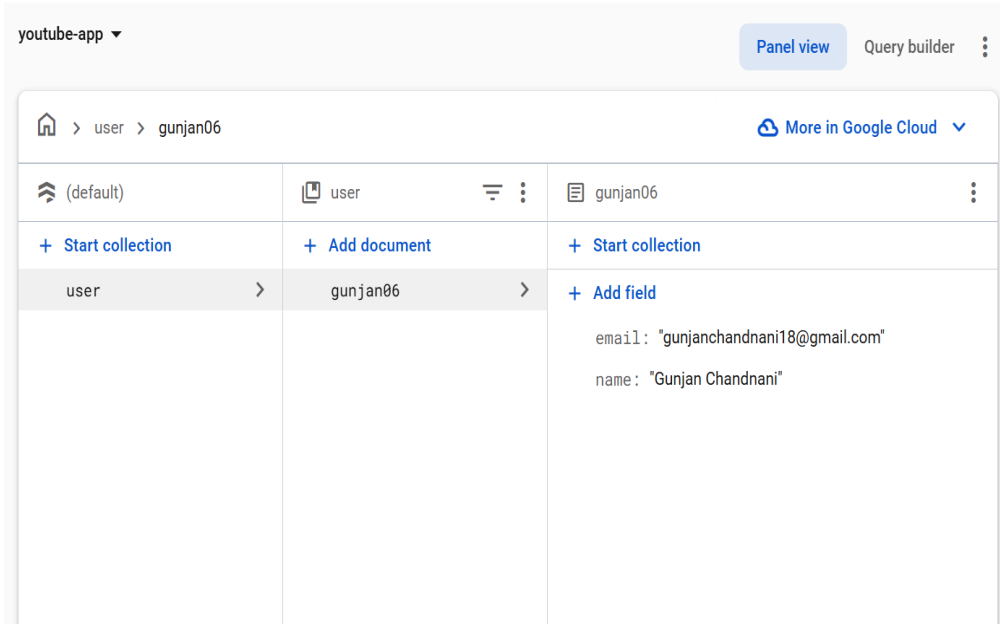
```dart
// ignore_for_file: invalid_use_of_visible_for_testing_member

import 'package:cached_network_image/cached_network_image.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:path/path.dart';
import 'package:youtube_clone/cores/screens/error_page.dart';
import 'package:youtube_clone/cores/screens/loader.dart';
import 'package:youtube_clone/cores/widgets/image_button.dart';
import 'package:youtube_clone/features/account/account_page.dart';
import 'package:youtube_clone/features/auth/provider/user_provider.dart';
import 'package:youtube_clone/features/content/bottom_navigation.dart';
import 'package:youtube_clone/features/upload/upload_bottom_sheet.dart';
import 'package:youtube_clone/pages_list.dart';

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

 @override
 State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
 int currentIndex = 0;
 @override
 Widget build(BuildContext context) {
   return Scaffold(
     backgroundColor: const Color(0xffFFFFFF),
     body: SafeArea(
       child: Column(
         children: [
           Row(
             children: [
               Image.asset(
                 "assets/images/youtube.jpg",
                 height: 36,
               ),
               const SizedBox(width: 4),
               const Spacer(),
               Padding(
                 padding: const EdgeInsets.only(right: 12),
                 child: SizedBox(
                   height: 42,
                   child: ImageButton(
                     image: "cast.png",
                     onPressed: () {},
                     haveColor: false,
                   ),
```

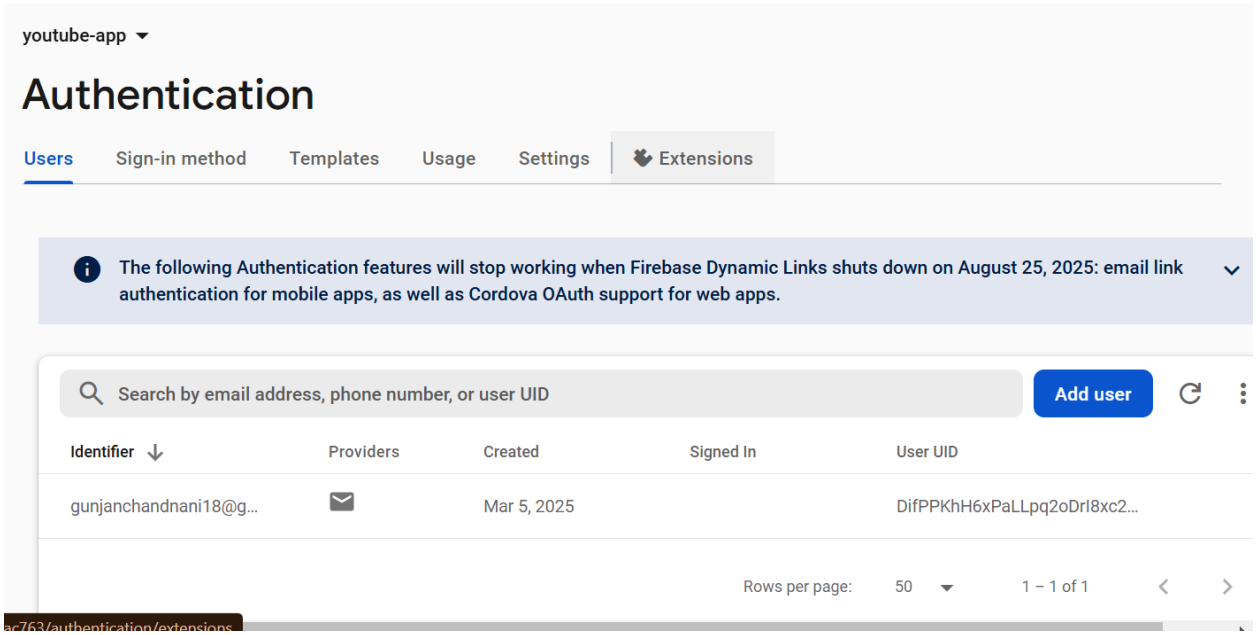
```
            ),
          ),
          SizedBox(
            height: 38,
            child: ImageButton(
              image: "notification.png",
              onPressed: () {},
              haveColor: false,
            ),
          ),
          Padding(
            padding: const EdgeInsets.only(left: 12, right: 15),
            child: SizedBox(
              height: 41.5,
              child: ImageButton(
                image: "search.png",
                onPressed: () {},
                haveColor: false,
              ),
            ),
          ),
          Consumer(
            builder: (context, ref, child) {
              return ref.watch(currentUserProvider).when(
                    data: (currentUser) => Padding(
                      padding: const EdgeInsets.only(right: 12),
                      child: GestureDetector(
                        onTap: () {
                          Navigator.push(
                            context,
                            MaterialPageRoute(
                              builder: (context) => AccountPage(
                                user: currentUser,
                              ),
                            ),
                          );
                        },
                        child: CircleAvatar(
                          radius: 14,
                          backgroundColor: Colors.grey,
                          backgroundImage: CachedNetworkImageProvider(
                            currentUser.profilePic,
                          ),
                        ),
                      ),
                    ),
                    error: (error, stackTrace) => const ErrorPage(),
                    loading: () => const Loader(),
                  );
```

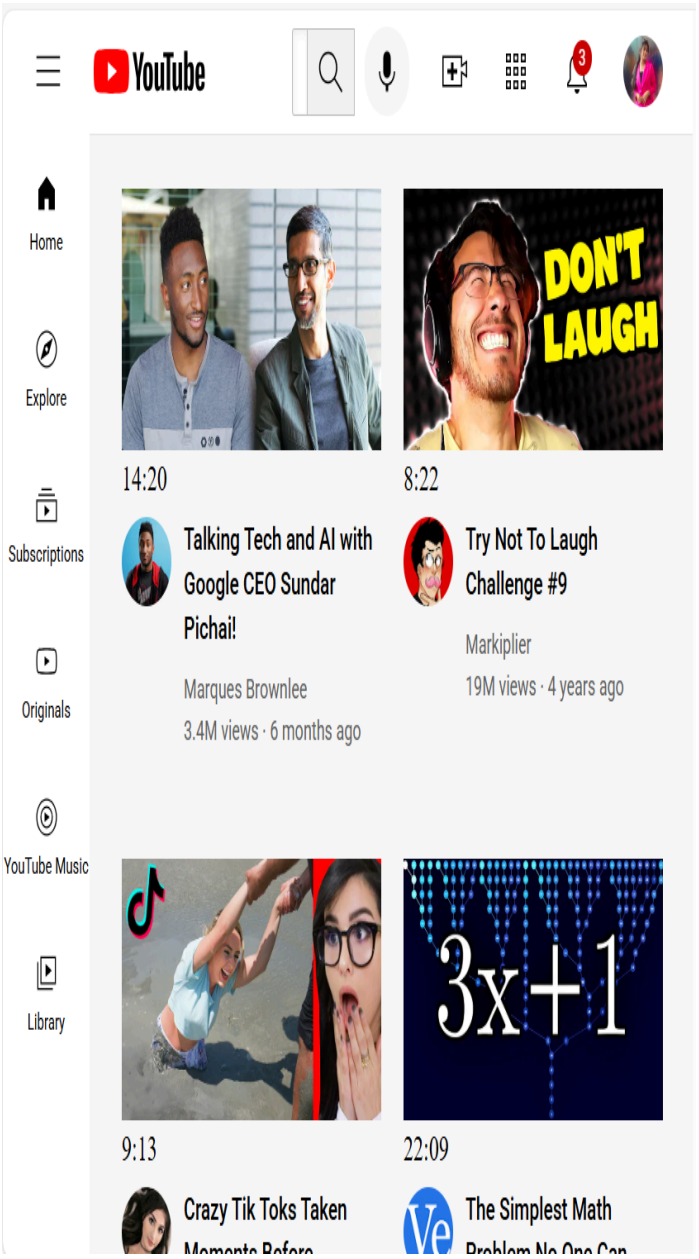
```
                },
              ),
            ],
          ),
          Expanded(
            child: pages[currentIndex],
          ),
        ],
      ),
    ),
    bottomNavigationBar: BottomNavigation(
      onPressed: (index) {
        if (index != 2) {
          currentIndex = index;
          setState(() {});
        } else {
          showModalBottomSheet(
            context: context,
            builder: (context) => const CreateBottomSheet(),
          );
        }
      },
    ),
  );
 }
}
```

OutPut:


YouTube
Sign in
Use your Google Account
gunjanchandnani18@gmail.com
Forgot password?
Next
Create account


youtube-app
Panel view
Query builder
user
gunjan06
More in Google Cloud
(default)
user
gunjan06
Start collection
Add document
Start collection
user
gunjan06
Add field
email: "gunjanchandnani18@gmail.com"
name: "Gunjan Chandnani"


youtube-app
Authentication
Users
Sign-in method
Templates
Usage
Settings
Extensions
The following Authentication features will stop working when Firebase Dynamic Links shuts down on August 25, 2025: email link authentication for mobile apps, as well as Cordova OAuth support for web apps.
Search by email address, phone number, or user UID
Add user
Identifier
Providers
Created
Signed In
User UID
gunjanchandnani18@g...
Mar 5, 2025
DifPPKhH6xPaLLpq2oDrI8xc2...
Rows per page:
50
1 – 1 of 1


YouTube
3
Home
Explore
Subscriptions
Originals
YouTube Music
Library
14:20
Talking Tech and AI with Google CEO Sundar Pichai!
Marques Brownlee
3.4M views · 6 months ago
8:22
DON'T LAUGH
Try Not To Laugh Challenge #9
Markiplier
19M views · 4 years ago
9:13
Crazy Tik Toks Taken
22:09
3x+1
The Simplest Math

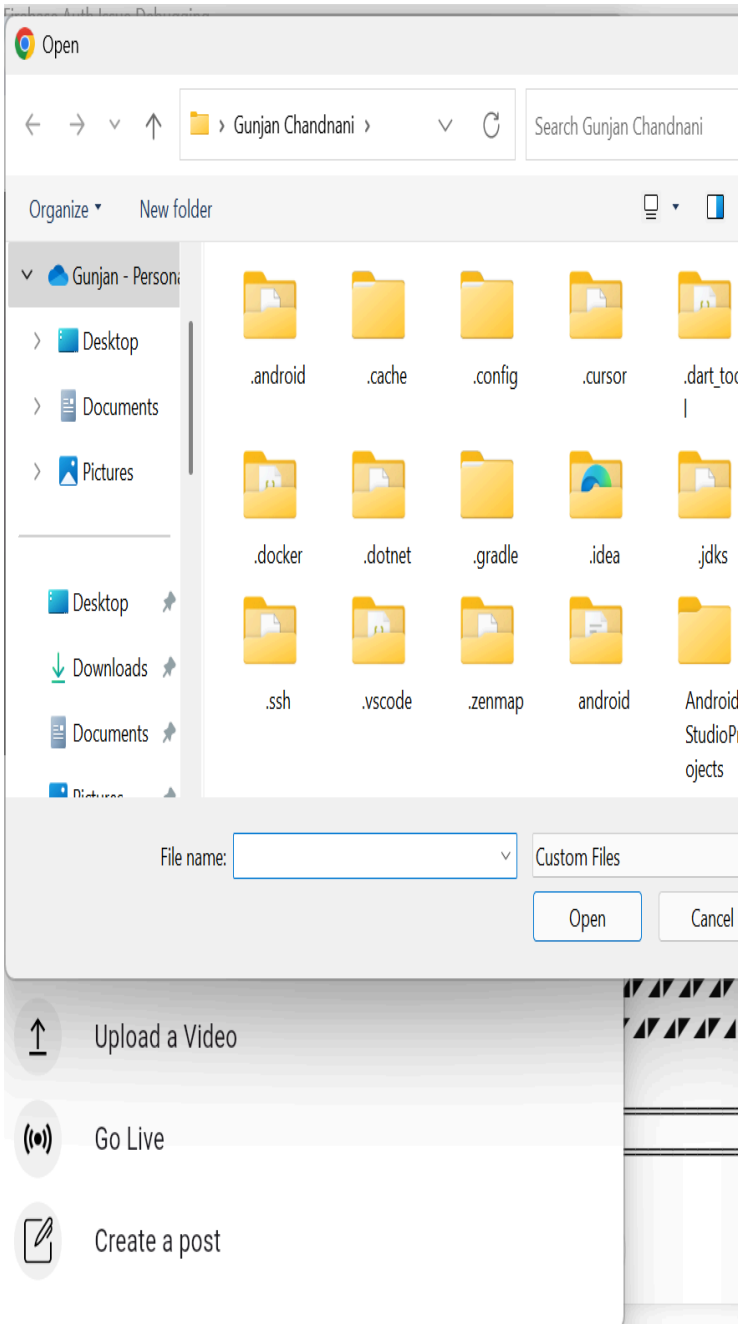
Create

Create a Short

Upload a Video

Go Live

Create a post


Open
Gunjan Chandnani
Search Gunjan Chandnani
Organize
New folder
Gunjan - Persona
Desktop
Documents
Pictures
Desktop
Downloads
Documents
.android
.cache
.config
.cursor
.dart_tool
.docker
.dotnet
.gradle
.idea
.jdks
.ssh
.vscode
.zenmap
android
AndroidStudioProjects
File name:
Custom Files
Open
Cancel
Upload a Video
Go Live
Create a post

Experiment No. 7

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

Theory:

- **Regular Web App**

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

- **Progressive Web App**

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

- **Difference between PWAs vs. Regular Web Apps:**

  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.

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:**

1. Progressive — They work for every user, regardless of the browser chosen because they are built at the base with progressive improvement principles.
2. Responsive — They adapt to the various screen sizes: desktop, mobile, tablet, or dimensions that can later become available.
3. Updated — Information is always up-to-date thanks to the data update process offered by service workers.
4. Secure — Exposed over HTTPS protocol to prevent the connection from displaying information or altering the contents.
5. Searchable — They are identified as “applications” and are indexed by search engines.
6. Reactivable — Make it easy to reactivate the application thanks to capabilities such as web notifications.
7. 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.
8. Linkable — Easily shared via URL without complex installations.
9. 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.

Code:

1. **Manifest.json**

```
{
  "name": "Shopping App",
  "short_name": "MyApp",
  "description": "An online shopping platform with a wide range of
products.",
  "start_url": "/",
  "scope": "/",
  "display": "standalone",
  "background_color": "#ffffff",
  "theme_color": "#ff6600",
  "icons": [
    {
      "src": "assets/images/banner-1.png",
      "sizes": "192x192",
      "type": "image/png",
      "purpose": "any maskable"
    },
    {
      "src": "assets/images/banner-2.png",
      "sizes": "512x512",
      "type": "image/png",
      "purpose": "any maskable"
    }
  ],
  "shortcuts": [
    {
      "name": "Shop Now",
      "short_name": "Shop",
      "description": "Go directly to shopping",
      "url": "/index.html",
      "icons": [
        {
          "src": "assets/images/shortcut-icon.png",
          "sizes": "96x96",
          "type": "image/png"
        }
      ]
    }
  ]
}
```

2. Service-worker.js

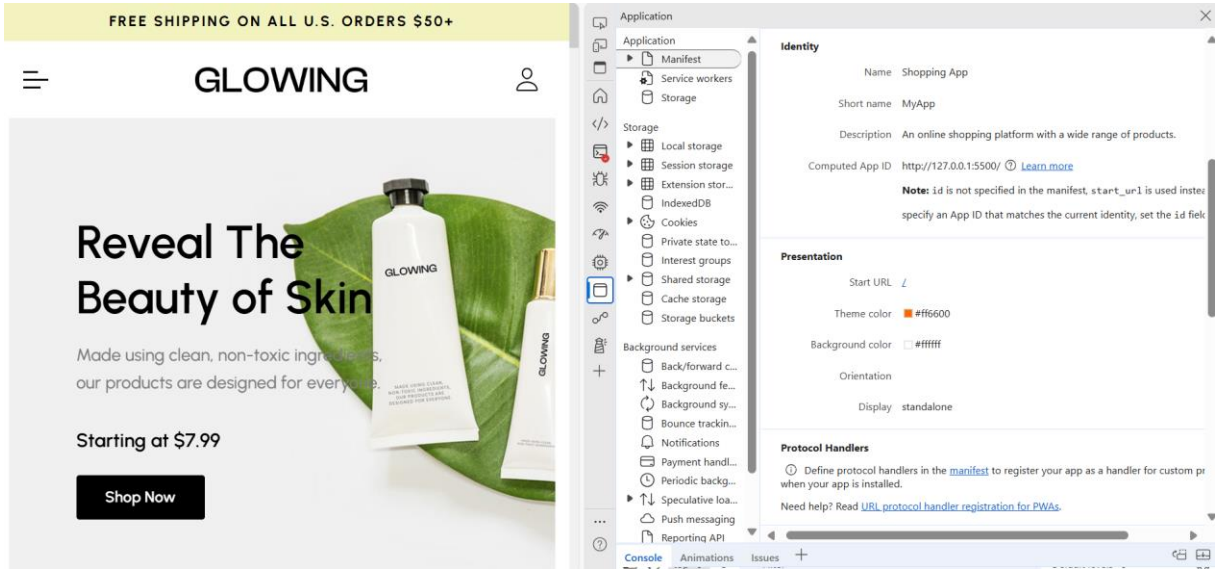
```
self.addEventListener("install", (event) => {
    console.log("Service Worker Installed");
  });

  self.addEventListener("fetch", (event) => {
    event.respondWith(fetch(event.request));
  });
```

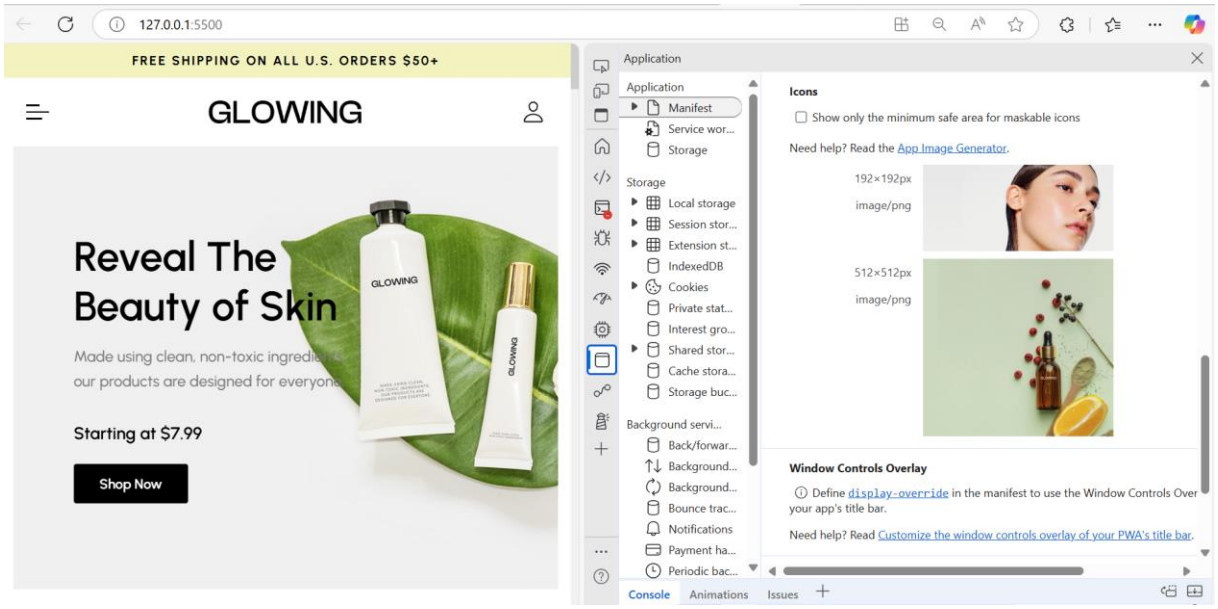
3. Script.js file:

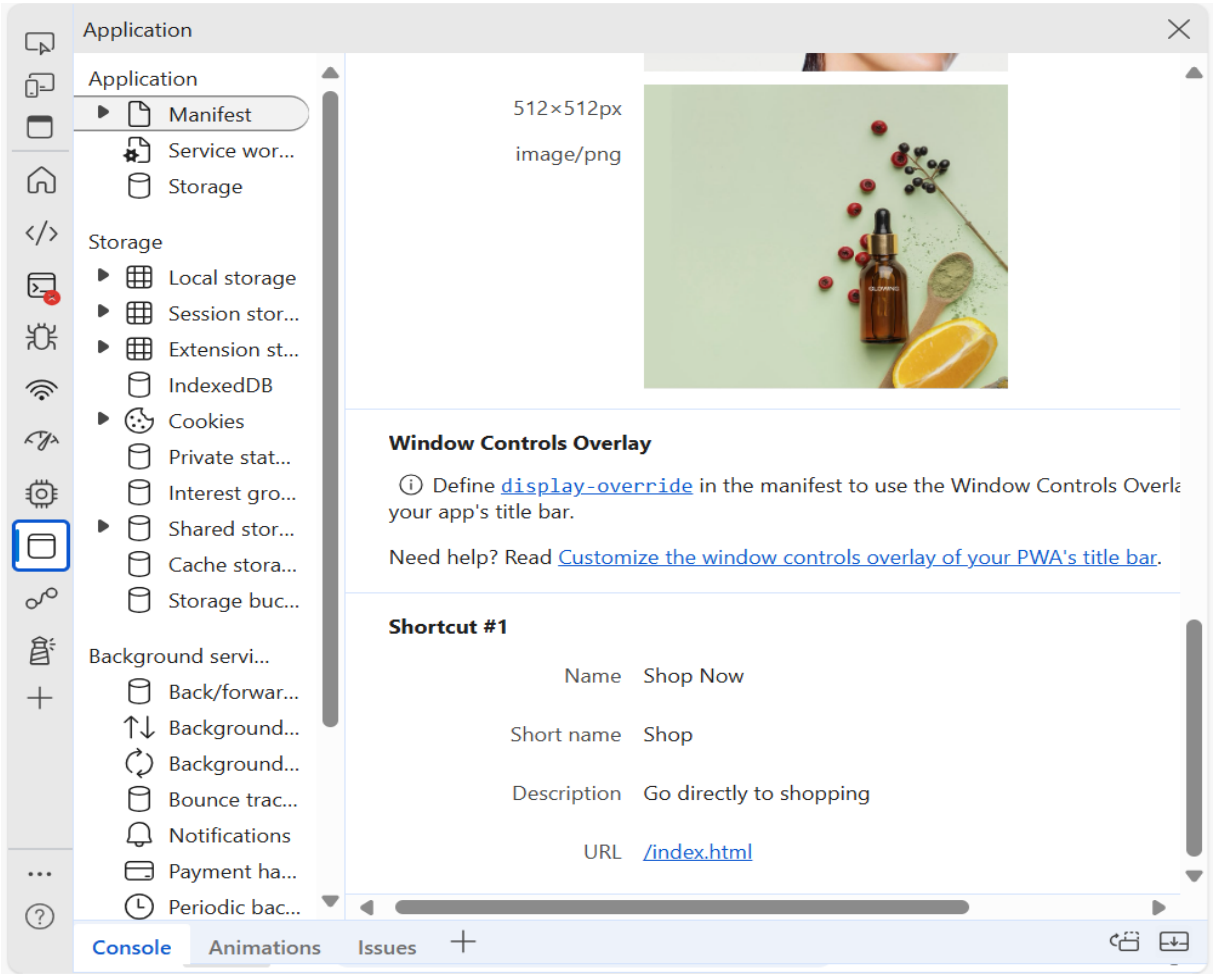
```
'use strict';

/**
 * add event on element
 */

const addEventOnElem = function (elem, type, callback) {
  if (elem.length > 1) {
    for (let i = 0; i < elem.length; i++) {
      elem[i].addEventListener(type, callback);
    }
  } else {
    elem.addEventListener(type, callback);
  }
}


/**
 * navbar toggle
 */

const navTogglers = document.querySelectorAll("[data-nav-toggler]");
const navbar = document.querySelector("[data-navbar]");
const navbarLinks = document.querySelectorAll("[data-nav-link]");
const overlay = document.querySelector("[data-overlay]");

const toggleNavbar = function () {
  navbar.classList.toggle("active");
  overlay.classList.toggle("active");
}

addEventOnElem(navTogglers, "click", toggleNavbar);

const closeNavbar = function () {
  navbar.classList.remove("active");
  overlay.classList.remove("active");
}

addEventOnElem(navbarLinks, "click", closeNavbar);
```

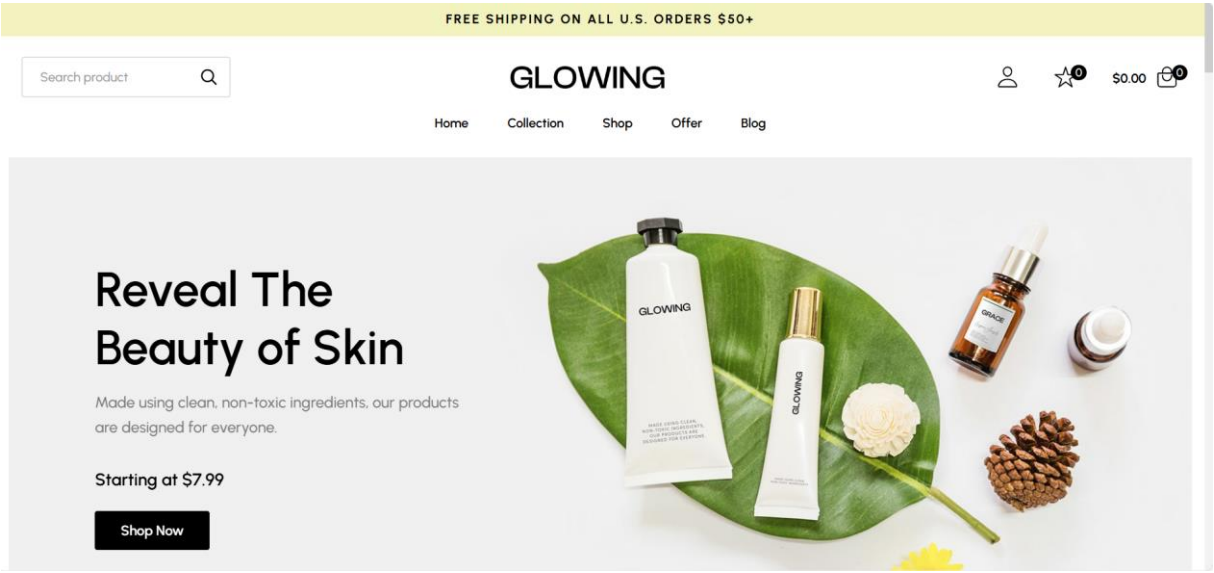
```
/**
 * header sticky & back top btn active
 */

const header = document.querySelector("[data-header]");
const backTopBtn = document.querySelector("[data-back-top-btn]");

const headerActive = function () {
  if (window.scrollY > 150) {
    header.classList.add("active");
    backTopBtn.classList.add("active");
  } else {
    header.classList.remove("active");
    backTopBtn.classList.remove("active");
  }
}

addEventOnElem(window, "scroll", headerActive);

let lastScrolledPos = 0;

const headerSticky = function () {
  if (lastScrolledPos >= window.scrollY) {
    header.classList.remove("header-hide");
  } else {
    header.classList.add("header-hide");
  }

  lastScrolledPos = window.scrollY;
}

addEventOnElem(window, "scroll", headerSticky);


/**
 * scroll reveal effect
 */

const sections = document.querySelectorAll("[data-section]");

const scrollReveal = function () {
  for (let i = 0; i < sections.length; i++) {
    if (sections[i].getBoundingClientRect().top < window.innerHeight / 2) {
      sections[i].classList.add("active");
    }
  }
}

scrollReveal();

addEventOnElem(window, "scroll", scrollReveal);
```

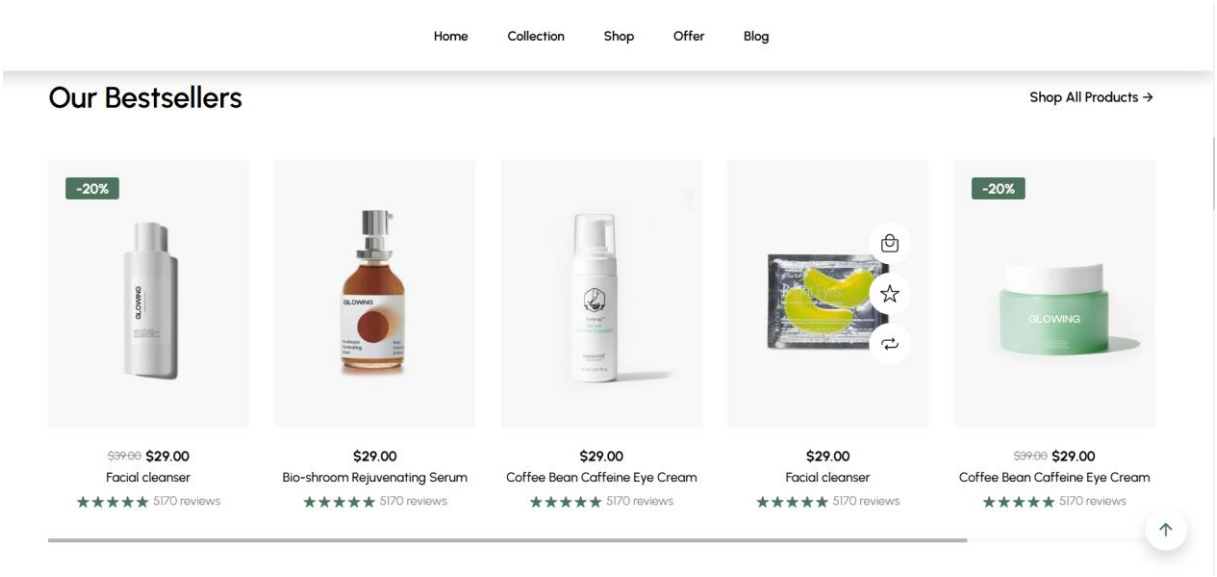
Output:


FREE SHIPPING ON ALL U.S. ORDERS $50+
GLOWING
Reveal The Beauty of Skin
Made using clean, non-toxic ingredients,
our products are designed for everyone.
Starting at $7.99
Shop Now
Application
Manifest
Service workers
Storage
Identity
Name Shopping App
Short name MyApp
Description An online shopping platform with a wide range of products.
Presentation
Start URL /
Theme color #ff6600
Background color #ffffff
Orientation
Display standalone
Protocol Handlers
Console
Animations
Issues


127.0.0.1:5500
FREE SHIPPING ON ALL U.S. ORDERS $50+
GLOWING
Reveal The Beauty of Skin
Starting at $7.99
Shop Now
Application
Manifest
Storage
Icons
Show only the minimum safe area for maskable icons
Need help? Read the App Image Generator.
192×192px
image/png
512×512px
image/png
Window Controls Overlay
Console
Animations
Issues


Application
Application
Manifest
Service wor...
Storage
Storage
Local storage
Session stor...
Extension st...
IndexedDB
Cookies
Private stat...
Interest gro...
Shared stor...
Cache stora...
Storage buc...
Background servi...
Back/forwar...
Background...
Background...
Bounce trac...
Notifications
Payment ha...
Periodic bac...
512×512px
image/png
Window Controls Overlay
Define display-override in the manifest to use the Window Controls Overla your app's title bar.
Need help? Read Customize the window controls overlay of your PWA's title bar.
Shortcut #1
Name Shop Now
Short name Shop
Description Go directly to shopping
URL /index.html
Console
Animations
Issues


FREE SHIPPING ON ALL U.S. ORDERS $50+
Search product
GLOWING
$0.00
Home
Collection
Shop
Offer
Blog
Reveal The Beauty of Skin
Made using clean, non-toxic ingredients, our products are designed for everyone.
Starting at $7.99
Shop Now


Home
Collection
Shop
Offer
Blog
Our Bestsellers
Shop All Products →
-20%
GLOWING
$39.00 $29.00
Facial cleanser
★★★★★ 5170 reviews
$29.00
Bio-shroom Rejuvenating Serum
★★★★★ 5170 reviews
$29.00
Coffee Bean Caffeine Eye Cream
★★★★★ 5170 reviews
$29.00
Facial cleanser
★★★★★ 5170 reviews
-20%
GLOWING
$39.00 $29.00
Coffee Bean Caffeine Eye Cream
★★★★★ 5170 reviews

Vanshika Ambwani (02), Gunjan Chandnani (06), Akruti Dabas (11)

Experiment number
08 MAD and PWA ab

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

Theory:

Service Worker

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

Things to note about Service Worker:

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

What can we do with Service Workers?

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

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

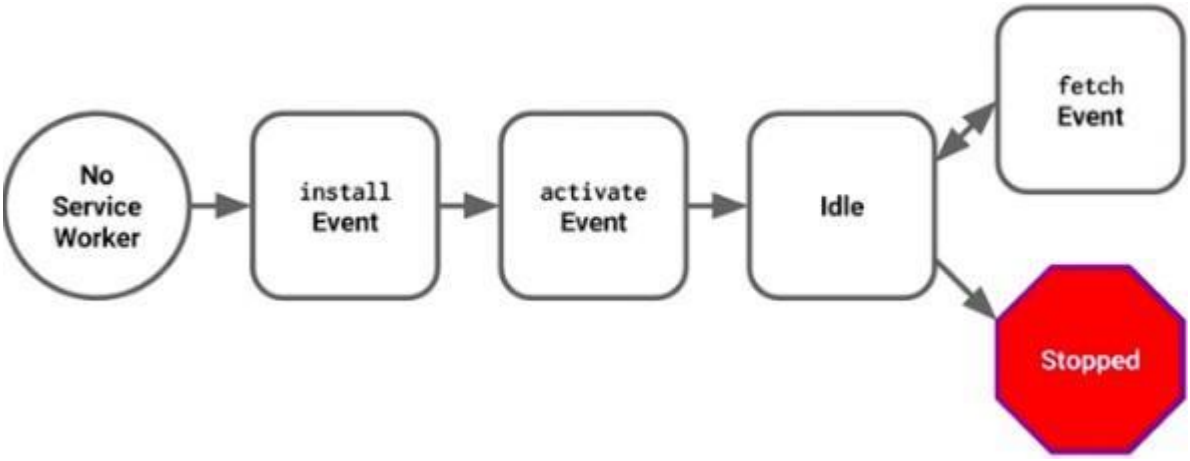
- You can manage **Push Notifications**
  You can manage push notifications with Service Worker and show any information message to the user.
- You can **Continue**
  Although Internet connection is broken, you can start any process with Background Sync of Service Worker.

What can’t we do with Service Workers?

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

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

Service Worker Cycle



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

- Registration
- Installation
- Activation

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

main.js

```
if ('serviceWorker' in navigator) {
  navigator.serviceWorker.register('/service-worker.js')
  .then(function(registration) {
    console.log('Registration successful, scope is:', registration.scope);
  })
  .catch(function(error) {
    console.log('Service worker registration failed, error:', error);
  });
}
```

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

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

You can also set an arbitrary scope by passing in an additional parameter when registering. For example:

main.js

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

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

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

main.js

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

Installation

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

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

service-worker.js

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

Activation

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

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

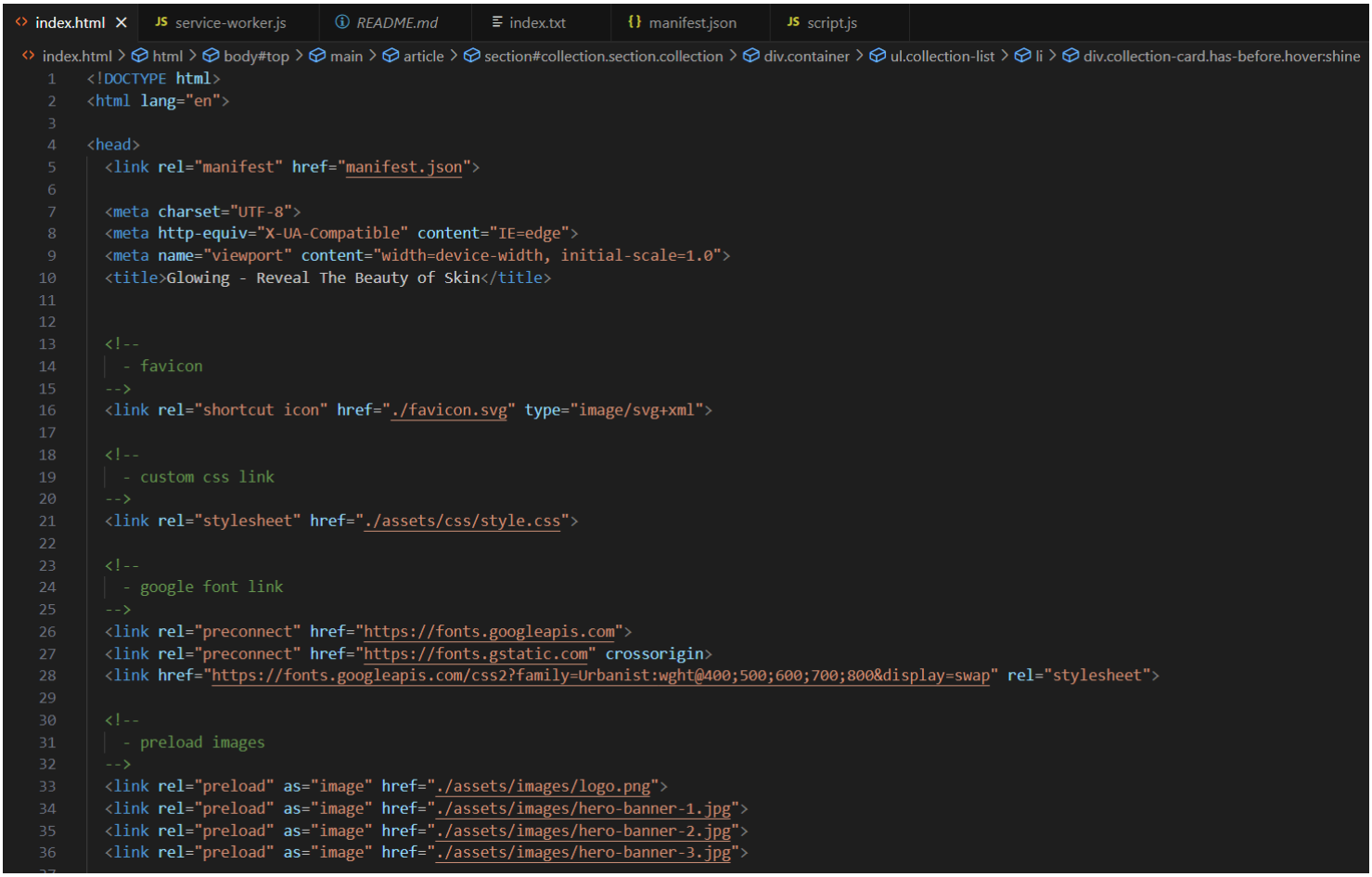
service-worker.js

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

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

Code

index.html



```
<!DOCTYPE html>
<html lang="en">

<head>
  <link rel="manifest" href="manifest.json">

  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Glowing - Reveal The Beauty of Skin</title>


  <!--
    - favicon
  -->
  <link rel="shortcut icon" href="./favicon.svg" type="image/svg+xml">

  <!--
    - custom css link
  -->
  <link rel="stylesheet" href="./assets/css/style.css">

  <!--
    - google font link
  -->
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link href="https://fonts.googleapis.com/css2?family=Urbanist:wght@400;500;600;700;800&display=swap" rel="stylesheet">

  <!--
    - preload images
  -->
  <link rel="preload" as="image" href="./assets/images/logo.png">
  <link rel="preload" as="image" href="./assets/images/hero-banner-1.jpg">
  <link rel="preload" as="image" href="./assets/images/hero-banner-2.jpg">
  <link rel="preload" as="image" href="./assets/images/hero-banner-3.jpg">
```

service-worker.js

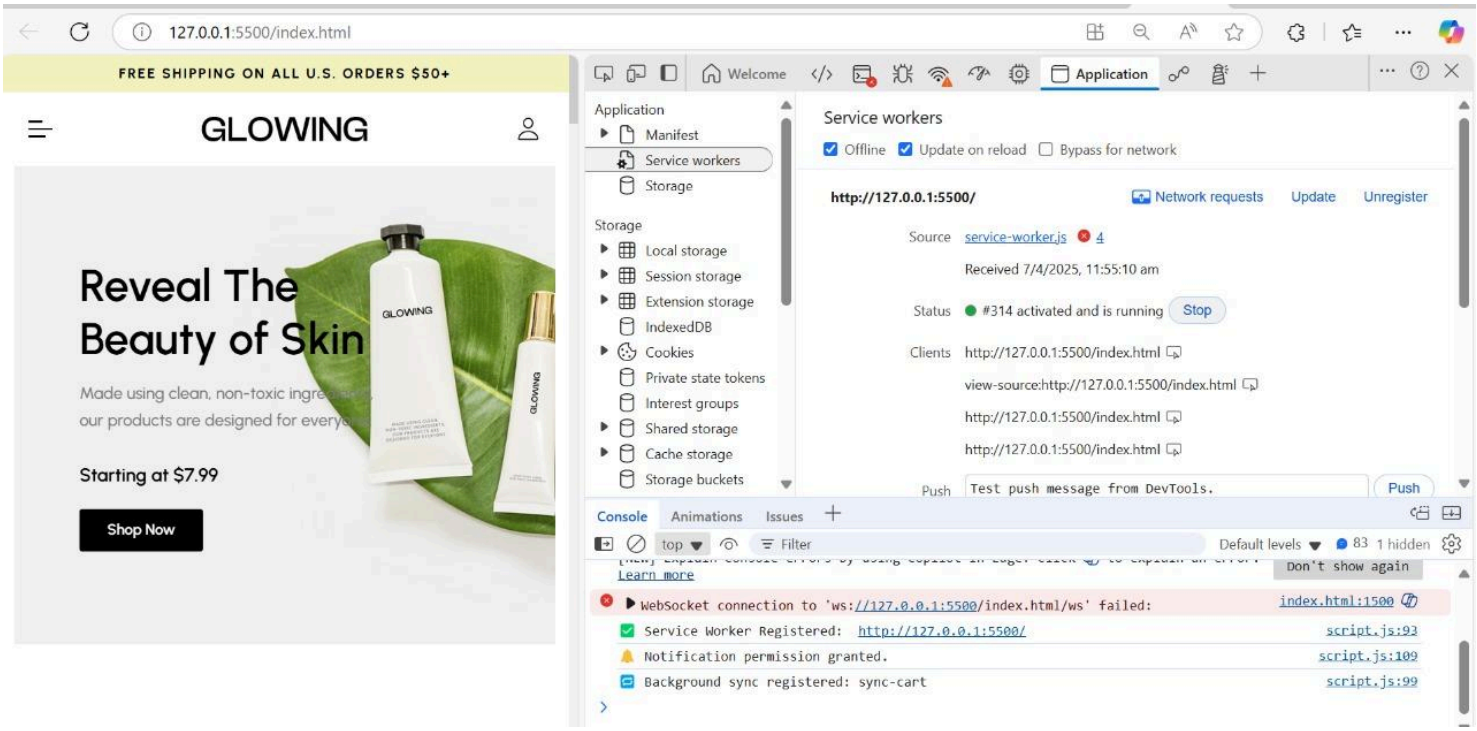
```
self.addEventListener("install", (event) => {
    console.log("Service Worker Installed");
  });

  self.addEventListener("fetch", (event) => {
    event.respondWith(fetch(event.request));
  });
```

Vanshika Ambwani (02), Gunjan Chandnani (06), Akruti Dabas (11)

manifest.json

```
{
  "name": "Shopping App",
  "short_name": "MyApp",
  "description": "An online shopping platform with a wide range of products.",
  "start_url": "/",
  "scope": "/",
  "display": "standalone",
  "background_color": "#ffffff",
  "theme_color": "#ff6600",
  "icons": [
    {
      "src": "assets/images/banner-1.png",
      "sizes": "192x192",
      "type": "image/png",
      "purpose": "any maskable"
    },
    {
      "src": "assets/images/banner-2.png",
      "sizes": "512x512",
      "type": "image/png",
      "purpose": "any maskable"
    }
  ],
  "shortcuts": [
    {
      "name": "Shop Now",
      "short_name": "Shop",
      "description": "Go directly to shopping",
      "url": "/index.html",
      "icons": [
        {
          "src": "assets/images/shortcut-icon.png",
          "sizes": "96x96",
          "type": "image/png"
        }
      ]
    }
  ]
}
```

Vanshika Ambwani (02), Gunjan Chandnani (06), Akruti Dabas (11)



EXPERIMENT NO. 9

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

Theory:

Service Worker

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

Things to note about Service Worker:

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

Fetch Event

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

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

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

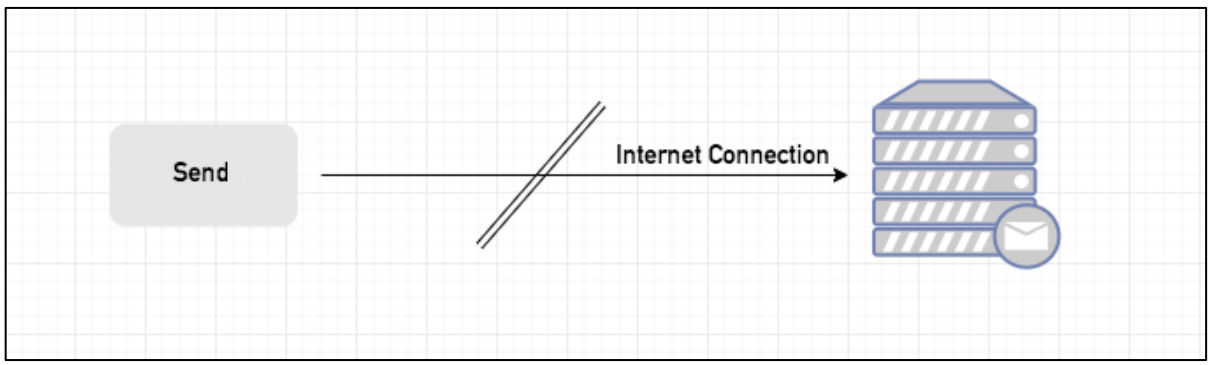
```
self.addEventListener("fetch", function (event) {
    const req = event.request;
    const url = new URL(req.url);

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

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

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

Sync Event

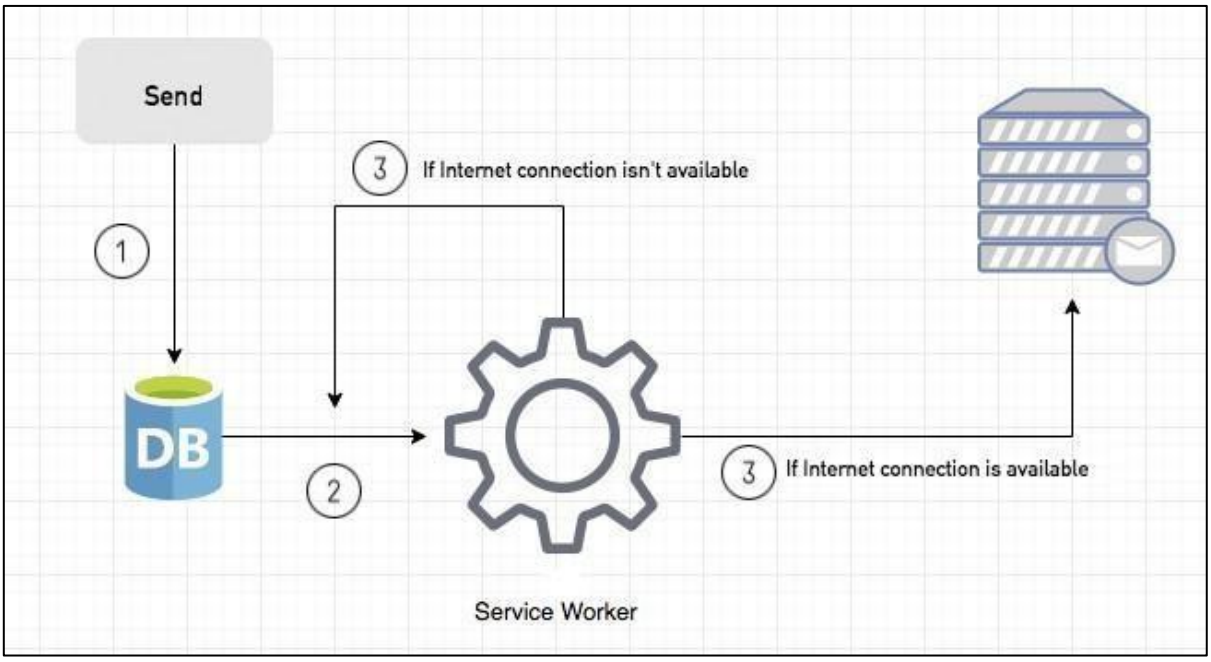
Background Sync is a Web API that is used to delay a process until the Internet connection is stable. We can adapt this definition to the real world; there is an e-mail client application that works on the browser and we want to send an email with this tool. Internet connection is broken while we are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

Here is a job for the Background Sync.

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



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



1. When we click the “send” button, email content will be saved to IndexedDB.
2. Background Sync registration.
3. **If the Internet connection is available**, all email content will be read and sent to Mail Server.

If the Internet connection is unavailable, the service worker waits until the connection is available even though the window is closed. When it is available, email content will be sent to Mail Server.

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

Event Listener for Background Sync Registration

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

Event Listener for sw.js

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

Push Event

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

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

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

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

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

Code:

1. Push Notification code:

```
self.addEventListener("install", (event) => {
  console.log("✅ Service Worker Installed");
});

self.addEventListener("fetch", (event) => {
  event.respondWith(fetch(event.request));
});

// ✅ Background Sync Event
self.addEventListener("sync", (event) => {
  console.log("🔁 Sync Event Triggered:", event.tag);
  if (event.tag === "sync-cart") {
    event.waitUntil(syncCartData());
  }
});

async function syncCartData() {
  try {
    console.log("🛒 Syncing cart data...");
    // Add your actual sync logic here
  } catch (err) {
    console.error("❌ Failed to sync cart data:", err);
  }
}

self.addEventListener("message", (event) => {
  if (event.data && event.data.type === "trigger-push") {
    const { title, body } = event.data;
    if (Notification.permission === "granted") {
      self.registration.showNotification(title, {
        body,
        icon: "/images/banner-1.png",
        badge: "/images/"
      });
    }
  }
});

// ✅ Push Notification Event
self.addEventListener("push", (event) => {
  console.log("📲 Push Event Received");

  let data = {};
  if (event.data) {
    data = event.data.json();
  }

  const title = data.title || "New Notification!";
  const options = {
    body: data.body || "You have a new message.",
```

```
    icon: "/images/logo.png",
    badge: "/images/logo.png"
  };

  // Check permission before showing notification
  if (Notification.permission === "granted") {
    event.waitUntil(
      self.registration.showNotification(title, options)
    );
  } else {
    console.warn("⃠ Notification not shown. Permission not granted.");
  }
});
```

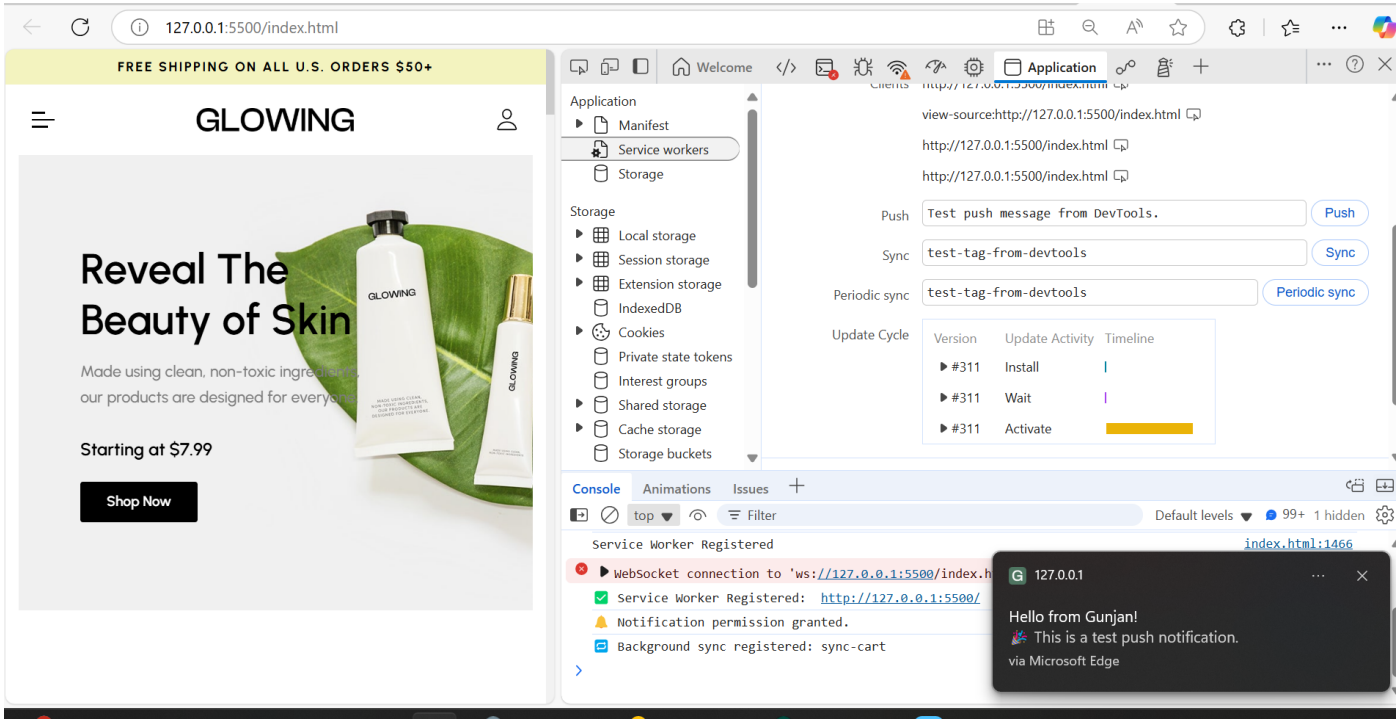
2. **Fetch and Sync Code:**

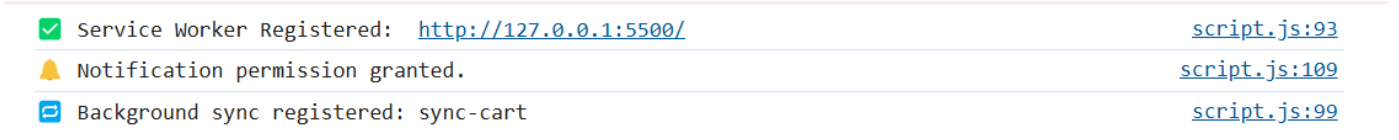
```
'use strict';

/**
 * Add event on element
 */
const addEventOnElem = function (elem, type, callback) {
  if (elem.length > 1) {
    for (let i = 0; i < elem.length; i++) {
      elem[i].addEventListener(type, callback);
    }
  } else {
    elem.addEventListener(type, callback);
  }
};

/**
 * Navbar toggle
 */
const navTogglers = document.querySelectorAll("[data-nav-toggler]");
const navbar = document.querySelector("[data-navbar]");
const navbarLinks = document.querySelectorAll("[data-nav-link]");
const overlay = document.querySelector("[data-overlay]");

const toggleNavbar = function () {
  navbar.classList.toggle("active");
  overlay.classList.toggle("active");
};

addEventOnElem(navTogglers, "click", toggleNavbar);

const closeNavbar = function () {
  navbar.classList.remove("active");
  overlay.classList.remove("active");
};

addEventOnElem(navbarLinks, "click", closeNavbar);

/**
 * Header sticky & back top btn active
 */
const header = document.querySelector("[data-header]");
const backTopBtn = document.querySelector("[data-back-top-btn]");

const headerActive = function () {
  if (window.scrollY > 150) {
    header.classList.add("active");
    backTopBtn.classList.add("active");
  } else {
```

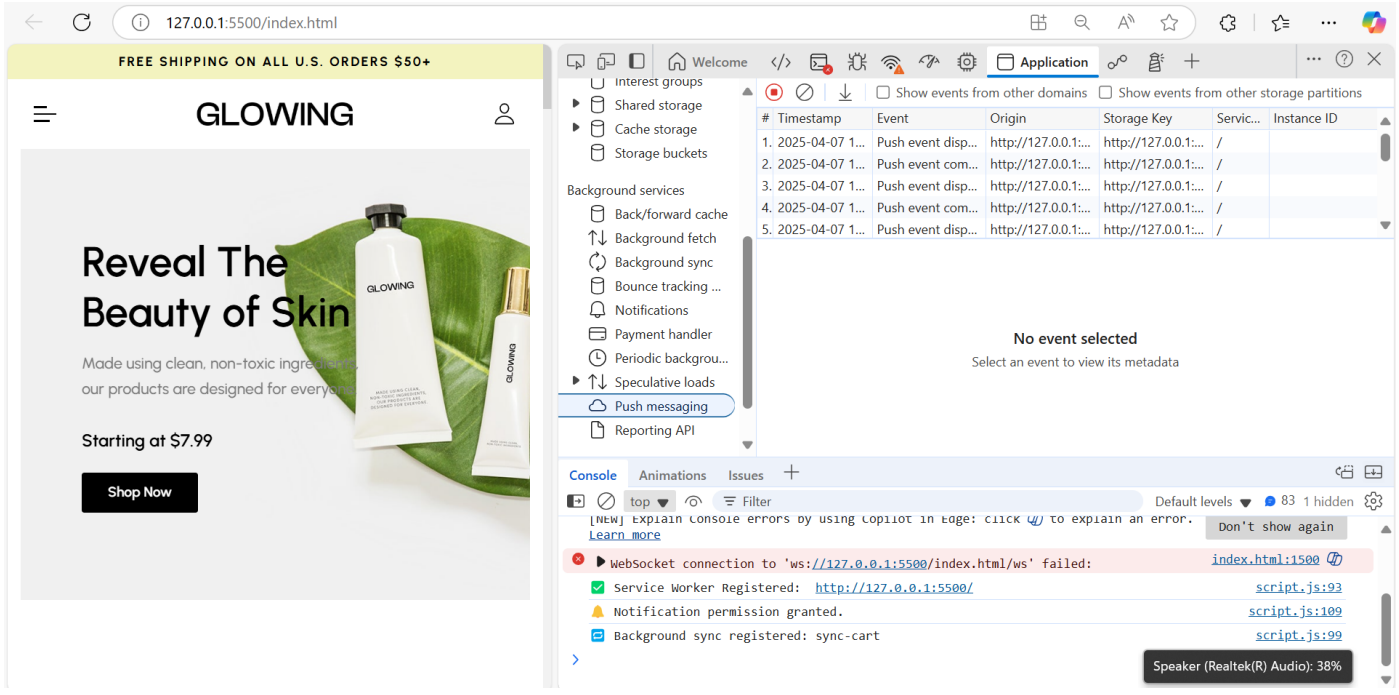
```
    header.classList.remove("active");
    backTopBtn.classList.remove("active");
  }
};

addEventOnElem(window, "scroll", headerActive);

let lastScrolledPos = 0;

const headerSticky = function () {
  if (lastScrolledPos >= window.scrollY) {
    header.classList.remove("header-hide");
  } else {
    header.classList.add("header-hide");
  }

  lastScrolledPos = window.scrollY;
};

addEventOnElem(window, "scroll", headerSticky);

/**
 * Scroll reveal effect
 */
const sections = document.querySelectorAll("[data-section]");

const scrollReveal = function () {
  for (let i = 0; i < sections.length; i++) {
    if (sections[i].getBoundingClientRect().top < window.innerHeight / 2) {
      sections[i].classList.add("active");
    }
  }
};

scrollReveal();
addEventOnElem(window, "scroll", scrollReveal);

/**
 * ✅ Register Service Worker and Notification Permission
 */
if ('serviceWorker' in navigator && 'Notification' in window) {
  window.addEventListener("load", () => {
    navigator.serviceWorker.register('/service-worker.js')
      .then((reg) => {
        console.log("✅ Service Worker Registered: ", reg.scope);

        // Optional: Register background sync
        if ('sync' in reg) {
          reg.sync.register('sync-cart')
            .then(() => {
```

```
              console.log("🔁 Background sync registered: sync-cart");
            })
            .catch((err) => {
              console.error("❌ Sync registration failed:", err);
            });
        }

        // Ask for notification permission
        Notification.requestPermission().then((permission) => {
          if (permission === "granted") {
            console.log("🔔 Notification permission granted.");

            // Optional: Trigger test push manually
            setTimeout(() => {
              reg.active.postMessage({
                type: 'trigger-push',
                title: 'Hello from Gunjan!',
                body: '🎉 This is a test push notification.'
              });
            }, 3000); // Trigger after 3s
          } else {
            console.warn("🔕 Notification permission denied.");
          }
        });
      })
      .catch((err) => {
        console.error("❌ Service Worker registration failed:", err);
      });
  });
}

```

Output:


127.0.0.1:5500/index.html
FREE SHIPPING ON ALL U.S. ORDERS $50+
GLOWING
Reveal The Beauty of Skin
Starting at $7.99
Shop Now
Application
Manifest
Service workers
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Storage buckets
Push
Test push message from DevTools.
Sync
test-tag-from-devtools
Periodic sync
Update Cycle
Version
Update Activity
Timeline
#311 Install
#311 Wait
#311 Activate
Console
Animations
Issues
Service Worker Registered
WebSocket connection to 'ws://127.0.0.1:5500/index.h
Service Worker Registered: http://127.0.0.1:5500/
Notification permission granted.
Background sync registered: sync-cart
127.0.0.1
Hello from Gunjan!
This is a test push notification.
via Microsoft Edge


Service Worker Registered: http://127.0.0.1:5500/ script.js:93
Notification permission granted. script.js:109
Background sync registered: sync-cart script.js:99


127.0.0.1:5500/index.html
FREE SHIPPING ON ALL U.S. ORDERS $50+
GLOWING
Reveal The Beauty of Skin
Starting at $7.99
Shop Now
Application
Shared storage
Cache storage
Storage buckets
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking ...
Notifications
Payment handler
Periodic backgrou...
Speculative loads
Push messaging
Reporting API
Show events from other domains
Show events from other storage partitions
Timestamp
Event
Origin
Storage Key
Instance ID
No event selected
Select an event to view its metadata
Console
Animations
Issues
Don't show again
WebSocket connection to 'ws://127.0.0.1:5500/index.html/ws' failed:
Service Worker Registered: http://127.0.0.1:5500/
Notification permission granted.
Background sync registered: sync-cart
Speaker (Realtek(R) Audio): 38%

EXPERIMENT NO: - 10

Name:- Gunjan Chandnani/ Akruti Dabas/ Vanshika Ambwani **Class:-** D15A

Roll:No: - 06/ 11 /02

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

Theory: -

GitHub Pages: Static Website Hosting Made Simple

GitHub Pages is a free hosting service that allows users to publish **public webpages directly from a GitHub repository**. It is particularly useful for **static websites, project documentation, and blogs**.

Key Features

- **Jekyll Integration**: Built-in support for Jekyll enables easy blogging.
- **Custom Domains**: Allows users to configure their own URLs.
- **Automatic Page Deployment**: Simply push your changes to the repository, and the updates go live.

Why Choose GitHub Pages?

- **Completely Free**: No hosting charges.
- **Seamless GitHub Integration**: Works directly with your repositories.
- **Quick Setup**: Just create a repository, push your files, and your site is live.

Who Uses GitHub Pages?

Companies like **Lyft, CircleCI, and HubSpot** use GitHub Pages for their documentation and static sites. It is widely adopted, appearing in **775 company stacks and 4,401 developer stacks**.

Pros & Cons

Pros

- Familiar interface for GitHub users.

- Simple deployment via the gh-pages branch.
- Supports custom domains with easy DNS configuration.

Cons

- Repositories need to be public unless you have a paid plan.
- Limited HTTPS support for custom domains (expected to improve).
- Jekyll plugins have limited support.

Firebase: A Full-Featured Real-Time Backend

Firebase is a cloud-based **real-time application platform** developed by Google. It enables developers to build **dynamic, collaborative applications** with ease.

Key Features

- **Real-Time Database**: Automatically syncs data across all connected clients.
- **Cloud-Based Storage**: JSON-based storage accessible via REST APIs.
- **Scalable Infrastructure**: Works well with existing services and scales automatically.
- **Authentication & Cloud Messaging**: Secure login and push notifications.

Why Choose Firebase?

- **Instant Backend Setup**: No need to build a separate backend.
- **Fast & Responsive**: Real-time data synchronization.
- **Built-in HTTPS**: Free SSL certificates for custom domains.

Who Uses Firebase?

Companies like **Instacart, 9GAG, and Twitch** rely on Firebase for their backend needs. Firebase is widely adopted, appearing in **1,215 company stacks and 4,651 developer stacks**.

Pros & Cons

Pros

- **Hosted by Google**, ensuring reliability and security.
- **Comes with authentication, messaging, and real-time database services.**
- **Free HTTPS support** for all custom domains.

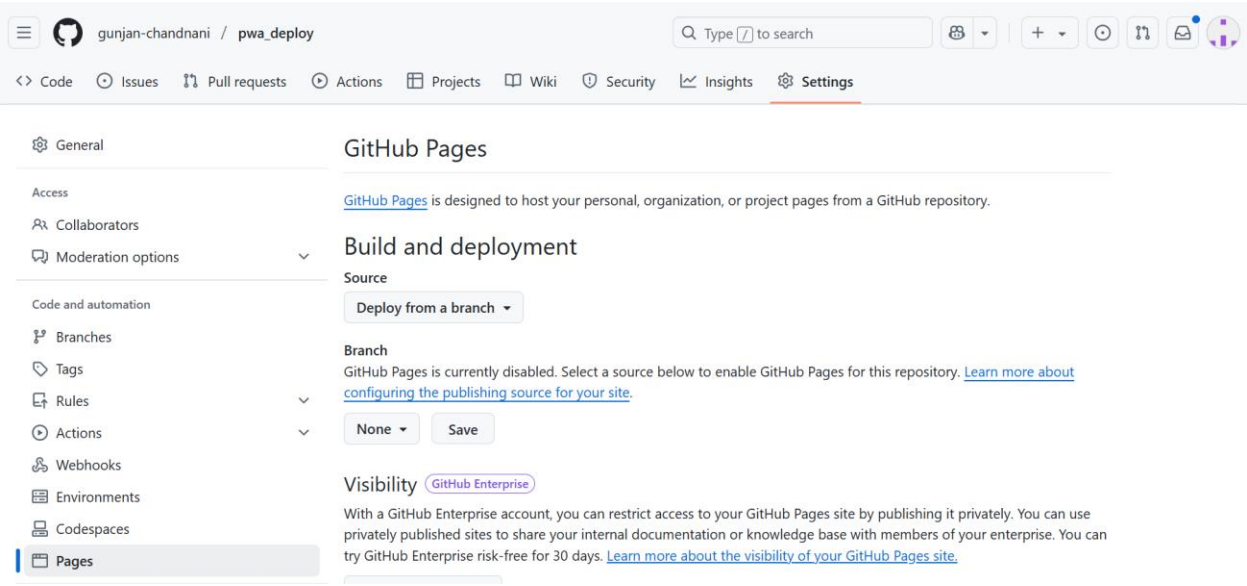
Cons

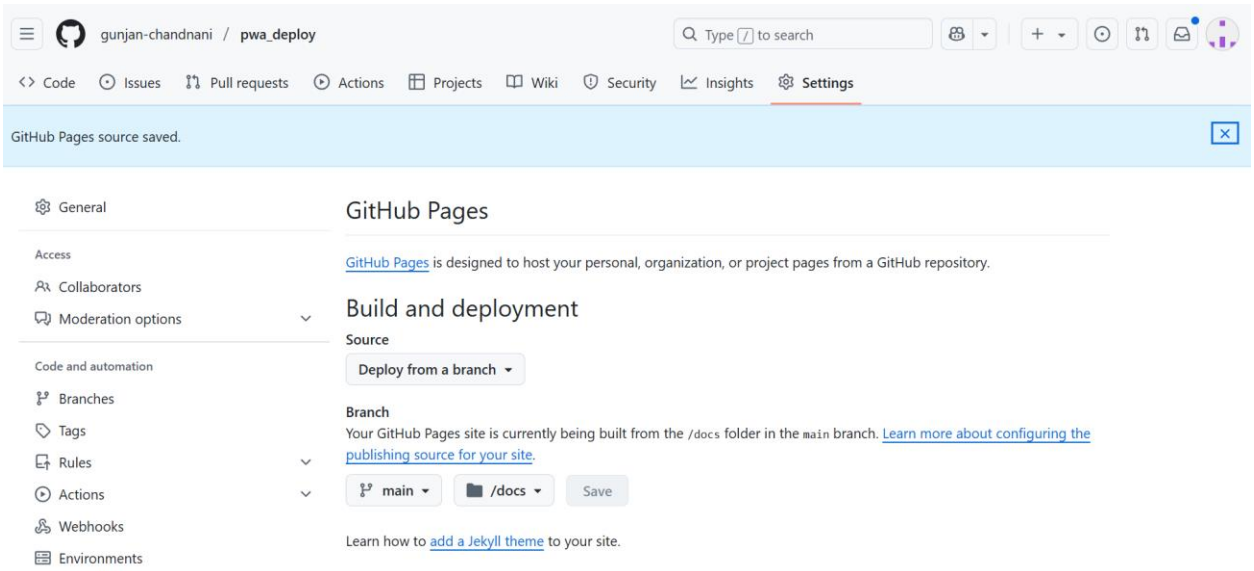
- **Limited Free Plan**: 10 GB of data transfer per month (can be mitigated with a CDN).
- **Command-Line Deployment**: No GUI for hosting.
- **No Built-in Static Site Generator Support**: Unlike GitHub Pages, Firebase doesn't natively support static site generators like Jekyll.

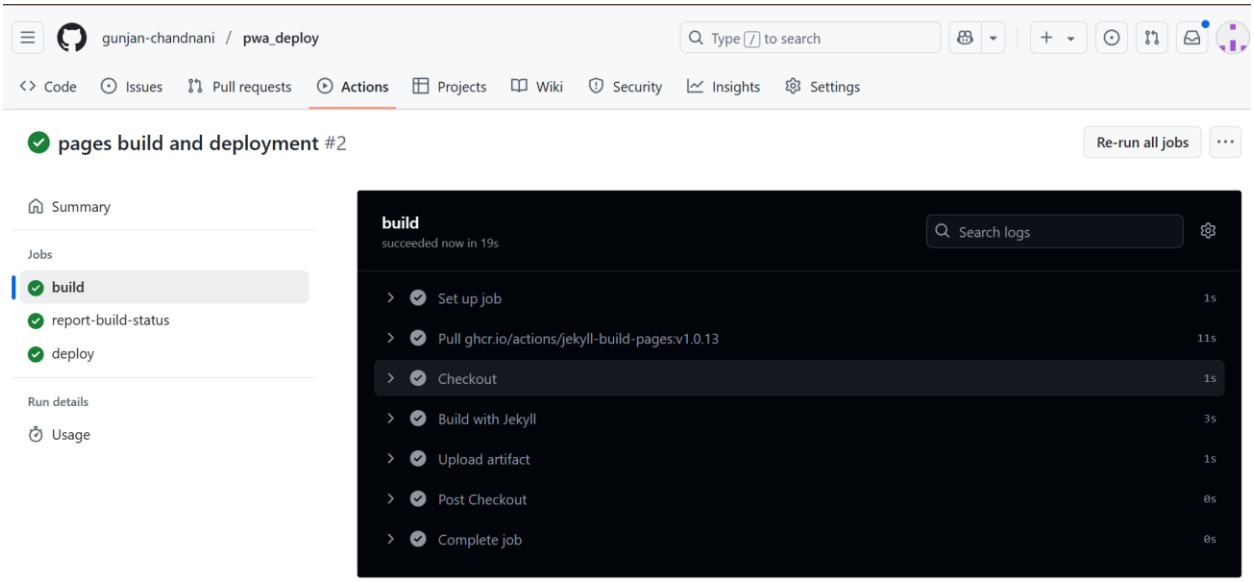
Link to our GitHub repository:

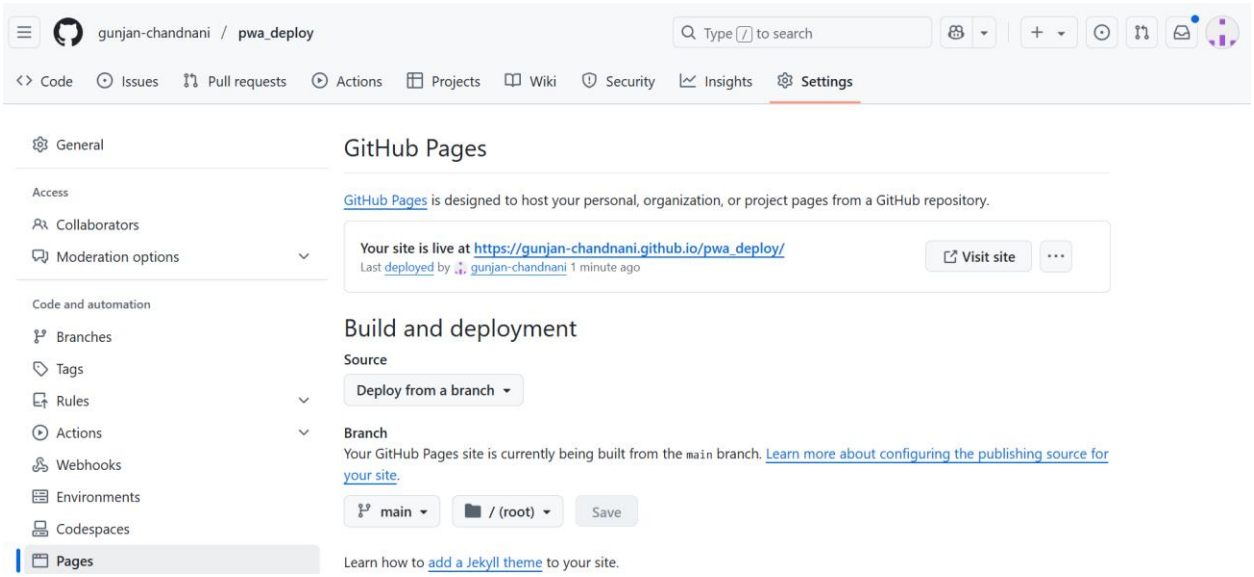
https://gunjan-chandnani.github.io/pwa_deploy/

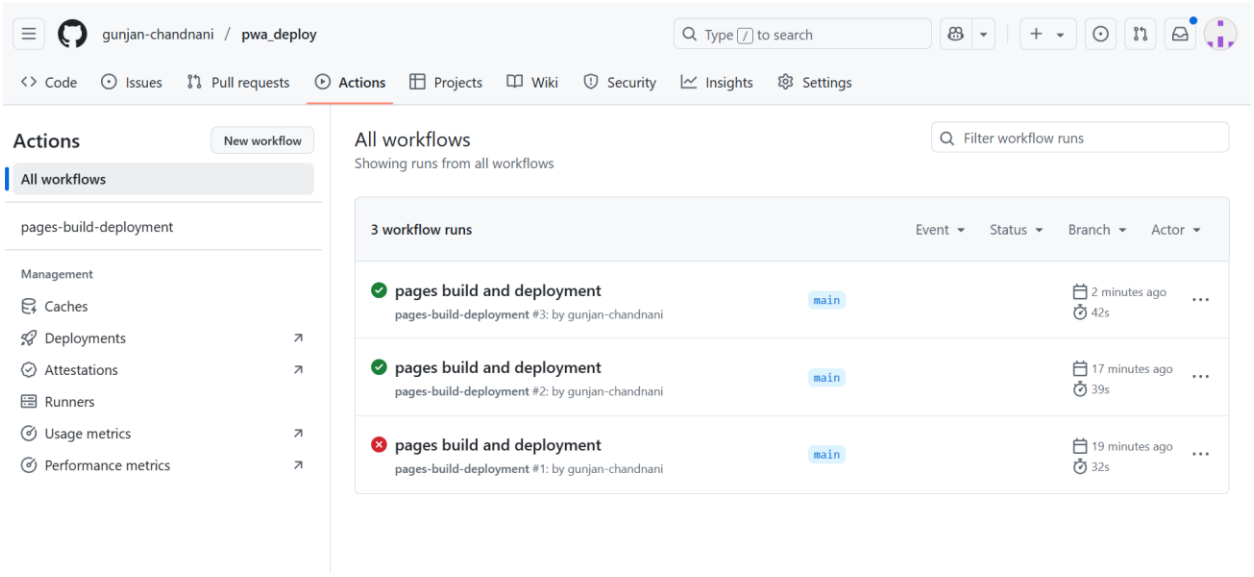
Github Screenshot:

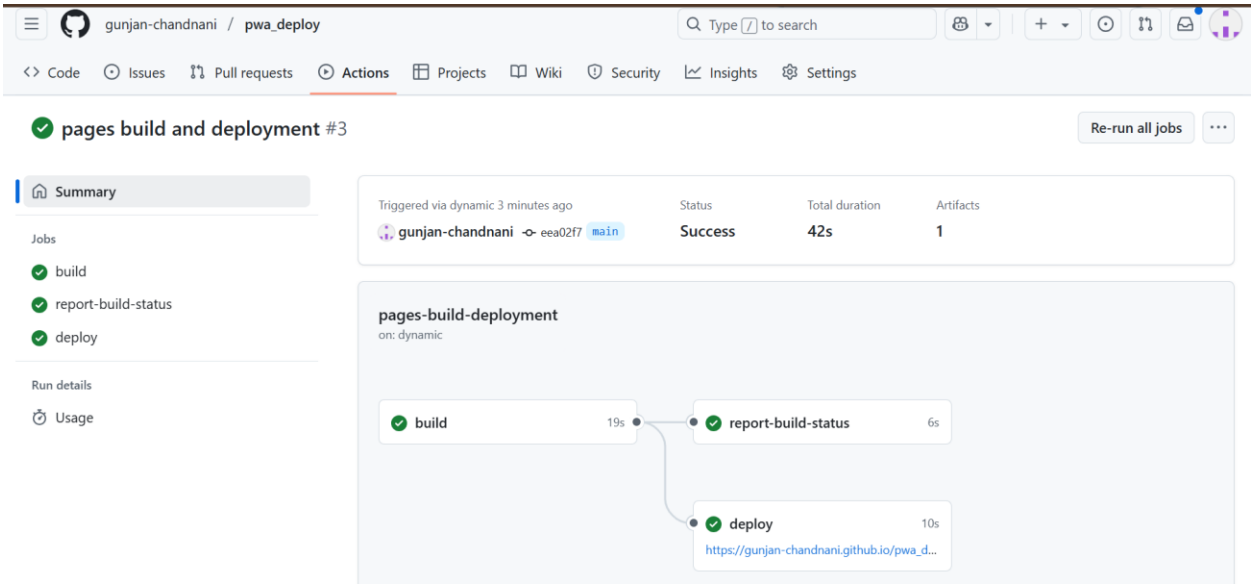


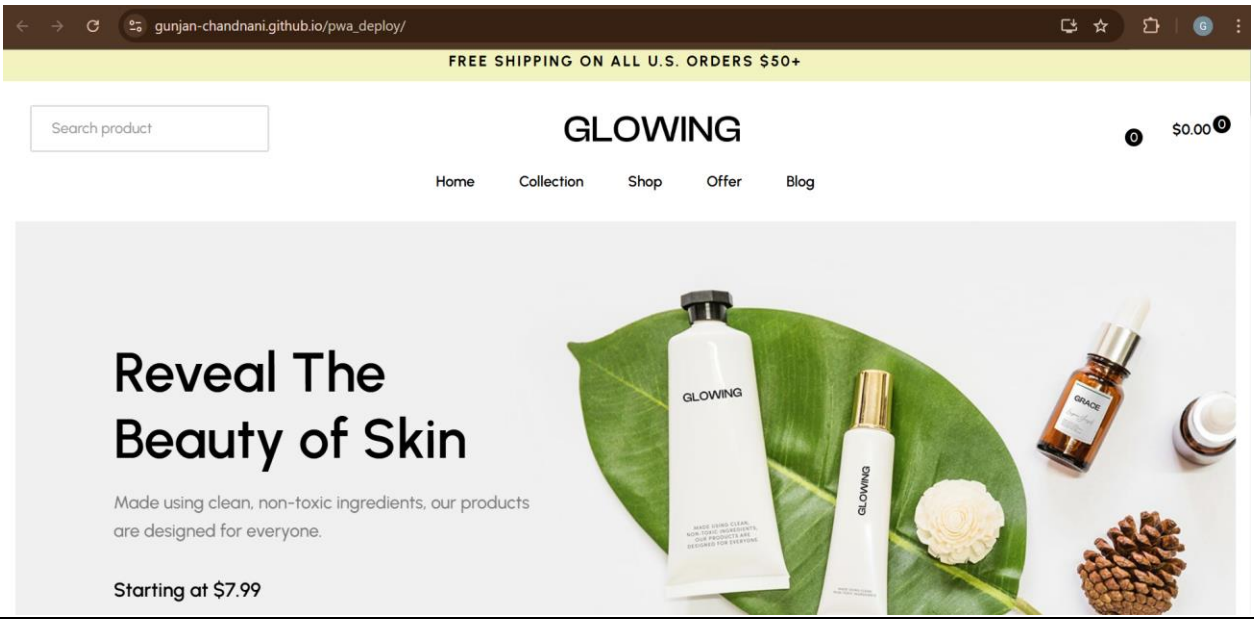

gunjan-chandnani / pwa_deploy
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
GitHub Pages source saved.
General
Access
Collaborators
Moderation options
Code and automation
Branches
Tags
Rules
Actions
Webhooks
Environments
GitHub Pages
GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.
Build and deployment
Source
Deploy from a branch
Branch
Your GitHub Pages site is currently being built from the /docs folder in the main branch. Learn more about configuring the publishing source for your site.
main
/docs
Save
Learn how to add a Jekyll theme to your site.


gunjan-chandnani / pwa_deploy
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
pages build and deployment #2
Re-run all jobs
Summary
Jobs
build
report-build-status
deploy
Run details
Usage
build
succeeded now in 19s
Search logs
Set up job 1s
Pull ghcr.io/actions/jekyll-build-pages:v1.0.13 11s
Checkout 1s
Build with Jekyll 3s
Upload artifact 1s
Post Checkout 0s
Complete job 0s


gunjan-chandnani / pwa_deploy
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
General
Access
Collaborators
Moderation options
Code and automation
Branches
Tags
Rules
Actions
Webhooks
Environments
Codespaces
Pages
GitHub Pages
GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.
Your site is live at https://gunjan-chandnani.github.io/pwa_deploy/
Last deployed by gunjan-chandnani 1 minute ago
Visit site
Build and deployment
Source
Deploy from a branch
Branch
Your GitHub Pages site is currently being built from the main branch. Learn more about configuring the publishing source for your site.
main
/ (root)
Save
Learn how to add a Jekyll theme to your site.


gunjan-chandnani / pwa_deploy
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Actions
New workflow
All workflows
pages-build-deployment
Management
Caches
Deployments
Attestations
Runners
Usage metrics
Performance metrics
All workflows
Showing runs from all workflows
Filter workflow runs
3 workflow runs
Event
Status
Branch
Actor
pages build and deployment
pages-build-deployment #3: by gunjan-chandnani
main
2 minutes ago
42s
pages build and deployment
pages-build-deployment #2: by gunjan-chandnani
main
17 minutes ago
39s
pages build and deployment
pages-build-deployment #1: by gunjan-chandnani
main
19 minutes ago
32s


gunjan-chandnani / pwa_deploy
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
pages build and deployment #3
Re-run all jobs
Summary
Jobs
build
report-build-status
deploy
Run details
Usage
Triggered via dynamic 3 minutes ago
gunjan-chandnani eea02f7 main
Status
Success
Total duration
42s
Artifacts
1
pages-build-deployment
on: dynamic
build
19s
report-build-status
6s
deploy
10s
https://gunjan-chandnani.github.io/pwa_d...


gunjan-chandnani.github.io/pwa_deploy/
FREE SHIPPING ON ALL U.S. ORDERS $50+
Search product
GLOWING
$0.00
Home
Collection
Shop
Offer
Blog
Reveal The Beauty of Skin
Made using clean, non-toxic ingredients, our products are designed for everyone.
Starting at $7.99
GLOWING
GLOWING
GRACE

More to Discover

Find a Store

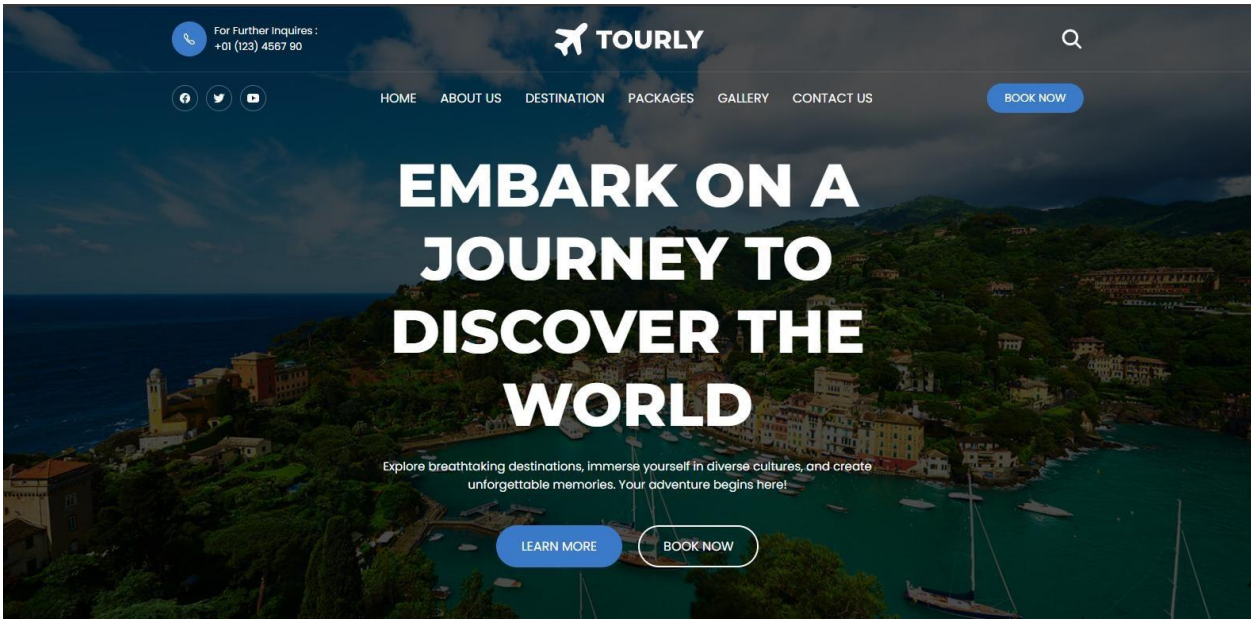
Our Store

From Our Blog

Our Store

Our Story

Our Store


For Further Inquires :
+01 (123) 4567 90
TOURLY
HOME
ABOUT US
DESTINATION
PACKAGES
GALLERY
CONTACT US
BOOK NOW
EMBARK ON A JOURNEY TO DISCOVER THE WORLD
Explore breathtaking destinations, immerse yourself in diverse cultures, and create unforgettable memories. Your adventure begins here!
LEARN MORE
BOOK NOW

EXPERIMENT NO: - 11

Name:- Vanshika, Gunajn, Akruti **Class:-** D15A **Roll:No: -** 2,6,11

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

Theory: -

Google Lighthouse is an open-source tool designed to evaluate and optimize web applications based on multiple key parameters, including performance, accessibility, Progressive Web App (PWA) implementation, and best practices. It provides a detailed automated audit that helps developers improve their websites efficiently. Unlike traditional manual audits that can take days or weeks, Lighthouse generates insights within minutes with minimal setup.

One of Lighthouse's biggest advantages is its ease of use—simply run it on a webpage or provide a URL, and it will generate a comprehensive report. Lighthouse supports audits for both desktop and mobile versions of a website, ensuring an optimal user experience across different devices.

Key Features and Audit Metrics

1. **Performance**

This metric evaluates how efficiently a webpage loads and becomes interactive. It considers several factors, including:

- Page Load Speed – Measures how quickly content appears to users.
- First Contentful Paint (FCP) – Time taken for the first visible content to load.
- Largest Contentful Paint (LCP) – Measures how long the largest visible content takes to fully render.
- Cumulative Layout Shift (CLS) – Ensures content does not shift unexpectedly, improving user experience.
- Time to Interactive (TTI) – The time taken for the webpage to become fully functional.

Lighthouse assigns a performance score from 0 to 100, where:

- 100 → Top 2% of websites (98th percentile)
- 50 → Around the 75th percentile
- Lower scores → Indicate areas needing optimization

2. Progressive Web App (PWA) Score

With the increasing adoption of PWAs, web applications now strive to deliver an app-like experience. Lighthouse evaluates a website's PWA readiness based on Google's Baseline PWA Checklist, which includes:

- Service Worker Implementation – Enables offline functionality and background synchronization.
- App Manifest Compliance – Provides metadata for mobile app-like integration.
- Viewport Configuration – Ensures responsiveness across different screen sizes.
- Performance in Script-Disabled Environments – Verifies that the page functions properly even if JavaScript is disabled.

A high PWA score indicates that the application meets essential criteria for speed, reliability, and mobile usability.

3. Accessibility

This metric determines how well a website supports users with **visual, cognitive, or physical disabilities**. Lighthouse evaluates accessibility based on:

- **ARIA Attributes** – Enhances screen reader support (e.g., aria-required).
- **Text Alternatives for Media** – Ensures images, audio, and video content are accessible.
- **Semantic HTML Usage** – Encourages proper use of elements like <section>, <article>, and <button> for better screen-reader compatibility.

Unlike other metrics, **accessibility follows a pass/fail approach**—missing key features can **significantly impact the overall score**. Improving accessibility ensures **inclusivity for all users**.

4. Best Practices

Lighthouse also assesses whether the website follows **modern web development best practices**, including:

- **Use of HTTPS** – Ensures secure data transmission.
- **Avoiding Deprecated Code** – Prevents the use of outdated elements, directives, and libraries.
- **Secure Password Inputs** – Disables paste-into fields to reduce security risks.
- **User Security Alerts** – Provides notifications for **geo-location access and cookie usage**.

A high **Best Practices score** means the website meets industry standards, leading to better **security, maintainability, and overall performance**.

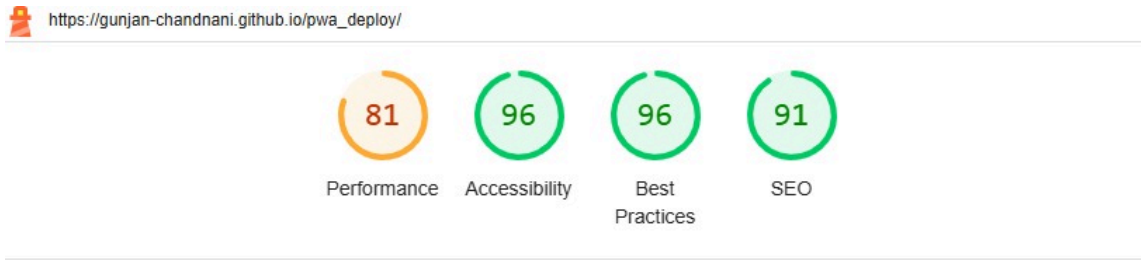
Manifest.json


```
{
  "name": "Shopping App",
  "short_name": "MyApp",
  "description": "An online shopping platform with a wide range of products.",
  "start_url": "/",
  "scope": "/",
  "display": "standalone",
  "background_color": "#ffffff",
  "theme_color": "#ff6600",
  "icons": [
    {
      "src": "assets/images/banner-1.png",
      "sizes": "192x192",
      "type": "image/png",
      "purpose": "any maskable"
    },
    {
      "src": "assets/images/banner-2.png",
      "sizes": "512x512",
      "type": "image/png",
      "purpose": "any maskable"
    }
  ],
  "shortcuts": [
    {
      "name": "Shop Now",
      "short_name": "Shop",
```

```
    "description": "Go directly to shopping",
    "url": "/index.html",
    "icons": [
      {
        "src": "assets/images/shortcut-icon.png",
        "sizes": "96x96",
        "type": "image/png"
      }
    ]
  }
 ]
}
```

Output: -

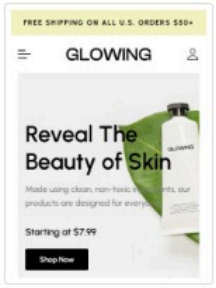
A] For Mobile



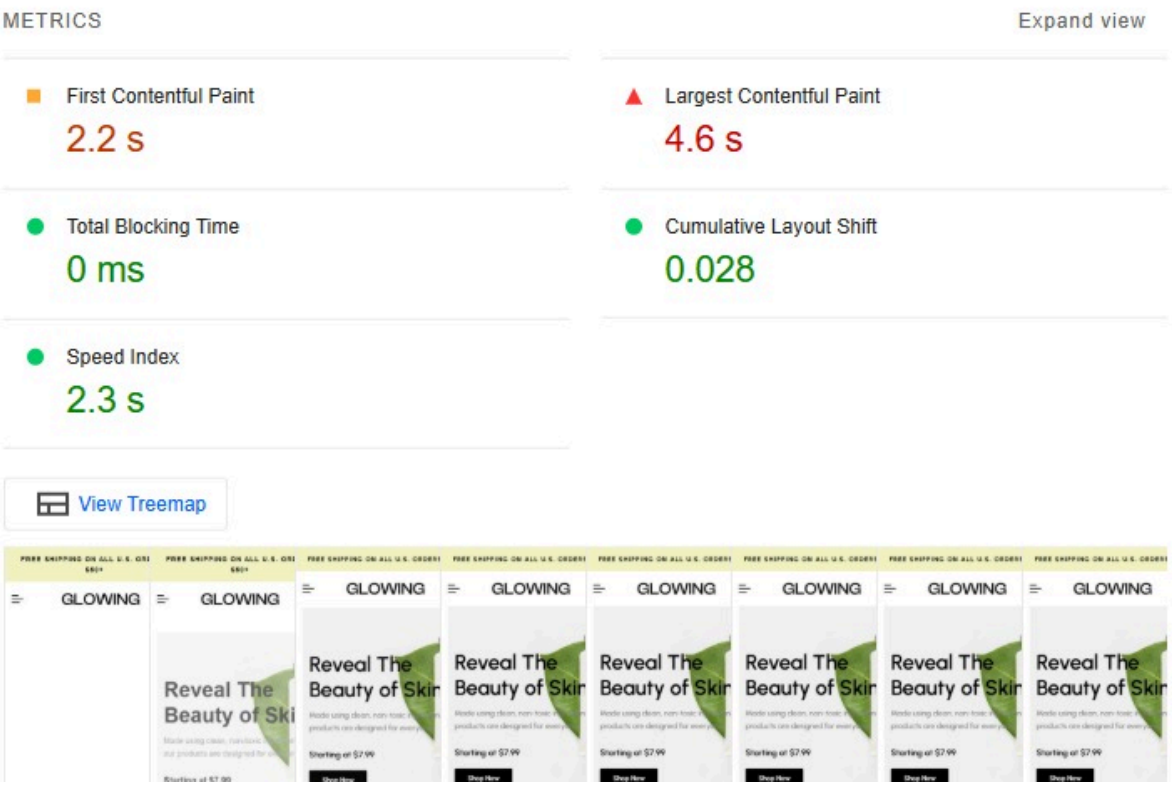
Performance

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

▲ 0–49 ■ 50–89 ● 90–100



METRICS Expand view



DIAGNOSTICS

- Largest Contentful Paint element — 4,610 ms
- Serve images in next-gen formats — Potential savings of 139 KiB
- Enable text compression — Potential savings of 62 KiB
- Use HTTP/2 — 16 requests not served via HTTP/2
- Defer offscreen images — Potential savings of 36 KiB
- Eliminate render-blocking resources — Potential savings of 170 ms
- Minify CSS — Potential savings of 6 KiB
- Serve static assets with an efficient cache policy — 10 resources found
- Avoid an excessive DOM size — 1,071 elements

https://gunjan-chandnani.github.io/pwa_deploy/

Accessibility

These checks highlight opportunities to improve the accessibility of your web app. Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so manual testing is also encouraged.

ARIA

- Elements use prohibited ARIA attributes

These are opportunities to improve the usage of ARIA in your application which may enhance the experience for users of assistive technology, like a screen reader.

ADDITIONAL ITEMS TO MANUALLY CHECK (10) Show

These items address areas which an automated testing tool cannot cover. Learn more in our guide on conducting an accessibility

ADDITIONAL ITEMS TO MANUALLY CHECK (10) Show

These items address areas which an automated testing tool cannot cover. Learn more in our guide on conducting an accessibility review.

PASSED AUDITS (24) Show

NOT APPLICABLE (32) Show

Best Practices

B]For Desktop

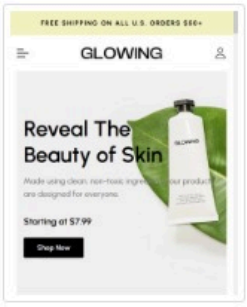
https://gunjan-chandnani.github.io/pwa_deploy/

Performance Accessibility Best Practices SEO

Performance

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

▲ 0–49 ■ 50–89 ● 90–100

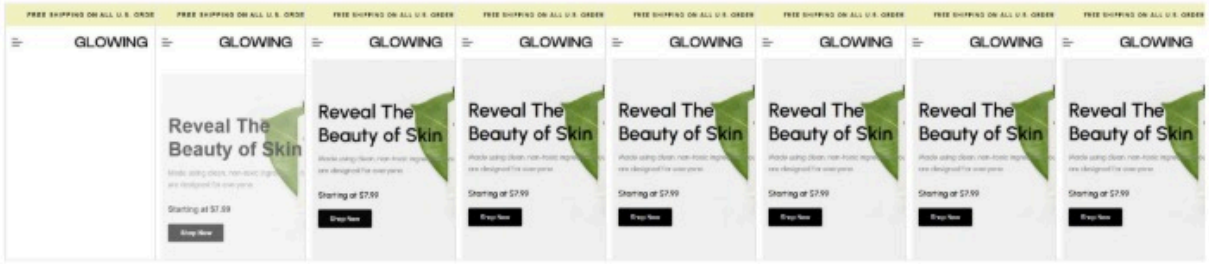


METRICS Expand view

97 96 96 91

- First Contentful Paint
  0.7 s
- Largest Contentful Paint
  1.2 s
- Total Blocking Time
  0 ms
- Cumulative Layout Shift
  0.001
- Speed Index
  0.9 s

View Treemap



Show audits relevant to: All FCP LCP TBT CLS