Vivekanand Education Society's Institute of Technology

An Autonomous Institute Affiliated to University of Mumbai
Hashu Advani Memorial Complex, Collector Colony, Chembur East, Mumbai - 400074.

Department of Information Technology

CERTIFICATE

This is to certify that **Himesh Pathai** of **D15A/D15B** semester **VI,** have successfully completed necessary experiments in the **MAD & PWA Lab** under my supervision in **VES Institute of Technology** during the academic year **2024-2025.**

Lab Assistant

Subject Teacher

Mrs. Kajal Joseph

Principal

Head of Department

Dr. Mrs. Shalu Chopra

Name of the Course : MAD & PWA Lab

Course Code : ITL604

Year/Sem/Class **:** D15A/D15B **A.Y.:** 24-25

Faculty Incharge : Mrs. Kajal Joseph.

Lab Teachers **:** Mrs. Kajal Joseph.

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

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

Program specific Outcomes

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

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

Lab Objectives:

Sr. No.	Lab Objectives
The Lab experiments aims:	
1	Learn the basics of the Flutter framework.
2	Develop the App UI by incorporating widgets, layouts, gestures and animation
3	Create a production ready Flutter App by including files and firebase backend service.
4	Learn the Essential technologies, and Concepts of PWAs to get started as quickly and efficiently as possible
5	Develop responsive web applications by combining AJAX development techniques with the jQuery JavaScript library.
6	Understand how service workers operate and also learn to Test and Deploy PWA.

Lab Outcomes:

Sr. No.	Lab Outcomes	Cognitive levels of attainment as per Bloom's Taxonomy
On Completion of the course the learner/student should be able to:		
1	Understand cross platform mobile application development using Flutter framework	L1, L2
2	Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation	L3
3	Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS	L3, L4
4	Understand various PWA frameworks and their requirements	L1, L2
5	Design and Develop a responsive User Interface by applying PWA Design techniques	L3
6	Develop and Analyse PWA Features and deploy it over app hosting solutions	L3, L4

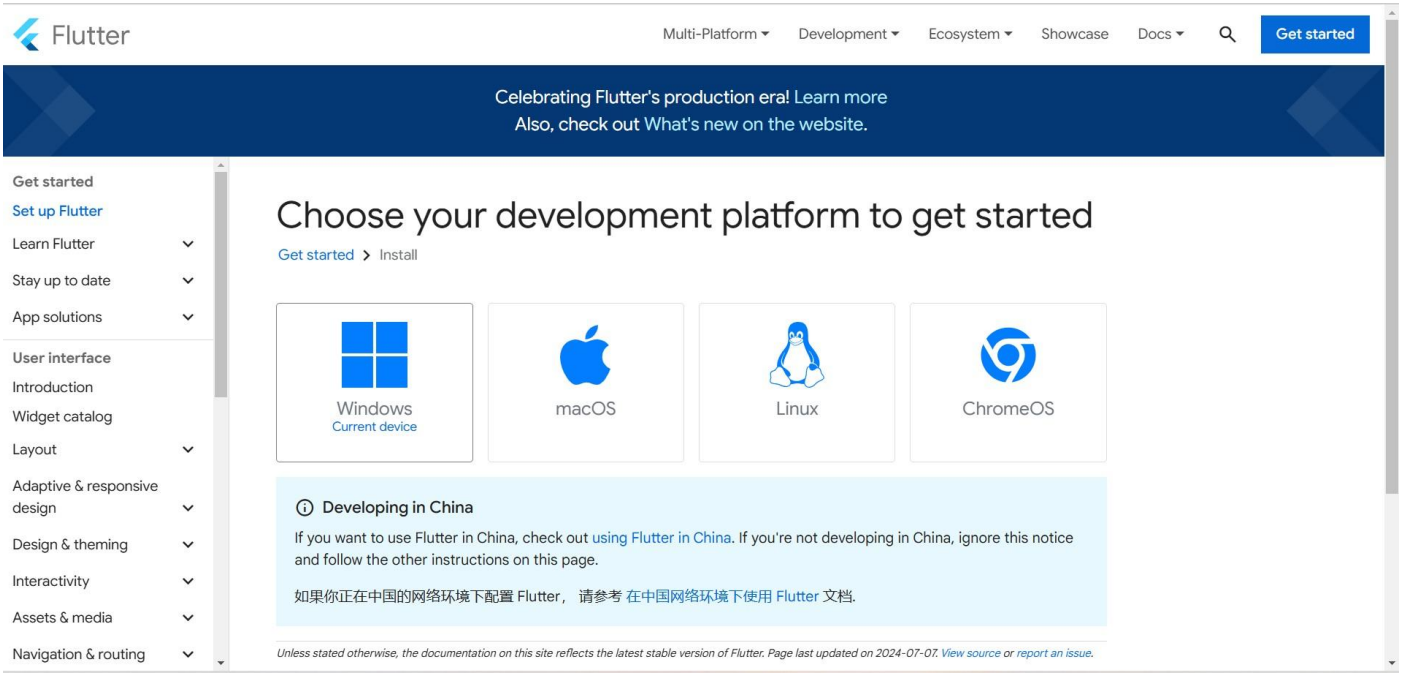
Index

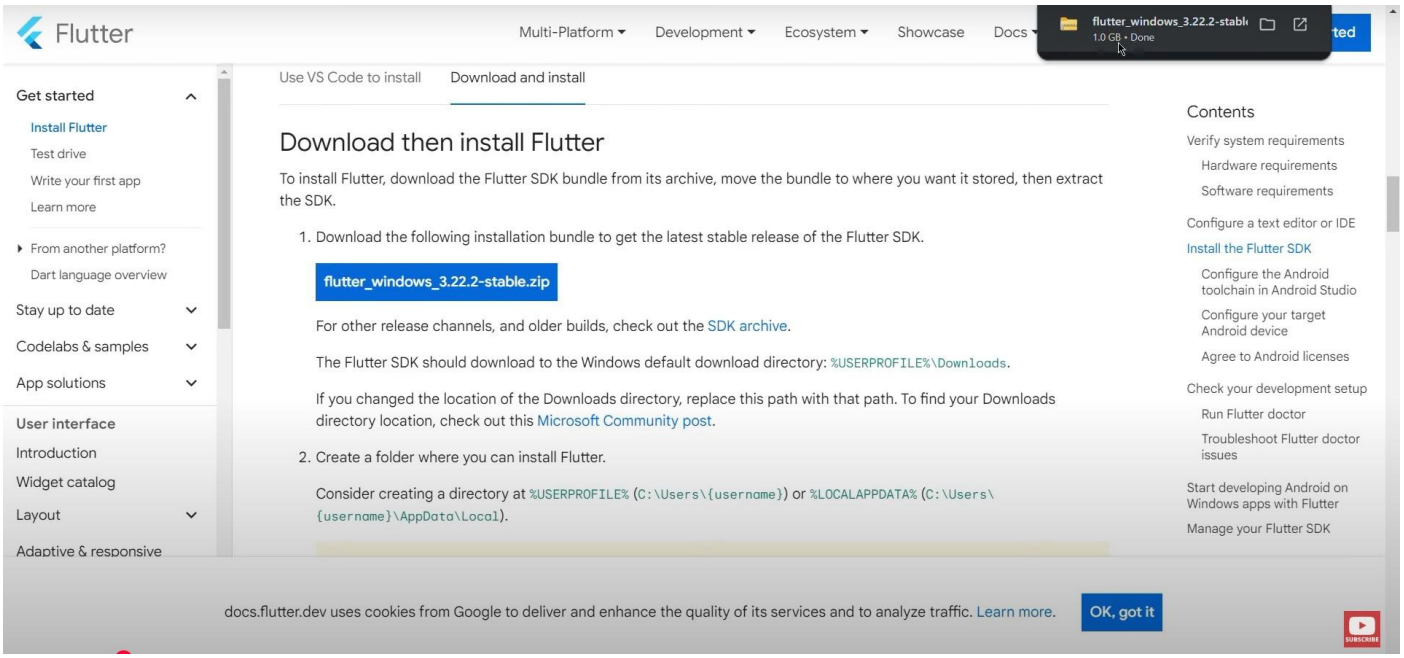
Sr. No	Experiment Title	LO	DOP	DOS	Grade
1.	To install and configure the Flutter Environment	LO1			
2.	To design Flutter UI by including common widgets.	LO2			
3.	To include icons, images, fonts in Flutter app	LO2			
4.	To create an interactive Form using form widget	LO2			
5.	To apply navigation, routing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

MAD & PWA Lab
Journal

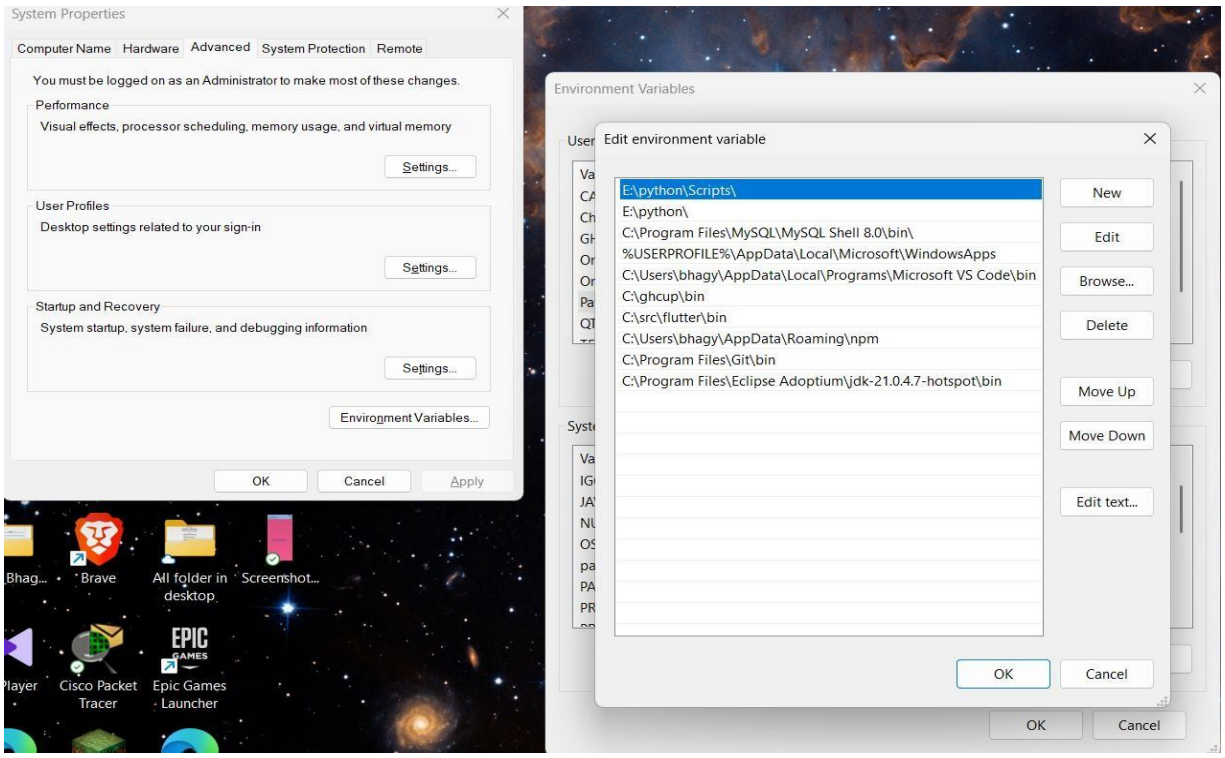
Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	34
Name	Himesh Pathai
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	

- Install flutter





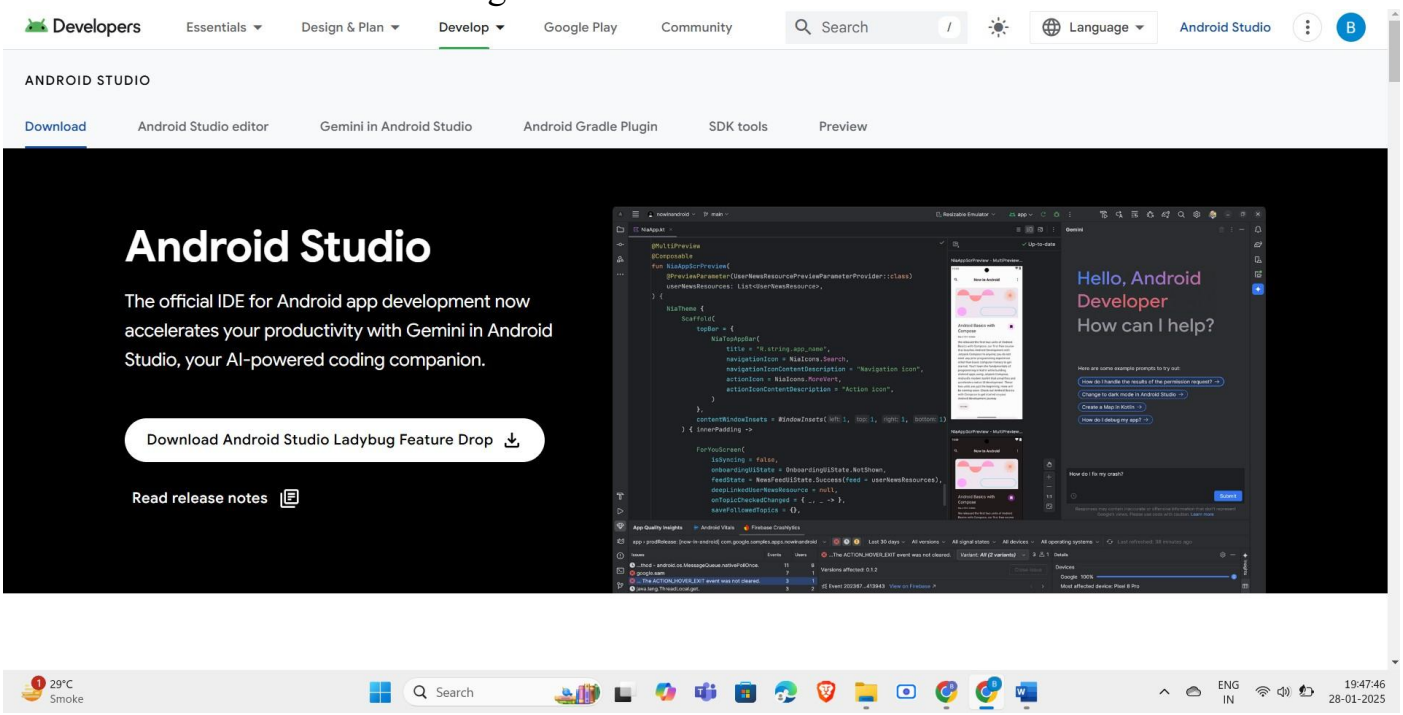
- Set Environment variables and paste the flutter bin path in path option

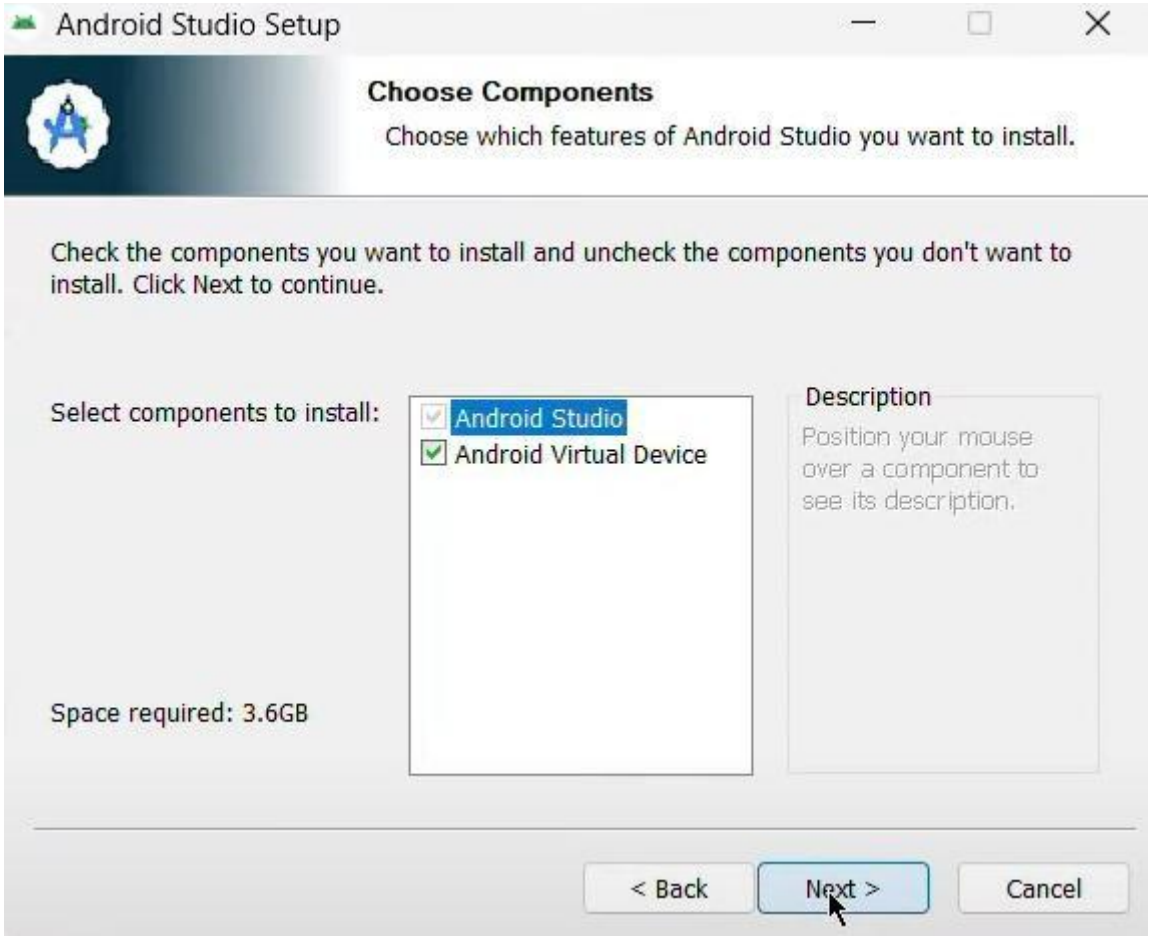


- In Cmd type flutter doctor

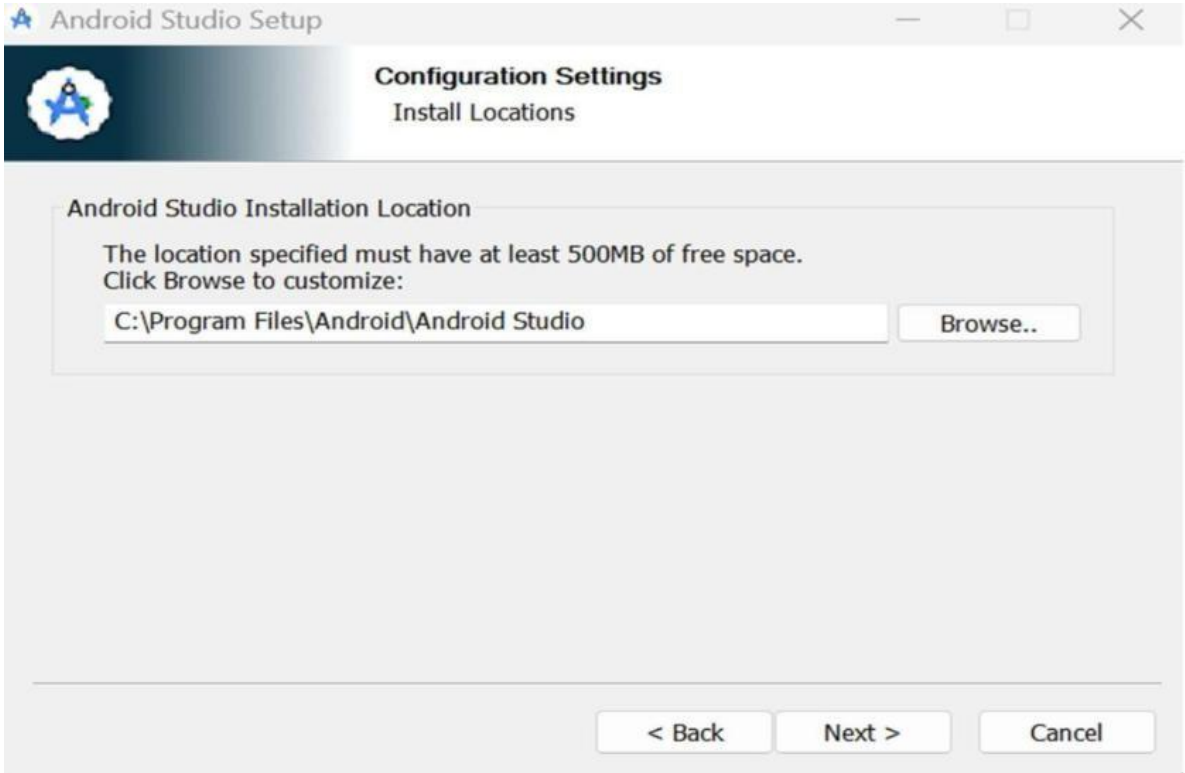
```
                Welcome to Flutter! - https://flutter.dev

The Flutter tool uses Google Analytics to anonymously report feature usage
statistics and basic crash reports. This data is used to help improve
Flutter tools over time.

Flutter tool analytics are not sent on the very first run. To disable
reporting, type 'flutter config --no-analytics'. To display the current
setting, type 'flutter config'. If you opt out of analytics, an opt-out
event will be sent, and then no further information will be sent by the
Flutter tool.

By downloading the Flutter SDK, you agree to the Google Terms of Service.
The Google Privacy Policy describes how data is handled in this service.

Moreover, Flutter includes the Dart SDK, which may send usage metrics and
crash reports to Google.

Read about data we send with crash reports:
https://flutter.dev/docs/reference/crash-reporting

See Google's privacy policy:
https://policies.google.com/privacy

To disable animations in this tool, use
'flutter config --no-cli-animations'.
```

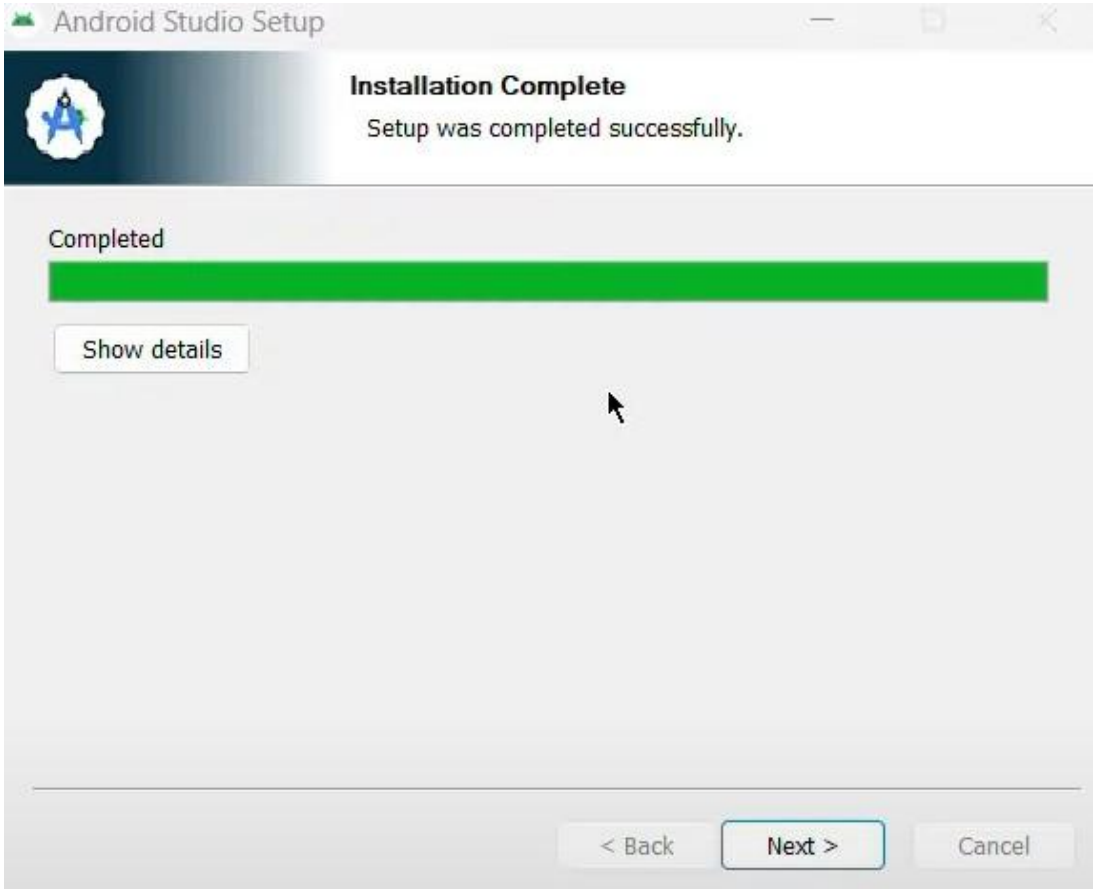
- Download the android studio go the official website



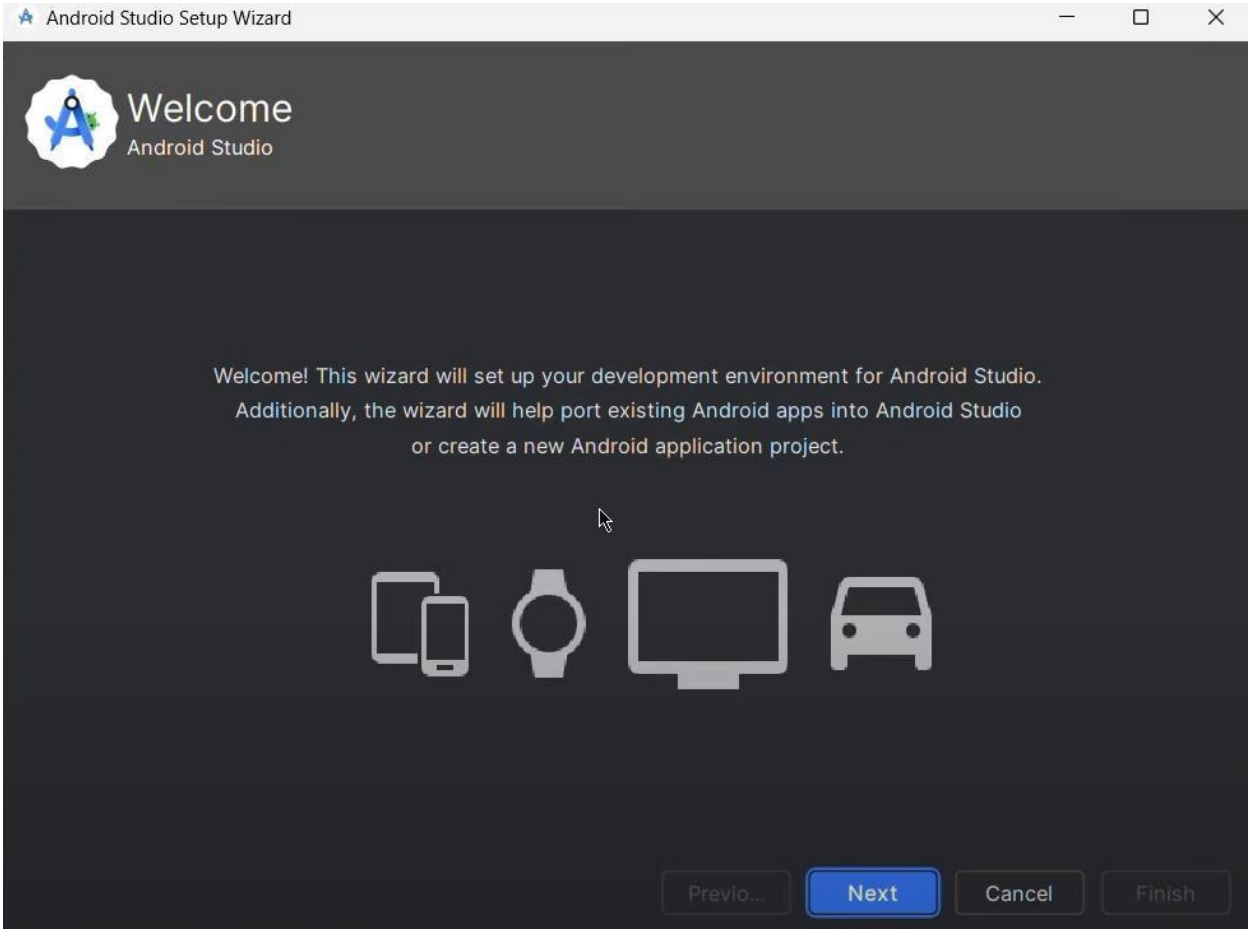


- Downloading Android SDK

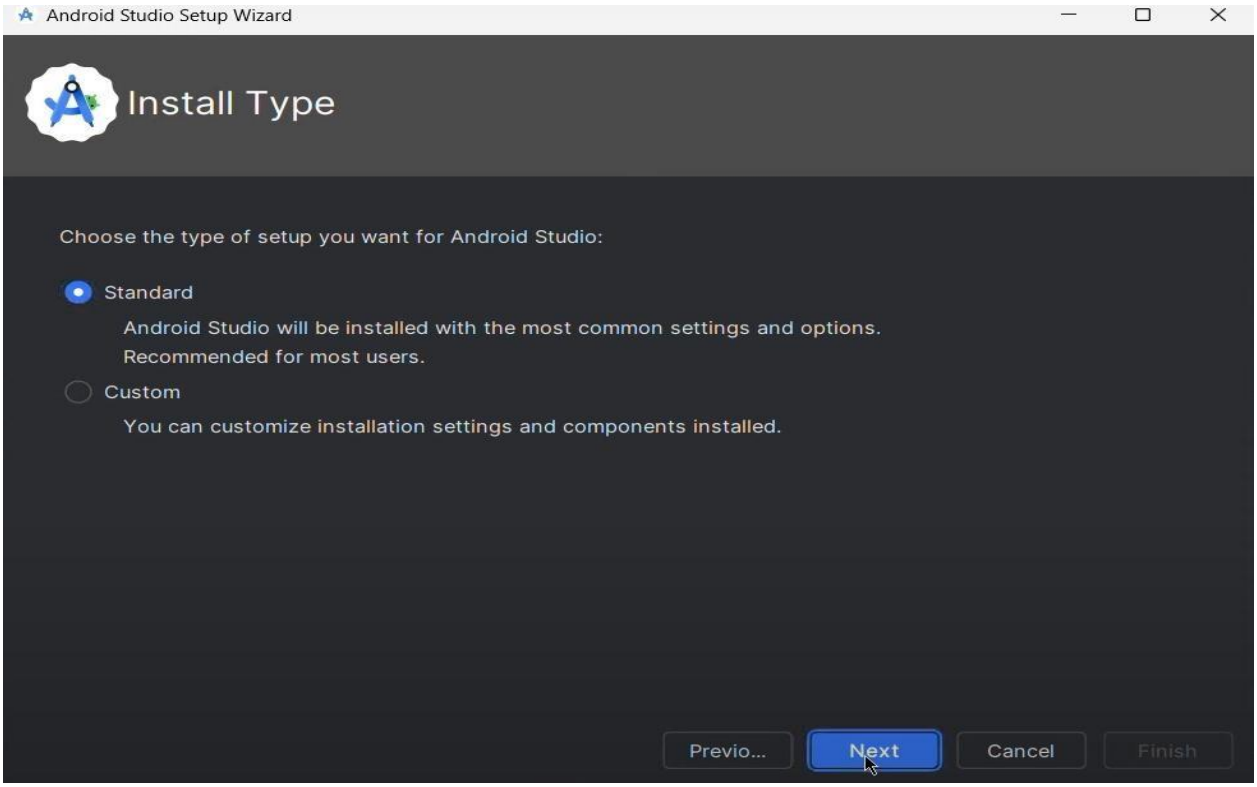




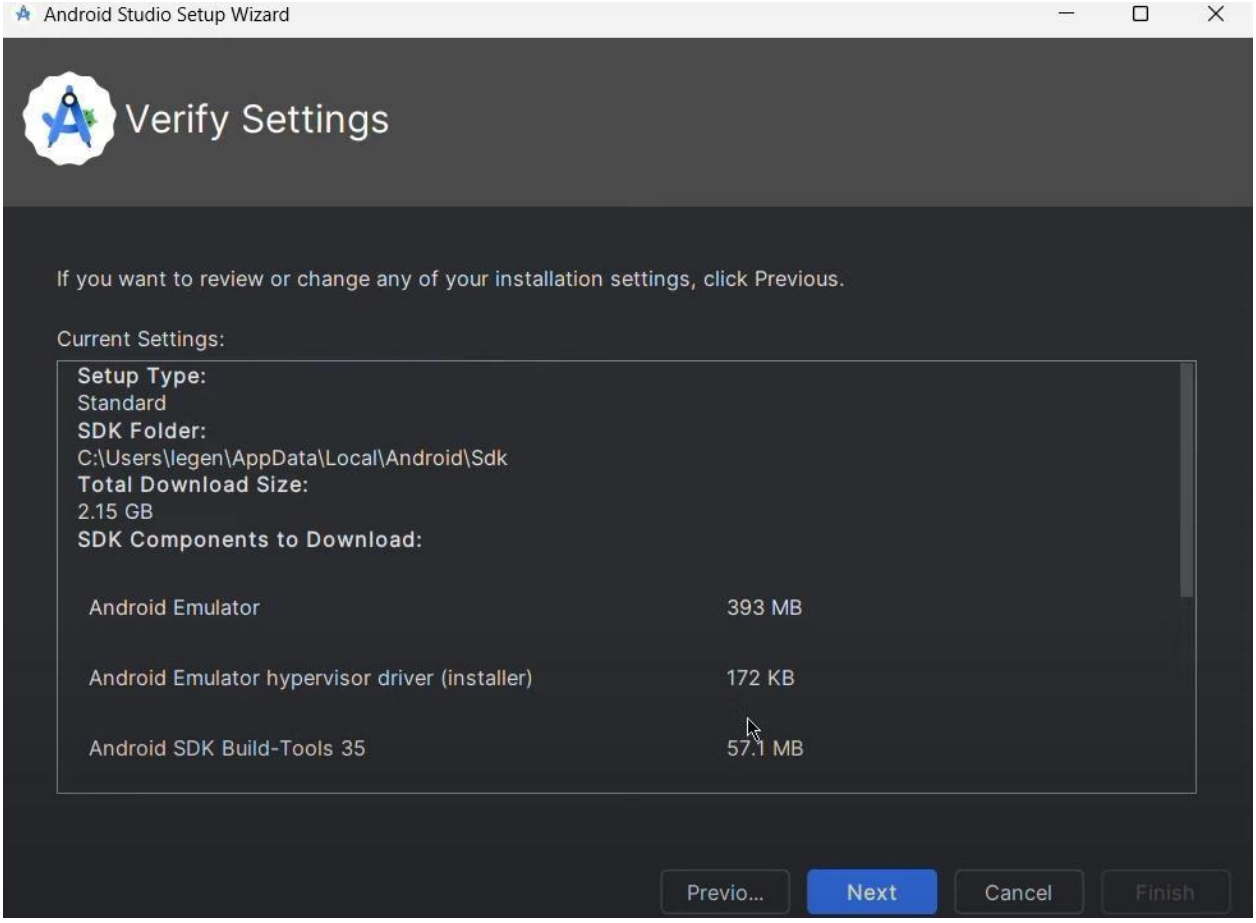
- After downloading the android studio this page come



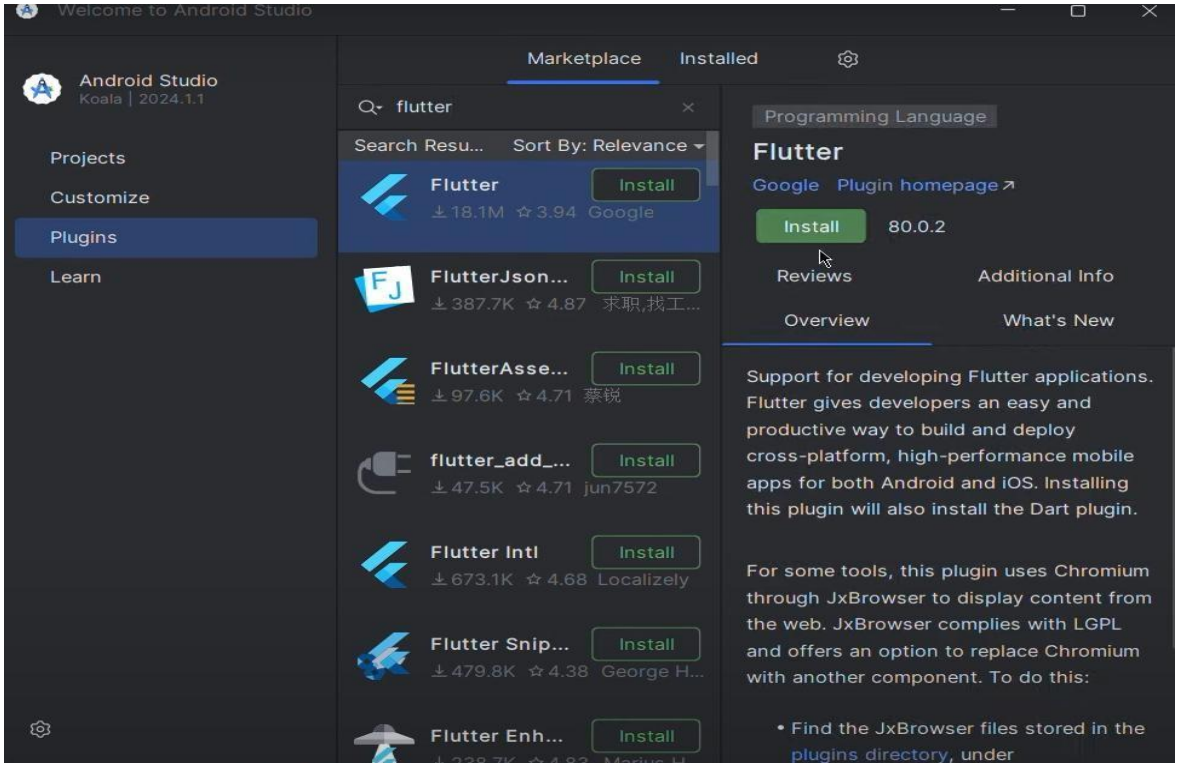
- Select Standard and click next



- Download the SDK and android emulator



- Install Plugin in android studio


Welcome to Android Studio
Android Studio
Koala | 2024.1.1
Projects
Customize
Plugins
Learn
Marketplace
Installed
flutter
Search Resu...
Sort By: Relevance
Flutter
Install
18.1M 3.94 Google
FlutterJson...
Install
387.7K 4.87 求职,找工...
FlutterAsse...
Install
97.6K 4.71 蒸锐
flutter_add_...
Install
47.5K 4.71 jun7572
Flutter Intl
Install
673.1K 4.68 Localizely
Flutter Snip...
Install
479.8K 4.38 George H...
Flutter Enh...
Install
Programming Language
Flutter
Google Plugin homepage
Install
80.0.2
Reviews
Additional Info
Overview
What's New
Support for developing Flutter applications. Flutter gives developers an easy and productive way to build and deploy cross-platform, high-performance mobile apps for both Android and iOS. Installing this plugin will also install the Dart plugin.
For some tools, this plugin uses Chromium through JxBrowser to display content from the web. JxBrowser complies with LGPL and offers an option to replace Chromium with another component. To do this:
• Find the JxBrowser files stored in the plugins directory, under

Code:-

main.dart | pubspec.yaml | pubspec.lock

first_hello > lib > main.dart > MyApp > build

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

Run | Debug | Profile | Codeium: Refactor | Explain | Generate Function Comment | ×
void main() {
  runApp(app: const MyApp());
}

Codeium: Refactor | Explain
class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Codeium: Refactor | Explain | Generate Function Comment | ×
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Hello Bhagyesh',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.blue),
        useMaterial3: true,
      ), // ThemeData
      home: const MyHomePage(title: 'Hello Bhagyesh'),
    ); // MaterialApp
  }
}
```

```dart
class MyHomePage extends StatelessWidget {
  const MyHomePage({super.key, required this.title});

  final String title;

  @override
  Codeium: Refactor | Explain | Generate Function Comment | ×
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Theme.of(context: context).colorScheme.inversePrimary,
        title: Text(data: title),
      ), // AppBar
      body: Container(
        color: const Color(value: 0xFFF75990),
        child: Center(
          child: Text(
            data: 'Hello Bhagyesh',
            style: TextStyle(
              fontSize: 32,
              fontWeight: FontWeight.bold,
              color: Color.fromARGB(a: 255, r: 32, g: 73, b: 194),
            ), // TextStyle
          ), // Text
        ), // Center
      ), // Container
    ); // Scaffold
  }
}
```


Android Emulator - flutter_emulator:5554
11:08
DEBUG
Hello World App
Hello, Himesh!

MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	34
Name	Himesh Pathai
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

EXPERIMENT NO: - 02

Name:- Himesh Pathai Class:- D15A Roll:No: - 34

AIM: - To design Flutter UI by including common widgets.

Theory: -

Understanding Widgets in Flutter

In Flutter, every UI element is a widget, and the entire app structure is a tree of nested widgets. The appearance and functionality of a screen are determined by the selection and arrangement of these widgets. Whenever code changes occur, Flutter recalculates the differences between the previous and updated widgets to apply minimal UI updates efficiently. Since widgets are deeply nested, the root of the app itself is a widget, with every subsequent element also being a widget. These can be responsible for rendering visuals, defining layout structures, handling interactions, and more.

Types of Layout Widgets

Single-Child Layout Widgets

Single-child widgets allow only one child element within the parent widget. These widgets often provide specialized layout functionalities that enhance UI design, improve readability, and optimize development time.

Multi-Child Layout Widgets

Multi-child widgets contain multiple child elements and follow unique layout patterns.

- Row – Arranges child widgets horizontally.
- Column – Arranges child widgets vertically.
- Combination of Row & Column – Enables complex UI structures by nesting these widgets together.

Types of Widgets StatefulWidget

- Maintains state information that can change during the widget's lifecycle.
- Comprises two key classes: the widget itself and a separate state object.
- Uses createState() instead of build(), which returns a class extending Flutter’s State class.
- Examples: Checkbox, Radio, Slider, InkWell, Form, TextField.

StatelessWidget

- Does not hold any state; remains constant throughout its lifecycle.
- Uses the build() method directly to define UI elements.
- Examples: Text, Row, Column, Container.

Commonly Used Widgets

- Container – A box-like widget for layout styling (padding, margins, colors, borders).
- Row & Column – Used for horizontal and vertical alignment of widgets.
- Stack – Overlaps widgets to create layered designs.
- Text – Displays stylized text.
- Image – Loads images from assets, networks, or memory.
- Scaffold – Provides a basic screen layout with an app bar, body, and navigation elements.
- ListView – Creates a scrollable list, ideal for dynamic content.
- GridView – Arranges content in a grid, useful for galleries and dashboards.
- SizedBox – Defines space between elements or sets fixed dimensions.
- ElevatedButton – A button with elevation and customization options.
- TextField – Accepts user text input with keyboard configurations.
- AppBar – A top navigation bar with a title and actions.
- BottomNavigationBar – A bottom navigation panel for switching between app sections.
- Drawer – A side menu panel for navigation.
- Card – A material design component for displaying content in an elevated box.

Code:

main_screen.dart

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

import 'package:get/get.dart';

import 'package:inventory/lumina/src/common_widgets/app_bar.dart';

import 'package:inventory/lumina/src/common_widgets/bottom_navigation_bar.dart';

import 'package:inventory/lumina/src/common_widgets/side_drawer.dart';

import 'package:inventory/lumina/src/features/authentication/controllers/emailcontroller.dart';

import 'package:inventory/lumina/src/features/main_app/dashboard/dashboard_screen.dart';

import 'package:inventory/lumina/src/features/main_app/menu_screen/menu_Screen.dart';

import 'package:inventory/lumina/src/features/main_app/search_screen/search_screen.dart';

import 'package:inventory/lumina/src/features/main_app/more.dart';

import
'package:inventory/lumina/src/features/main_app/transactions_screen/transaction_screen.dart';

import 'package:supabase_flutter/supabase_flutter.dart';


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


  static  List<Widget> _screenOptions = <Widget>[
    Dashboard(),
    SearchScreen(),
    MenuScreen()
   , MoreScreen()
  ];


  @override
  State<MainScreen> createState() => _DashboardState();
}


class _DashboardState extends State<MainScreen> {
  int _selectedIndex = 0;
```

```
final _supabase = Supabase.instance.client;
final Emailcontroller emailGet = Get.put(Emailcontroller());

void _onItemTapped(int index) {
  if (index == 2) {
    // Custom action for the middle "Add" button
     Navigator.of(context).push(MaterialPageRoute(builder: (context)=>TransactionScreen()));
  } else {
    setState(() {
      _selectedIndex = index >= 2 ? index - 1 : index;
    });
  }
}

void naamkaran() async {
  final response = await _supabase
      .from('admins')
      .select()
      .eq('emailid', emailGet.emailget.value);
  final data = response;
  emailGet.Namefrommail.value ;
}

@override
void initState() {
  // TODO: implement initState
  super.initState();
  naamkaran();
}

@override
```

```
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: CustomAppBar(),
      drawer: CustomSideDrawer(),
      body: Center(
        child: MainScreen._screenOptions.elementAt(_selectedIndex),
      ),
      bottomNavigationBar: CustomBottomNavigationBar(
        currentIndex: _selectedIndex >= 2 ? _selectedIndex + 1 : _selectedIndex,
        onTap: _onItemTapped,
      ),
    );
  }
}
```

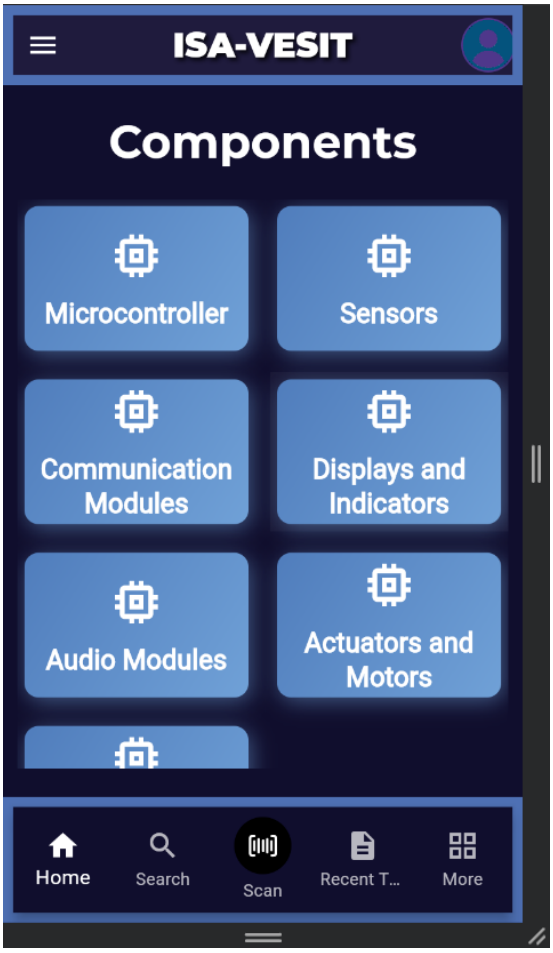
dashboard_screen.dart

```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:inventory/lumina/src/common_widgets/app_bar.dart';
import 'package:inventory/lumina/src/common_widgets/bottom_navigation_bar.dart';
import 'package:inventory/lumina/src/common_widgets/side_drawer.dart';
import 'package:inventory/lumina/src/features/authentication/controllers/emailcontroller.dart';
import 'package:inventory/lumina/src/features/main_app/dashboard/dashboard_screen.dart';
import 'package:inventory/lumina/src/features/main_app/menu_screen/menu_Screen.dart';
import 'package:inventory/lumina/src/features/main_app/search_screen/search_screen.dart';
import 'package:inventory/lumina/src/features/main_app/more.dart';
import
'package:inventory/lumina/src/features/main_app/transactions_screen/transaction_screen.dart';
import 'package:supabase_flutter/supabase_flutter.dart';

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

```
 static  List<Widget> _screenOptions = <Widget>[
   Dashboard(),
   SearchScreen(),
   MenuScreen()
  , MoreScreen()
 ];

 @override
 State<MainScreen> createState() => _DashboardState();
}

class _DashboardState extends State<MainScreen> {
 int _selectedIndex = 0;

 final _supabase = Supabase.instance.client;
 final Emailcontroller emailGet = Get.put(Emailcontroller());

 void _onItemTapped(int index) {
  if (index == 2) {
    // Custom action for the middle "Add" button
     Navigator.of(context).push(MaterialPageRoute(builder: (context)=>TransactionScreen()));
  } else {
    setState(() {
      _selectedIndex = index >= 2 ? index - 1 : index;
    });
  }
 }

 void naamkaran() async {
  final response = await _supabase
      .from('admins')
```

```
      .select()
      .eq('emailid', emailGet.emailget.value);
  final data = response;
  emailGet.Namefrommail.value ;
 }

 @override
 void initState() {
  // TODO: implement initState
  super.initState();



  naamkaran();
 }

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: CustomAppBar(),
   drawer: CustomSideDrawer(),
   body: Center(
    child: MainScreen._screenOptions.elementAt(_selectedIndex),
   ),
   bottomNavigationBar: CustomBottomNavigationBar(
    currentIndex: _selectedIndex >= 2 ? _selectedIndex + 1 : _selectedIndex,
    onTap: _onItemTapped,
   ),
  );
 }
}
```

Output :


ISA-VESIT
Components
Microcontroller
Sensors
Communication Modules
Displays and Indicators
Audio Modules
Actuators and Motors
Home
Search
Scan
Recent T...
More

MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	34
Name	Himesh Pathai
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

EXPERIMENT NO: - 03

Name:- Himesh Pathai Class:- D15A Roll:No: - 34

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

Theory: -

Incorporating Visual Elements in Flutter: Icons, Images, and Custom Fonts

Flutter is a powerful open-source UI framework that enables developers to build natively compiled applications for mobile, web, and desktop platforms—all from a single codebase. One of Flutter's greatest strengths is its flexibility in crafting highly customizable UIs.

This practical guide focuses on integrating essential visual elements—icons, images, and custom fonts—into a Flutter application. These elements enhance visual appeal, improve usability, and create a more engaging user experience.

Importance of Visual Elements in App Development

- Enhanced User Experience – Icons and images make applications more visually appealing and userfriendly.
- Efficient Communication – Well-designed icons convey information quickly, reducing the need for lengthy text.
- Brand Identity – Custom icons and images reinforce branding, making an app more memorable.

Managing Assets in a Flutter App

When a Flutter app is built, it consists of both code and assets. Assets include static files such as images, icons, fonts, and configuration files, which are deployed and available at runtime. Flutter supports multiple image formats, including JPEG, WebP, PNG, GIF, BMP, and WBMP.

Adding Icons in Flutter

Flutter provides built-in Material Design icons through the Icons class. Custom icons can also be integrated using third-party packages like flutter_launcher_icons and font_awesome_flutter. Example (Built-in Material Icons):

Code :
Main-app/dashboard

```
import 'package:flutter/material.dart';
import 'package:get/get.dart'; // Get for dark mode support
import 'package:inventory/lumina/src/features/authentication/controllers/componentController.dart';
import 'package:inventory/lumina/src/features/main_app/dashboard/classScreen.dart';

class ClassContainer extends StatelessWidget {
  final String label;

  ClassContainer({required this.label});

  final ComponentController componentController =
      Get.put(ComponentController());

  @override
  Widget build(BuildContext context) {
    bool isDarkMode = Get.isDarkMode;

    return InkWell(
      onTap: () {
        componentController.ClassName.value = label;
        Navigator.of(context).push(
          MaterialPageRoute(builder: (context) => Classscreen(title: label)),
        );
      },
      splashColor: const Color(0xFFD3DCF2),
      child: Container(
        margin: const EdgeInsets.all(5),
        padding: const EdgeInsets.symmetric(horizontal: 10),
        decoration: BoxDecoration(
          borderRadius: BorderRadius.circular(10),
          gradient: isDarkMode
              ? const LinearGradient(
                  colors: [Color(0xFF507DBC), Color(0xFF70A1D7)],
                  begin: Alignment.topLeft,
                  end: Alignment.bottomRight,
                )
              : const LinearGradient(
                  colors: [
                    Color.fromARGB(255, 154, 210, 255),
                    Color.fromARGB(255, 213, 245, 252),
                  ],
                  begin: Alignment.topLeft,
                  end: Alignment.bottomRight,
                ),
          boxShadow: [
            BoxShadow(
              color: isDarkMode
                  ? Color(0xFF70A1D7).withOpacity(0.5)
```

```
                : Colors.black.withOpacity(0.2),
            offset: Offset(2, 3),
            blurRadius: isDarkMode ? 12 : 6,
          ),
        ],
      ),
      child: Column(
        mainAxisAlignment:
            MainAxisAlignment.center, // Center contents vertically
        crossAxisAlignment:
            CrossAxisAlignment.center, // Center contents horizontally
        children: [
          // Centered Icon
          Icon(
            Icons.memory,
            color: isDarkMode ? Colors.white : Colors.black,
            size: 40, // Adjust size as needed
          ),
          const SizedBox(height: 8), // Spacing between icon and label
          // Label at the bottom
          Text(
            label,
            style: Theme.of(context).textTheme.titleLarge?.copyWith(
                  color: isDarkMode ? Colors.white : Colors.black,
                  fontWeight: FontWeight.bold,
                  fontSize: 19,
                ),
            textAlign: TextAlign.center, // Center text
            maxLines: 2, // Allow up to two lines
            overflow: TextOverflow.ellipsis, // Show "..." if text is too long
          )
        ],
      ),
    ),
  );
 }
}
```

Profile Screen:

```
import 'dart:io';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:image_picker/image_picker.dart';
import 'package:inventory/lumina/src/features/authentication/controllers/emailcontroller.dart';
import 'package:inventory/lumina/src/features/authentication/screens/log_out_widget.dart';
import 'package:qr_flutter/qr_flutter.dart';
import 'package:supabase_flutter/supabase_flutter.dart';
import 'dart:convert';
import 'package:inventory/lumina/src/features/authentication/screens/verify_old_pass.dart';

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

  @override
  State<ProfileScreen> createState() => _ProfileScreenState();
}

class _ProfileScreenState extends State<ProfileScreen> {
  final _supabase = Supabase.instance.client;
  final Emailcontroller emailGet = Get.put(Emailcontroller());
  File? _image;
  final ImagePicker _picker = ImagePicker();

  @override
  void initState() {
    super.initState();
    naamkaran();
    _loadImage();
  }

  Future<void> naamkaran() async {
    while (emailGet.emailget.value.isEmpty) {
      print("Waiting for email to be set...");
      await Future.delayed(Duration(milliseconds: 200)); // Small wait loop
    }

    try {
      final response = await _supabase
          .from('Members')
          .select()
          .eq('Email Id', emailGet.emailget.value)
          .maybeSingle();

      if (response != null) {
        emailGet.Namefrommail.value = response['Name'] ?? 'Guest';
        emailGet.idfrommail.value = response['ISA Login ID'] ?? '';
```

```
      emailGet.qrData.value = jsonEncode({
        'member_id': response['ISA Login ID'] ?? '',
        'email_id': response['Email Id'] ?? '',
        'division': response['Division'] ?? '',
        'name': response['Name'] ?? 'Guest',
        'phone_number': response['Phone Number'] ?? '',
      });

      print("User data fetched successfully!");
    } else {
      print("No user data found.");
    }
  } catch (e) {
    print('Error in naamkaran: $e');
  }
}

Future<void> _loadImage() async {
  final user = _supabase.auth.currentUser;
  if (user == null) return;

  try {
    // Query the database for the profile image URL
    final response = await _supabase
        .from('profiles')
        .select('profile_image_url')
        .eq('id', user.id)
        .maybeSingle();

    // Check if response contains data
    if (response != null && response['profile_image_url'] != null) {
      final imageUrl = response['profile_image_url'] as String;
      emailGet.profileImageUrl.value =
          '$imageUrl?timestamp=${DateTime.now().millisecondsSinceEpoch}';
    } else {
      // No profile image URL found, set default image
      emailGet.profileImageUrl.value =
          ''; // Or assign a default image URL if you have one
    }
  } catch (e) {
    print('Error loading profile image: $e');
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('Failed to load profile image.')),
    );
  }
}

Future<void> _pickImage() async {
  final user = _supabase.auth.currentUser;
  if (user == null) {
    ScaffoldMessenger.of(context).showSnackBar(
```

```
    const SnackBar(content: Text('No user is currently logged in.')),
   );
   return;
  }

  final pickedFile = await _picker.pickImage(source: ImageSource.gallery);

  if (pickedFile != null) {
   File file = File(pickedFile.path);

   try {
    // First, get the ISA Login ID from Members table
    final memberResponse = await _supabase
      .from('Members')
      .select('"ISA Login ID"')
      .eq('Email Id', emailGet.emailget.value)
      .single();

    final String isaLoginId = memberResponse['ISA Login ID'] as String;

    // Upload the file with the same filename
    String fileName = 'profile_${user.id}.jpg';

    // Delete the old file (optional but recommended)
    await _supabase.storage.from('profile-images').remove([fileName]);

    // Upload the new file
    await _supabase.storage.from('profile-images').upload(fileName, file);

    // Get the public URL
    final imageUrl =
      _supabase.storage.from('profile-images').getPublicUrl(fileName);

    // Update the database with both image URL and member_id
    await _supabase.from('profiles').upsert({
     'id': user.id,
     'profile_image_url': imageUrl,
     'member_id': isaLoginId, // Adding the member_id from ISA Login ID
    });

    // Update the observable and invalidate the cache by appending a timestamp
    emailGet.profileImageUrl.value =
      '$imageUrl?timestamp=${DateTime.now().millisecondsSinceEpoch}';

    setState(() {
     _image = file;
    });

    ScaffoldMessenger.of(context).showSnackBar(
     const SnackBar(content: Text('Profile image updated successfully!')),
    );
```

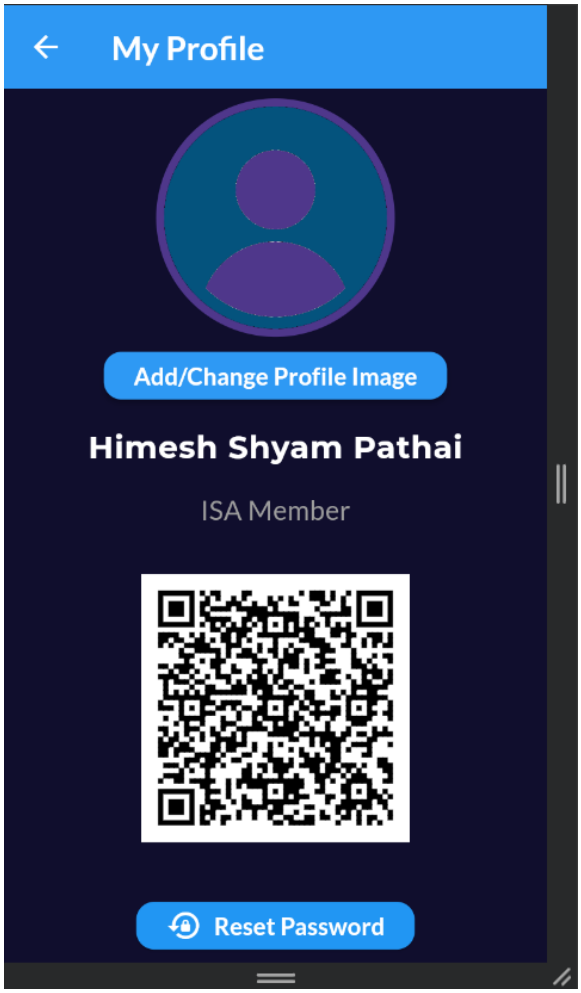
```
    } catch (e) {
      print('Error: $e');
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('An unexpected error occurred.')),
      );
    }
  }
}

@override
Widget build(BuildContext context) {
  final theme = Theme.of(context);
  bool isDarkMode = Get.isDarkMode;

  return Scaffold(
    appBar: AppBar(
      backgroundColor: const Color(0xff2196F3),
      title: Text(
        'My Profile',
        style: GoogleFonts.lato(
          fontSize: 22,
          fontWeight: FontWeight.bold,
          color: Colors.white,
        ),
      ),
      leading: IconButton(
        icon: const Icon(Icons.arrow_back, color: Colors.white),
        onPressed: () => Navigator.pop(context),
      ),
    ),
    body: SingleChildScrollView(
      child: Padding(
        padding: const EdgeInsets.symmetric(horizontal: 20.0, vertical: 30.0),
        child: Column(
          children: [
            Center(
              child: Obx(() {
                String profileImageUrl = emailGet.profileImageUrl.value;
                return CircleAvatar(
                  backgroundImage: profileImageUrl.isNotEmpty
                      ? NetworkImage(profileImageUrl) as ImageProvider
                      : const AssetImage(
                          "assets/images/default_profile_image.png"),
                  radius: 80,
                );
              }),
            ),
            const SizedBox(height: 10),
            ElevatedButton(
              onPressed: _pickImage,
              style: ElevatedButton.styleFrom(
```

```
    backgroundColor: const Color(0xff2196F3),
    shape: RoundedRectangleBorder(
     borderRadius: BorderRadius.circular(12),
    ),
    padding: const EdgeInsets.symmetric(
     horizontal: 20,
     vertical: 12,
    ),
   ),
   child: Text(
    'Add/Change Profile Image',
    style: GoogleFonts.lato(
     color: Colors.white,
     fontSize: 16,
     fontWeight: FontWeight.w600,
    ),
   ),
  ),
 ),
 const SizedBox(height: 5),
 Obx(
  () {
    String name = emailGet.Namefrommail.value.isNotEmpty
        ? emailGet.Namefrommail.value
        : 'Guest';
    return Container(
     height: 55,
     width: 500,
     child: Center(
        child: Text(
       name,
       style: theme.textTheme.bodyLarge?.copyWith(
        fontSize: 21,
        fontWeight: FontWeight.bold,
        color: isDarkMode ? Colors.white : Colors.black,
       ),
      )),
    );
  },
 ),
 const SizedBox(height: 1),
 Text(
  "ISA Member",
  style: GoogleFonts.lato(
   fontSize: 18,
   fontWeight: FontWeight.w400,
   color: Colors.grey,
  ),
 ),
 const SizedBox(height: 30),
 Obx(
  () => QrImageView(
```

```
      data: emailGet.qrData.value,
      version: QrVersions.auto,
      size: 180.0,
      backgroundColor: Colors.white,
      padding: const EdgeInsets.all(10),
    ),
  ),
  const SizedBox(height: 40),
  ElevatedButton.icon(
    onPressed: () {
      final userEmail = emailGet.emailget.value;
      if (userEmail.isNotEmpty) {
        Navigator.push(
          context,
          MaterialPageRoute(
            builder: (context) => VerifyOldPasswordScreen(
              email: userEmail,
            ),
          ),
        );
      } else {
        ScaffoldMessenger.of(context).showSnackBar(
          const SnackBar(
            content: Text('No email found. Please try again.'),
          ),
        );
      }
    },
    style: ElevatedButton.styleFrom(
      backgroundColor: const Color(0xff2196F3),
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(12),
      ),
      padding: const EdgeInsets.symmetric(
        horizontal: 20,
        vertical: 12,
      ),
    ),
    icon: const Icon(Icons.lock_reset, color: Colors.white),
    label: Text(
      'Reset Password',
      style: GoogleFonts.lato(
        color: Colors.white,
        fontSize: 16,
        fontWeight: FontWeight.w600,
      ),
    ),
  ),
  const SizedBox(height: 20),
  const LogOutWidget(),
],
```

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

ScreenShots:



MAD & PWA Lab

Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	34
Name	Himesh Pathai
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

EXPERIMENT NO: - 04

Name:- Himesh Pathai Class:- D15A Roll:No: - 34

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

Forms in Flutter

Forms in Flutter are essential components used to collect and manage user input efficiently. They are widely used in login screens, registration pages, and feedback forms. Flu er provides a Form widget that works alongside TextFormField and other input elements, offering features such as validation, error handling, and state management to improve user experience.

Key Components of a Form in Flu er

1.Form Widget

The Form widget acts as a container that groups multiple input fields and manages their validation. Requires a GlobalKey<FormState> to uniquely identify the form and interact with it.

- Helps in structuring form fields and handling user input efficiently.

2.Form Fields (TextFormField)

The TextFormField widget is used for user input, such as entering names, emails, or phone numbers.

- It supports input validation using the validator property.
- Allows customization with InputDecoration (e.g., labels, icons, borders, hint text).
- Different TextInputType op ons can be set for appropriate keyboard input (e.g., TextInputType.emailAddress for emails).

3.Validation in Forms

Ensuring valid user input is crucial. The validator property in TextFormField helps check whether the entered data meets specified criteria before submission.

- Valida on can be triggered manually using formKey.currentState!.validate().
- The autovalidateMode property can enable automatic validation during user input.

4. State Management in Forms

To ensure data persistence and processing, proper state management is required.

- The FormState class provides methods like validate(), save(), and reset() to manage form behavior.
- The save() method stores user input when validation is successful.
- The reset() method clears the form fields and restores the initial state.

5.Submit Button

A submit button is necessary to trigger form validation and submit user data.

When pressed, it checks validation using formKey.currentState!.validate().

If validation succeeds, the form data is saved and processed accordingly.

Important Properties & Methods of Form Widget

Proper es

key → A GlobalKey<FormState> that uniquely identifies the form.

child → Contains form fields, typically wrapped in a Column or ListView.

autovalidateMode → Defines when the form should auto-validate.

Methods

validate() → Checks if all form fields are valid and returns true or false.

save() → Stores the current values of form fields a er successful validation.

reset() → Clears user input and resets the form to its initial state.

Code :

```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:inventory/lumina/src/fea-
tures/main_app/main_screen/main_screen.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'email_input_screen.dart';
import 'package:inventory/lumina/src/features/authentication/controllers/emailcontrol-
ler.dart';
import 'package:supabase_flutter/supabase_flutter.dart';

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

  @override
  State<LoginForm> createState() => _LoginFormState();
}

class _LoginFormState extends State<LoginForm> {
  final supabase = Supabase.instance.client;

  final TextEditingController emailcontroller = TextEditingController();
  final TextEditingController passwordcontroller = TextEditingController();

  final Emailcontroller emailGet = Get.put(Emailcontroller());

  bool isTermsAccepted = false;

  Future<void> emailsignin() async {
    if (!isTermsAccepted) {
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(
            content:
                Text("Please accept the terms and conditions to proceed.")),
      );
      return;
    }

    try {
      final response = await supabase.auth.signInWithPassword(
        email: emailcontroller.text,
        password: passwordcontroller.text,
      );

      if (response.user != null) {
        emailGet.emailget.value = emailcontroller.text;
```

```
      emailcontroller.clear();
      passwordcontroller.clear();
      Get.to(MainScreen(), transition: Transition.fade);
      final session = response.session;
      await supabase.auth.setSession(session as String);
      final prefs = await SharedPreferences.getInstance();
      prefs.setString('email', emailGet.emailget.value);
    }
  } on AuthException catch (e) {
    // Handle Supabase AuthException explicitly
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text(e.message)),
    );
  } on Exception catch (e) {
    // Handle other exceptions
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(
          content: Text("Something went wrong! Please try again.")),
    );
  }
}

@override
void initState() {
  super.initState();

  supabase.auth.refreshSession().then((session) {
    if (session != null) {
      emailGet.emailget.value = session.user!.email!;
      emailGet.mailchecker();
    }
  });
}

TextStyle termsTextStyle = TextStyle(
  fontSize: 16.0,
  color: Colors.black,
);

@override
Widget build(BuildContext context) {
  return Form(
    child: Container(
      padding: EdgeInsets.symmetric(vertical: 20),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          TextFormField(
            controller: emailcontroller,
```

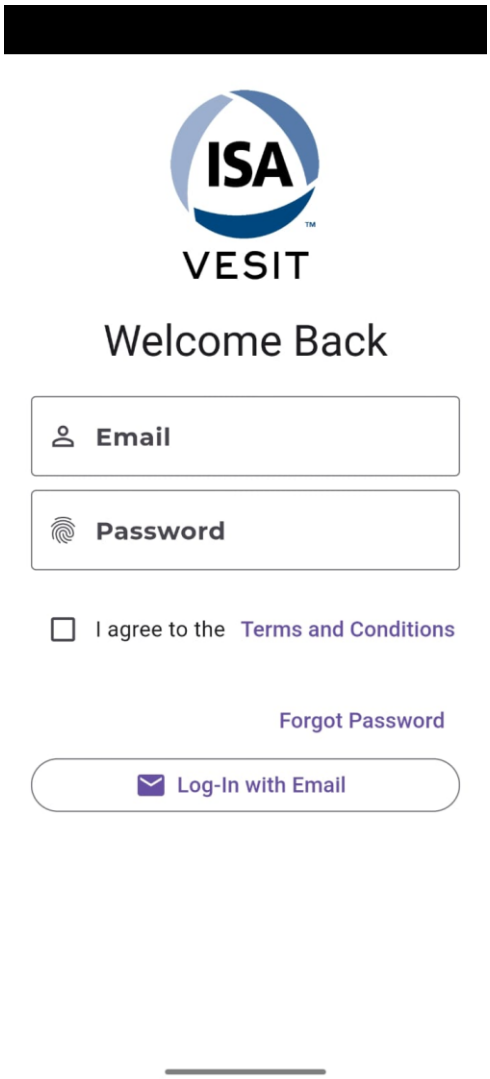
```
  decoration: const InputDecoration(
      prefixIcon: Icon(Icons.person_outline_outlined),
      labelText: "Email",
      hintText: "Email",
      border: OutlineInputBorder()),
),
SizedBox(
  height: 10,
),
TextFormField(
  obscureText: true,
  controller: passwordcontroller,
  decoration: const InputDecoration(
      prefixIcon: Icon(Icons.fingerprint),
      labelText: "Password",
      hintText: "Password",
      border: OutlineInputBorder()),
),
const SizedBox(height: 20),
Row(
  children: [
    Checkbox(
      value: isTermsAccepted,
      onChanged: (bool? value) {
        setState(() {
          isTermsAccepted = value ?? false;
        });
      },
    ),
    const Text('I agree to the'),
    TextButton(
      onPressed: () {
        _showTermsAndConditions();
      },
      child: Text("Terms and Conditions"),
    ),
  ],
),
const SizedBox(height: 20),
Align(
  alignment: Alignment.centerRight,
  child: TextButton(
    onPressed: () {
      Navigator.push(
        context,
        MaterialPageRoute(
            builder: (context) => const ForgotPasswordScreen()),
      );
    },
```

```
            child: const Text("Forgot Password"),
          ),
        ),
        Padding(
          padding:
              const EdgeInsets.symmetric(vertical: 20.0, horizontal: 10.0),
          child: SizedBox(
            width: double.infinity,
            child: Container(
              decoration: BoxDecoration(
                gradient: const LinearGradient(
                  colors: [Color(0xFF507DBC), Color(0xFF70A1D7)],
                  begin: Alignment.topLeft,
                  end: Alignment.bottomRight,
                ),
                borderRadius: BorderRadius.circular(
                    16),
              ),
              child: OutlinedButton.icon(
                style: OutlinedButton.styleFrom(
                  padding: const EdgeInsets.symmetric(
                      vertical: 16.0,
                      horizontal: 24.0),
                  side: BorderSide(
                      color: Colors
                          .transparent),
                  shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.circular(
                          16)),
                ),
                icon: Icon(Icons.email, color: Colors.white),
                onPressed: () {
                  emailsignin();
                },
                label: const Text(
                  "Log-In with Email",
                  style: TextStyle(color: Colors.white, fontSize: 18.0),
                ),
              ),
            ),
          ),
        )
      ],
    ),
  ),
 );
}

void _showTermsAndConditions() {
```

```
    bool isDarkMode = Theme.of(context).brightness == Brightness.dark;
    showModalBottomSheet(
      context: context,
      builder: (context) => Container(
        padding: const EdgeInsets.all(20),
        child: SingleChildScrollView(
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Text(
                "Terms and Conditions",
                style: Theme.of(context).textTheme.headlineMedium,
              ),
              SizedBox(height: 10),
              Text(
                '1. Objective\n'
                'The purpose of these guidelines is to provide a clear understanding of the rules and procedures for the ISA-VESIT inventory system.\n\n'
                '2. Issuance of Components\n'
                '● Eligibility: Only students who are currently enrolled for ISA Memberships are eligible to borrow components.\n'
                '● Issuance Procedure:\n'
                '  - Components will be issued based on availability and necessity.\n'
                '  - A record of issued components will be maintained by the ISA Council.\n\n'
                '3. Student Responsibilities\n'
                '● Care: Students are responsible for the proper care and handling of the components.\n'
                '● Usage: Components must be used only for their intended educational or project purposes.\n'
                '● Return: Components must be returned by the due date specified at the time of issuance.\n'
                '● Modification: Modification of the components is not allowed.\n'
                '● Damage: No damage is allowed. Students will have to pay the entire amount if the component is damaged.\n'
                '● Loss: If a component is lost, the student is responsible for it and has to pay the entire amount of the component as listed below.\n'
                '● Issuance/Reissuance: Components must be issued or reissued in the presence of and with the approval of a council member only.\n'
                '● Reissue: The component should be reissued within 1 month of time after issuing the component.\n'
                '● Timing: For issuance/reissuance of the component the timings are 1) 1:00 pm - 1:30 pm 2) 3:30 pm - 4:00 pm\n\n'
                '4. Return Policy\n'
                '● Due Date: Components must be returned by the due date specified during issuance.\n'
                '● Condition: Components must be returned in the same condition as they were issued.\n'
                '● All the components should be returned to the council before the end semester exam.\n'
```

```
            '● Refer Fine Structure for more terms and conditions regarding the fine pay-
ment.\n\n'
            '5. Fine Structure and Payment\n'
            '● Late Returns:\n'
            '  - A fine will be imposed for each day the component is returned late.\n'
            '  - The maximum late fine will not exceed 2500 Rs.\n'
            '● Component Damage:\n'
            '  - No damage to the components would be accepted.\n'
            '  - In case of damage: Replacement Cost of the new component.\n'
            '● Loss of Component:\n'
            '  - Full replacement cost of the component will be charged.\n'
            '● Payment:\n'
            '  - Fines must be paid within 5 days of notification.\n'
            '  - Payment should be made online.\n\n'
            '6. Consequences of Non-Payment\n'
            '● Failure to pay fines may result in:\n'
            '  - Suspension of borrowing privileges.\n'
            '  - Membership Suspension.\n'
            '  - Clearance for collecting Leaving Certificate would not be provided by the
Central Library of College.\n\n'
            '7. Dispute Resolution\n'
            '● Students who wish to dispute a fine may do so by submitting a written ap-
peal to the ISA committee within 3 days of fine notification.\n'
            '● The decision of the ISA committee will be final.\n\n'
            '8. Policy Review\n'
            '● This policy will be reviewed annually and is subject to change. Updates
will be communicated to all students via email and WhatsApp.\n\n'
            '9. Component Price List:\n'
            'https://bit.ly/ComponentPrice\n'
            'Note: Subject to Change of Price\n\n'
            '10. Contact Information\n'
            'For any questions or concerns regarding this policy, please contact:\n'
            'Sr. Treasurer: Atishkar Singh\n'
            'Phone No: 9049120954',
            style: termsTextStyle.copyWith(
              color: isDarkMode
                  ? Colors.white
                  : Colors.black), // Set color to white
          ),
        ],
      ),
    ),
  ),
 );
}
```

Output :


ISA
VESIT
Welcome Back
Email
Password
I agree to the Terms and Conditions
Forgot Password
Log-In with Email

MAD & PWA Lab

Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	34
Name	Himesh Pathai
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

EXPERIMENT NO: - 05

Name:- Himesh Pathai Class:- D15A Roll:No: - 34

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

Theory: -

Navigation, Routing, and Gesture Handling in Flutter

In Flutter, screens or pages are referred to as routes, and each route is essentially a widget. This concept is similar to Activities in Android. Navigating between pages defines an app's workflow, and the mechanism for handling this is known as routing.

Flutter provides a built-in routing system using MaterialPageRoute, along with the Navigator.push() and Navigator.pop() methods to move between routes.

Additionally, gestures allow apps to respond to user interactions like taps, swipes, and drags, making applications more dynamic and user-friendly.

Navigation and Routing in Flutter

1. Using the Navigator Widget

Flutter's Navigator widget manages a stack of routes, enabling seamless navigation between screens.

- Pushing a Route: Moves to a new screen using Navigator.push().
- Popping a Route: Returns to the previous screen using Navigator.pop().

Example:

```
ElevatedButton(
  onPressed: () {
    Navigator.push(
      context,
      MaterialPageRoute(builder: (context) => SecondScreen()),
    );
  },
  child: Text('Go to Second Screen'),
);
```

---

2. Using Named Routes

For larger applications, named routes provide a cleaner and more structured way to manage navigation.

Step 1: Define Routes in MaterialApp

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

Step 2: Navigate Using Navigator.pushNamed()

```
Navigator.pushNamed(context, '/second');
```

Handling Gestures in Flutter

Gestures enable user interaction through taps, swipes, pinches, and drags. Flutter provides various widgets and gesture detectors to manage these interactions effectively.

1. Tap Gestures

Taps are one of the most common interactions and can be handled using:

- GestureDetector
- InkWell
- ElevatedButton

Example (Tap Gesture using GestureDetector):

```
GestureDetector(
  onTap: () {
    print("Tapped!");
  },
  child: Container(
    padding: EdgeInsets.all(20),
    color: Colors.blue,
    child: Text('Tap Me'),
  ),
);
```

2. Long Press Gestures

Long-press interactions can be captured using the onLongPress callback in GestureDetector or InkWell.

```
InkWell(
  onLongPress: () {
    print("Long Pressed!");
  },
  child: Container(
    padding: EdgeInsets.all(20),
    color: Colors.red,
    child: Text('Long Press Me'),
  ),
);
```

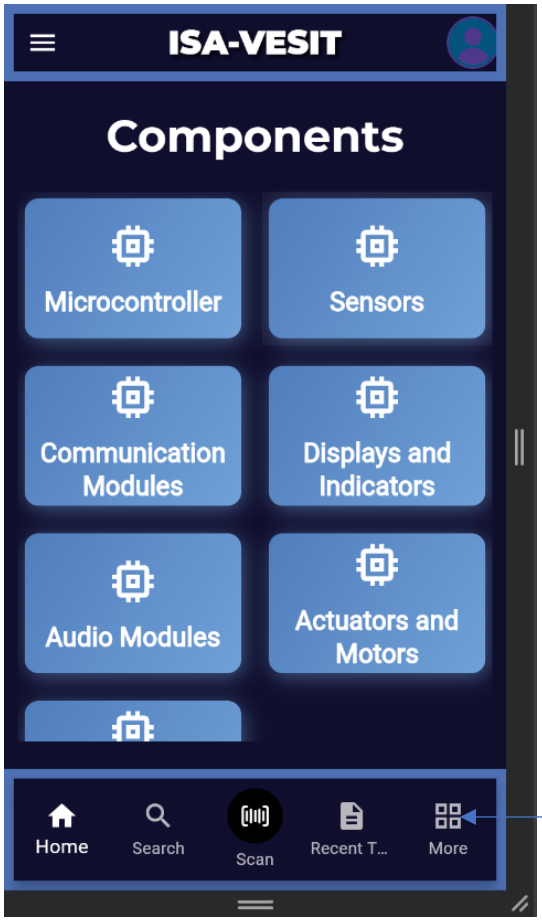
---

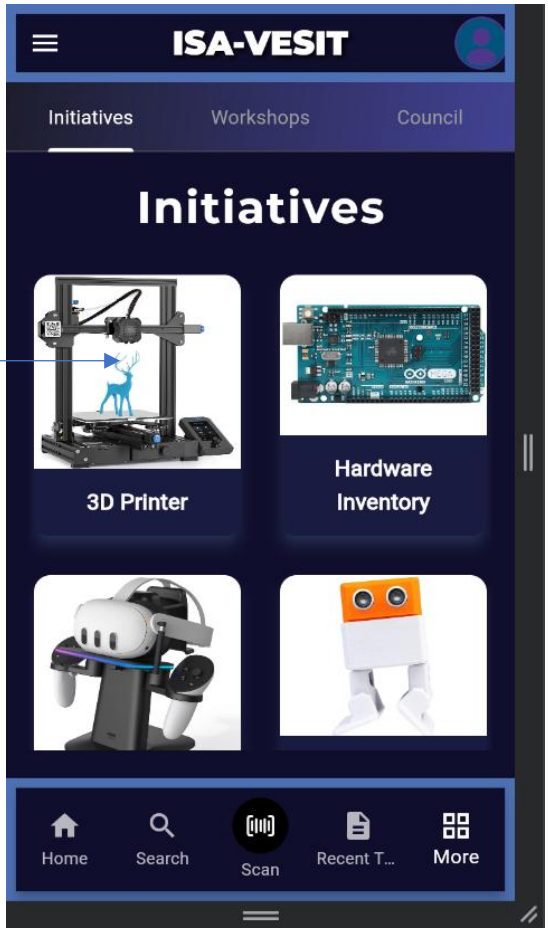
3. Swipe and Drag Gestures

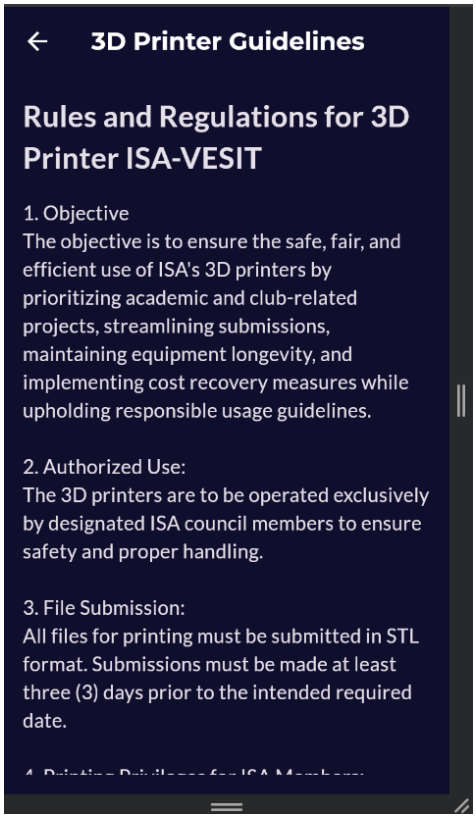
Flutter provides built-in methods like onHorizontalDragUpdate and onVerticalDragUpdate to detect swipe and drag actions.

Example (Swipe Detection):

```
GestureDetector(
  onHorizontalDragUpdate: (details) {
    if (details.primaryDelta! > 0) {
      print("Swiped Right!");
    } else {
      print("Swiped Left!");
    }
  },
  child: Container(
    padding: EdgeInsets.all(20),
    color: Colors.green,
    child: Text('Swipe Me'),
  ),
);
```


ISA-VESIT
Components
Microcontroller
Sensors
Communication Modules
Displays and Indicators
Audio Modules
Actuators and Motors
Home
Search
Scan
Recent T...
More


ISA-VESIT
Initiatives
Workshops
Council
Initiatives
3D Printer
Hardware Inventory
Home
Search
Scan
Recent T...
More


3D Printer Guidelines
Rules and Regulations for 3D Printer ISA-VESIT
1. Objective
The objective is to ensure the safe, fair, and efficient use of ISA's 3D printers by prioritizing academic and club-related projects, streamlining submissions, maintaining equipment longevity, and implementing cost recovery measures while upholding responsible usage guidelines.
2. Authorized Use:
The 3D printers are to be operated exclusively by designated ISA council members to ensure safety and proper handling.
3. File Submission:
All files for printing must be submitted in STL format. Submissions must be made at least three (3) days prior to the intended required date.

MAD & PWA Lab

Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	34
Name	Himesh Pathai
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	

EXPERIMENT NO: - 06

Name:- Himesh Pathai Class:- D15A Roll:No: - 34

AIM: - To connect Flutter UI with Firebase database.

Theory: -

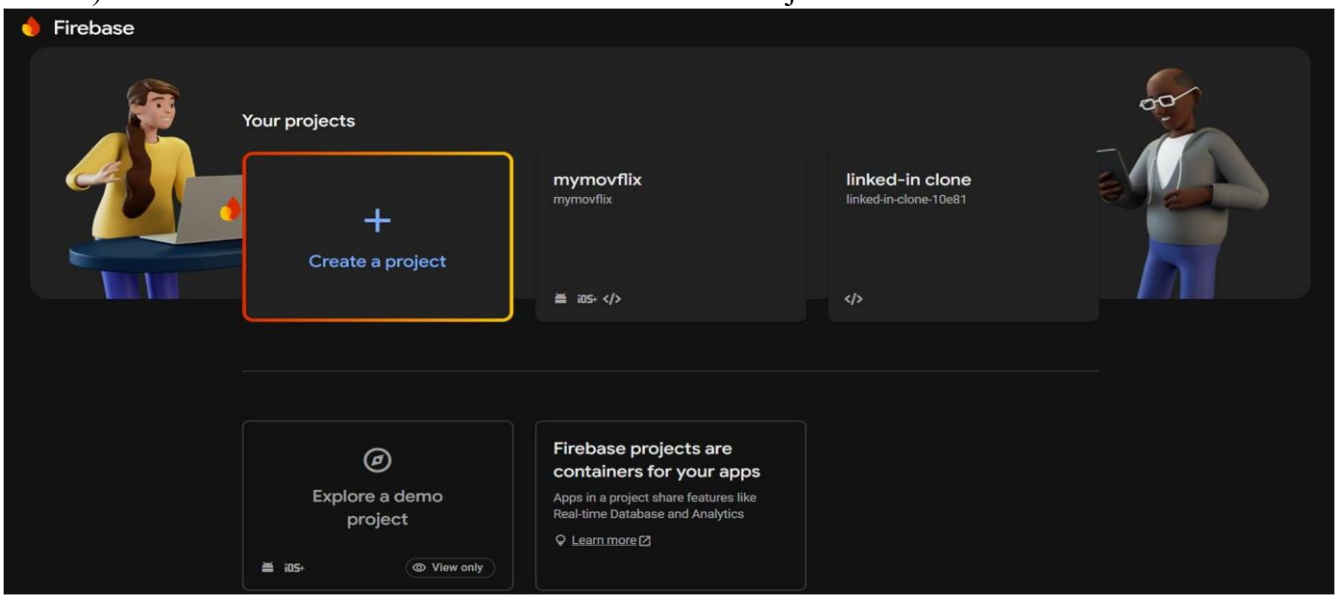
Flutter is an open-source UI toolkit developed by Google for building natively compiled applications for mobile, web, and desktop from a single codebase. Firebase, a Backend-as-a-Service (BaaS) platform, provides real-time database, authentication, and cloud storage services, making it a powerful backend 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



1.2) Click on Create a Project and give it a suitable name.

Step 2:- Add Firebase to Your Flutter App

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

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

```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:-

Sign_in_page.dart

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

import 'sign_up_page.dart'; import
'package:movflix/screens/homescreen.dart';
import
'package:movflix/widgets/bottom_bar_nav.dart';

class SignInPage extends StatefulWidget {
  @override
  _SignInPageState createState() =>
_SignInPageState();
}

class _SignInPageState
extends State<SignInPage> {
final FirebaseAuth _auth =
FirebaseAuth.instance;
  final FirebaseFirestore _firestore =
FirebaseFirestore.instance;
  final TextEditingController _emailController =
TextEditingController();   final
TextEditingController
_passwordController = TextEditingController();
String _errorMessage = "";   bool _isLoading =
false;

  Future<void>      _signIn()      async      {
setState(() {
     _isLoading = true;
     _errorMessage = "";
   });

   try {
     // Firebase Authentication
       email: _emailController.text.trim(),
       password: _passwordController.text.trim(),
     );

     User? user = userCredential.user;
if (user != null) {
       // Ensure user document exists
before updating        var userDoc = await
_firestore.collection("users").doc(user.uid).get()
;
       if (userDoc.exists) {
await
_firestore.collection("users").doc(user.uid).upda
te({
          "lastLogin":
FieldValue.serverTimestamp(),
        });
       }

       Navigator.pushAndRemoveUntil(
context,
        MaterialPageRoute(builder: (context) =>
HomeScreen()),
           (route) => false,
       );
       Navigator.of(context).pushReplacement(
MaterialPageRoute(builder: (context) => const
BottomNavBar()),
       );
     }
   } on FirebaseAuthException catch (e) {
setState(() {
       _errorMessage = e.message ?? "Error
signing in";
       _isLoading = false;
     });
   }        finally        {
setState(() {
       _isLoading = false;
     });
   }
```

```
}


  @override
  Widget build(BuildContext context) {
return Scaffold(       body: Container(
padding: EdgeInsets.symmetric(horizontal:
15, vertical: 15),
child: Column(
children: [
          _headerWidget(),
SizedBox(
height: 10,
          ),
          _formWidget(),
        ],
      ),
    ),
  );
 }

  Widget _headerWidget() {
return Row(       children: [
InkWell(          onTap: () {
           Navigator.pop(context);
         },
         child: Icon(Icons.arrow_back),
       ),
       SizedBox(
width: 10,
       ),
       Container(          height: 40,
child: Image.asset('assets/logo.png'),
       )
     ],
  );
 }

  Widget _formWidget() {
return Expanded(
child: Column(
mainAxisAlignment:
MainAxisAlignment.center,
      children: [
Container(
padding:
EdgeInsets.symmetric(horizontal: 8),
decoration: BoxDecoration(
color: Colors.grey[800],
           borderRadius: BorderRadius.all(
             Radius.circular(5),
           ),
         ),
         child: TextFormField(
controller: _emailController,
decoration: InputDecoration(
labelStyle: TextStyle(fontSize: 14, color:
Colors.white),              border:
InputBorder.none,              labelText: "Email
or phone number",
           ),
         ),
       ),
       SizedBox(
height: 10,
       ),
       Container(
padding:
EdgeInsets.symmetric(horizontal: 8),
decoration: BoxDecoration(
color: Colors.grey[800],
           borderRadius: BorderRadius.all(
             Radius.circular(5),
           ),
         ),
         child: TextFormField(
           controller: _passwordController,
obscureText: true,              decoration:
InputDecoration(              labelStyle:
TextStyle(fontSize: 14, color: Colors.white),
border: InputBorder.none,
labelText: "Password",
           ),
         ),
       ),
       SizedBox(
height: 15,
```

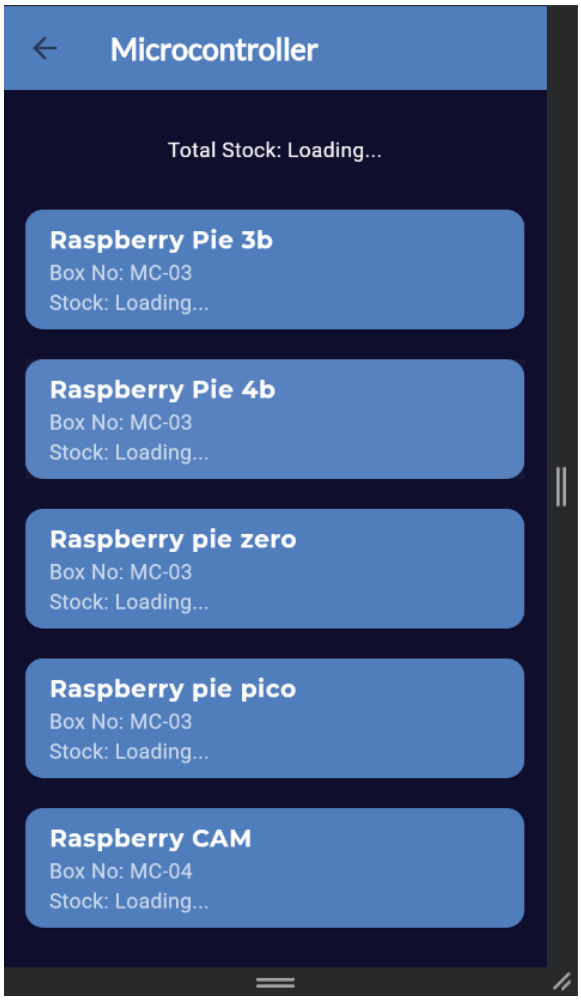
```
        InkWell(            onTap: _isLoading
? null : _signIn,            child: Container(
alignment: Alignment.center,
padding:
EdgeInsets.symmetric(vertical: 15),
width: double.maxFinite,
decoration: BoxDecoration(
color: _isLoading ? Colors.grey :
Colors.transparent,                border:
Border.all(                color:
Colors.grey[600] ??
Colors.grey, width: 2),
            ),
            child: _isLoading
                ? CircularProgressIndicator(color:
Colors.white)
                : Text("Sign In"),
          ),
        ),
        if (_errorMessage.isNotEmpty)
Padding(              padding: const
EdgeInsets.all(8.0),              child: Text(
_errorMessage,              style:
TextStyle(color: Colors.red),
            ),
          ),
        SizedBox(
height: 15,
        ),
        Text("Need Help?"),
        SizedBox(
          height: 15,
        ),
        GestureDetector(
onTap: () {
Navigator.push(
context,
              MaterialPageRoute(builder: (context)
=> SignUpPage()),
            );
},            child:
Text(
              "New to Netflix? Sign up now.",
style: TextStyle(fontWeight:
FontWeight.bold, color: Colors.blue),
            ),
        ),
        SizedBox(
height: 20,
        ),
        Text(
          "Sign-in is protected by Google reCAPTCHA
to ensure you're not a bot. Learn more.",
          style: TextStyle(
fontSize: 12,
          ),
          textAlign: TextAlign.center,
        )
      ],
    ),
  );
 }
}
```

ign_up_page.dart

```
import 'package:flutter/material.dart';
import 'package:flutter/rendering.dart';
import
'package:movflix/widgets/header_widget.da
rt';
import 'sign_in_page.dart';
import
'package:movflix/screens/homescreen.dart';
import
'package:movflix/widgets/bottom_bar_nav.d
art';
import
'package:firebase_auth/firebase_auth.dart';
import
'package:cloud_firestore/cloud_firestore.dart
';

class SignUpPage extends StatefulWidget {
  @override
  _SignUpPageState createState() =>
_SignUpPageState();
}

class _SignUpPageState
extends State<SignUpPage> {
final FirebaseAuth _auth =
FirebaseAuth.instance;
  final FirebaseFirestore _firestore
= FirebaseFirestore.instance;   final
TextEditingController
_emailController = TextEditingController();
final TextEditingController
_passwordController =
TextEditingController();
bool _isCheck = false;
String _errorMessage = "";
  bool _isLoading = false;

  Future<void>      _signUp()      async      {
setState(() {
     _isLoading = true;
     _errorMessage = "";
   });
   var existingUser = await _firestore

  .collection('users')
        .where('email', isEqualTo:
_emailController.text.trim())
        .get();

    if       (existingUser.docs.isNotEmpty)       {
setState(() {
        _errorMessage = "User already exists.
Please log in.";
        _isLoading = false;
      });
return;
    }

    // Create user in Firebase Authentication
    UserCredential userCredential = await
_auth.createUserWithEmailAndPassword(
email: _emailController.text.trim(),
      password: _passwordController.text.trim(),
    );

    // Store user details in Firestore
await
_firestore.collection('users').doc(userCredential.
user!.uid).set({
      'email': _emailController.text.trim(),
      'password':_passwordController.text.trim(),
      'createdAt': FieldValue.serverTimestamp(),
    });

    // Navigate to Home Page
    Navigator.pushAndRemoveUntil(
context,
      MaterialPageRoute(builder: (context) =>
HomeScreen()),
          (route) => false,
    );

    Navigator.of(context).pushReplacement(
MaterialPageRoute(builder: (context) =>
const BottomNavBar()),
    );
```

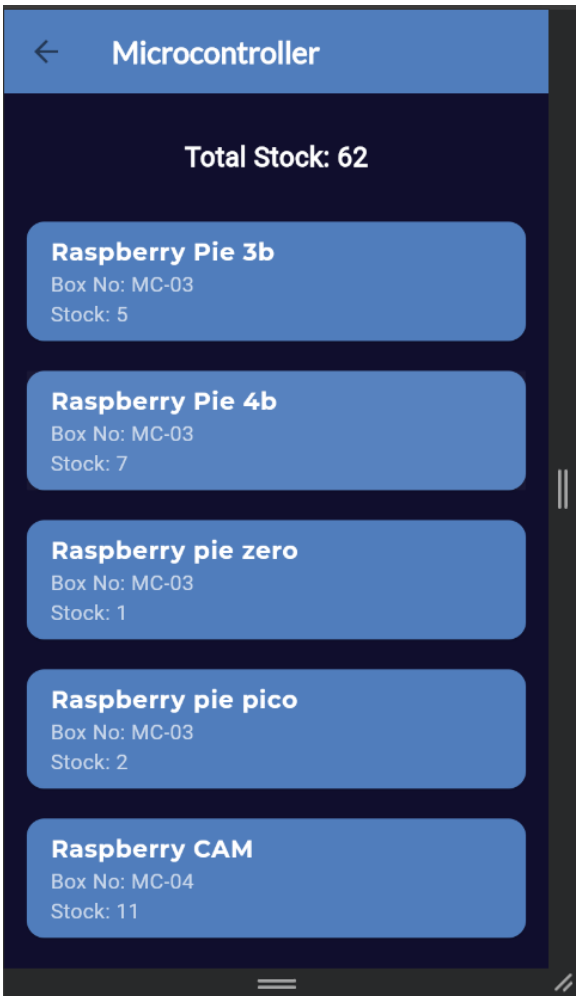
```
    try {
      // Check if the user already exists in
Firestore
    } on FirebaseAuthException catch (e) {
setState(() {
        _errorMessage = e.message ?? "Error
signing up";
      });
```

Output:



Fetching Data From the Database

'Stock: Loading…'



Once the Data is Fetched

'Stock: Nos'

MAD & PWA Lab

Journal

Experiment No.	07
Experiment Title.	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.
Roll No.	34
Name	Himesh Pathai
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	

PWA Experiment -7

Himesh Pathai 34/D15A

❖ **AIM**

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

❖ **Theory**:-

Regular Web Application

A regular web application is a website designed to be accessible on various devices, ensuring that content adjusts dynamically to different screen sizes. Built using web technologies such as HTML, CSS, JavaScript, and Ruby, these applications function through a browser. While they can leverage some native device features, their functionality is dependent on browser compatibility. For instance, a feature may work on Google Chrome but not on Safari or Mozilla Firefox due to browser limitations.

Progressive Web Application (PWA)

A Progressive Web Application (PWA) is an advanced version of a regular web app, integrating additional features to enhance the user experience. PWAs combine the best elements of desktop and mobile applications, delivering a seamless experience across platforms.

❖ Key Differences Between PWAs and Regular Web Apps

1. Native-Like Experience

While both PWAs and regular web apps utilize standard web technologies like HTML, CSS, and JavaScript, PWAs offer a user experience similar to native mobile applications. PWAs can access device-specific features such as push notifications without relying on a particular browser, creating a smoother, more integrated experience.

2. Ease of Access

Unlike traditional mobile apps that require time-consuming downloads and storage space, PWAs can be installed directly via a URL. Users can add a PWA to their home screen with a simple link, eliminating installation complexities and ensuring easy access while keeping brand presence strong.

3. Enhanced Performance

PWAs utilize caching mechanisms to pre-load content, such as text, stylesheets, and images, allowing for faster loading times. This significantly improves user engagement and retention by reducing waiting periods, ultimately benefiting businesses by increasing interaction rates.

4. Improved User Engagement

PWAs efficiently leverage push notifications and native device features to keep users engaged. Unlike regular web apps, their functionality is not restricted by browser dependencies, enabling businesses to notify users about updates, offers, and promotions without disruptions.

5. Real-Time Updates

A major advantage of PWAs is their ability to update automatically without requiring users to download new versions from an app store. Developers can push updates directly from the server, ensuring that users always access the latest features and improvements instantly.

6. Search Engine Optimization (SEO) Benefits

Since PWAs function within web browsers, they can be indexed by search engines, improving their visibility in search results. This gives them a strategic advantage over native apps, which are limited to app store searches.

7. Cost-Effective Development

Unlike native mobile apps, PWAs do not require approval or submission to app stores, reducing development and maintenance costs.

❖ Advantages and Limitations of PWAs

➢ Advantages:

- **Progressive:** Compatible with all browsers and devices, following the principle of progressive enhancement.
- **Responsive:** Adapts to different screen sizes, including desktops, tablets, and smartphones.
- **App-Like Feel:** Mimics the experience of native applications in navigation and interaction.
- **Always Updated:** Service workers ensure real-time updates without requiring user intervention.
- **Secure:** Delivered over HTTPS, ensuring secure data transfer and protection against cyber threats.
- **SEO-Friendly:** Can be indexed by search engines, enhancing discoverability.
- **Re-Engagement:** Enables push notifications to encourage continued user interaction.
- **Installable:** Allows users to add the app to their home screen without app store downloads.
- **Offline Functionality:** Can function in low or no connectivity conditions using cached content.

➢ Limitations:

- **Higher Battery Consumption:** PWAs tend to consume more battery due to constant background processes.
- **Limited Hardware Access:** Some device features, such as advanced sensors and Bluetooth, may not be fully accessible.
- **Offline Mode Constraints:** Some offline capabilities remain limited, depending on browser support.
- **No App Store Presence:** PWAs cannot generate traffic from app store searches.
- **Lack of Centralized Control:** Unlike native apps, PWAs do not undergo an official approval process, potentially affecting credibility.

CODE:

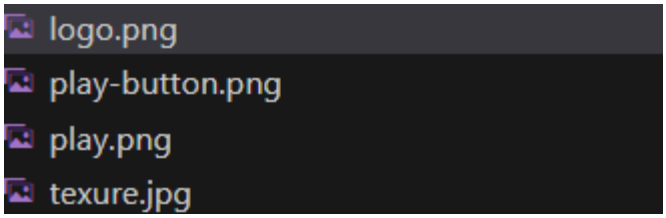
Index.html :

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

    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />

    <link  rel="apple-touch-icon"  href="%PUBLIC_URL%/apple-touch-icon.png">
    <link rel="apple-touch-icon" sizes="192x192" href="%PUBLIC_URL%/logo.png">
    <link rel="apple-touch-icon" sizes="512x512" href="%PUBLIC_URL%/logo.png">

    <link rel="shortcut icon" href="%PUBLIC_URL%/favicon.ico" type="image/x-icon"
/>

    <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.15.4/css/all.css"
        integrity="sha384-
DyZ88mC6Up2uqS4h/KRgHuoeGwBcD4Ng9SiP4dIRy0EXTlnuz47vAwmeGwVChigm
"
        crossorigin="anonymous" />

    <title>Streamo - Netflix</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

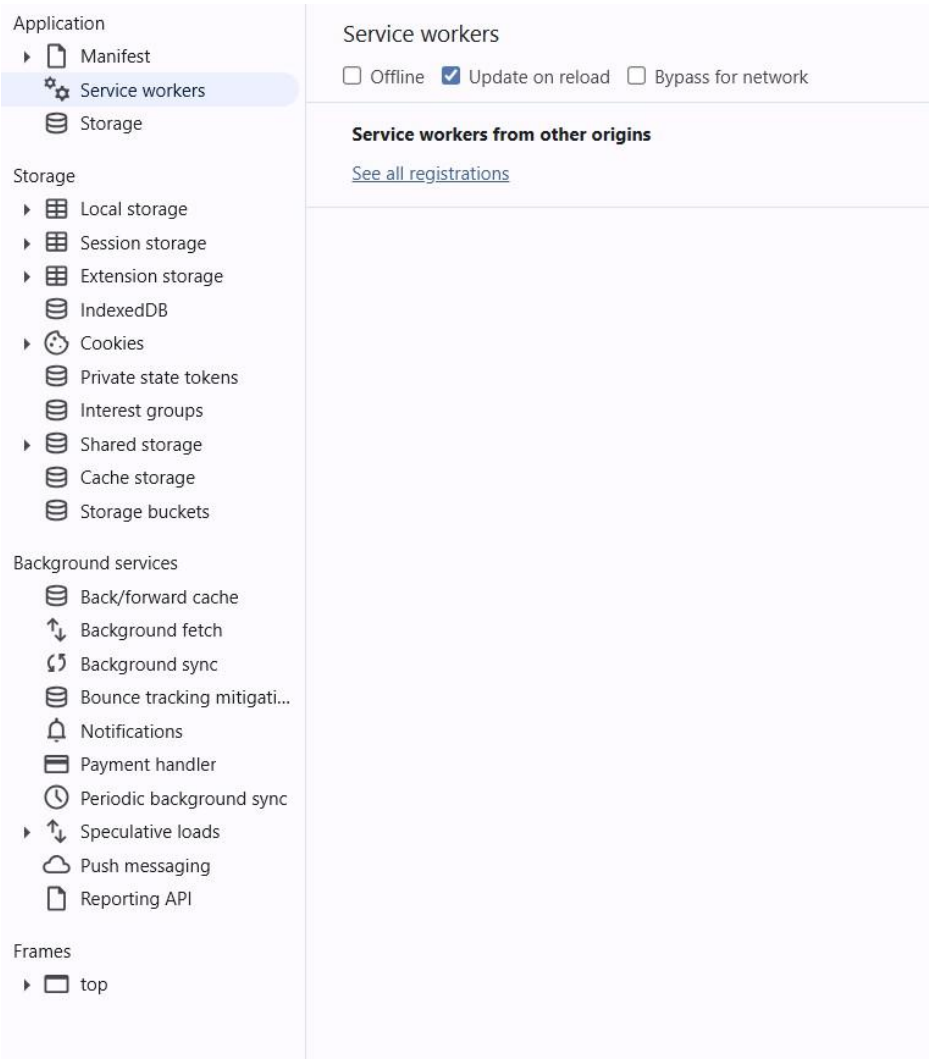
Manifest.json


```
{
  "name": "Streamo - Netflix",
  "short_name": "Streamo",
  "start_url": "/",
  "display": "standalone",
  "background_color": "#000000",
  "theme_color": "#000000",
  "description": "Watch unlimited movies & TV shows.",
  "icons": [
    {
      "src": "/logo.png",
      "type": "image/png",
      "sizes": "192x192"
    },
    {
      "src": "/logo.png",
      "type": "image/png",
      "sizes": "512x512"
    }
  ]
}
```

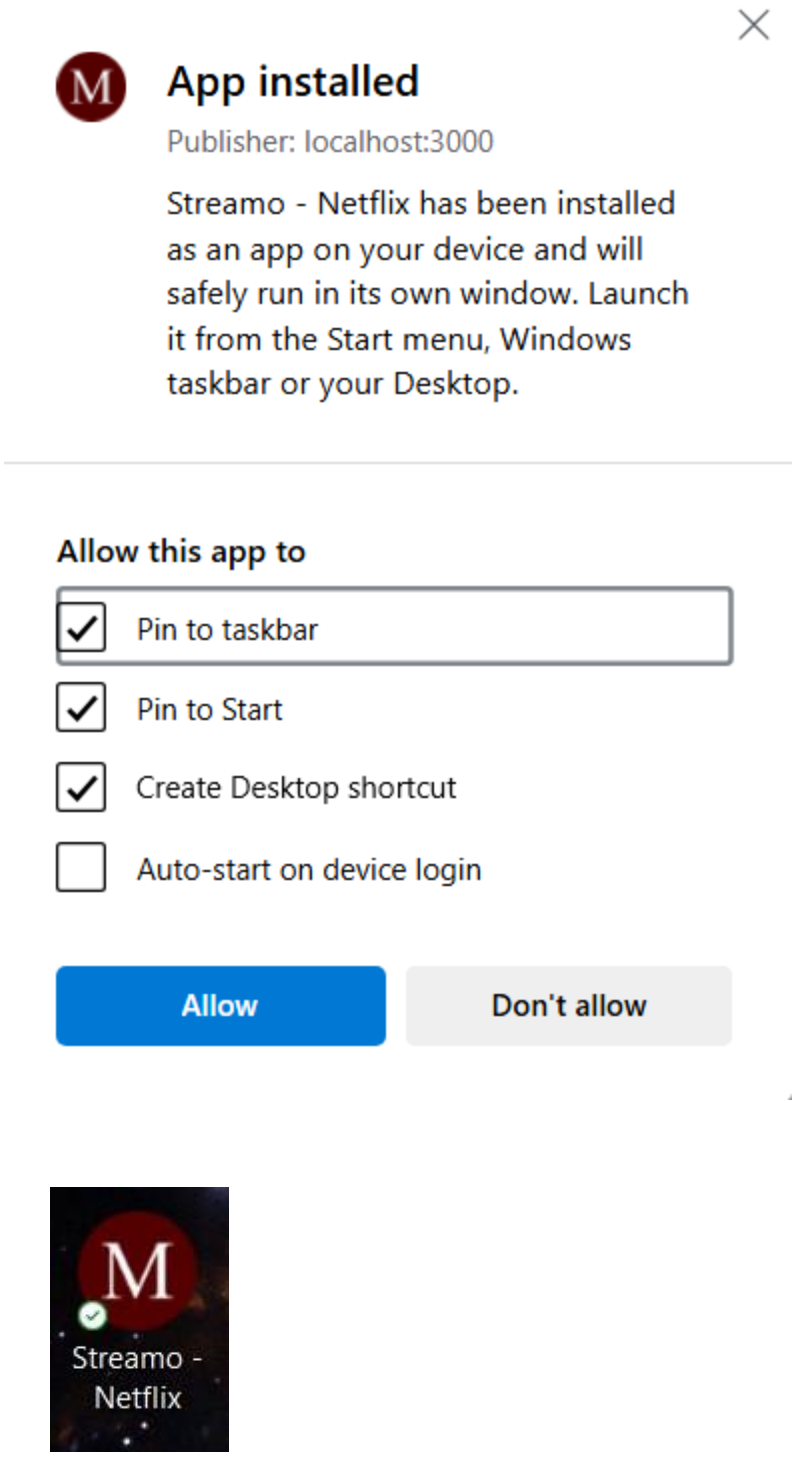
Icons



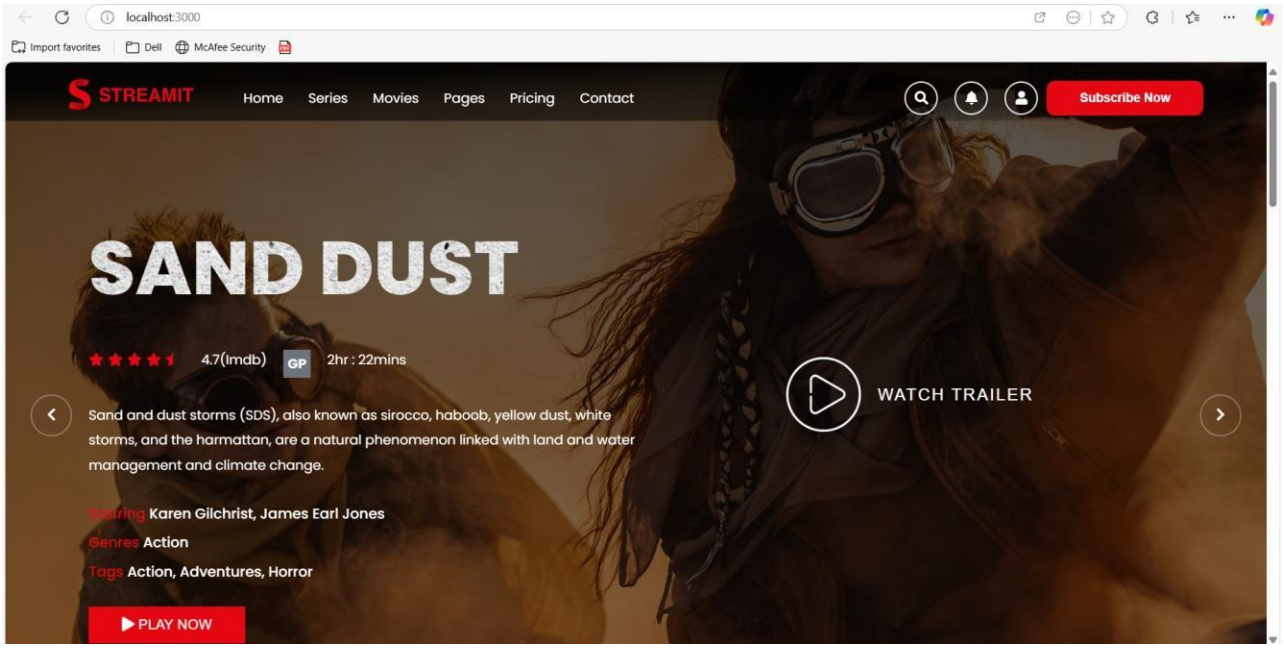
Google Dev

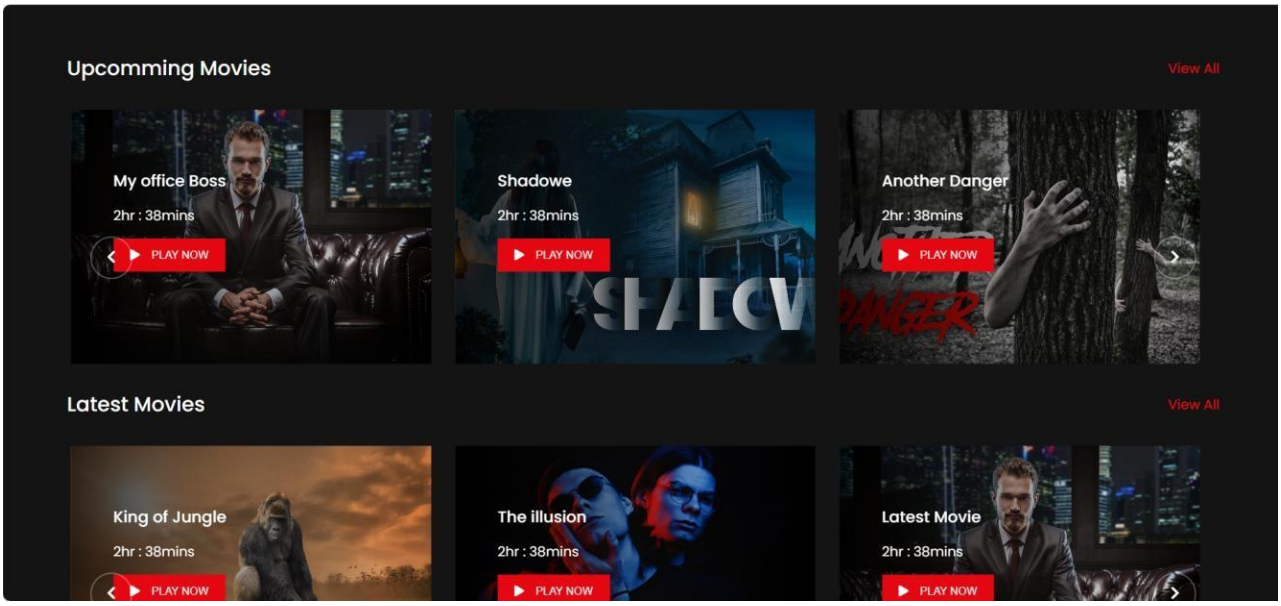


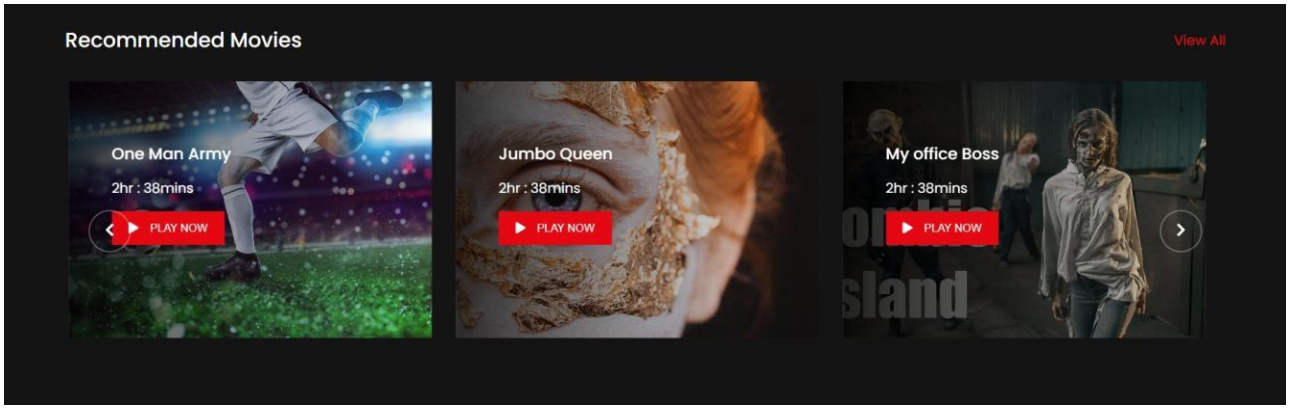
❖ **Output:-**


App installed
Publisher: localhost:3000
Streamo - Netflix has been installed as an app on your device and will safely run in its own window. Launch it from the Start menu, Windows taskbar or your Desktop.
Allow this app to
Pin to taskbar
Pin to Start
Create Desktop shortcut
Auto-start on device login
Allow
Don't allow

Streamo - Netflix


localhost:3000
Import favorites
Dell
McAfee Security
STREAMIT
Home
Series
Movies
Pages
Pricing
Contact
Subscribe Now
SAND DUST
4.7(Imdb)
GP
2hr : 22mins
Sand and dust storms (SDS), also known as sirocco, haboob, yellow dust, white storms, and the harmattan, are a natural phenomenon linked with land and water management and climate change.
Karen Gilchrist, James Earl Jones
Action
Action, Adventures, Horror
PLAY NOW
WATCH TRAILER


Upcomming Movies
View All
My office Boss
2hr : 38mins
PLAY NOW
Shadowe
2hr : 38mins
PLAY NOW
Another Danger
2hr : 38mins
PLAY NOW
Latest Movies
View All
King of Jungle
2hr : 38mins
PLAY NOW
The illusion
2hr : 38mins
PLAY NOW
Latest Movie
2hr : 38mins
PLAY NOW


Recommended Movies
View All
One Man Army
2hr : 38mins
PLAY NOW
Jumbo Queen
2hr : 38mins
PLAY NOW
My office Boss
2hr : 38mins
PLAY NOW

MAD & PWA Lab

Journal

Experiment No.	08
Experiment Title.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA
Roll No.	34
Name	Himesh Pathai
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

MPL Experiment 8 (PWA)

Name: Himesh Pathai **Class:** D15A **Roll no:** 34

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 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.

Codes:

```
//Serviceworker.js

// sw.js - Complete Service Worker for E-commerce PWA
const CACHE_NAME = 'ecommerce-pwa-v2';
const API_CACHE = 'ecommerce-api-v1';
const ASSETS_TO_CACHE = [
  '/',
  '/index.html',
  '/manifest.json',
  '/offline.html',
  '/css/main.min.css',
  '/js/app.min.js',
  '/icons/icon-192x192.png',
  '/icons/icon-512x512.png',
  '/images/placeholder-product.jpg'
];

// ======================
// Install Event
// ======================
self.addEventListener('install', (event) => {
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then((cache) => {
        console.log('[Service Worker] Cache opened');
        return cache.addAll(ASSETS_TO_CACHE);
      })
```

```
      .then(() => self.skipWaiting())
  );
});

// ======================
// Activate Event
// ======================
self.addEventListener('activate', (event) => {
  event.waitUntil(
    caches.keys().then((cacheNames) => {
      return Promise.all(
        cacheNames.map((cacheName) => {
          if (cacheName !== CACHE_NAME && cacheName !== API_CACHE) {
            console.log('[Service Worker] Deleting old cache:',
cacheName);
            return caches.delete(cacheName);
          }
        })
      );
    })

    .then(() => self.clients.claim())
  );
});

// ======================
// Fetch Event
// ======================
self.addEventListener('fetch', (event) => {
  const { request } = event;
  const url = new URL(request.url);

  // 1. Skip non-GET requests and chrome-extension
  if (request.method !== 'GET' || url.protocol ===
'chrome-extension:') {
    return;
  }

  // 2. API Requests (Network First with Cache Fallback)
```

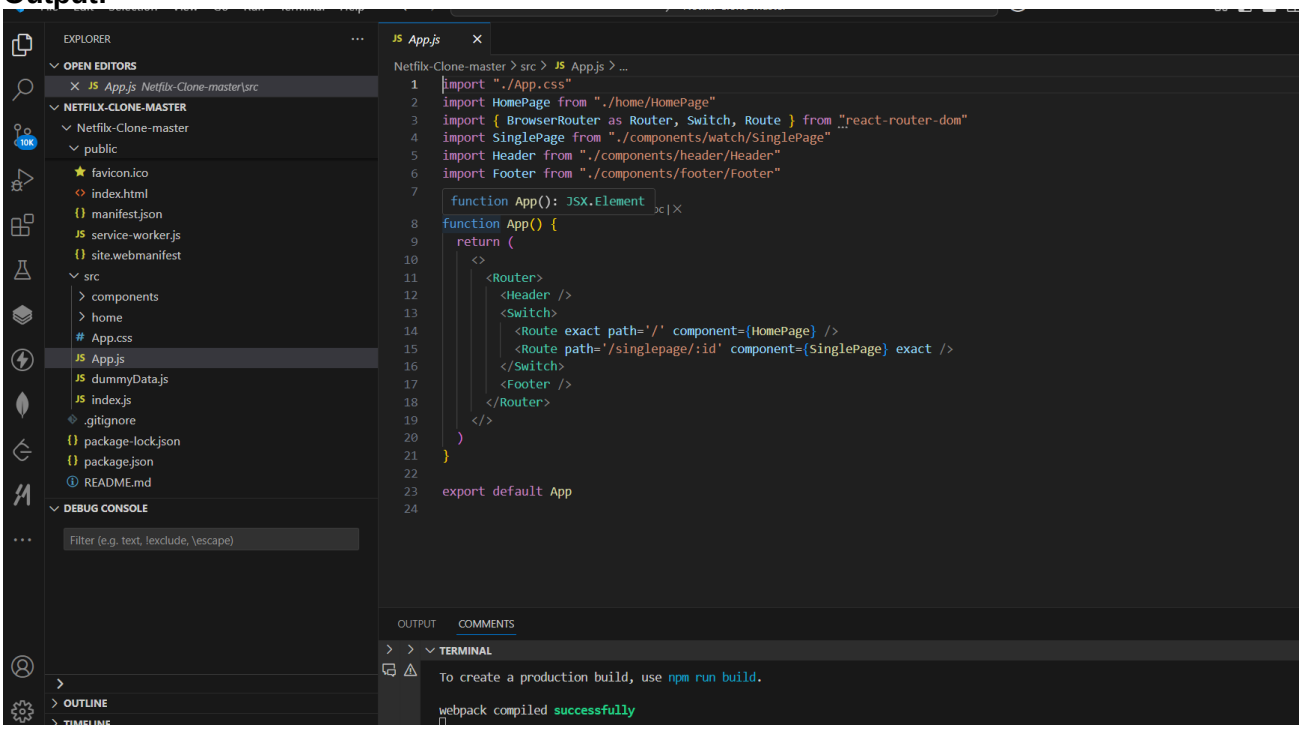
```
if (url.pathname.startsWith('/api/')) {
  event.respondWith(
    fetch(request)
      .then(networkResponse => {
        // Cache successful API responses
        if (networkResponse.ok) {
          const clone = networkResponse.clone();
          caches.open(API_CACHE)
            .then(cache => cache.put(request, clone));
        }
        return networkResponse;
      })
      .catch(() => {
        // Return cached version if available
        return caches.match(request)
          .then(cachedResponse => cachedResponse || Response.json(
            { error: 'Network error' },
            { status: 503 }
          ));
      })
  );
  return;
}

// 3. Static Assets (Cache First with Network Fallback)
event.respondWith(
  caches.match(request)
    .then(cachedResponse => {
      // Return cached version if found
      if (cachedResponse) {
        return cachedResponse;
      }

      // Otherwise fetch from network
      return fetch(request)
        .then(networkResponse => {
          // Cache successful responses
          if (networkResponse.ok) {
```

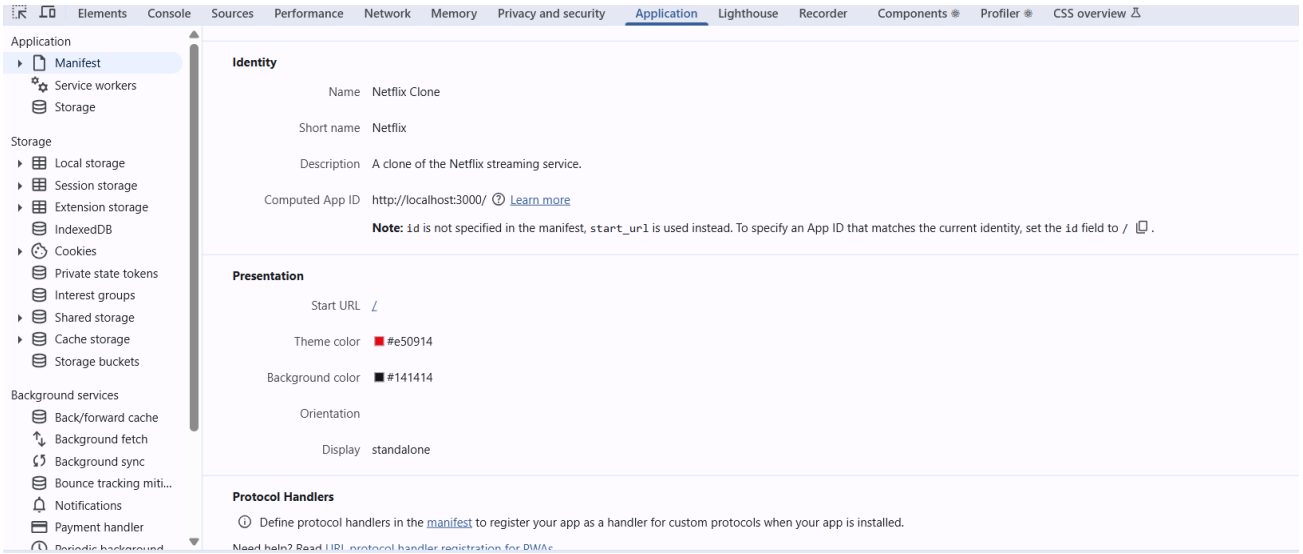
```
              const clone = networkResponse.clone();
              caches.open(CACHE_NAME)
                .then(cache => cache.put(request, clone));
            }
            return networkResponse;
          })
          .catch(() => {
            // Special handling for HTML pages
            if (request.headers.get('accept').includes('text/html')) {
              return caches.match('/offline.html');
            }
            // Return placeholder for images
            if (request.headers.get('accept').includes('image')) {
              return caches.match('/images/placeholder-product.jpg');
            }
          });
      })
  );
});


// ======================
// Background Sync
// ======================
self.addEventListener('sync', (event) => {
  if (event.tag === 'sync-cart') {
    event.waitUntil(
      // Get cart data from IndexedDB
      getCartData()
        .then(cartItems => {
          return fetch('/api/cart/sync', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify(cartItems)
          });
        })
        .then(() => {
          return showNotification('Cart Synced', 'Your cart has been
updated');
```

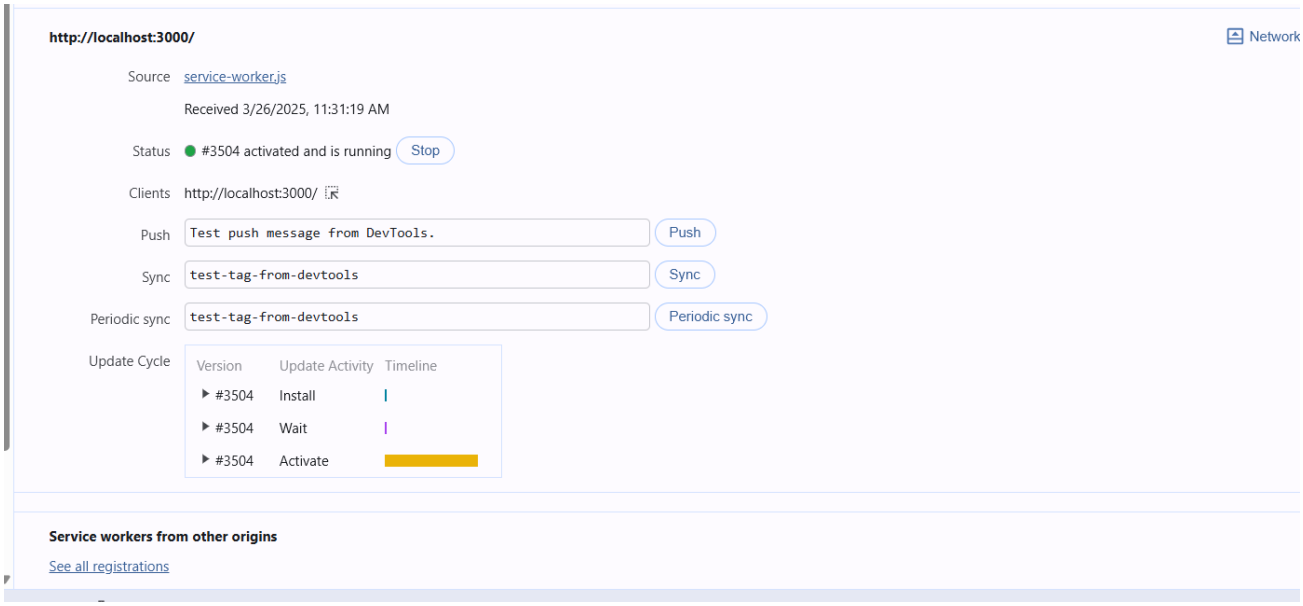
```
        })
        .catch(err => {
          console.error('Sync failed:', err);
        })
    );
  }
});


// ======================
// Push Notifications
// ======================
self.addEventListener('push', (event) => {
  let data = {};
  try {
    data = event.data.json();
  } catch (e) {
    data = {
      title: 'New Update',
      body: 'Check out our latest products!',
      icon: '/icons/icon-192x192.png',
      url: '/'
    };
  }

  const options = {
    body: data.body,
    icon: data.icon || '/icons/icon-192x192.png',
    badge: '/icons/icon-96x96.png',
    data: {
      url: data.url || '/'
    }
  };

  event.waitUntil(
    self.registration.showNotification(data.title, options)
  );
});
```

```
self.addEventListener('notificationclick', (event) => {
  event.notification.close();
  event.waitUntil(
    clients.matchAll({ type: 'window' })
      .then(clientList => {
        for (const client of clientList) {
          if (client.url === event.notification.data.url && 'focus' in
client) {
            return client.focus();
          }
        }

        if (clients.openWindow) {
          return clients.openWindow(event.notification.data.url);
        }
      })
  );
});

// ======================
// Helper Functions
// ======================
async function getCartData() {
  // In a real app, you would use IndexedDB
  return new Promise(resolve => {
    resolve([]);
  });
}

async function showNotification(title, body) {
  return self.registration.showNotification(title, { body });
}
```

Output:


EXPLORER
OPEN EDITORS
App.js Netflix-Clone-master\src
NETFILX-CLONE-MASTER
Netflix-Clone-master
public
favicon.ico
index.html
manifest.json
service-worker.js
site.webmanifest
src
components
home
App.css
App.js
dummyData.js
index.js
.gitignore
package-lock.json
package.json
README.md
DEBUG CONSOLE
Filter (e.g. text, !exclude, \escape)
OUTLINE
App.js
Netflix-Clone-master > src > App.js > ...
import "./App.css"
import HomePage from "./home/HomePage"
import { BrowserRouter as Router, Switch, Route } from "react-router-dom"
import SinglePage from "./components/watch/SinglePage"
import Header from "./components/header/Header"
import Footer from "./components/footer/Footer"
function App(): JSX.Element
function App() {
return (
<>
<Router>
<Header />
<Switch>
<Route exact path='/' component={HomePage} />
<Route path='/singlepage/:id' component={SinglePage} exact />
</Switch>
<Footer />
</Router>
</>
)
}
export default App
OUTPUT
COMMENTS
TERMINAL
To create a production build, use npm run build.
webpack compiled successfully


Elements
Console
Sources
Performance
Network
Memory
Privacy and security
Application
Lighthouse
Recorder
Components
Profiler
CSS overview
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Storage buckets
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking miti...
Notifications
Payment handler
Identity
Name Netflix Clone
Short name Netflix
Description A clone of the Netflix streaming service.
Computed App ID http://localhost:3000/ Learn more
Note: id is not specified in the manifest, start_url is used instead. To specify an App ID that matches the current identity, set the id field to /.
Presentation
Start URL /
Theme color #e50914
Background color #141414
Orientation
Display standalone
Protocol Handlers
Define protocol handlers in the manifest to register your app as a handler for custom protocols when your app is installed.


http://localhost:3000/
Network
Source
service-worker.js
Received 3/26/2025, 11:31:19 AM
Status
#3504 activated and is running
Stop
Clients
http://localhost:3000/
Push
Test push message from DevTools.
Push
Sync
test-tag-from-devtools
Sync
Periodic sync
test-tag-from-devtools
Periodic sync
Update Cycle
Version
Update Activity
Timeline
#3504
Install
#3504
Wait
#3504
Activate
Service workers from other origins
See all registrations

MAD & PWA Lab

Journal

Experiment No.	09
Experiment Title.	To implement Service worker events like fetch, sync and push for E-commerce PWA
Roll No.	34
Name	Himesh Pathai
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

MPL Experiment 9 (PWA)

Name: Himesh Pathai **Class:** D15A **Roll no:** 34

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

Theory:

Service Worker

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

Things to note about Service Worker:

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

Fetch Event

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

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

- **CacheFirst** - In this function, if the received request has cached before, the cached response is returned to the page. But if not, a new response requested from the network.
- **NetworkFirst** - In this function, firstly we can try getting an updated response from the network, if this process completed successfully, the new response will be cached and returned. But if this process fails, we check whether the request has been cached

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

Sync Event

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

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.

Code:

```
//Serviceworker.js
// sw.js - Complete Service Worker for E-commerce PWA
const CACHE_NAME = 'ecommerce-pwa-v2';
const API_CACHE = 'ecommerce-api-v1';
const ASSETS_TO_CACHE = [
  '/',
  '/index.html',
  '/manifest.json',
  '/offline.html',
  '/css/main.min.css',
  '/js/app.min.js',
  '/icons/icon-192x192.png',
  '/icons/icon-512x512.png',
```

```
  '/images/placeholder-product.jpg'
];

// ======================
// Install Event
// ======================
self.addEventListener('install', (event) => {
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then((cache) => {
        console.log('[Service Worker] Cache opened');
        return cache.addAll(ASSETS_TO_CACHE);
      })
      .then(() => self.skipWaiting())
  );
});

// ======================
// Activate Event
// ======================
self.addEventListener('activate', (event) => {
  event.waitUntil(
    caches.keys().then((cacheNames) => {
      return Promise.all(
        cacheNames.map((cacheName) => {
          if (cacheName !== CACHE_NAME && cacheName !== API_CACHE) {
            console.log('[Service Worker] Deleting old cache:',
cacheName);
                                        return caches.delete(cacheName);
          }
        })
      );
    })

    .then(() => self.clients.claim())
  );
});

// ======================
```

```
// Fetch Event
// ======================
self.addEventListener('fetch', (event) => {
  const { request } = event;
  const url = new URL(request.url);

  // 1. Skip non-GET requests and chrome-extension
  if (request.method !== 'GET' || url.protocol ===
'chrome-extension:') {
    return;
  }

  // 2. API Requests (Network First with Cache Fallback)
  if (url.pathname.startsWith('/api/')) {
    event.respondWith(
      fetch(request)
        .then(networkResponse => {
          // Cache successful API responses
          if (networkResponse.ok) {
            const clone = networkResponse.clone();
            caches.open(API_CACHE)
              .then(cache => cache.put(request, clone));
          }
          return networkResponse;
        })
        .catch(() => {
          // Return cached version if available
          return caches.match(request)
            .then(cachedResponse => cachedResponse || Response.json(
              { error: 'Network error' },
              { status: 503 }
            ));
        })
    );
    return;
  }

  // 3. Static Assets (Cache First with Network Fallback)
```

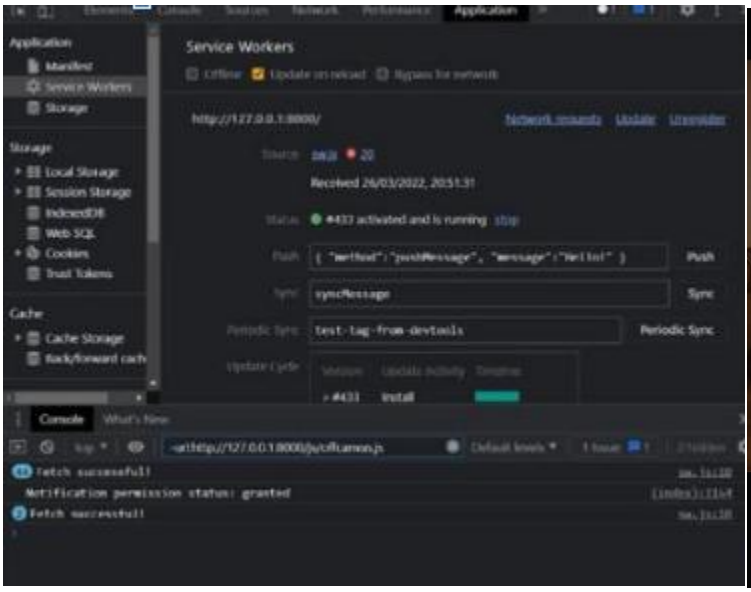
```
  event.respondWith(
    caches.match(request)
      .then(cachedResponse => {
        // Return cached version if found
        if (cachedResponse) {
          return cachedResponse;
        }

        // Otherwise fetch from network
        return fetch(request)
          .then(networkResponse => {
            // Cache successful responses
            if (networkResponse.ok) {
              const clone = networkResponse.clone();
              caches.open(CACHE_NAME)
                .then(cache => cache.put(request, clone));
            }
            return networkResponse;
          })
          .catch(() => {
            // Special handling for HTML pages
            if (request.headers.get('accept').includes('text/html')) {
              return caches.match('/offline.html');
            }
            // Return placeholder for images
            if (request.headers.get('accept').includes('image')) {
              return caches.match('/images/placeholder-product.jpg');
                                    }
                                    }
                                    )
                                    ;
  );                                })
});


// ======================
// Background Sync
// ======================
self.addEventListener('sync', (event) => {
  if (event.tag === 'sync-cart') {
```

```
    event.waitUntil(
      // Get cart data from IndexedDB
      getCartData()
        .then(cartItems => {
          return fetch('/api/cart/sync', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify(cartItems)
          });
        })
        .then(() => {
          return showNotification('Cart Synced', 'Your cart has been
updated');
        })
        .catch(err => {
          console.error('Sync failed:', err);
        })
    );
  }
});

// ======================
// Push Notifications
// ======================
self.addEventListener('push', (event) => {
  let data = {};
  try {
    data = event.data.json();
  } catch (e) {
    data = {
      title: 'New Update',
      body: 'Check out our latest products!',
      icon: '/icons/icon-192x192.png',
      url: '/'
    };
  }

  const options = {
```

```
    body: data.body,
    icon: data.icon || '/icons/icon-192x192.png',
    badge: '/icons/icon-96x96.png',
    data: {
      url: data.url || '/'
    }
  };

  event.waitUntil(
    self.registration.showNotification(data.title, options)
  );
});

self.addEventListener('notificationclick', (event) => {
  event.notification.close();
  event.waitUntil(
    clients.matchAll({ type: 'window' })
      .then(clientList => {
        for (const client of clientList) {
          if (client.url === event.notification.data.url && 'focus' in
client) {
            return client.focus();
          }
        }
        if (clients.openWindow) {
          return clients.openWindow(event.notification.data.url);
        }
      })
  );
});

// ======================
// Helper Functions
// ======================
async function getCartData() {
  // In a real app, you would use IndexedDB
  return new Promise(resolve => {
    resolve([]);
```

```
  });
}

async function showNotification(title, body) {
  return self.registration.showNotification(title, { body });
}
```

Output:



t

MAD & PWA Lab

Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	34
Name	Himesh Pathai
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

MPL Experiment 10 (PWA)

Name: Himesh Pathai **Class:** D15A **Roll no:** 34

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

Theory:

GitHub Pages

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

GitHub Pages provides the following key features:

- Blogging with Jekyll
- Custom URL
- Automatic Page Generator

Reasons for favoring this over Firebase:

- Free to use
- Right out of github
- Quick to set up

Companies Using GitHub Pages:

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

Pros

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

Cons

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

Firebase

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

Some of the features offered by Firebase are:

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

Reasons for favoring over GitHub Pages:

- Realtime backend made easy
- Fast and responsive

Companies Using Firebase:

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

Pros

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

Cons

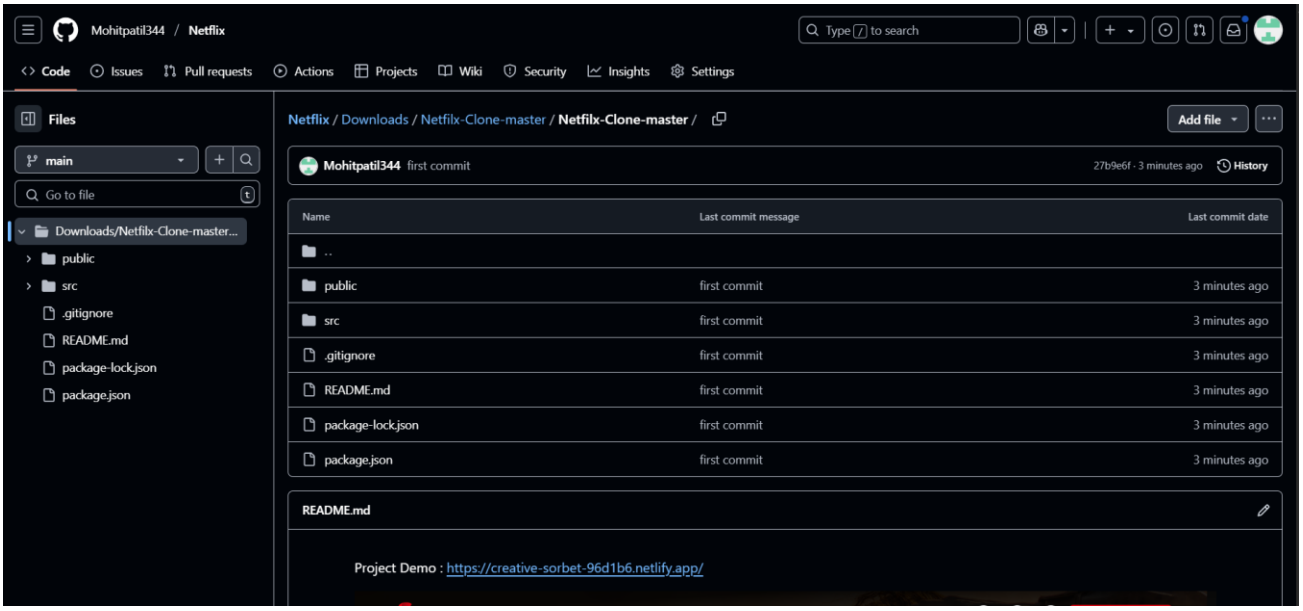
- Only 10 GB of data transfer is allowed per month. But this is not really a big problem, if you use a CDN or AMP.
- Command-line interface only.
- No in-built support for any static site generator.

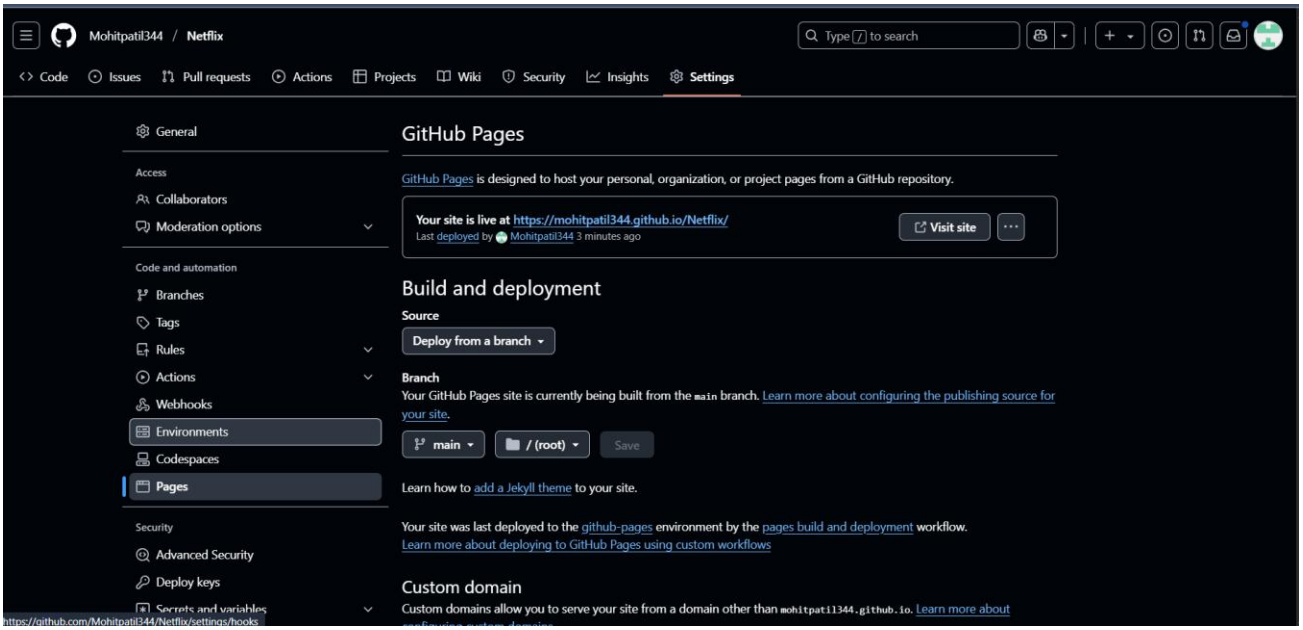
Link to our GitHub repository:

https://github.com/Mohitpatil344/Netflix

Link to our Hosted website:
https://creative-sorbet-96d1b6.netlify.app/

Github Screenshot:





MAD & PWA Lab

Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	34
Name	Himesh Pathai
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	

PWA Experiment -11

Himesh Pathai 34/D15A

✞ Aim:

To use Google Lighthouse PWA Analysis Tool to test the PWA functioning.

✞ Theory:

Google Lighthouse: Overview

Google Lighthouse is an open-source automated tool developed by Google to audit web applications based on multiple parameters, including performance, accessibility, SEO, best practices, and Progressive Web App (PWA) implementations. It provides an in-depth analysis of a webpage by running different tests and generating a detailed report highlighting areas for improvement.

Lighthouse can be executed via **Chrome DevTools**, **Node.js command line**, or as a **browser extension**. It helps developers optimize their applications to enhance user experience, improve mobile responsiveness, and ensure compliance with best web development practices.

Key Features of Google Lighthouse

Lighthouse audits web pages for both **desktop** and **mobile** versions. The key metrics analyzed during an audit are as follows:

- **Performance:**

This metric evaluates how fast a webpage loads and becomes interactive for users. The score is based on various factors, including:

- **First Contentful Paint (FCP):** Measures the time taken to render the first visible content.
- **Largest Contentful Paint (LCP):** Measures the time taken for the largest visible element to load.
- **Time to Interactive (TTI):** Measures how long the page takes to become fully interactive.
- **Speed Index:** Indicates how quickly content is visually displayed.
- **Total Blocking Time (TBT):** Calculates the time a page remains unresponsive due to heavy JavaScript execution.

A **high performance score (closer to 100)** means the website loads quickly and delivers a smooth user experience.

- **Progressive Web App (PWA) Analysis:**

Google Lighthouse checks whether a web application follows the **Baseline PWA Checklist** set by Google. It verifies essential PWA components like:

- **Service Workers:** Ensuring offline functionality and background synchronization.
- **Web App Manifest:** Proper implementation of manifest.json for home screen installation.
- **Viewport Handling:** Ensuring mobile-friendliness with <meta name="viewport">.
- **HTTPS:** Ensuring a secure connection for user safety.
- **Responsive Design:** Optimizing layout and content for different screen sizes.
- **Offline Support:** Verifying if key resources are cached to enable offline access.

A high **PWA score** ensures that the application provides an **app-like experience** on mobile devices.

- **Accessibility:**

Accessibility measures how well a web page supports users with disabilities, including those using screen readers and assistive technologies. Lighthouse evaluates accessibility based on:

- **ARIA Attributes:** Proper use of aria-label, aria-required, etc. for better screen reader support.
- **Text Contrast:** Ensuring readable text against the background color.
- **Keyboard Navigation:** Ensuring all elements are accessible via keyboard (no mouse required).
- **Form Labels:** Ensuring form fields have proper labels and descriptions.
- **Alt Text for Images:** Checking if images have alt attributes for visually impaired users.

Accessibility scores are calculated based on pass/fail criteria. A **low score** means that the website is not user-friendly for people with disabilities.

- **Best Practices:**

Lighthouse evaluates whether a web application follows industry-recommended best practices to ensure security, efficiency, and maintainability. It checks for:

- **Use of HTTPS:** Ensuring a secure connection.
- **Deprecated Code:** Identifying outdated HTML tags, CSS styles, and JavaScript APIs.

- **Password Protection:** Verifying that users can securely input passwords (e.g., disabling "paste" for password fields).
- **Safe JavaScript Execution:** Identifying possible security risks and performance issues in JavaScript code.
- **Geo-Location and Cookie Alerts:** Ensuring compliance with privacy regulations like GDPR by displaying necessary permission prompts.

A **high Best Practices score** ensures the website is built using modern, secure, and efficient coding techniques.

- **SEO (Search Engine Optimization):**

SEO audits help determine how well a webpage is optimized for search engines. Lighthouse checks:

- **Meta Tags:** Ensuring title and meta description are properly set.
- **Mobile-Friendliness:** Verifying that the website is optimized for mobile devices.
- **Canonical URLs:** Preventing duplicate content issues.
- **Crawlability:** Ensuring search engines can index the website properly.

A high **SEO score** improves a website's ranking on search engines like Google.

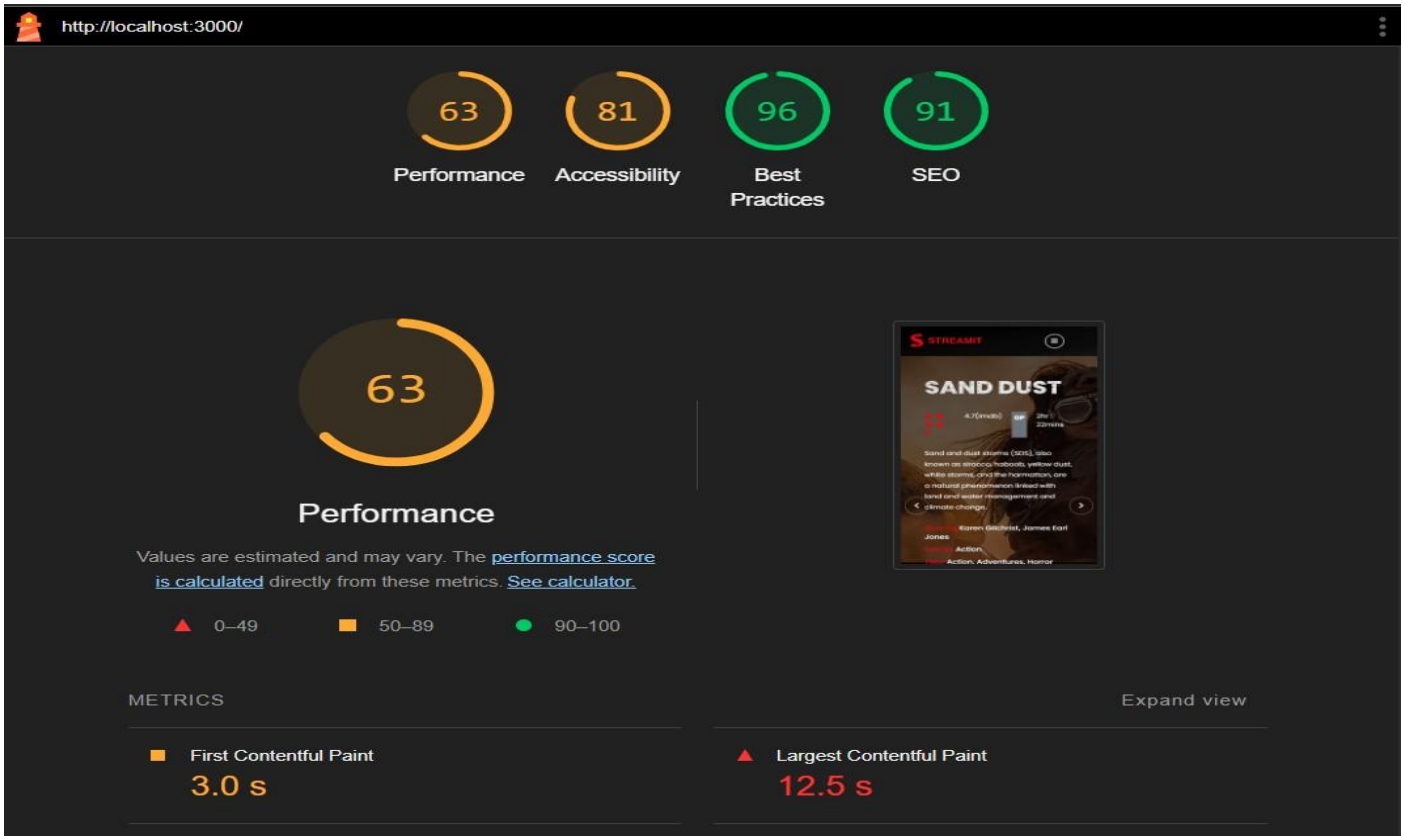
manifest.json

```
{
  "name": "Streamo - Netflix",
  "short_name": "Streamo",
  "start_url": "/",
  "display": "standalone",
  "background_color": "#000000",
  "theme_color": "#000000",
  "description": "Watch unlimited movies & TV shows.",
  "icons": [
    {
      "src": "/logo.png",
```

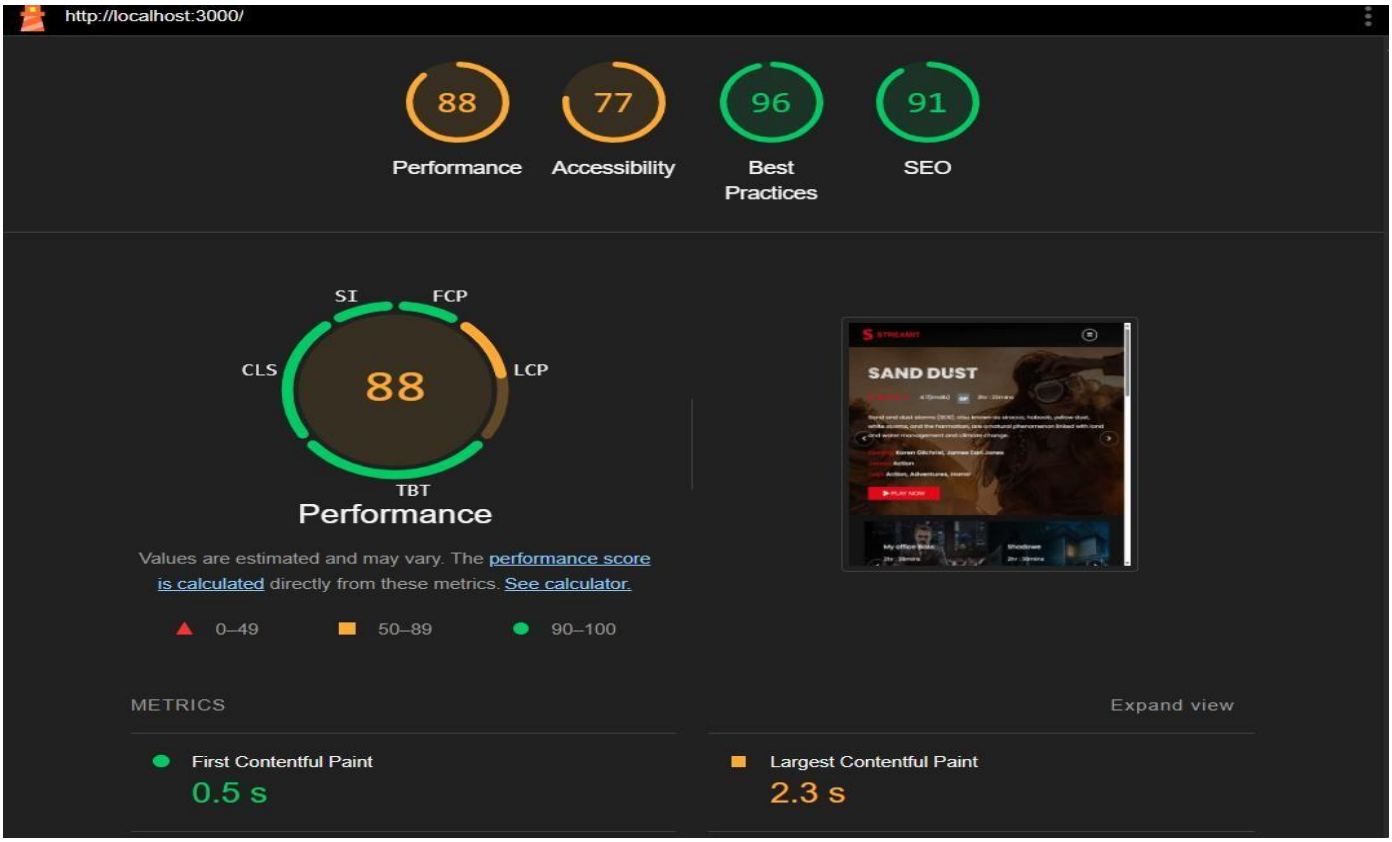
```
    "type": "image/png",
    "sizes": "192x192"
  },
  {
    "src": "/logo.png",
    "type": "image/png",
    "sizes": "512x512"
  }
]
```

✞ Output

- Before Code change



- After code change


http://localhost:3000/
88
Performance
77
Accessibility
96
Best Practices
91
SEO
SI
FCP
CLS
88
LCP
TBT
Performance
Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.
0–49
50–89
90–100
SAND DUST
METRICS
Expand view
First Contentful Paint
0.5 s
Largest Contentful Paint
2.3 s