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

Lab Assistant

Subject Teacher

Mrs. Kajal Joseph

Principal

Head of Department

Dr. Mrs. Shalu Chopra

Name of the Course : MAD & PWA Lab

Course Code : ITL604

Project Title: Discord Flutter Clone/ PWA : To Do List **Roll No. 52**

Year/Sem/Class **:** D15A/D15B **A.Y.:** 23-24

Faculty Incharge : Mrs. Kajal Joseph.

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

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

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

Program specific Outcomes

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

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

Project Title: Discord Flutter Clone/ PWA : To Do List **Roll No. 52**

Lab Objectives:

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

Lab Outcomes:

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

Project Title: Discord Flutter Clone/ PWA : To Do List **Roll No. 52**

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.	52
Name	Susmita Santi
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	15

Project Title: Discord Flutter Clone/ PWA : To Do List **Roll No. 52**

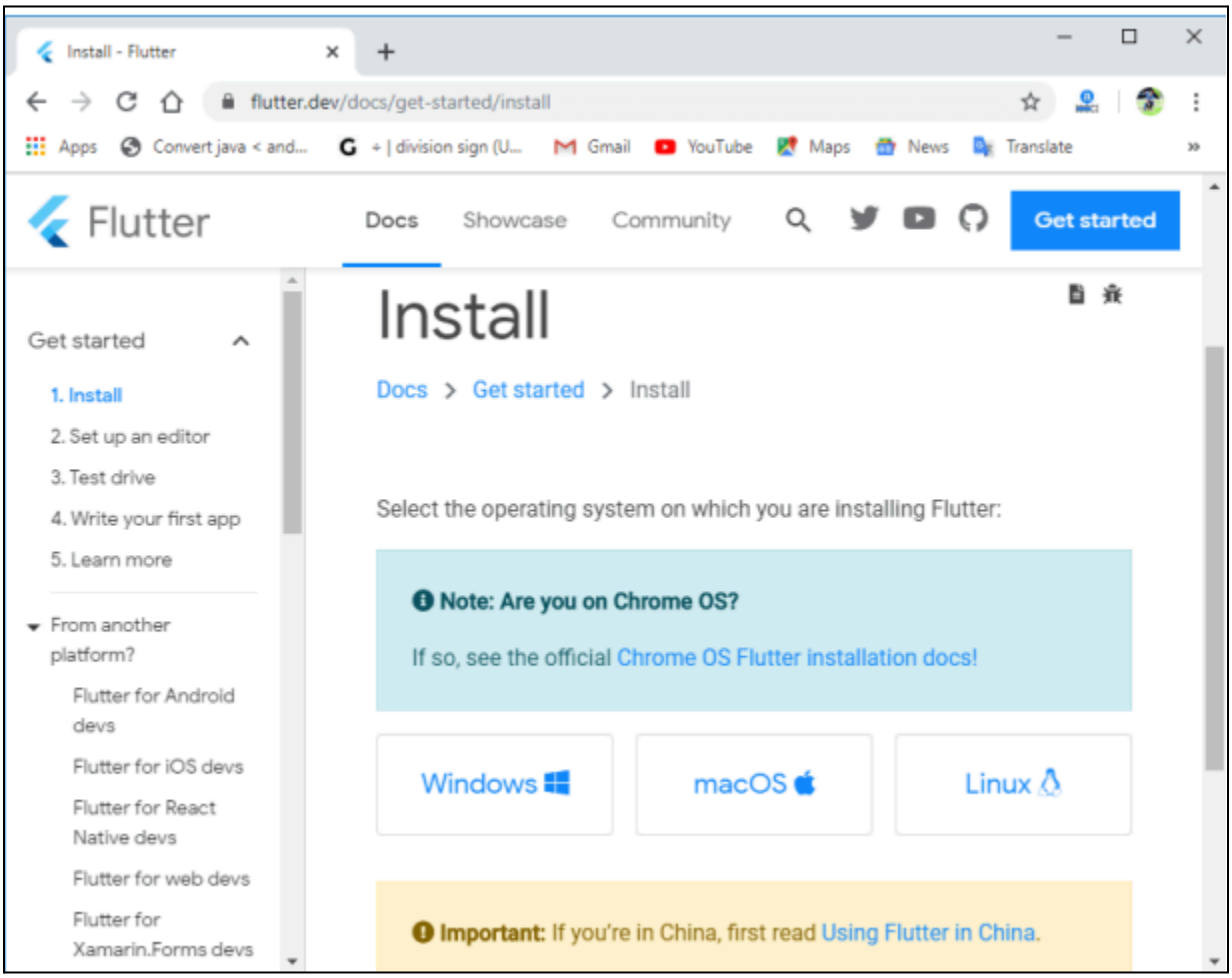
NAME: SUSMITA SANTI **ROLL NO.:52**
SUBJECT: MAD LAB **CLASS:D15A/BATCH C**

EXP 1:Installation and Configuration of Flutter Environment.

AIM: Installation and Configuration of Flutter Environment.

STEPS FOR INSTALLATION:

Step 1: Download the installation bundle of the Flutter Software Development Kit for windows. To download Flutter SDK, Go to its official website,
you will get the following screen.

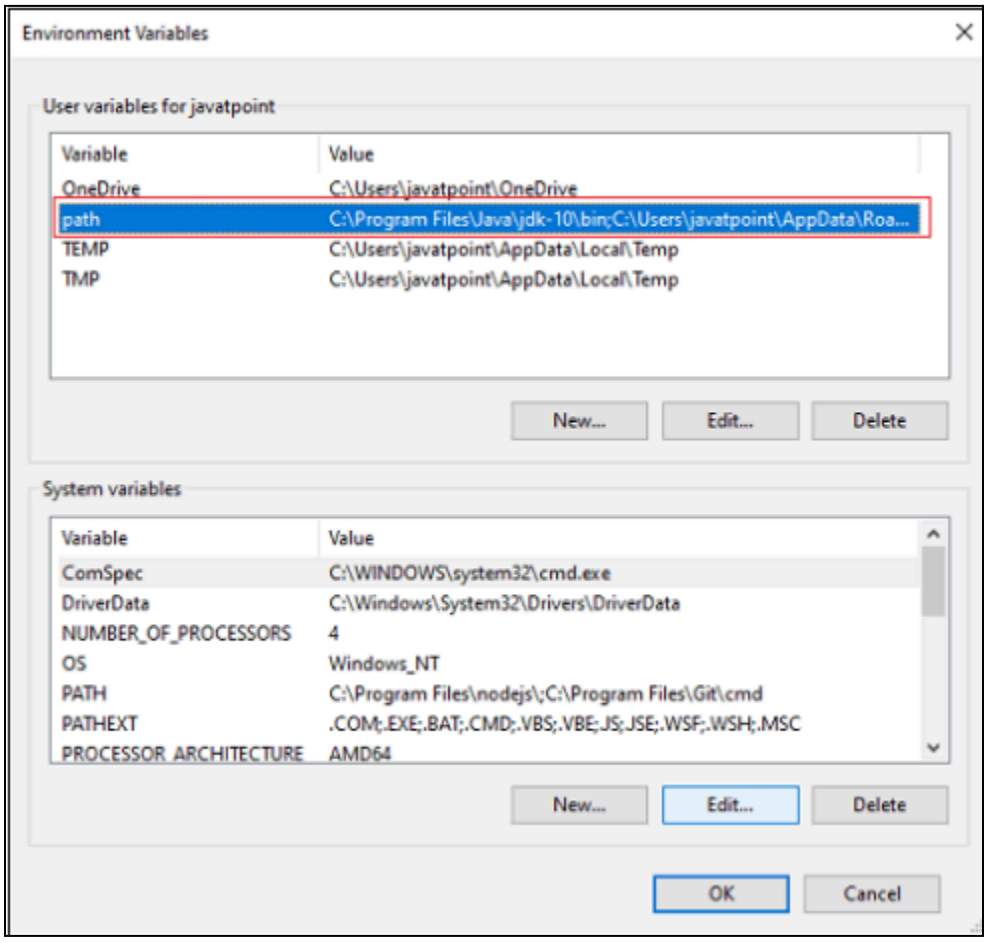


Step 2: Next, to download the latest Flutter SDK, click on the Windows icon. Here, you will find the download link for SDK.

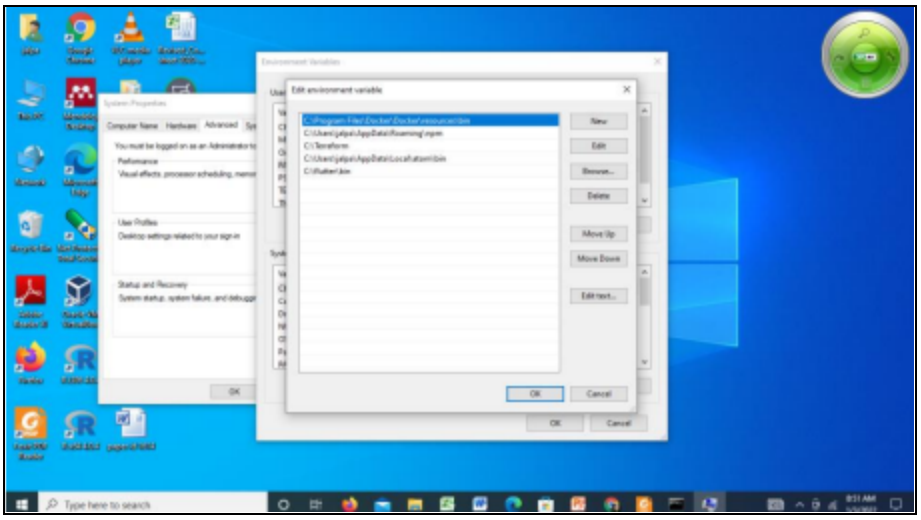
Step 3: When your download is complete, extract the zip file and place it in the desired installation folder or location, for example, C: /Flutter.

Step 4: To run the Flutter command in regular windows console, you need to update the system path to include the flutter bin directory. The following steps are required to do this:

Step 4.1: Go to MyComputer properties -> advanced tab -> environment variables. You will get the following screen.



Step 4.2: Now, select path -> click on edit. The following screen appears

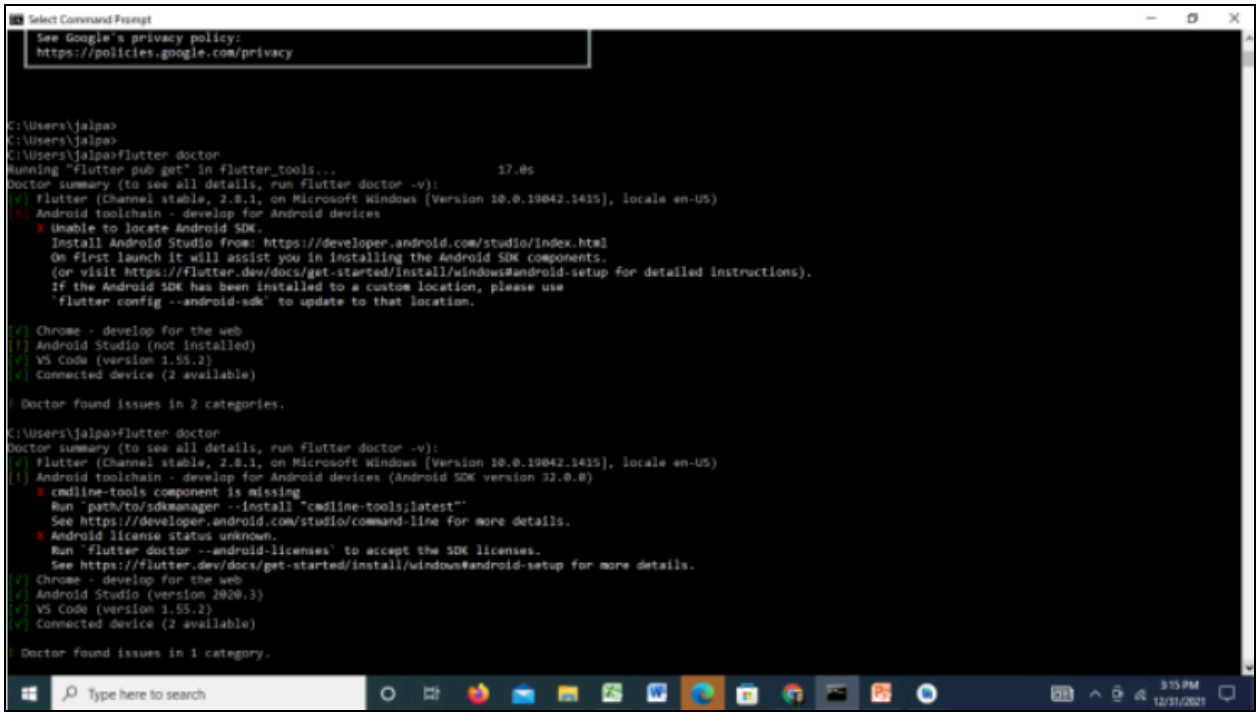


Step 4.3: In the above window, click on New->write path of Flutter bin folder in variable value -> ok -> ok -> ok.

Step 5: Now, run the $ flutter -version command in the command prompt.

```
C:\Users\acer>flutter --version

 A new version of Flutter is available!

 To update to the latest version, run "flutter upgrade".

Flutter 3.16.7 • channel stable • https://github.com/flutter/flutter.git
Framework • revision ef1af02aea (8 days ago) • 2024-01-11 15:19:26 -0600
Engine • revision 4a585b7929
Tools • Dart 3.2.4 • DevTools 2.28.5
```

Now, run the $ flutter doctor command. This command checks for all the requirements of Flutter app development and displays a report of the status of your Flutter installation.

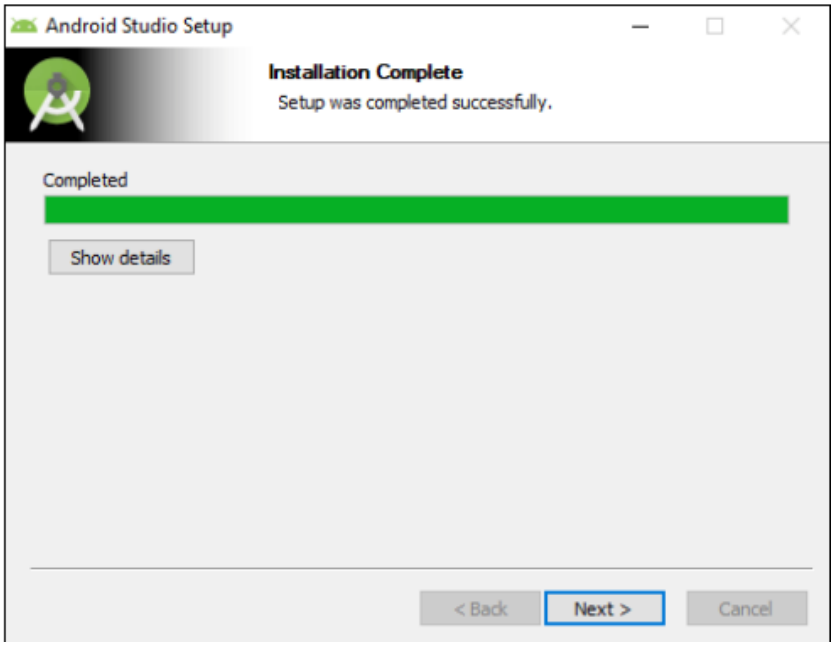


Step 6: When you run the above command, it will analyze the system and show its report, as shown in the below image. Here, you will find the details of all missing tools, which required to run Flutter as well as the development tools that are available but not connected with the device.

Step 7: Install the Android SDK. If the flutter doctor command does not find the Android SDK tool in your system, then you need first to install the Android Studio IDE. To install Android Studio IDE, do the following steps.

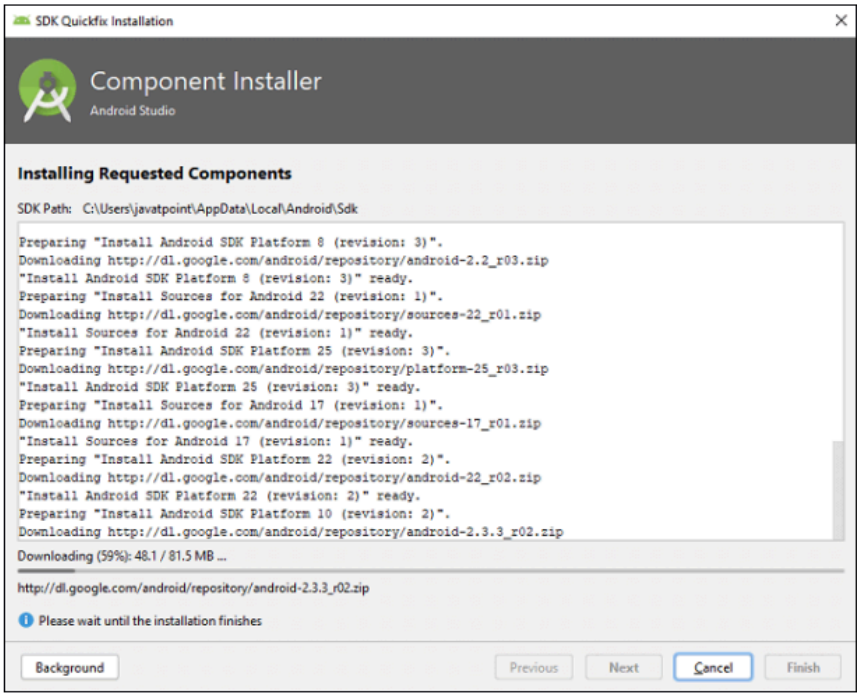
Step 7.1: Download the latest Android Studio executable or zip file from the official site.
Step 7.2: When the download is complete, open the .exe file and run it. You will get the following dialog box.

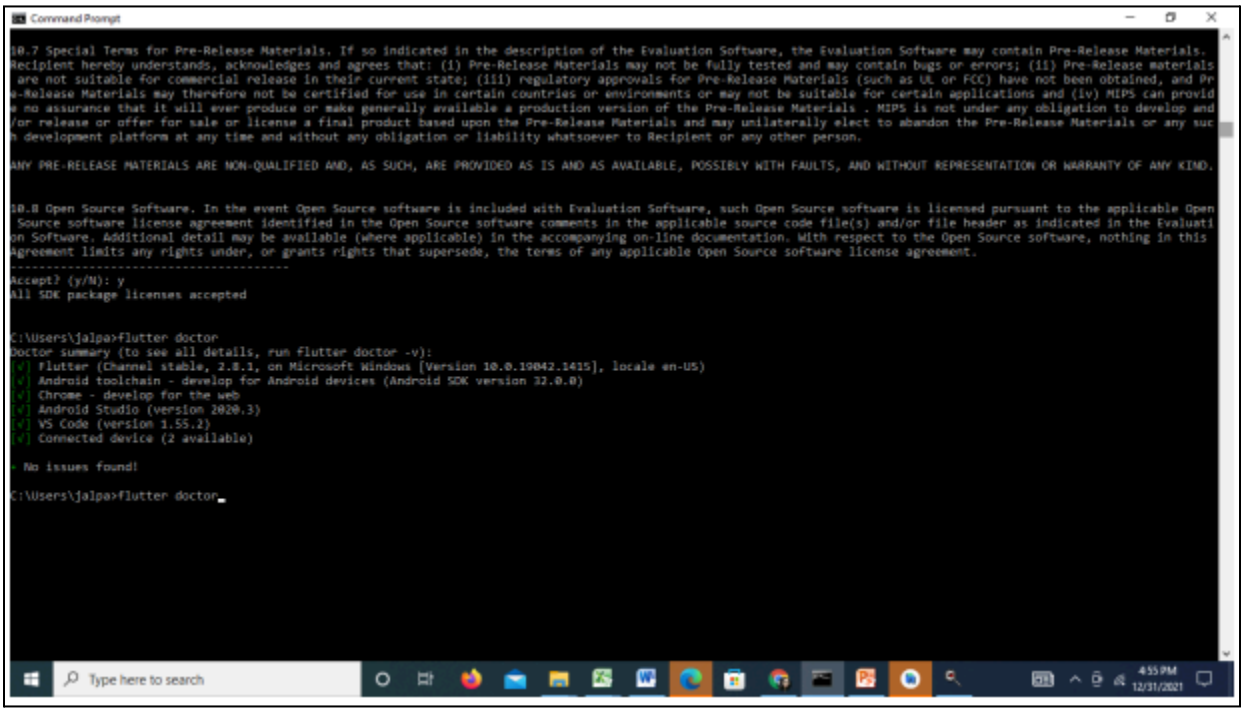
Step 7.3: Follow the steps of the installation wizard. Once the installation wizard completes, you will get the following screen.



Step 7.4: In the above screen, click Next-> Finish. Once the Finish button is clicked, you need to choose the 'Don't import Settings option' and click OK. It will start the Android Studio.

Step 7.5 run the $ flutter doctor command and Run flutter doctor --android-licenses command.

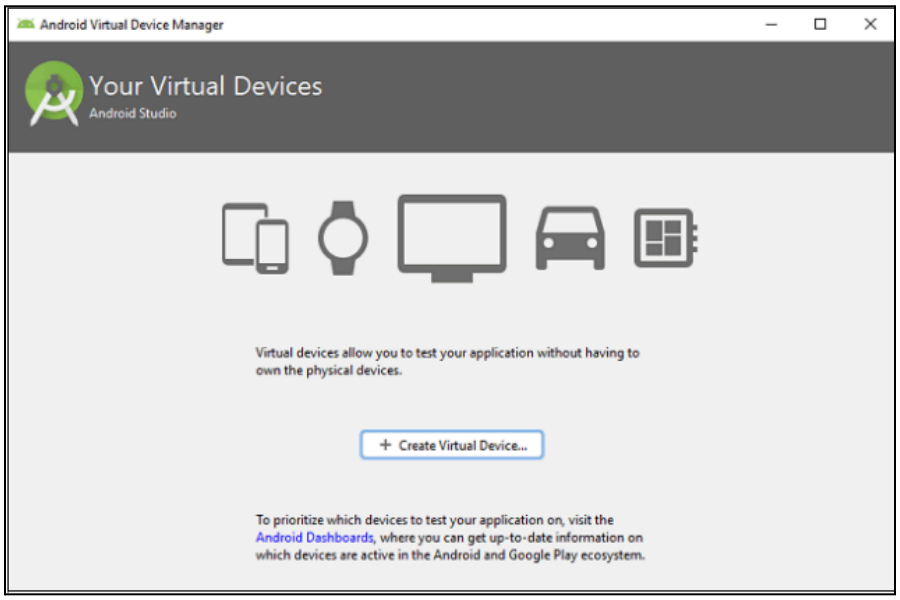




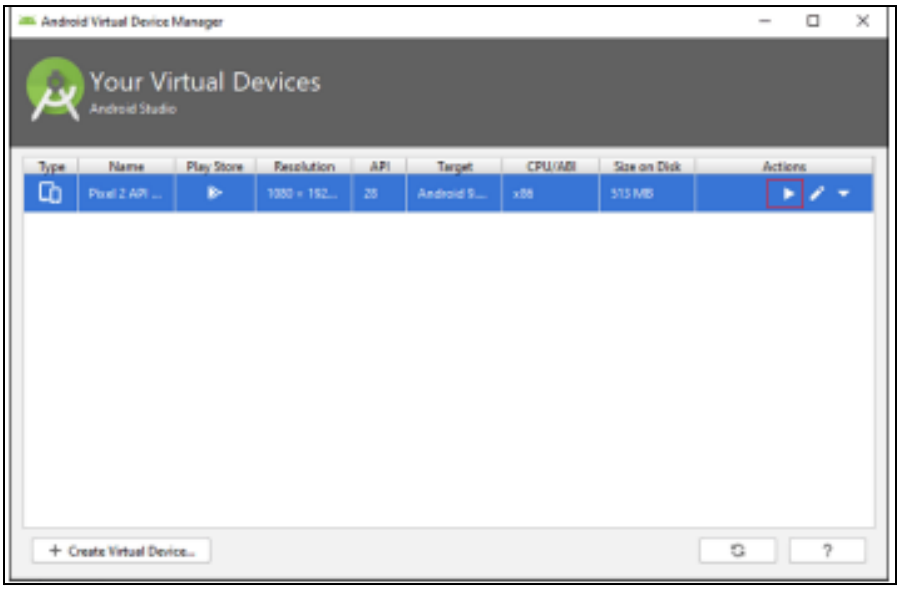
Step 8: Next, you need to set up an Android emulator. It is responsible for running and testing the Flutter application.

Step 8.1: To set an Android emulator, go to Android Studio > Tools > Android > AVD Manager and select Create Virtual Device. Or, go to Help->Find Action->Type Emulator in the search

box. You will get the following screen.



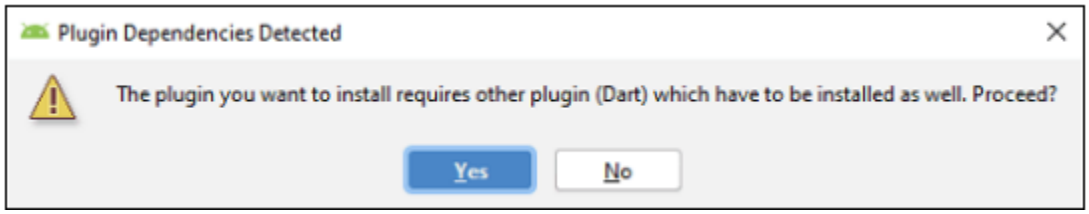
Step 8.2: Choose your device definition and click on Next.
Step 8.3: Select the system image for the latest Android version and click on Next.
Step 8.4: Now, verify the all AVD configuration. If it is correct, click on Finish. The following screen appears.



Step 8.5: Last, click on the icon pointed into the red color rectangle. The Android emulator displayed as below screen.

Step 9: Now, install Flutter and Dart plugin for building Flutter applications in Android Studio. These plugins provide a template to create a Flutter application, give an option to run and debug Flutter application in the Android Studio itself. Do the following steps to install these plugins.

Step 9.1: Open the Android Studio and then go to File->Settings->Plugins.
Step 9.2: Now, search the Flutter plugin. If found, select Flutter plugin and click install. When you click on install, it will ask you to install Dart plugin as below screen. Click yes to proceed.



Step 9.3: Restart the Android Studio.

STEPS FOR CREATING AN APP:

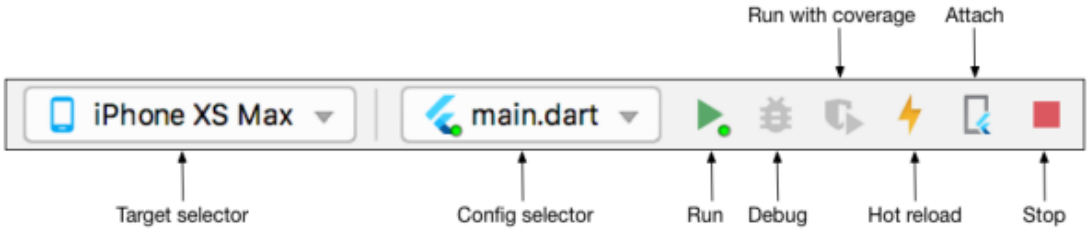
Step 1 : Create the app

1. Open the IDE and select Create New Flutter Project.
2. Select Flutter Application as the project type. Then click Next.
3. Verify the Flutter SDK path specifies the SDK‟s location (select Install SDK... if the text field is blank).
4. Enter a project name (for example, myapp). Then click Next.
5. Click Finish.
6. Wait for Android Studio to install the SDK and create the project.

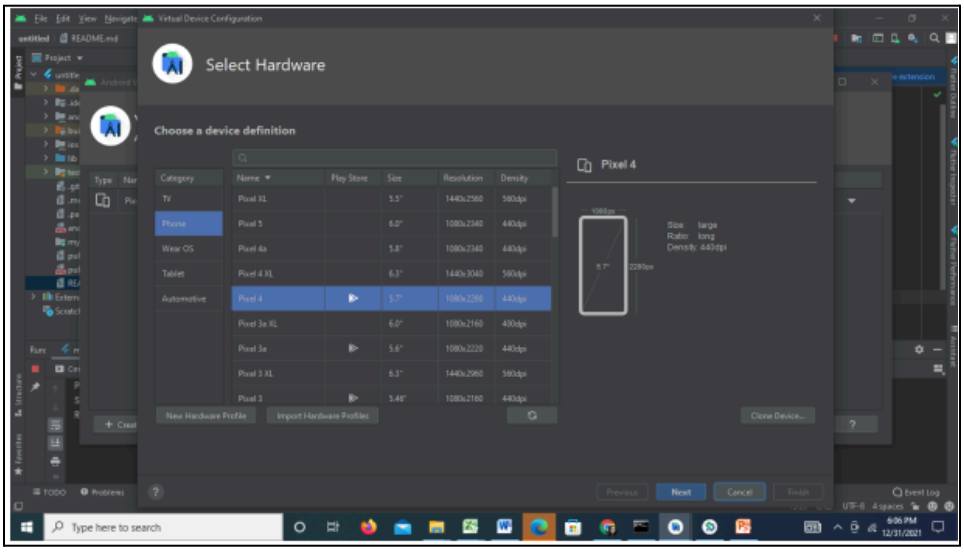
Step 2: Run the app

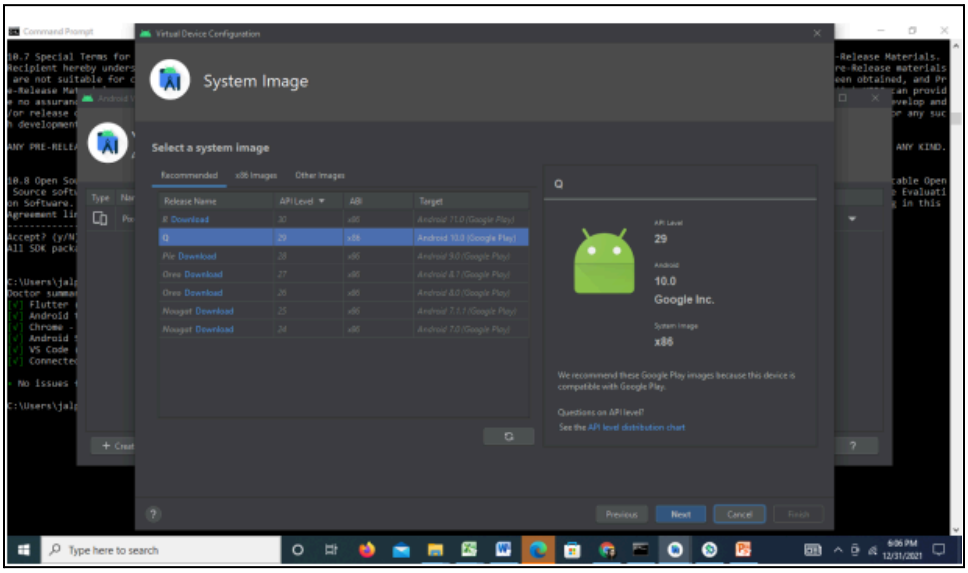
Project Title: Discord Flutter Clone/ PWA : To Do List **Roll No. 52**

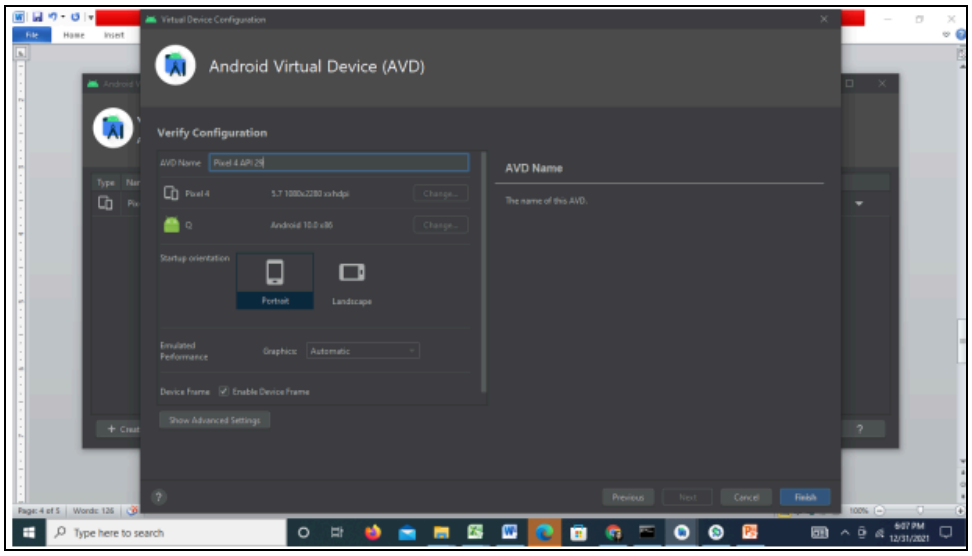
1. Locate the main Android Studio toolbar:



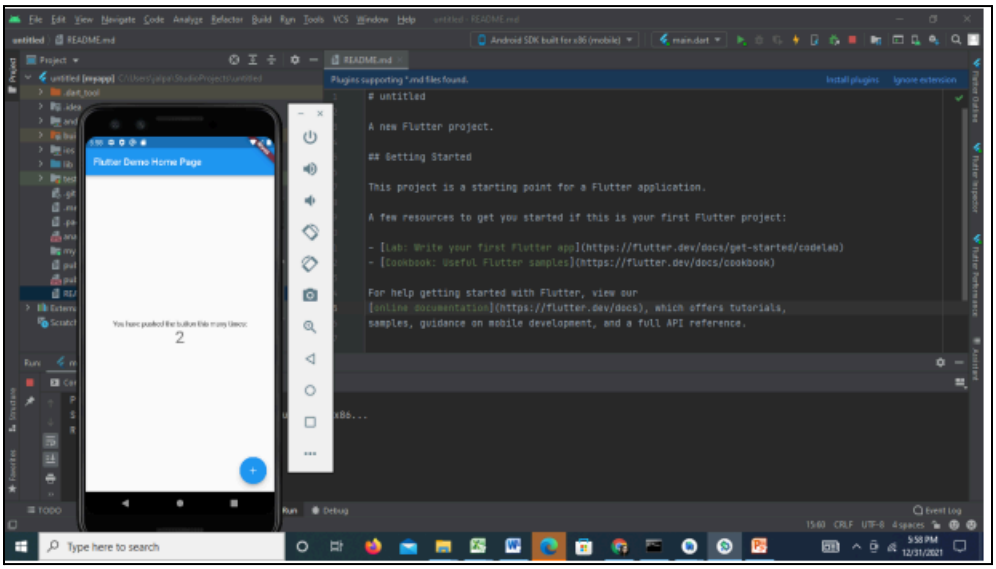
2. In the target selector, select an Android device for running the app. If none are listed as available, select Tools > AVD Manager and create one there.







3. Click the run icon in the toolbar, or invoke the menu item Run > Run.



Step 3 : Creating app

1. Replace the contents of lib/main.dart. - Delete all of the code from lib/main.dart.
2. Replace with the following code, which displays “Susmita Santi” in the center of the screen.

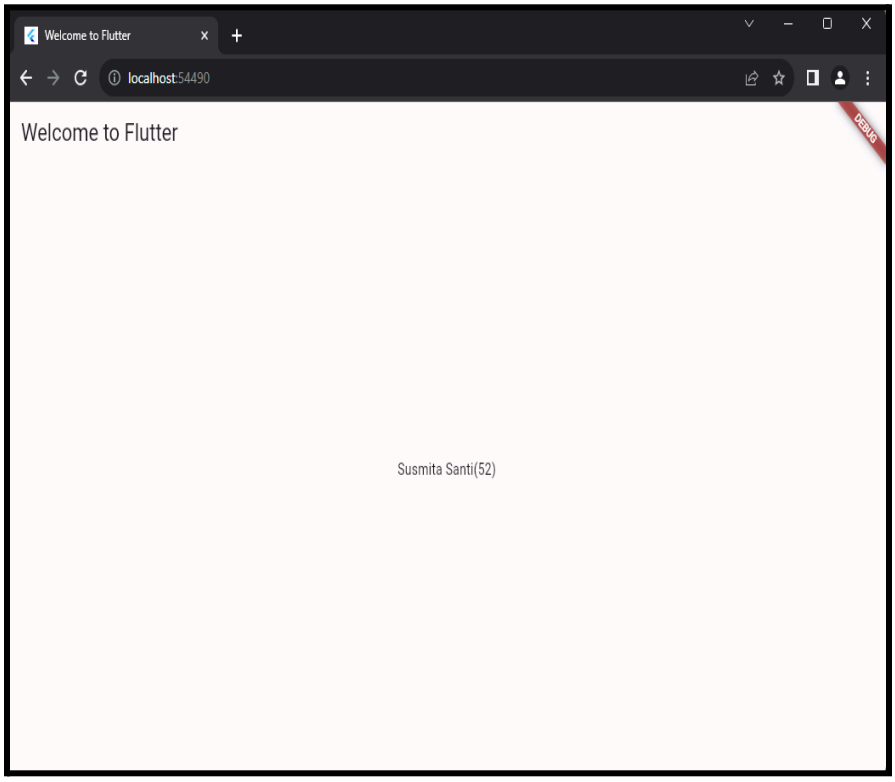
CODE:

//main.dart

```
import 'package:flutter/material.dart';
void main() {
runApp(const MyApp());
}
class MyApp extends StatelessWidget {
const MyApp({Key? key}) : super(key: key);
@override
Widget build(BuildContext context) {
return MaterialApp(
title: 'Welcome to Flutter',
home: Scaffold(
appBar: AppBar(
title: const Text('Welcome to Flutter'),
),
body: const Center(
child: Text('Susmita Santi'),
),
),
);
}
}
```

Project Title: Discord Flutter Clone/ PWA : To Do List **Roll No. 52**

OUTPUT:



CONCLUSION: We have successfully installed Flutter and implemented the first "Hello World" using Flutter

MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	52
Name	Susmita Santi
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

Project Title: Discord Flutter Clone/ PWA : To Do List **Roll No. 52**

NAME: SUSMITA SANTI **ROLL NO.:52**
SUBJECT: MAD LAB **CLASS:D15A/BATCH C**

EXP 2

AIM: To design Flutter UI by including common widgets.

THEORY:
Designing a Flutter UI involves using a variety of widgets – the basic building blocks of a Flutter application. Widgets in Flutter are organized into a tree, which can be thought of as the blueprint for the UI.

Designing a Flutter UI involves using a variety of widgets – the basic building blocks of a Flutter application. Widgets in Flutter are organized into a tree, which can be thought of as the blueprint for the UI. Here's a theoretical overview to help you understand the process:

Widgets:

Flutter uses widgets for everything. They describe what their view should look like given their current configuration and state.

Types of Widgets:

- Stateless Widgets: Unchanging widgets that do not depend on any data change or user interaction.
- Stateful Widgets: Dynamic widgets that manage state and can change during runtime based on user interaction or data changes.

Basic Widgets:

- Text: For displaying a string of text with style.
- Column and Row: For creating flexible layouts in the vertical and horizontal directions respectively.
- Container: A multi-purpose widget which can be used for padding, margins, borders, or to give a specific height/width to a widget.
- Image: For displaying images.
- Icon: To show icons from a predefined set of icons like Material or Cupertino.
- Form, TextField: For input forms and text fields.
- Buttons (FlatButton, RaisedButton, IconButton, etc.): For user interactions.

App Structure Widgets:

Scaffold: Provides the basic material design visual layout structure.

CODE:

// **main.dart**

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Discord',
      themeMode: ThemeMode.dark,
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ),
      darkTheme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity: VisualDensity.adaptivePlatformDensity,
        scaffoldBackgroundColor: const Color.fromARGB(255, 21, 21, 21),
        appBarTheme: const AppBarTheme(
          color: Colors.black,
        ),
      ),
      home: const HomePage(),
    );
  }
}
```

// home.dart

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

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

  @override
```

```
Widget build(BuildContext context) {
  // Get screen width
  double screenWidth = MediaQuery.of(context).size.width;

  return Scaffold(
    body: SingleChildScrollView(
      child: Column(
        children: [
          // Profile Banner Image
          Container(
            width: screenWidth,
            height: 200,
            decoration: const BoxDecoration(
              image: DecorationImage(
                image: AssetImage('assets/images/banner1.jpg'),
                fit: BoxFit.cover,
              ),
            ),
            child: Stack(
              children: [
                Container(
                  // Profile Image
                  padding: const EdgeInsets.fromLTRB(10, 50, 10, 10),
                  alignment: Alignment.bottomLeft,
                  child: const CircleAvatar(
                    backgroundImage:
                        AssetImage('assets/images/profile_image.jpg'),
                    radius: 40,
                  ),
                ),

                // Settings Icon
                Positioned(
                  top: 10,
                  right: 10,
                  child: Container(
                    decoration: BoxDecoration(
                      color: Colors.grey[400],
                      shape: BoxShape.circle,
                    ),
                    child: IconButton(
                      icon: const Icon(Icons.settings, color: Colors.white),
                      onPressed: () {},
                    ),
```

```
          ),
        ),
      ],
    ),
  ),

  // Bio Section
  Container(
    margin: const EdgeInsets.all(16.0),
    padding: const EdgeInsets.all(16.0),
    decoration: BoxDecoration(
      color:
          const Color.fromARGB(255, 39, 38, 38), // Dark shade color
      borderRadius: BorderRadius.circular(10.0), // Rounded borders
    ),
    alignment: Alignment.centerLeft,
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        const Text(
          'Susmita Santi',
          style: TextStyle(
            fontSize: 20,
            color: Colors.white,
            fontWeight: FontWeight.bold,
          ),
        ),
        const SizedBox(height: 4),
        const Text(
          'susmitasanti04',
          style: TextStyle(
            fontSize: 16,
            color: Colors.grey,
          ),
        ),
        const SizedBox(height: 10),
        const Text(
          'This is a short bio of myself. It like listening to music.',
          style: TextStyle(
            fontSize: 16,
            color: Colors.grey,
          ),
        ),
        const SizedBox(height: 10),
```

```
        Row(
          children: [
            Expanded(
              child: ElevatedButton.icon(
                onPressed: () {},
                icon: const Icon(Icons.edit),
                label: const Text('Edit Status'),
                style: ElevatedButton.styleFrom(
                  backgroundColor: Colors.grey.shade300,
                  foregroundColor: Colors.black,
                ),
              ),
            ),
            const SizedBox(width: 10),
            Expanded(
              child: ElevatedButton.icon(
                onPressed: () {},
                icon: const Icon(Icons.person),
                label: const Text('Edit Profile'),
                style: ElevatedButton.styleFrom(
                  backgroundColor: Colors.grey.shade300,
                  foregroundColor: Colors.black,
                ),
              ),
            ),
          ],
        ),
      ],
    ),
  ),

  // About Me Section
  Container(
    margin: const EdgeInsets.only(
        left: 16.0, right: 16.0, top: 8.0, bottom: 16.0),
    padding: const EdgeInsets.all(16.0),
    decoration: BoxDecoration(
      color: const Color.fromARGB(255, 39, 38, 38),
      borderRadius: BorderRadius.circular(10.0),
    ),
    alignment: Alignment.centerLeft,
    child: const Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
```

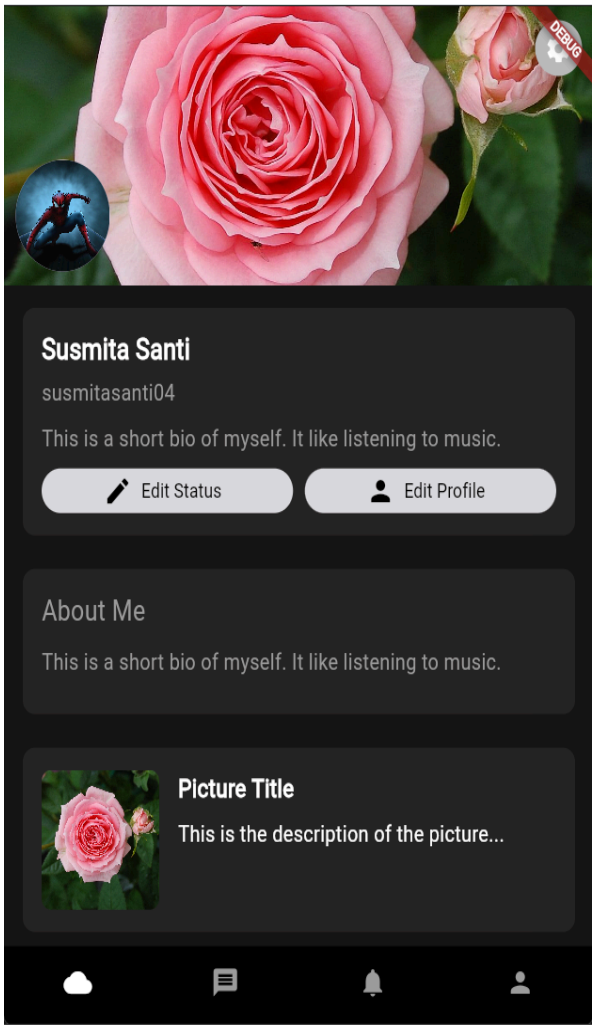
```
        Text(
          'About Me',
          style: TextStyle(
            fontSize: 20,
            color: Colors.grey,
          ),
        ),
        SizedBox(height: 10),
        Text(
          'This is a short bio of myself. It like listening to music.',
          style: TextStyle(
            fontSize: 16,
            color: Colors.grey,
          ),
        ),
        SizedBox(height: 10),
      ],
    ),
  ),

  // Picture and It's Description Section
  Container(
    margin: const EdgeInsets.only(
        left: 16.0, right: 16.0, top: 8.0, bottom: 16.0),
    padding: const EdgeInsets.all(16.0),
    decoration: BoxDecoration(
      color: const Color.fromARGB(255, 39, 38, 38),
      borderRadius: BorderRadius.circular(10.0),
    ),
    child: Row(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        // Picture
        Container(
          width: 100.0,
          height: 100.0,
          decoration: BoxDecoration(
            image: const DecorationImage(
              image: AssetImage('assets/images/banner1.jpg'),
              fit: BoxFit.cover,
            ),
            borderRadius: BorderRadius.circular(8.0),
          ),
        ),
```

```
            const SizedBox(width: 16.0),

            // Picture Description
            const Expanded(
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text(
                    'Picture Title',
                    style: TextStyle(
                      fontSize: 18.0,
                      fontWeight: FontWeight.bold,
                      color: Colors.white,
                    ),
                  ),
                  SizedBox(height: 8.0),
                  Text(
                    'The description of Rose in the corresponding picture.',
                    style: TextStyle(
                      fontSize: 16.0,
                      color: Colors.white,
                    ),
                  ),
                ],
              ),
            ),
          ],
        ),
      ),
    ],
  ),
),

// Bottom navigation bar
bottomNavigationBar: BottomNavigationBar(
  type: BottomNavigationBarType.fixed,
  backgroundColor: Colors.black87,
  selectedItemColor: Colors.white,
  unselectedItemColor: Colors.grey,
  showSelectedLabels: false,
  showUnselectedLabels: false,
  items: const [
    BottomNavigationBarItem(
```

```
            icon: Icon(Icons.cloud),
            label: 'Servers',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.message),
            label: 'Messages',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.notifications),
            label: 'Notifications',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.person),
            label: 'You',
          ),
        ],
      ),
    );
  }
}
```

Project Title: Discord Flutter Clone/ PWA : To Do List **Roll No. 52**

SCREENSHOT:



CONCLUSION:

Flutter's widget-based architecture offers great flexibility for building complex UIs. Understanding key widgets and concepts is essential for effective Flutter development.

MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	52
Name	Susmita Santi
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

NAME: SUSMITA SANTI ROLL NO.:52
SUBJECT: MAD LAB CLASS:D15A/BATCH C

EXP 3

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

THEORY:

Including icons, images, and custom fonts in a Flutter app is a common requirement to enhance the visual appeal and functionality of the application. Let's discuss each of these elements:

1. Icons:

Icons in Flutter are typically represented by the Icon widget. Flutter provides a set of built-in icons through the Icons class, but you can also use custom icons or those from external icon packs.

How to Include Icons:

Built-in Icons:
Icon(Icons.star);

Custom Icons:
Flutter allows you to use custom icons in various formats, such as SVG or PNG. You can use the Image.asset or Image.network widget to display custom icons.

2. Images:

Displaying images is a crucial part of app development. Flutter supports various image formats, including PNG, JPEG, GIF, and WebP.

How to Include Images:

Asset Images:

- Place your images in the assets folder of your project, and then use the Image.asset widget to display them.

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

3. Fonts:

Custom fonts allow you to create a unique typographic style for your app. Flutter supports TrueType (TTF) and OpenType (OTF) fonts.

How to Include Fonts:

Adding Fonts to pubspec.yaml:
Add your font files to the fonts section of the pubspec.yaml file.

```
flutter:
  fonts:
    - family: MyCustomFont
      fonts:
```

```
    - asset: assets/fonts/my_custom_font.ttf
```

Using Custom Fonts:
Apply custom fonts using the Text widget.

```
Text(
  'Hello, World!',
  style: TextStyle(
    fontFamily: 'MyCustomFont',
    fontSize: 20,
  ),
);
```

CODE:

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

class ChatPage extends StatefulWidget {
  const ChatPage({Key? key}) : super(key: key);

  @override
  _ChatPageState createState() => _ChatPageState();
}

class _ChatPageState extends State<ChatPage> {
  final TextEditingController _messageController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.black,
      appBar: AppBar(
        backgroundColor: Colors.black,
        title: Row(
          children: [
            const Text('Discord'),
            const SizedBox(width: 8.0),
            const Icon(Icons.arrow_drop_down),
          ],
        ),
        actions: [
          IconButton(
```

```
          onPressed: () {},
          icon: const Icon(Icons.search),
        ),
        IconButton(
          onPressed: () {},
          icon: const Icon(Icons.add),
        ),
        IconButton(
          onPressed: () {},
          icon: const Icon(Icons.settings),
        ),
      ],
    ),
    body: Column(
      children: [
        // Chat Messages
        Expanded(
          child: ListView.builder(
            reverse: true,
            itemCount: 20,
            itemBuilder: (context, index) {
              return ChatMessage(
                isCurrentUser: index % 2 == 0,
                username: 'Username $index',
                message: 'Message $index',
                index: index,
              );
            },
          ),
        ),

        // Chat Input Field
        Container(
          padding: const EdgeInsets.symmetric(horizontal: 8.0),
          color: Colors.black,
          child: Row(
            children: [
              IconButton(
                onPressed: () {
                  // Action for the plus icon
```

```
      },
      icon: const Icon(Icons.add, color: Colors.white),
    ),
    IconButton(
      onPressed: () {
        // Action for the gift icon
      },
      icon: const Icon(Icons.card_giftcard, color: Colors.white),
    ),
    Expanded(
      child: TextField(
        controller: _messageController,
        style: TextStyle(color: Colors.white), // Text color
        decoration: InputDecoration(
          hintText: 'Type your message...',
          hintStyle:
              TextStyle(color: Colors.white54), // Hint text color
          fillColor: Colors.grey[850],
          filled: true,
          border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(20.0),
            borderSide: BorderSide.none,
          ),
        ),
      ),
    ),
    IconButton(
      onPressed: () {
        // Action for the emoji icon
      },
      icon: const Icon(Icons.emoji_emotions_outlined,
          color: Colors.white),
    ),
    IconButton(
      onPressed: () {
        // Action for the mic icon
      },
      icon: const Icon(Icons.mic_none, color: Colors.white),
    ),
  ],
```

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

class ChatMessage extends StatelessWidget {
 final bool isCurrentUser;
 final String username;
 final String message;
 final int index;

 const ChatMessage({
  Key? key,
  required this.isCurrentUser,
  required this.username,
  required this.message,
  required this.index,
 }) : super(key: key);

 Color getUsernameColor(int index) {
  // You can customize this logic to generate different colors based on the index
  return index % 2 == 0 ? Colors.blue : Colors.green;
 }

 @override
 Widget build(BuildContext context) {
  return Container(
   margin: const EdgeInsets.symmetric(vertical: 8.0, horizontal: 16.0),
   child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
     Row(
      children: [
       Container(
        width: 32.0, // Overall size of the container
        height: 32.0,
        decoration: BoxDecoration(
```

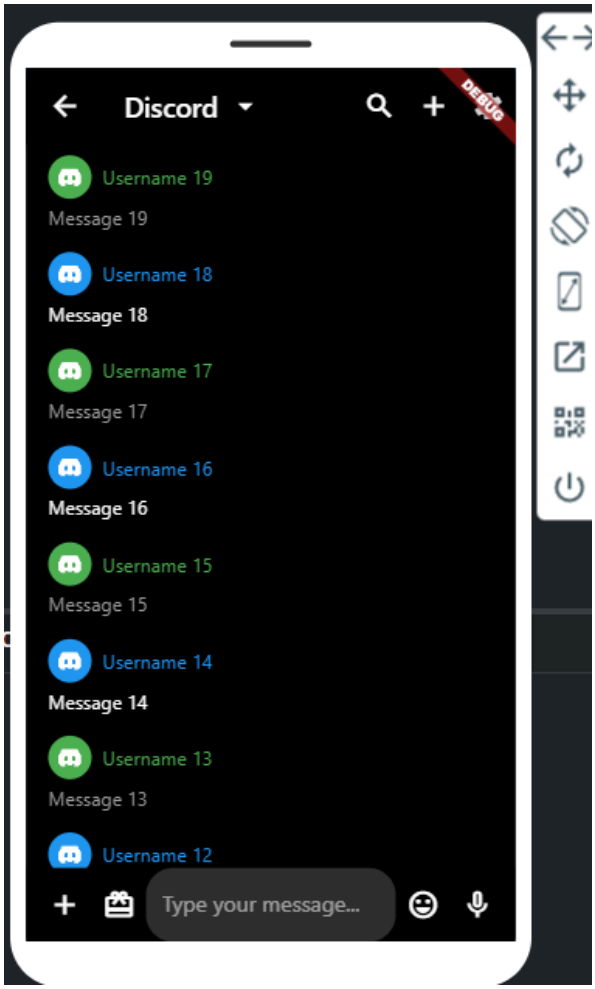
```
              color: getUsernameColor(index), // Background color
              shape: BoxShape.circle,
            ),
            child: Center(
              child: Image.asset(
                'assets/images/discord_logo1.png', // Make sure you have this asset in your project
                width: 25.0, // Size of the logo
                height: 25.0,
                color: Colors.white,
              ),
            ),
          ),
          const SizedBox(width: 8.0),
          Text(
            username,
            style: TextStyle(
              color: isCurrentUser ? Colors.blue : getUsernameColor(index),
            ),
          ),
        ],
      ),
      const SizedBox(height: 4.0),
      Text(
        message,
        style: TextStyle(
          color: isCurrentUser ? Colors.white : Colors.grey,
        ),
      ),
    ],
  ),
 );
 }
}
```

//pubspec.yaml

```
name: discord
description: A new Flutter project.
version: 1.0.0+1
environment:
  sdk: ">=3.0.0 <4.0.0"
dependencies:
```

```
  flutter:
    sdk: flutter

dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_lints: ^2.0.2

flutter:
  uses-material-design: true
  assets:
    - assets/images/discord_logo1.png
    - assets/images/banner1.jpg
    - assets/images/profile_image1.jpg
    - assets/images/profile_image.jpg
```

OUTPUT



CONCLUSION:

In conclusion, enhancing the visual appeal and functionality of a Flutter app involves incorporating icons, images, and custom fonts.

1. Icons: Utilize the Icon widget for built-in icons and explore custom icons using various formats.
2. Images: Display images with the Image.asset widget for local assets or Image.network for network images.
3. Fonts: Include custom fonts in the pubspec.yaml file and apply them using the TextStyle property in the Text widget.

MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	52
Name	Susmita Santi
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

NAME: SUSMITA SANTI **ROLL NO.:52**
SUBJECT: MAD LAB **CLASS:D15A/BATCH C**

EXP 4

AIM: To create an interactive Form using form widget

THEORY:

1. Form

The Form widget is used to group multiple form fields (e.g., TextFormField) and provides a way to validate all the fields in a single go. It uses a GlobalKey<FormState> to uniquely identify the form and manage its state, including validation.

2. TextFormField

TextFormField widgets are used to accept input from the user. In this example, there are two TextFormField widgets: one for the username and another for the password. Each field is styled with custom text colors and an outline border. Validators are attached to each field to ensure that they are not left empty.

3. Validators

Validators are functions that check the input of form fields for errors. If the input is invalid, a validator returns an error string that is displayed below the field. If the input is valid, it returns null. In this code, simple validators are used to ensure that the username and password fields are not left empty.

4. ElevatedButton

An ElevatedButton is used to trigger form submission. The button's onPressed callback checks if the form is valid by using the form's validate method. If the form is valid, the app could then proceed with the login process (not implemented in this snippet).

5. TextEditingController

TextEditingController objects are used to control and listen to the text that users type into TextFormField widgets. They can be used to fetch the current value of a text field, clear its content, or listen for changes.

Key Concepts

- State Management: The use of GlobalKey to manage form state and validation is a simple example of state management in Flutter.

- Validation: Demonstrates how to enforce input rules before processing data.

CODE:

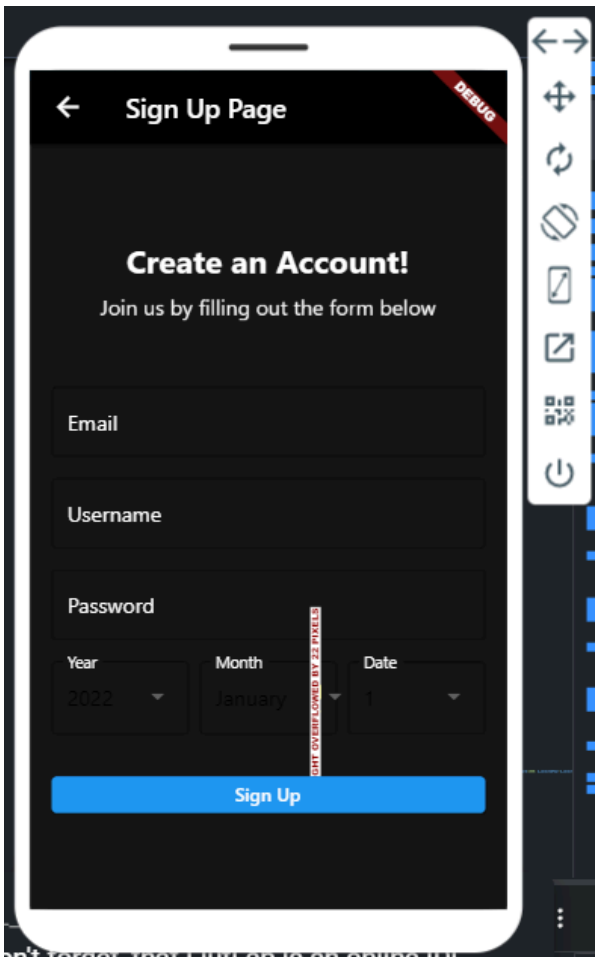
```
import 'package:flutter/material.dart';

class LoginPage extends StatelessWidget {
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
  TextEditingController _usernameController = TextEditingController();
  TextEditingController _passwordController = TextEditingController();

  @override
  Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
      title: Text('Login Page'),
     ),
     body: Padding(
      padding: EdgeInsets.all(16.0),
      child: Form(
       key: _formKey,
       child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
         Text(
          'Welcome Back!', // Welcome message
          style: TextStyle(
           fontSize: 24.0,
           fontWeight: FontWeight.bold,
           color: Colors.white,
          ),
         ),
         SizedBox(height: 8.0),
         Text(
          'Login with your email to start talking',
          style: TextStyle(
           fontSize: 16.0,
           color: Colors.white,
          ),
         ),
         SizedBox(height: 50.0),
```

```
TextFormField(
  controller: _usernameController,
  style: TextStyle(
    color: Colors.white, // Set text color to white
  ),
  decoration: InputDecoration(
    labelText: 'Username',
    labelStyle: TextStyle(
      color: Colors.white, // Set label color to white
    ),
    border:
        OutlineInputBorder(), // Set border to make it rectangular
  ),
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter your username';
    }
    return null;
  },
),
SizedBox(height: 16.0),
TextFormField(
  controller: _passwordController,
  obscureText: true,
  style: TextStyle(
    color: Colors.white, // Set text color to white
  ),
  decoration: InputDecoration(
    labelText: 'Password',
    labelStyle: TextStyle(
      color: Colors.white, // Set label color to white
    ),
    border:
        OutlineInputBorder(), // Set border to make it rectangular
  ),
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter your password';
    }
    return null;
```

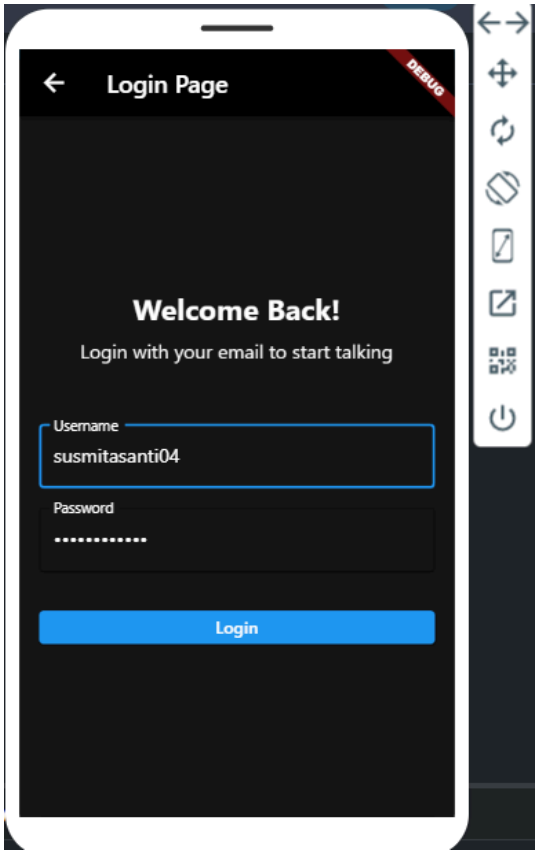
```
            },
          ),
          SizedBox(height: 32.0),
          Container(
            width: double.infinity, // Set width to match parent width
            child: ElevatedButton(
              onPressed: () {
                if (_formKey.currentState != null &&
                    _formKey.currentState!.validate()) {
                  // Form is valid, implement your login logic here
                  String username = _usernameController.text;
                  String password = _passwordController.text;
                  // Implement your authentication logic with username and password
                  print('Username: $username\nPassword: $password');
                }
              },
              child: Text('Login'),
            ),
          ),
        ],
      ),
    ),
  ),
 );
 }
}
```

Project Title: Discord Flutter Clone/ PWA : To Do List **Roll No. 52**

OUTPUT:



Project Title: Discord Flutter Clone/ PWA : To Do List **Roll No. 52**



CONCLUSION:

It utilizes GlobalKey<FormState> for form management, TextFormField for user input, and validators to ensure data validity. The ElevatedButton triggers form submission, while TextEditingController handles text input. The code introduces simple state management through GlobalKey, showcasing essential Flutter concepts for creating a functional form.

MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	52
Name	Susmita Santi
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	15

NAME: SUSMITA SANTI **ROLL NO.:52**
SUBJECT: MAD LAB **CLASS:D15A/BATCH C**

EXP 5

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

THEORY:

Navigation: In Flutter, navigation refers to the ability to move between different screens or "routes" within your app. Flutter provides a Navigator class which manages a stack of Route objects. Each route typically corresponds to a different screen or page in your app. You can push new routes onto the navigator's stack to navigate forward, and pop routes off the stack to navigate backward. Navigation can be triggered by user actions such as tapping on buttons or by programmatically invoking navigation methods.

Routing: Routing in Flutter involves defining the routes for different screens in your app and handling navigation between these routes. You can define routes using the MaterialApp widget's routes parameter, which maps route names to builder functions that create the corresponding screens. For example, you can define a route named "/home" that maps to a function that builds the home screen widget. When you want to navigate to a particular route, you use the Navigator to push that route onto the stack.

Gestures: Gestures in Flutter refer to user interactions such as tapping, swiping, dragging, pinching, etc. Flutter provides a rich set of widgets and classes for handling gestures, such as GestureDetector, InkWell, Draggable, Dismissible, etc. These widgets allow you to detect various types of gestures and respond to them accordingly. For exple, you can wrap a widget with a GestureDetector to detect taps and then perform some action in response to the tap event.

To apply these concepts in your Flutter app, you would typically do the following:

- Define the routes for different screens in your app using the MaterialApp widget's routes parameter.
- Implement navigation logic to navigate between screens using the Navigator class, either in response to user actions or programmatically.
- Use gesture detection widgets like GestureDetector to detect user gestures and trigger navigation or other actions in response to those gestures.

CODE:
//home.dart

```
import 'package:discord/pages/login.dart';
import 'package:flutter/material.dart';
import 'package:discord/pages/sidebar.dart';
import 'package:discord/pages/chatpage.dart';

class HomePage extends StatefulWidget {
  const HomePage({Key? key}) : super(key: key);

  @override
  _HomePageState createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  bool _isSidebarVisible = false;
  int _selectedIndex = 0;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.black,
        leading: GestureDetector(
          onTap: () {
            _toggleSidebarVisibility();
          },
          child: const Icon(Icons.menu),
        ),
        title: const Text('Home Page'),
      ),
      body: Row(
        children: [
          // Slim Sidebar
          Visibility(
            visible: _isSidebarVisible,
            child: Container(
              width: 80,
              color: Colors.grey[900],
              child: ListView(
                children: [
                  ChannelTile(imagePath: 'assets/images/profile_image1.jpg'),
```

```
        ChannelTile(imagePath: 'assets/images/banner1.jpg'),
        ChannelTile(imagePath: 'assets/images/profile_image.jpg'),
        // Add more ChannelTiles as needed
      ],
    ),
  ),
),

// Main Content
Expanded(
  child: SingleChildScrollView(
    child: Column(
      children: [
        // Profile Banner Image
        Container(
          width: MediaQuery.of(context).size.width,
          height: 200,
          decoration: const BoxDecoration(
            image: DecorationImage(
              image: AssetImage('assets/images/banner1.jpg'),
              fit: BoxFit.cover,
            ),
          ),
          child: Stack(
            children: [
              Container(
                // Profile Image
                padding: const EdgeInsets.fromLTRB(10, 50, 10, 10),
                alignment: Alignment.bottomLeft,
                child: const CircleAvatar(
                  backgroundImage:
                      AssetImage('assets/images/profile_image1.jpg'),
                  radius: 40,
                ),
              ),
              // Settings Icon
              Positioned(
                top: 10,
                right: 10,
                child: Container(
```

```
            decoration: BoxDecoration(
              color: Colors.grey[400],
              shape: BoxShape.circle,
            ),
            child: IconButton(
              icon: const Icon(Icons.settings,
                  color: Colors.white),
              onPressed: () {},
            ),
          ),
        ),
      ],
    ),
  ),
  // Bio Section
  Container(
    margin: const EdgeInsets.all(16.0),
    padding: const EdgeInsets.all(16.0),
    decoration: BoxDecoration(
      color: const Color.fromARGB(255, 39, 38, 38),
      borderRadius: BorderRadius.circular(10.0),
    ),
    alignment: Alignment.centerLeft,
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        const Text(
          'Susmita Santi',
          style: TextStyle(
            fontSize: 20,
            color: Colors.white,
            fontWeight: FontWeight.bold,
          ),
        ),
        const SizedBox(height: 4),
        const Text(
          'susmitasanti04',
          style: TextStyle(
            fontSize: 16,
            color: Colors.grey,
```

```
          ),
        ),
        const SizedBox(height: 10),
        const Text(
          'This is a short bio of myself. It like listening to music.',
          style: TextStyle(
            fontSize: 16,
            color: Colors.grey,
          ),
        ),
        const SizedBox(height: 10),
        Row(
          children: [
            Expanded(
              child: ElevatedButton.icon(
                onPressed: () {},
                icon: const Icon(Icons.edit),
                label: const Text('Edit Status'),
                style: ElevatedButton.styleFrom(
                  backgroundColor: Colors.grey.shade300,
                  foregroundColor: Colors.black,
                ),
              ),
            ),
            const SizedBox(width: 10),
            Expanded(
              child: ElevatedButton.icon(
                onPressed: () {},
                icon: const Icon(Icons.person),
                label: const Text('Edit Profile'),
                style: ElevatedButton.styleFrom(
                  backgroundColor: Colors.grey.shade300,
                  foregroundColor: Colors.black,
                ),
              ),
            ),
          ],
        ),
      ],
    ),
```

```
),
// About Me Section
Container(
  margin: const EdgeInsets.only(
      left: 16.0, right: 16.0, top: 8.0, bottom: 16.0),
  padding: const EdgeInsets.all(16.0),
  decoration: BoxDecoration(
    color: const Color.fromARGB(255, 39, 38, 38),
    borderRadius: BorderRadius.circular(10.0),
  ),
  alignment: Alignment.centerLeft,
  child: const Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      Text(
        'About Me',
        style: TextStyle(
          fontSize: 20,
          color: Colors.grey,
        ),
      ),
      SizedBox(height: 10),
      Text(
        'This is a short bio of myself. It like listening to music.',
        style: TextStyle(
          fontSize: 16,
          color: Colors.grey,
        ),
      ),
      SizedBox(height: 10),
    ],
  ),
),
// Picture and Its Description Section
Container(
  margin: const EdgeInsets.only(
      left: 16.0, right: 16.0, top: 8.0, bottom: 16.0),
  padding: const EdgeInsets.all(16.0),
  decoration: BoxDecoration(
    color: const Color.fromARGB(255, 39, 38, 38),
```

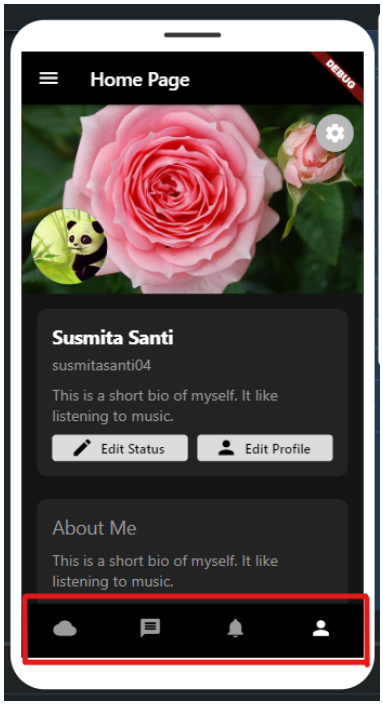
```
  borderRadius: BorderRadius.circular(10.0),
),
child: Row(
  crossAxisAlignment: CrossAxisAlignment.start,
  children: [
    // Picture
    Container(
      width: 100.0,
      height: 100.0,
      decoration: BoxDecoration(
        image: const DecorationImage(
          image: AssetImage('assets/images/banner1.jpg'),
          fit: BoxFit.cover,
        ),
        borderRadius: BorderRadius.circular(8.0),
      ),
    ),

    const SizedBox(width: 16.0),

    // Picture Description
    const Expanded(
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Text(
            'Picture Title',
            style: TextStyle(
              fontSize: 18.0,
              fontWeight: FontWeight.bold,
              color: Colors.white,
            ),
          ),
          SizedBox(height: 8.0),
          Text(
            'The description of Rose in the corresponding picture.',
            style: TextStyle(
              fontSize: 16.0,
              color: Colors.white,
            ),
```

```
                    ),
                  ],
                ),
              ),
            ),
          ],
        ),
      ),
    ),
  ),
],
),
bottomNavigationBar: BottomNavigationBar(
  type: BottomNavigationBarType.fixed,
  backgroundColor: Colors.black87,
  selectedItemColor: Colors.white,
  unselectedItemColor: Colors.grey,
  showSelectedLabels: false,
  showUnselectedLabels: false,
  currentIndex: _selectedIndex,
  onTap: (index) {
    // Handle page navigation based on index
    _handlePageNavigation(index);
  },
  items: [
    BottomNavigationBarItem(
      icon: Icon(Icons.cloud),
      label: 'Servers',
    ),
    BottomNavigationBarItem(
      icon: Icon(Icons.message),
      label: 'Messages',
    ),
    BottomNavigationBarItem(
      icon: Icon(Icons.notifications),
      label: 'Notifications',
    ),
    BottomNavigationBarItem(
      icon: Icon(Icons.person),
```

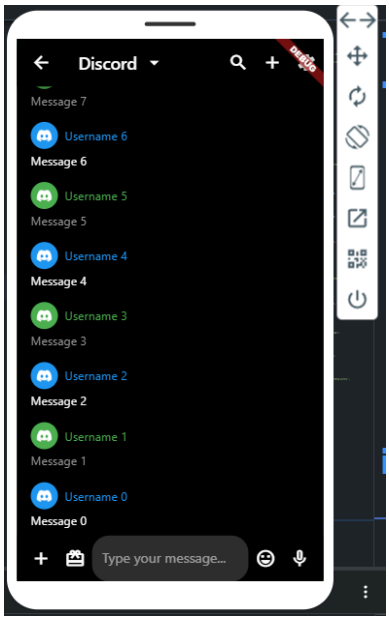
```
            label: 'You',
          ),
        ],
      ),
    );
  }

  void _handlePageNavigation(int index) {
    setState(() {
      _selectedIndex = index;
    });

    // Navigate to the selected page
    switch (_selectedIndex) {
      case 1:
        Navigator.push(
          context,
          MaterialPageRoute(builder: (context) => const ChatPage()),
        );
        break;

      case 3:
        Navigator.push(
          context,
          MaterialPageRoute(builder: (context) => LoginPage()),
        );
        break;
      // Add cases for other pages if needed
    }
  }

  void _toggleSidebarVisibility() {
    setState(() {
      _isSidebarVisible = !_isSidebarVisible;
    });
  }
}
```

SCREENSHOT:

Following is the Home Page:

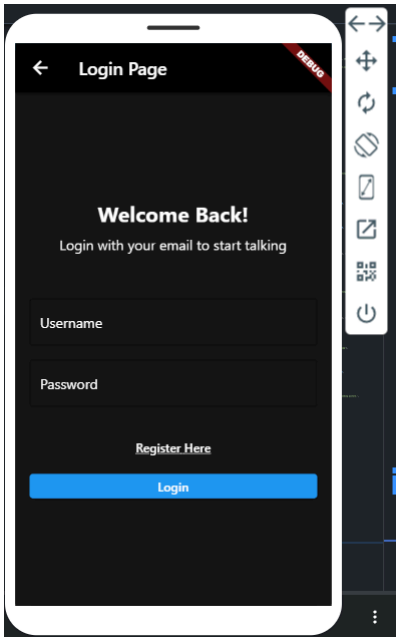


On selecting message icon, you get navigated to the following page:



On clicking Back Button, you go back to the Home Page.

Similarly, on selecting profile icon, you get navigated to the following page:



CONCLUSION:

In conclusion, navigation, routing, and gestures are essential aspects of building user-friendly and interactive Flutter apps

MAD & PWA Lab
Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	52
Name	Susmita Santi
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	15

Project Title: Discord Flutter Clone/ PWA : To Do List **Roll No. 52**

NAME: SUSMITA SANTI **ROLL NO.:52**
SUBJECT: MAD LAB **CLASS:D15A/BATCH C**

EXP 6

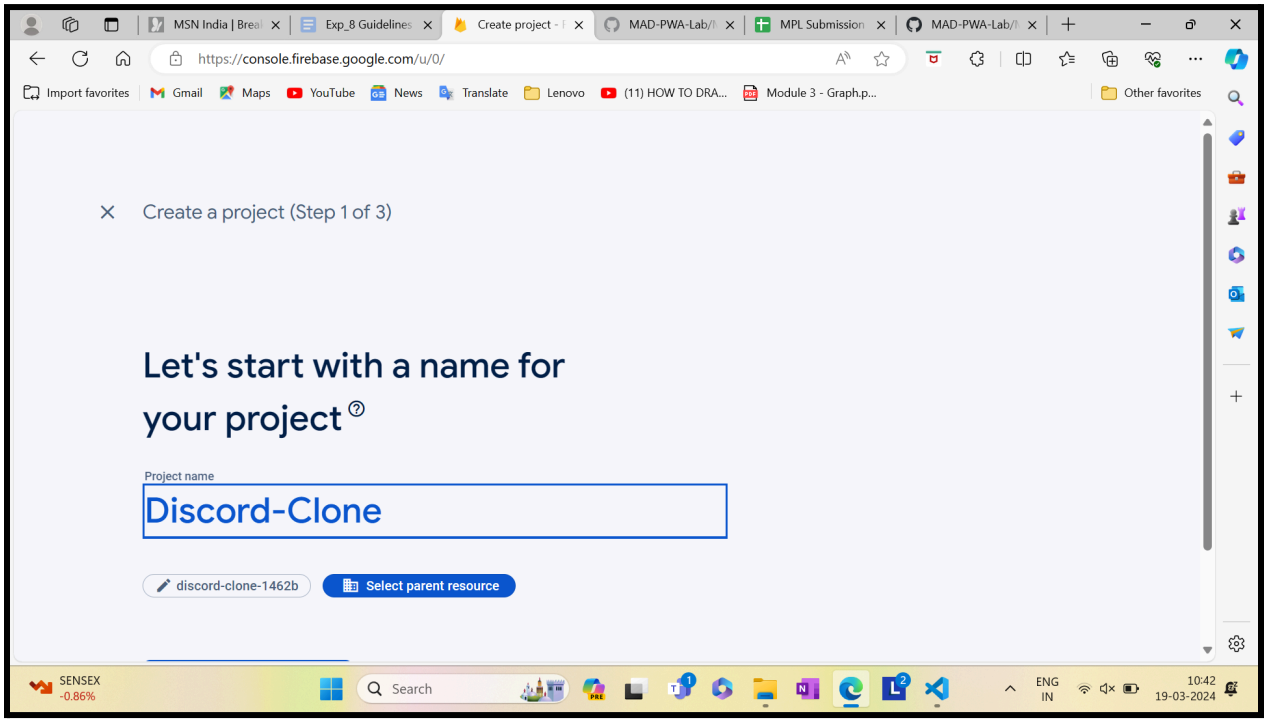
AIM: How To Set Up Firebase with Flutter for iOS and Android Apps

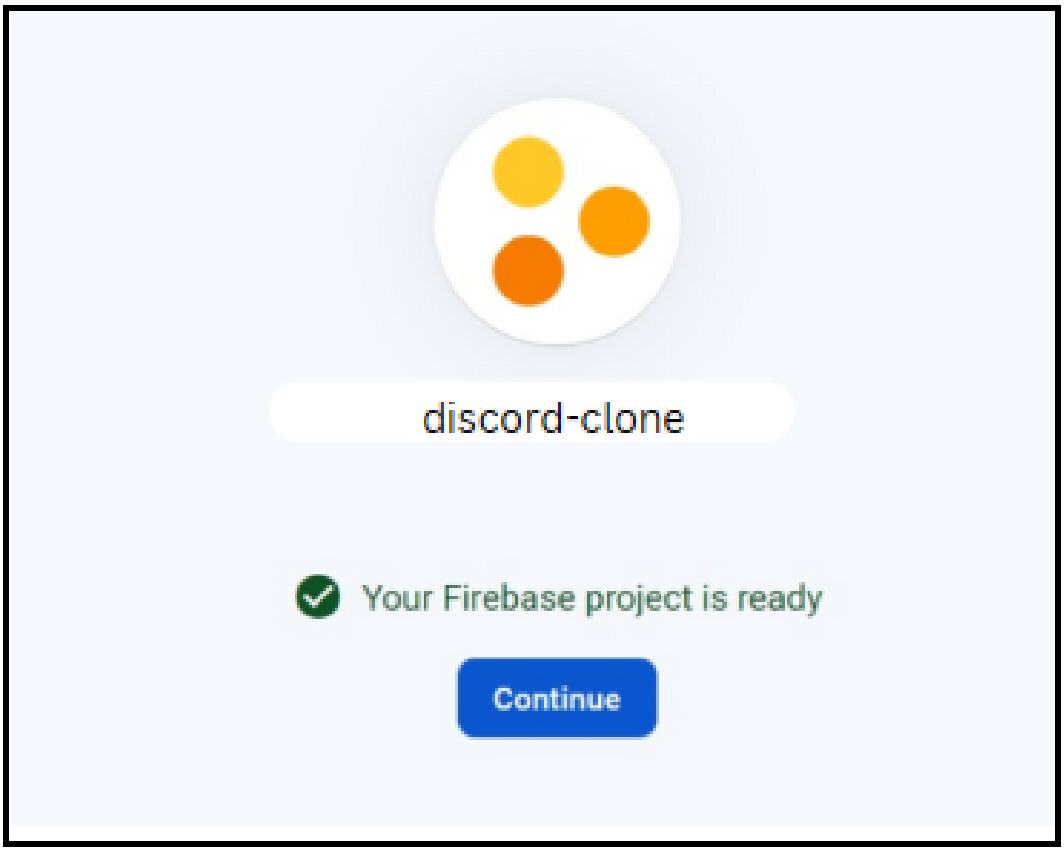
THEORY:

- Create a Firebase Project:
  - Go to the Firebase Console (https://console.firebase.google.com/) and create a new project.

Add Your App to Firebase:

- After creating your Firebase project, click on the "Add app" button.
- Choose the platform (iOS or Android) for which you want to add the app.



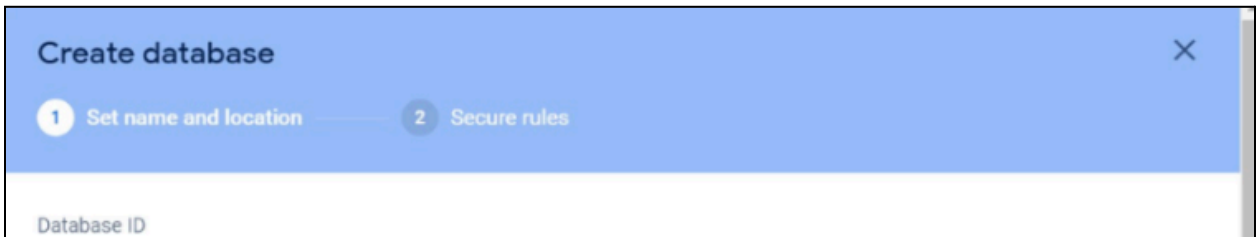


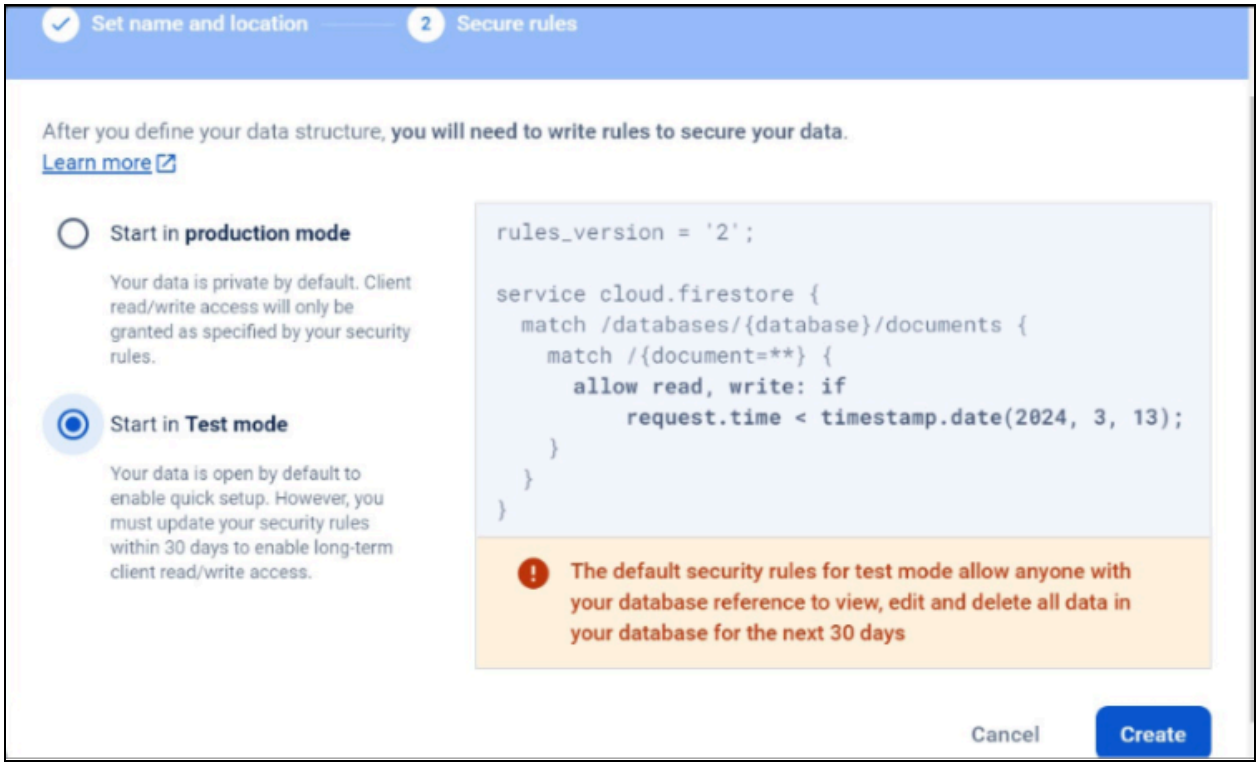
Go to the Terminal:

```
PS S:\Flutter\thread-clone-flutter> flutterfire configure
>>
i Found 2 Firebase projects.
✓ Select a Firebase project to configure your Flutter application with · discord-clone-42bfe (Discord-Clone)
✓ Which platforms should your configuration support (use arrow keys & space to select)? · android, ios
i Firebase android app com.example.thread_clone_flutter is not registered on Firebase project discord-clone-42bfe.
i Registered a new Firebase android app on Firebase project discord-clone-42bfe.
i Firebase ios app com.example.threadCloneFlutter is not registered on Firebase project discord-clone-42bfe.
i Registered a new Firebase ios app on Firebase project discord-clone-42bfe.
? Generated FirebaseAppID file S:\Flutter\thread-clone-flutter\ios\firebase_app_id_file.json already exists, do you want to o
✓ Generated FirebaseAppID file S:\Flutter\thread-clone-flutter\ios\firebase_app_id_file.json already exists, do you want to o
erride it? · yes
? The google-services.json file already exists but for a different Firebase project (thread-clone-tutorial). Do you want to r
? The google-services.json file already exists but for a different Firebase project (thread-clone-tutorial). Do you want to r
✓ The google-services.json file already exists but for a different Firebase project (thread-clone-tutorial). Do you want to r
place it with Firebase project discord-clone-42bfe? · yes

Firebase configuration file lib\firebase_options.dart generated successfully with the following Firebase apps:

Platform  Firebase App Id
android   1:969782907837:android:f9d1f3eaa99c955175f8d4
ios       1:969782907837:ios:28196051d80f51a375f8d4

Learn more about using this file and next steps from the documentation:
 > https://firebase.google.com/docs/flutter/setup
PS S:\Flutter\thread-clone-flutter>
```

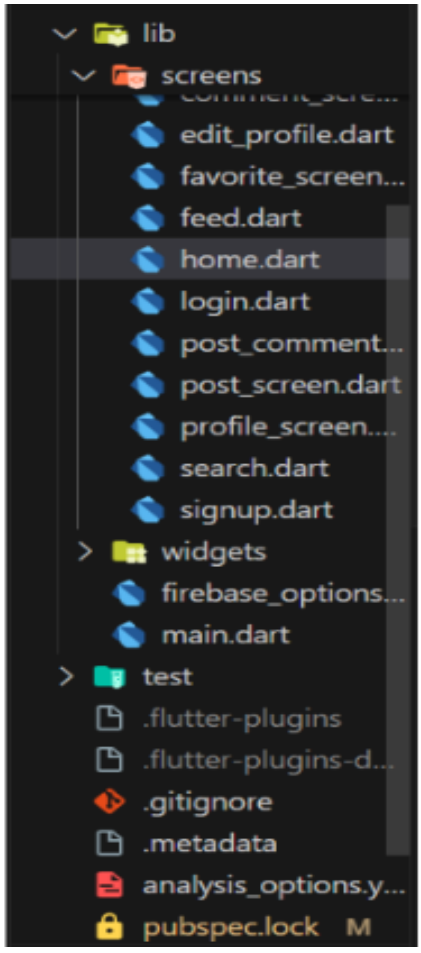
Ln 294, Col 1 Spaces: 2 UTF-8 CRLF {} Dart Go Live No Device Prettier

To create Database

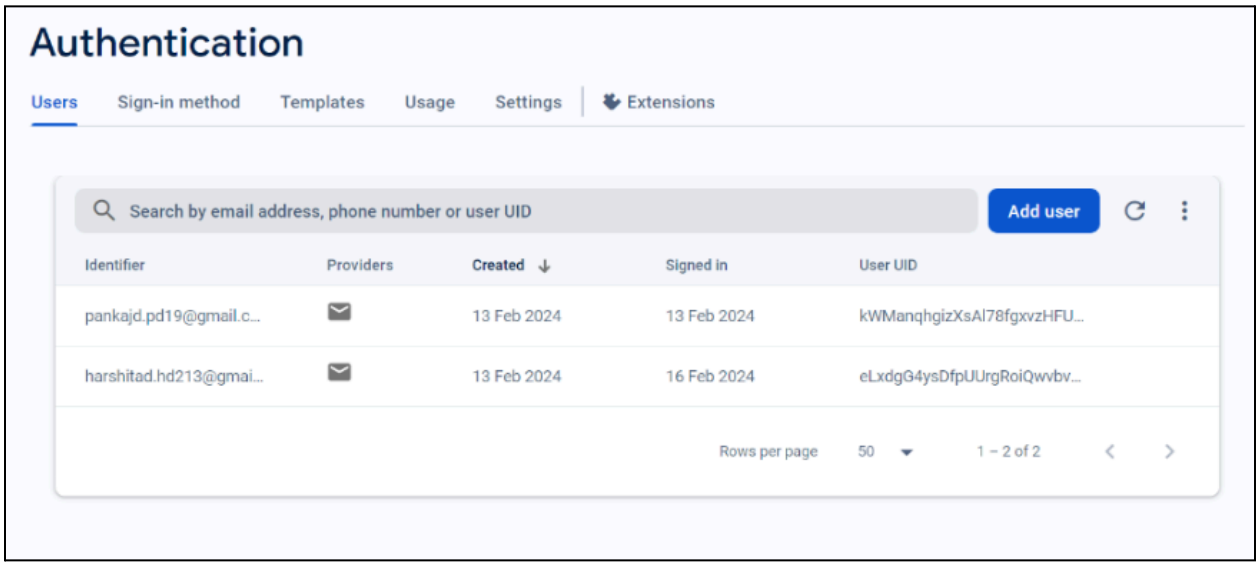


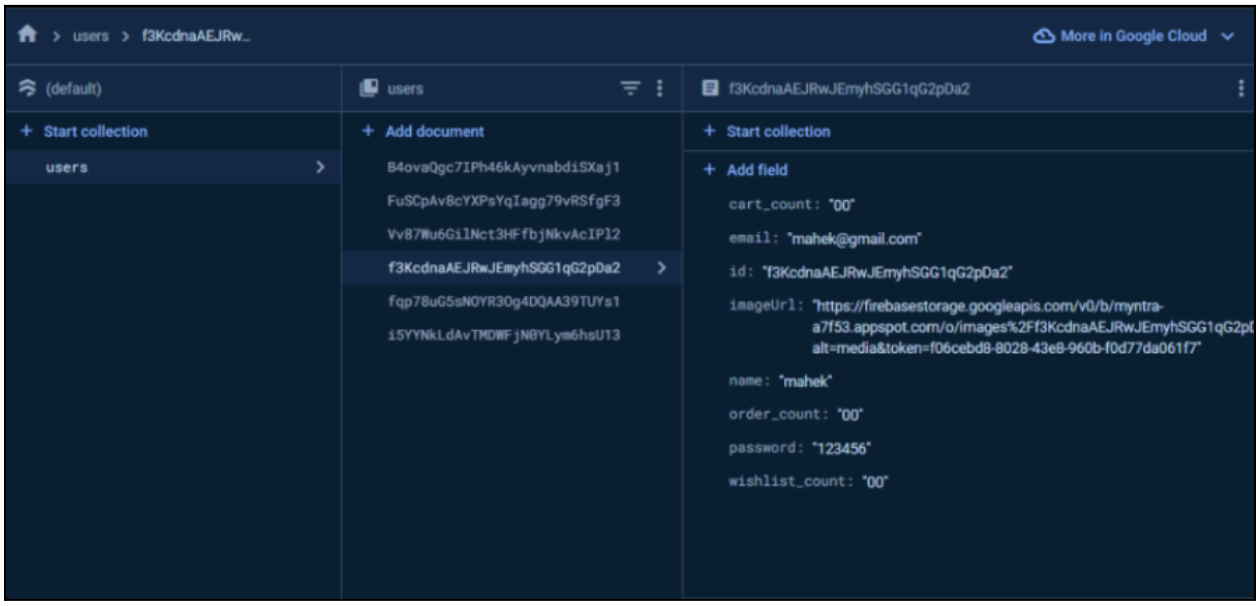


Folder Structure:



Authentication of User





CONCLUSION

I have successfully implemented my application with the firebase

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.	52
Name	Susmita Santi
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	15

Project Title: Discord Flutter Clone/ PWA : To Do List **Roll No. 52**

NAME: SUSMITA SANTI **ROLL NO.:52**
SUBJECT: MAD LAB **CLASS:D15A/BATCH C**

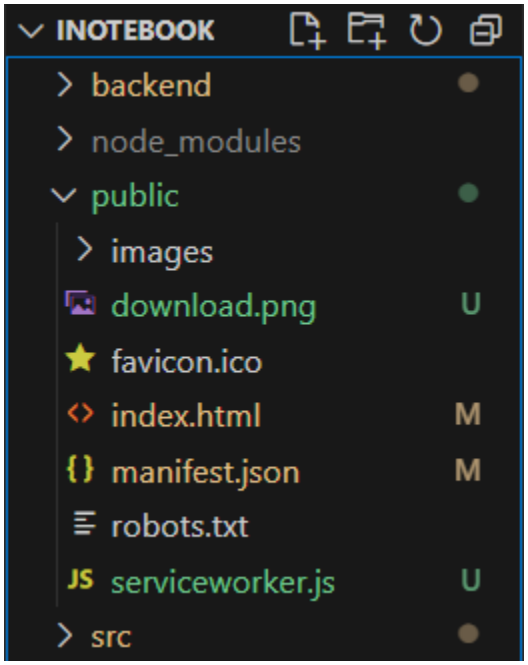
EXP 7

AIM: To write meta data of your Ecommerce PWA in a Web app manifest file to enable "add to homescreen feature"

THEORY:

Regular Web App

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



Progressive Web App

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

manifest.json

```
{
   "short_name": "React App",
   "name": "Create React App Sample",
   "icons": [
     {
       "src": "favicon.ico",
       "sizes": "64x64 32x32 24x24 16x16",
       "type": "image/x-icon"
     },
     {
       "src": "logo192.png",
       "type": "image/png",
       "sizes": "192x192"
     },
     {
       "src": "logo512.png",
       "type": "image/png",
       "sizes": "512x512"
     }
   ],
   "start_url": ".",
   "display": "standalone",
   "theme_color": "#000000",
   "background_color": "#ffffff"
 }
```

serviceworker.js

```
var staticCacheName = "pwa";

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

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

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

public/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="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
  <!--
     manifest.json provides metadata used when your web app is installed on a
     user's mobile device or desktop. See
https://developers.google.com/web/fundamentals/web-app-manifest/
   -->
  <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
  <!--
     Notice the use of %PUBLIC_URL% in the tags above.
     It will be replaced with the URL of the `public` folder during the build.
     Only files inside the `public` folder can be referenced from the HTML.

     Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
     work correctly both with client-side routing and a non-root public URL.
     Learn how to configure a non-root public URL by running `npm run build`.
   -->

  <!-- Bootstrap CSS -->
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.rtl.min.css"

integrity="sha384-PJsj/BTMqILvmcej7ulplguok8ag4xFTPryRq8xevL7eBYSmpXKcbNVuy+P0
RMgq" crossorigin="anonymous">
```

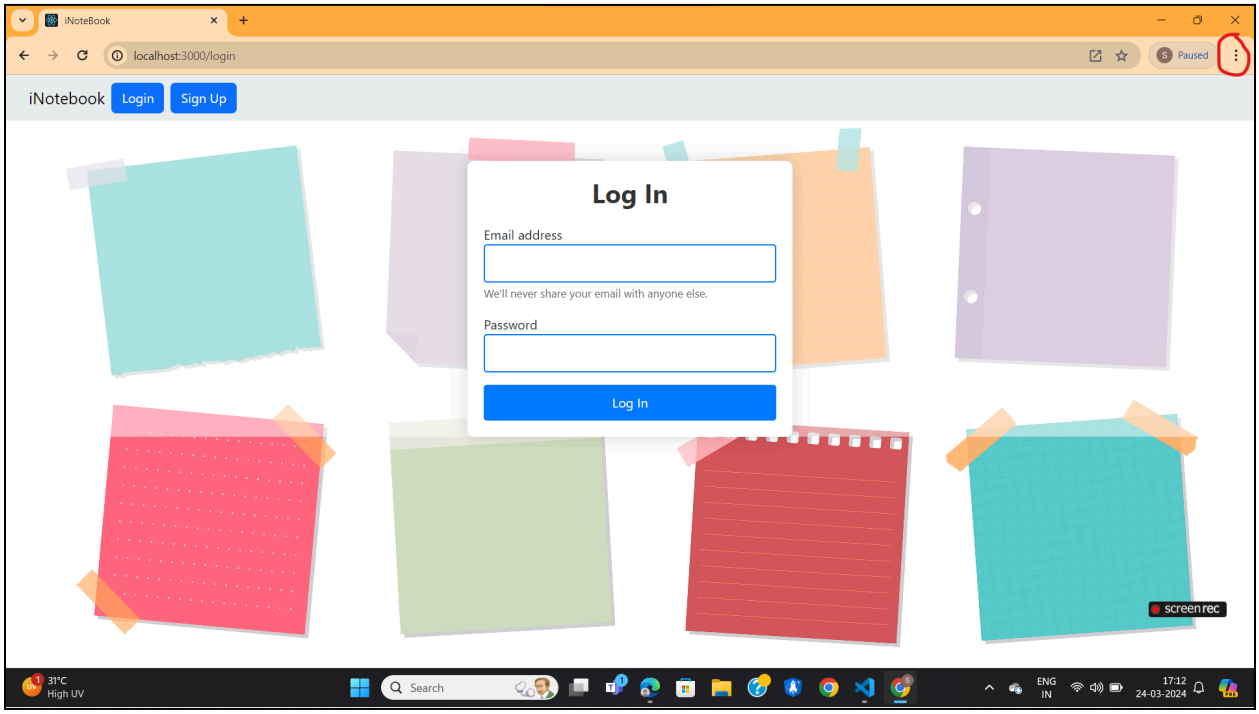
```
  <title>iNoteBook</title>
</head>

<body>
  <noscript>You need to enable JavaScript to run this app.</noscript>
  <div id="root"></div>
  <!--
     This HTML file is a template.
     If you open it directly in the browser, you will see an empty page.

     You can add webfonts, meta tags, or analytics to this file.
     The build step will place the bundled scripts into the <body> tag.

     To begin the development, run `npm start` or `yarn start`.
     To create a production bundle, use `npm run build` or `yarn build`.
    -->

  <!-- Option 1: Bootstrap Bundle with Popper -->
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"

integrity="sha384-geWF76RCwLtnZ8qwWowPQNguL3RmwHVBC9FhGdlKrxdiJJigb/j/68SIy
3Te4Bkz"
    crossorigin="anonymous"></script>

    <script src="https://kit.fontawesome.com/e7a7193d30.js" crossorigin="anonymous"></script>

    <script>
        window.addEventListener("load", () => {
          registerSW();
        });

        // Register the Service Worker
        async function registerSW() {
          if ("serviceWorker" in navigator) {
            try {
              await navigator;
              ServiceWorker;
              register("serviceworker.js");
            } catch (e) {
```

```
            console.log("SW Registration Failed.");
        }
    }
}
    </script>
</body>
</html>
```

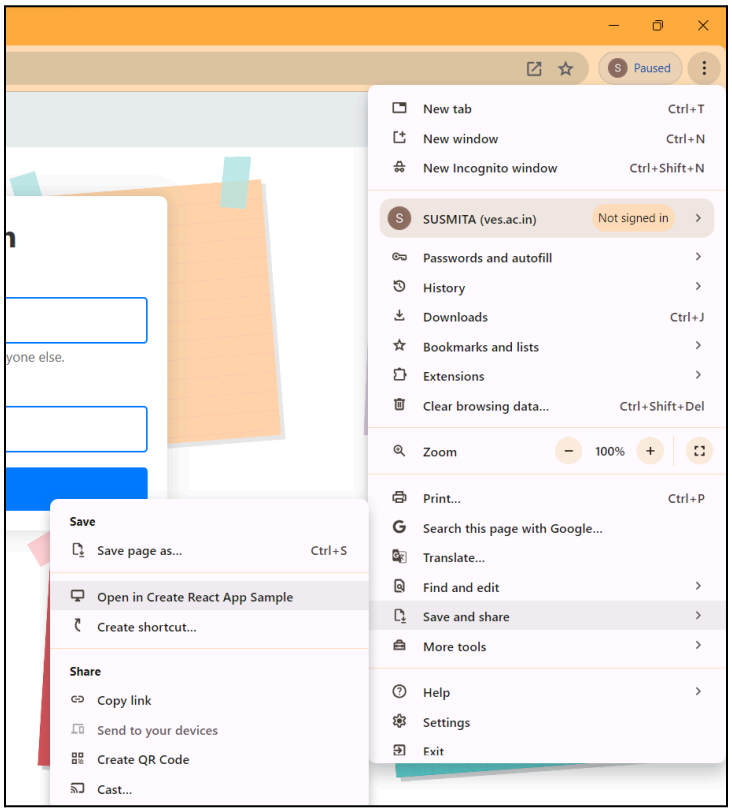
COMMAND: **npm run start**

```
You can now view inotebook in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.0.104:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

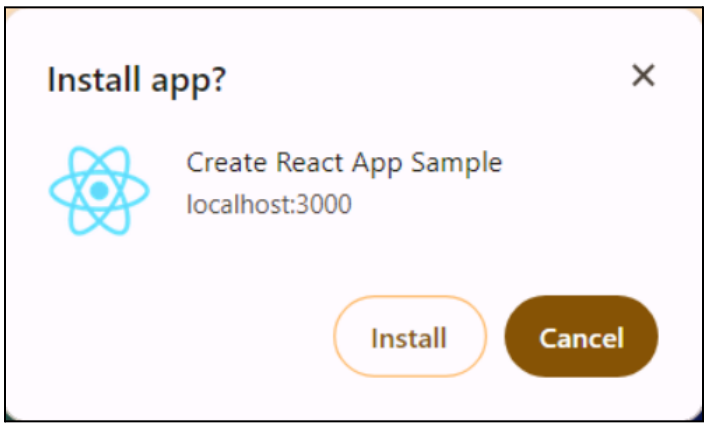
Open localhost:3000 on the browser



Click on 3 dots on the top right corner of the browser → go to **Save and Share** → Click on **Install Create React App Sample**

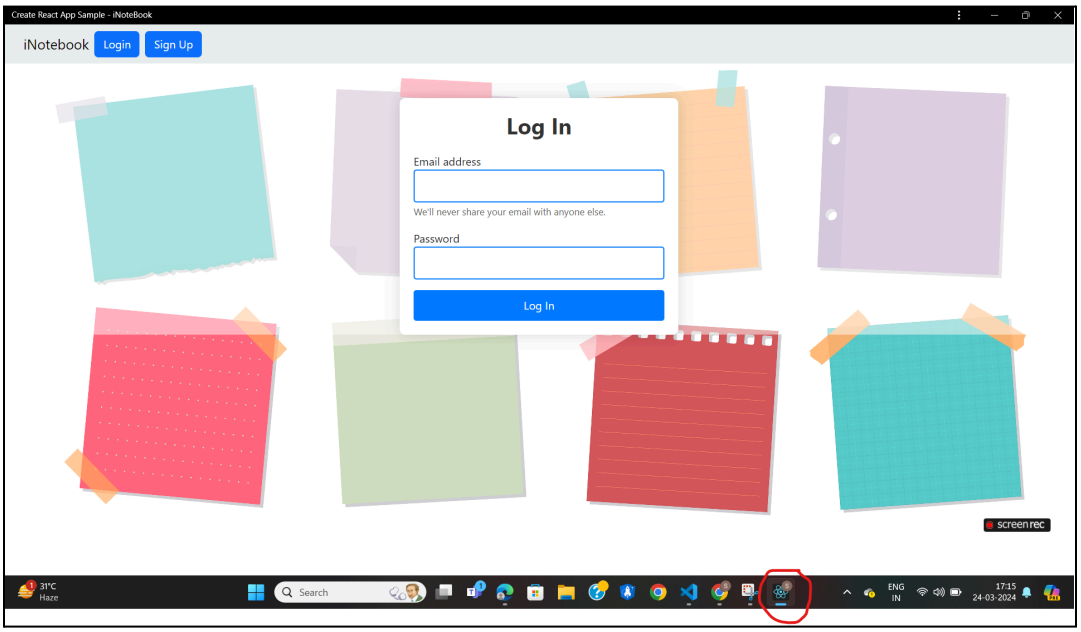


You will get the following pop-up:



Click on **Install**.

App Icon will appear at the bottom:



CONCLUSION:

Hence, we learnt how to write a metadata of our PWA in a Web App Manifest File to enable add to homescreen feature.

MAD & PWA Lab
Journal

Experiment No.	08
Experiment Title.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA
Roll No.	52
Name	Susmita Santi
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

NAME: SUSMITA SANTI ROLL NO.:52
SUBJECT: MAD LAB CLASS:D15A/BATCH C

EXP 8

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

Theory:

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

Things to note about Service Worker:

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

What can we do with Service Workers?

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

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

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

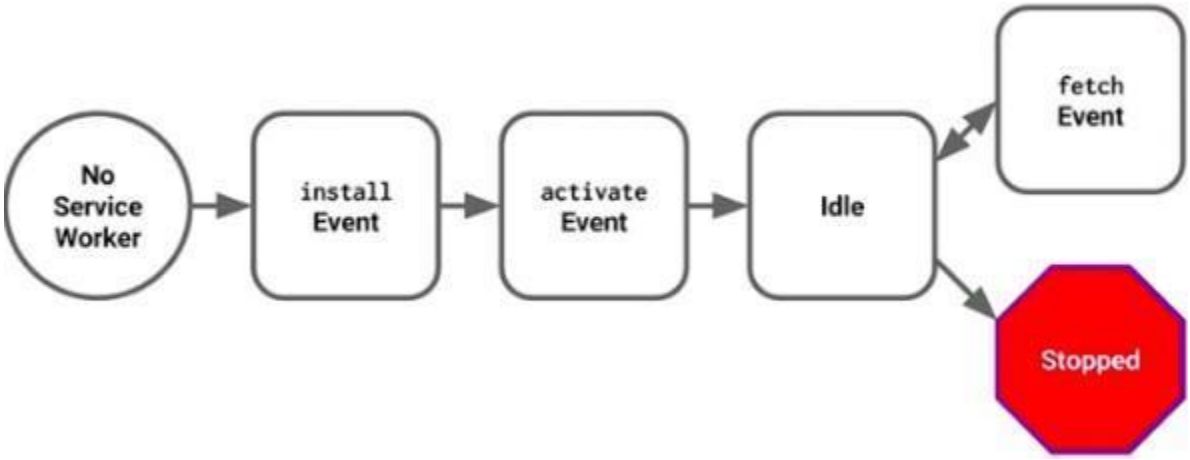
Sync of Service Worker.

What can't we do with Service Workers?

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

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

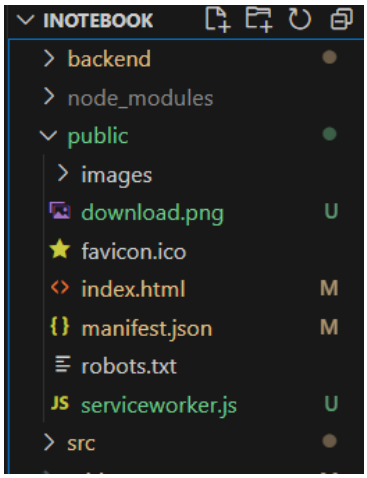
Service Worker Cycle



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

- Registration
- Installation
- Activation

My Project's folder structure:



Registration

To install a service worker, you need to register it in your main JavaScript code. Registration tells the browser where your service worker is located, and to start installing it in the background.

CODE:

public/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="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is installed on a
      user's mobile device or desktop. See
https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder during the build.
      Only files inside the `public` folder can be referenced from the HTML.
```

```
    Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
    work correctly both with client-side routing and a non-root public URL.
    Learn how to configure a non-root public URL by running `npm run build`.
  -->

  <!-- Bootstrap CSS -->
  <link
    rel="stylesheet"
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.rtl.min.css"

integrity="sha384-PJsj/BTMqILvmcej7ulplguok8ag4xFTPryRq8xevL7eBYSmpXKcbNVuy+P0
RMgq"
    crossorigin="anonymous"
  />

  <title>iNoteBook</title>
 </head>

 <body>
  <noscript>You need to enable JavaScript to run this app.</noscript>
  <div id="root"></div>
  <!--
    This HTML file is a template.
    If you open it directly in the browser, you will see an empty page.

    You can add webfonts, meta tags, or analytics to this file.
    The build step will place the bundled scripts into the <body> tag.

    To begin the development, run `npm start` or `yarn start`.
    To create a production bundle, use `npm run build` or `yarn build`.
  -->

  <!-- Option 1: Bootstrap Bundle with Popper -->
  <script
    src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"

integrity="sha384-geWF76RCwLtnZ8qwWowPQNguL3RmwHVBC9FhGdlKrxdiJJigb/j/68SIy
3Te4Bkz"
    crossorigin="anonymous"
  ></script>
```

```
    <script
      src="https://kit.fontawesome.com/e7a7193d30.js"
      crossorigin="anonymous"
    ></script>

    <script>
      window.addEventListener("load", () => {
        registerSW();
      });

      // Register the Service Worker
      async function registerSW() {
        if ("serviceWorker" in navigator) {
          try {
            const registration = await navigator.serviceWorker.register(
              "serviceworker.js"
            );
            console.log("Registered Service Worker:", registration);
          } catch (error) {
            console.log("SW Registration Failed:", error);
          }
        }
      }
    </script>
  </body>
</html>
```

Installation

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

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

interactivity is going to be even better in those cases. An example of an installation event listener looks like this:

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

~This is done in serviceworker.js

Activation

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

When the new service worker activates, an activate event is triggered in the activating service worker. This event listener is a good place to clean up outdated caches.

```
self.addEventListener('activate', event => {
  event.waitUntil(
    caches.keys().then(cacheNames => {
      return Promise.all(
        cacheNames.filter(name => {
          return name !== staticCacheName;
        }).map(name => {
          return caches.delete(name);
        })
      );
    })
  );
});
```

~This is done in serviceworker.js

Therefore, serviceworker.js finally looks like this:

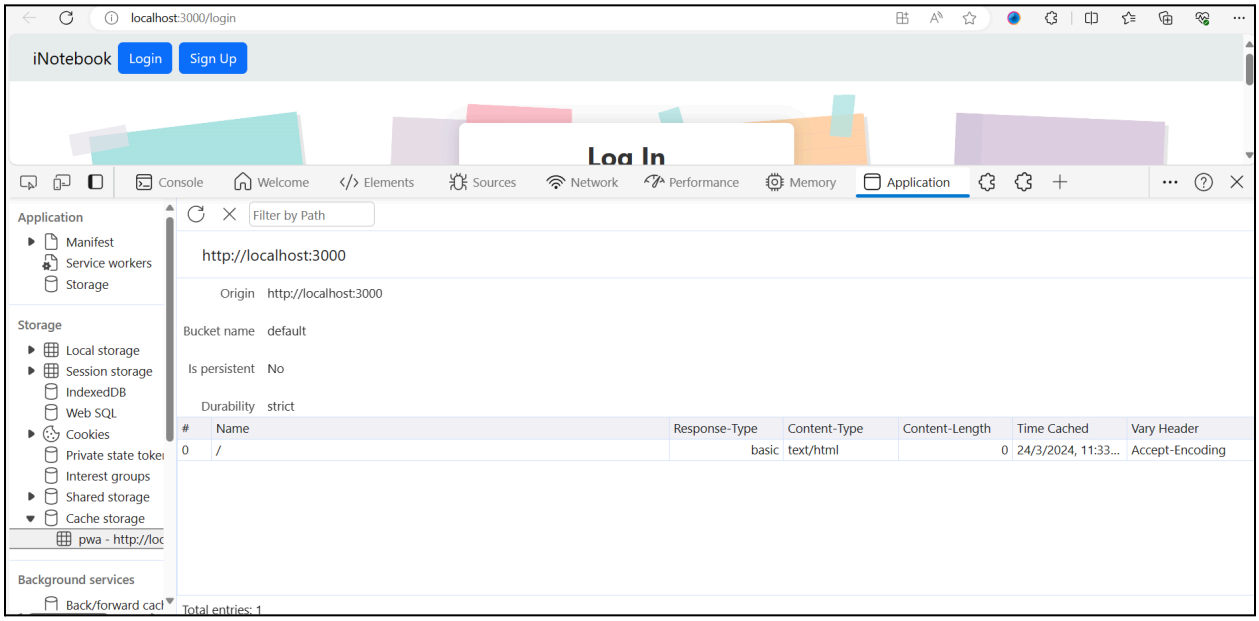
```
var staticCacheName = "pwa";
```

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

self.addEventListener('activate', event => {
    event.waitUntil(
        caches.keys().then(cacheNames => {
            return Promise.all(
                cacheNames.filter(name => {
                    return name !== staticCacheName;
                }).map(name => {
                    return caches.delete(name);
                })
            );
        })
    );
});

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

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

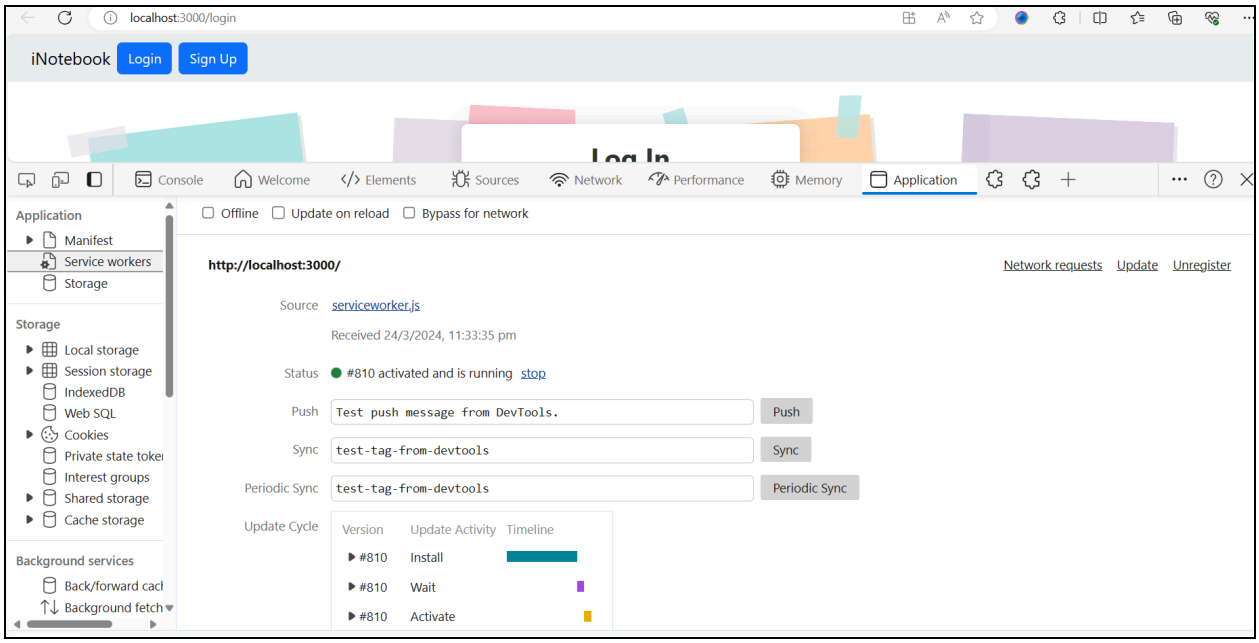
We now run the app using the following command:
> npm run start

```
You can now view inotebook in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.0.104:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

On the browser:



Project Title: Discord Flutter Clone/ PWA : To Do List **Roll No. 52**



Conclusion:

The aim was to implement a service worker for aProgressive Web Application (PWA), enabling offline functionality and improving performance. This involved coding and registering the service worker (serviceworker.js) to intercept network requests, cache essential resources, and enhance the offline experience. By completing the install and activation process, the PWA gained features like offline access and faster page loads, enhancing user experience and resilience to network issues.

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.	52
Name	Susmita Santi
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

NAME: SUSMITA SANTI ROLL NO.:52
SUBJECT: MAD LAB CLASS:D15A/BATCH C

EXP 8

Aim: To implement Service worker events like fetch, sync and push for a 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.

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

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

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

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

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.

serviceworker.js

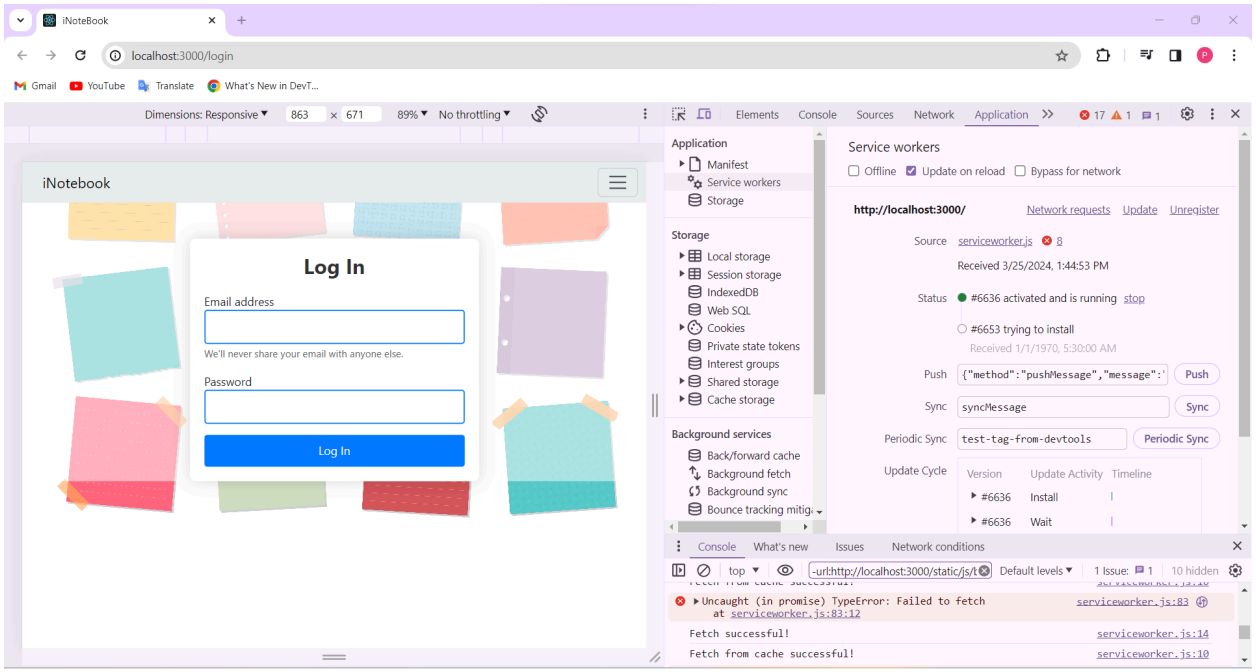
```
var staticCacheName = "pwa";

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

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

self.addEventListener("sync", (event) => {
  if (event.tag === "syncMessage") {
    console.log("Sync successful!");
  }
});

self.addEventListener("push", function (event) {
    if (event && event.data) {
      var data = event.data.json();
      if (data.method == "pushMessage") {
        console.log("Push notification requested");

        if (Notification.permission === "granted") {
          event.waitUntil(
            self.registration.showNotification("To Do List", {
              body: data.message,
            }).catch(function(error) {
              console.error("Error showing notification:", error);
            })
          );
```

```
      } else if (Notification.permission === "denied") {
        console.error("Notification permission denied.");
        // Handle the case where notification permission is denied, such as showing a message to
the user.
      } else {
        Notification.requestPermission().then(function(permission) {
          if (permission === "granted") {
            console.log("Notification permission granted, showing notification");
            self.registration.showNotification("To Do List", {
              body: data.message,
            });
          } else {
            console.error("Notification permission denied.");
            // Handle the case where notification permission is denied after requesting.
          }
        });
      }
    }
  }
});


var filesToCache = ["/login", "/Home", "/About", "/index.html"];

var preLoad = function () {
  return caches.open("index").then(function (cache) {
    return cache.addAll(filesToCache);
  });
};

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

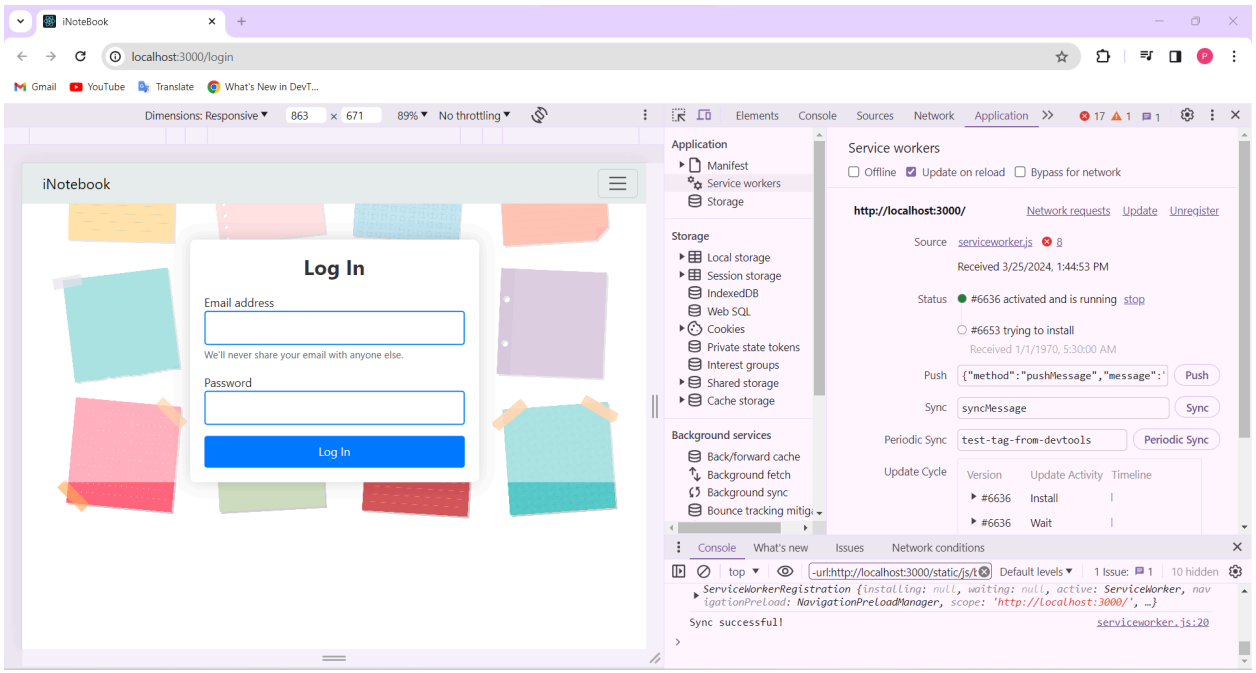
```
      .catch(reject);
  });
};

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

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

self.addEventListener("activate", (event) => {
  event.waitUntil(
    caches.keys().then((cacheNames) => {
      return Promise.all(
        cacheNames
          .filter((name) => {
            return name !== staticCacheName;
          })
          .map((name) => {
            return caches.delete(name);
          })
      );
    })
  );
});
```

Project Title: Discord Flutter Clone/ PWA : To Do List **Roll No. 52**

Fetch Event :



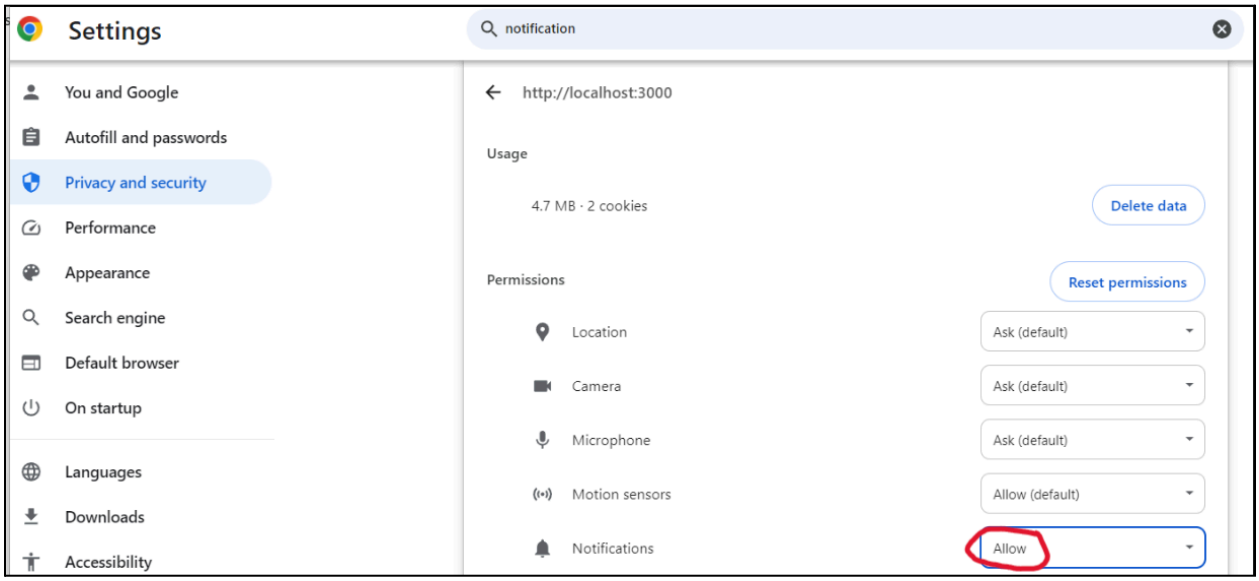
Sync Event :



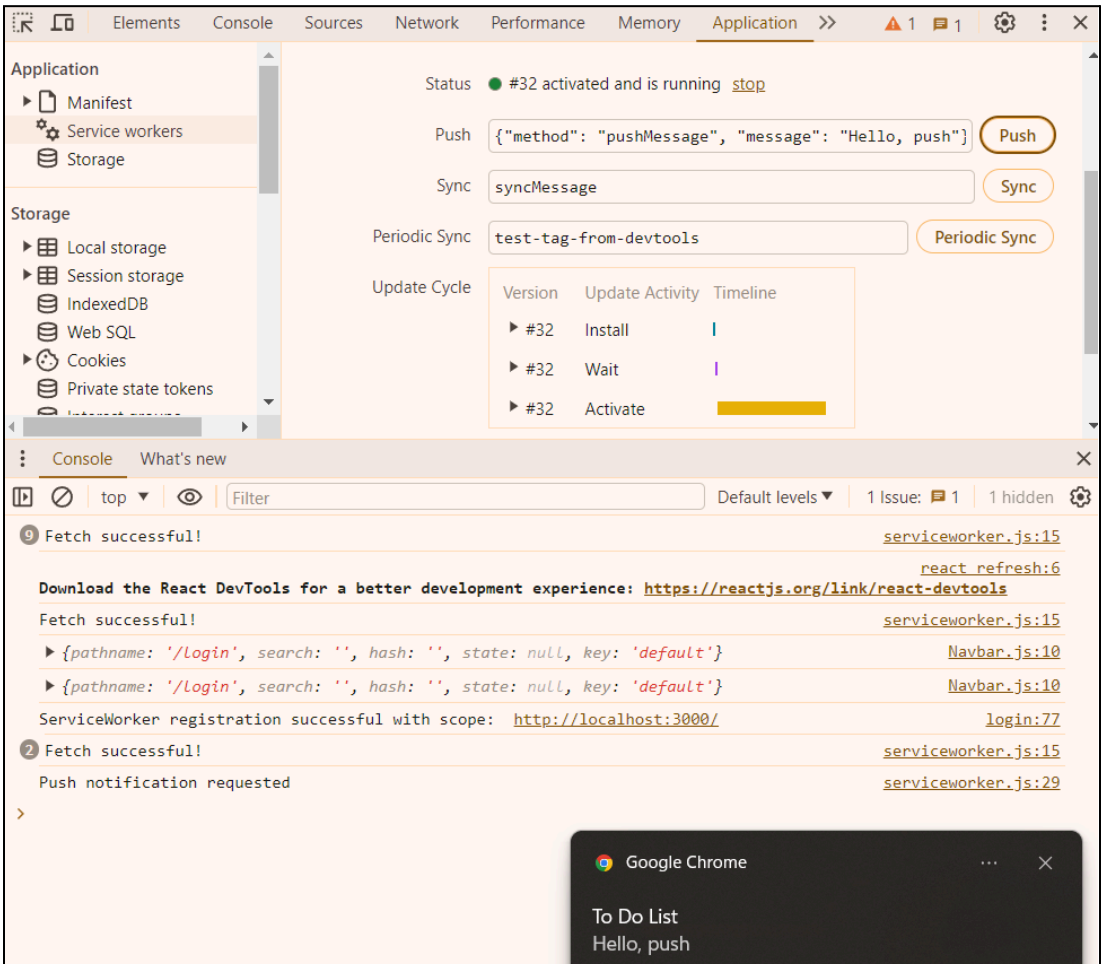
Push Event:

```
Push notification requested                                    serviceworker.js:29
▼Notification permission denied.                               serviceworker.js:42
(anonymous) @ serviceworker.js:42
```

Go to Site Settings and allow Notifications:



Now, we can see the desired result:



CONCLUSION:

- The aim is to enhance the functionality and user experience of a Progressive Web App (PWA) by implementing service worker events such as fetch, sync, and push.
- These events enable the app to efficiently cache resources for offline access, synchronize data in the background, and deliver timely notifications to users, thereby improving performance, reliability, and engagement.

MAD & PWA Lab
Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	52
Name	Susmita Santi
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	15

NAME: SUSMITA SANTI ROLL NO.:52
SUBJECT: MAD LAB CLASS:D15A/BATCH C

EXP 10

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

THEORY:

GitHub Pages

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

GitHub Pages provides the following key features:

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

Reasons for favoring this over Firebase:

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

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

Pros

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

Cons

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

Firebase

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

Some of the features offered by Firebase are:

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

Reasons for favoring over GitHub Pages:

1. Realtime backend made easy
2. Fast and responsive

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

Pros

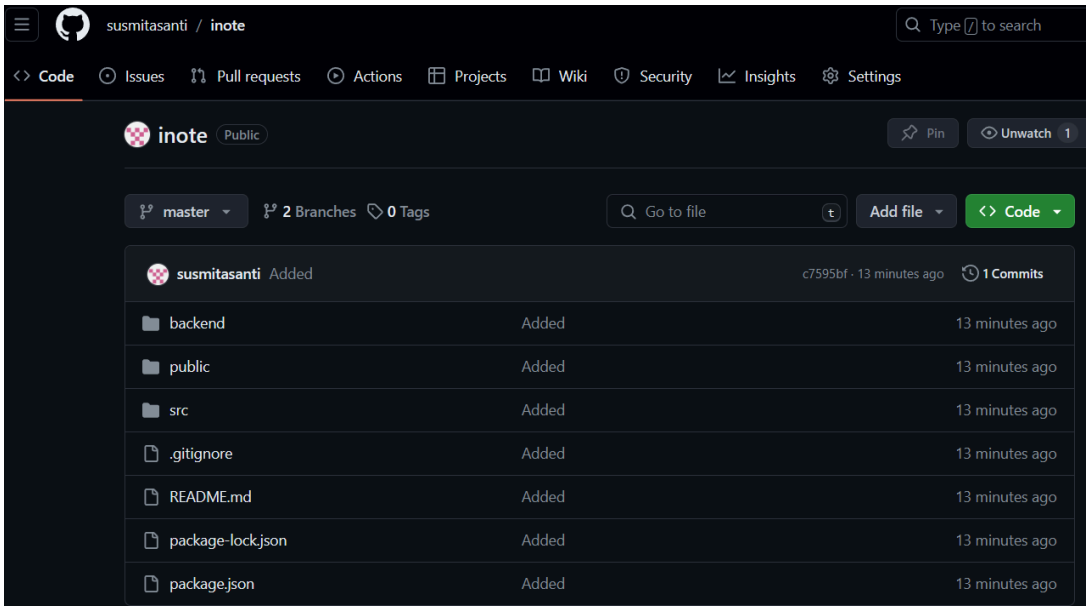
1. Hosted by Google. Enough said.
2. Authentication, Cloud Messaging, and a whole lot of other handy services will be available to you.
3. A real-time database will be available to you, which can store 1 GB of data.
4. You'll also have access to a blob store, which can store another 1 GB of data.
5.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

Cons

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

Project Title: Discord Flutter Clone/ PWA : To Do List **Roll No. 52**

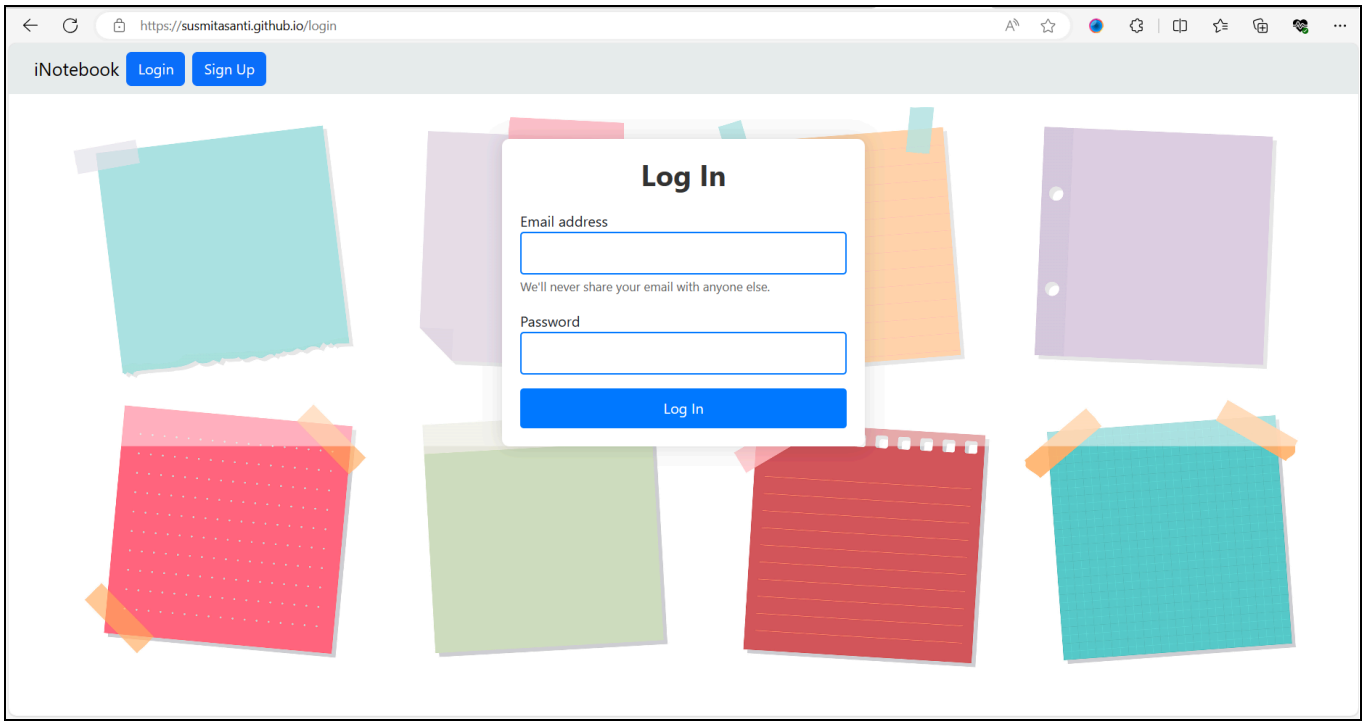
Link to our GitHub repository: [susmitasanti/inote (github.com)]



HOSTED LINK:
[iNoteBook (susmitasanti.github.io)]

HOSTED IMAGE:

Project Title: Discord Flutter Clone/ PWA : To Do List **Roll No. 52**



MAD & PWA Lab
Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	52
Name	Susmita Santi
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	15

NAME: SUSMITA SANTI ROLL NO.:52
SUBJECT: MAD LAB CLASS:D15A/BATCH C

EXP 11

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

THEORY:

Google Lighthouse :

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

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

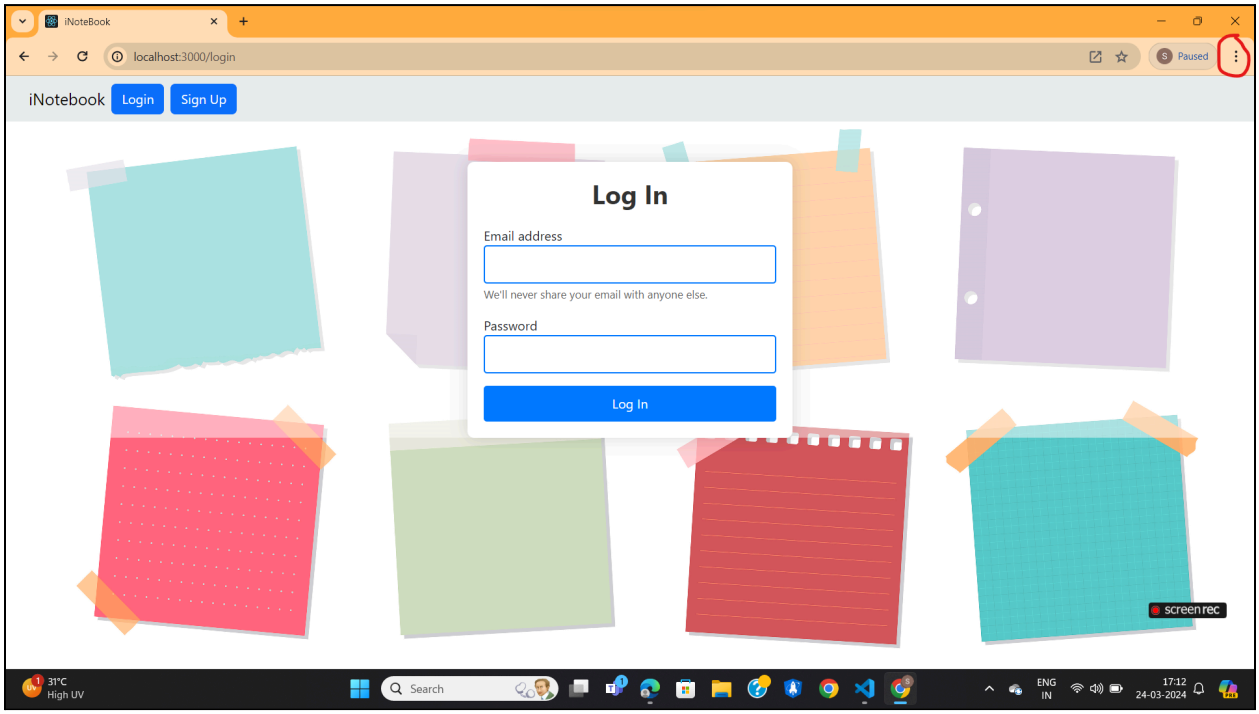
Key Features and Audit Metrics

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

1. **Performance:** This score is an aggregation of how the page fared in aspects such as (but not limited to) loading speed, time taken for loading for basic frame(s), displaying meaningful content to the user, etc. To a layman, this score is indicative of how decently the site performs, with a score of 100 meaning that you figure in the 98th percentile, 50 meaning that you figure in the 75th percentile and so on.
2. **PWA Score (Mobile):** Thanks to the rise of Service Workers, app manifests, etc., a lot of modern web applications are moving towards the PWA paradigm, where the objective is to make the application behave as close as possible to native mobile applications. Scoring points are based on the Baseline PWA checklist laid down by Google which includes Service Worker implementation(s), viewport handling, offline functionality, performance in script-disabled environments, etc.
3. **Accessibility:** As you might have guessed, this metric is a measure of how accessible your website is, across a plethora of accessibility features that can be implemented in your page (such as the 'aria-' attributes like aria-required, audio captions, button names, etc.). Unlike the other metrics though, Accessibility metrics score on a

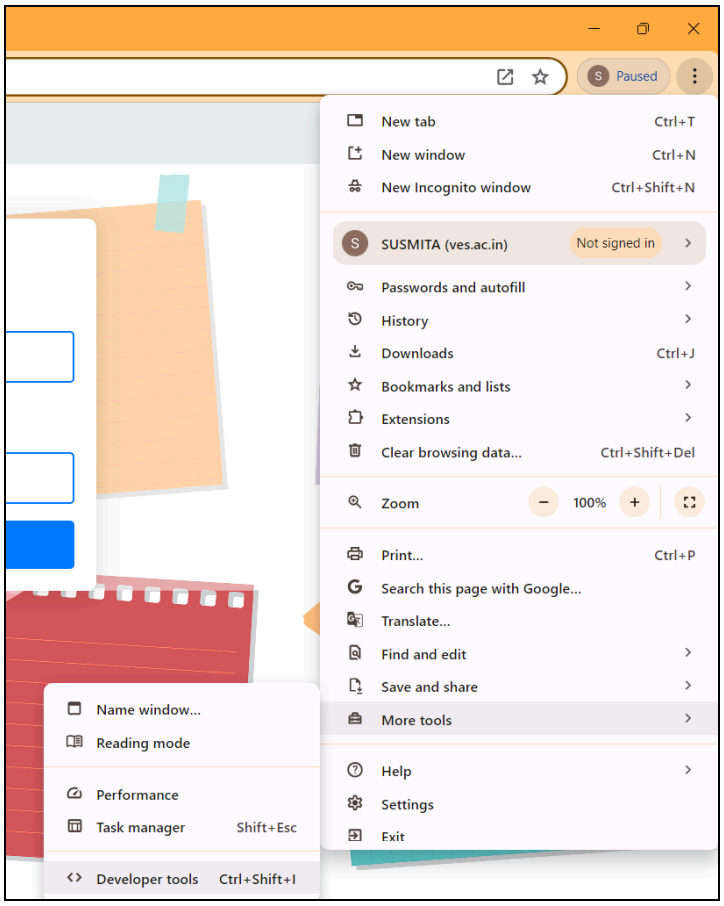
pass/fail basis i.e. if all possible elements of the page are not screen-reader friendly (HTML5 introduced features that would make pages easy to interpret for screen readers used by visually challenged people like tag names, tags such as <section>, <article>, etc.), you get a 0 on that score. The aggregate of these scores is your Accessibility metric score.

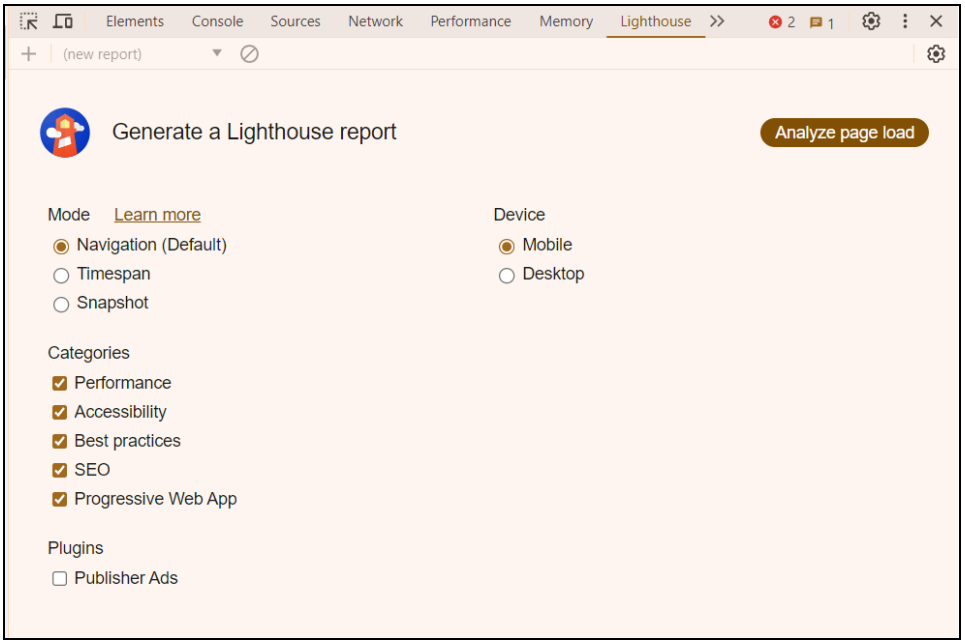
4. **Best Practices:** As any developer would know, there are a number of practices that have been deemed 'best' based on empirical data. This metric is an aggregation of many such points, including but not limited to:Use of HTTPS
   Avoiding the use of deprecated code elements like tags, directives, libraries, etc. Password input with paste-into disabled
   Geo-Location and cookie usage alerts on load, etc.



Click on the 3 dots on the top-right corner of the page → Click on **More Tools** → Click on **Developer Tools**

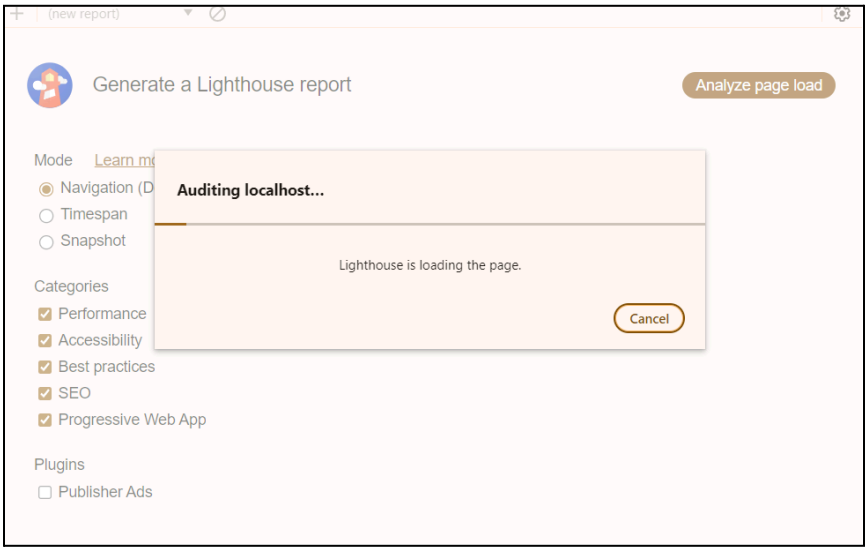
Project Title: Discord Flutter Clone/ PWA : To Do List **Roll No. 52**

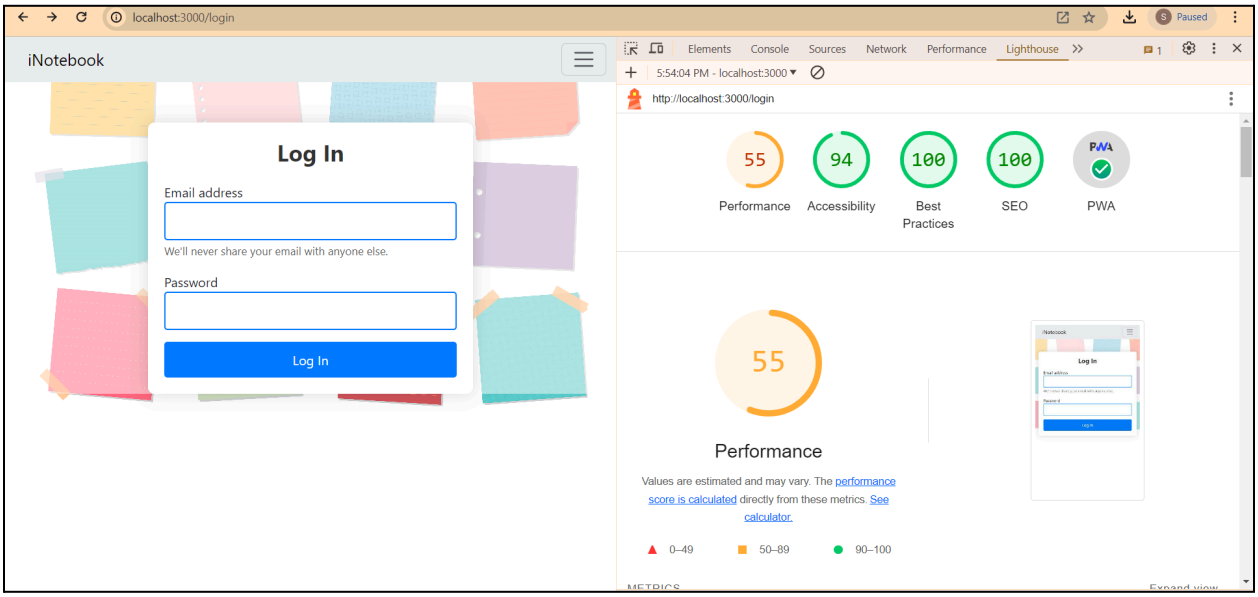




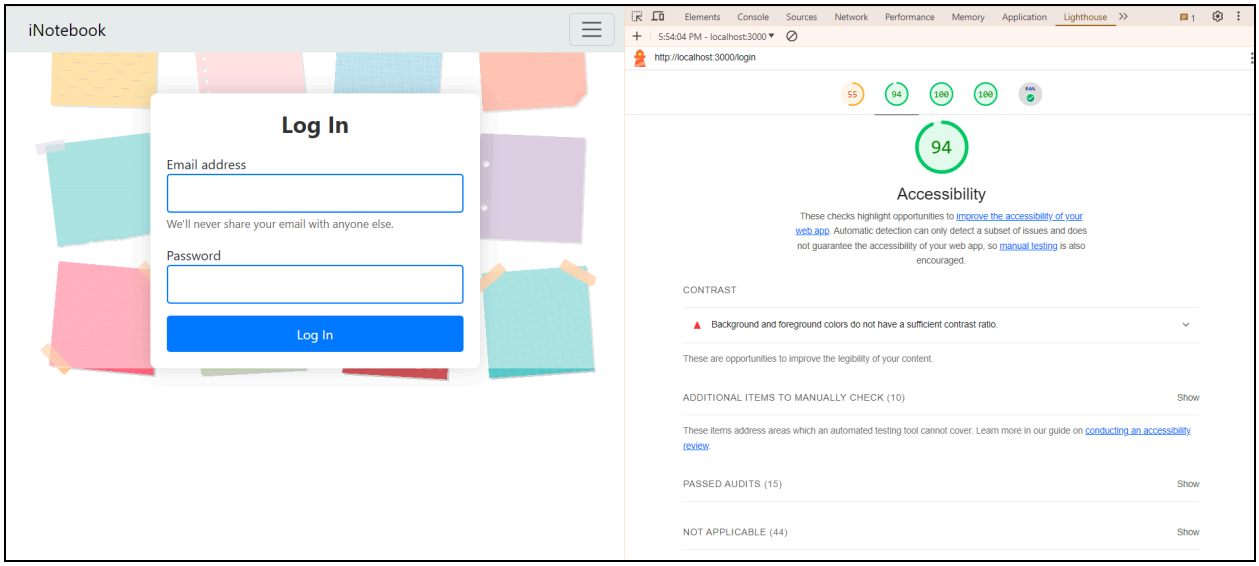
Click on **Analyze page load**

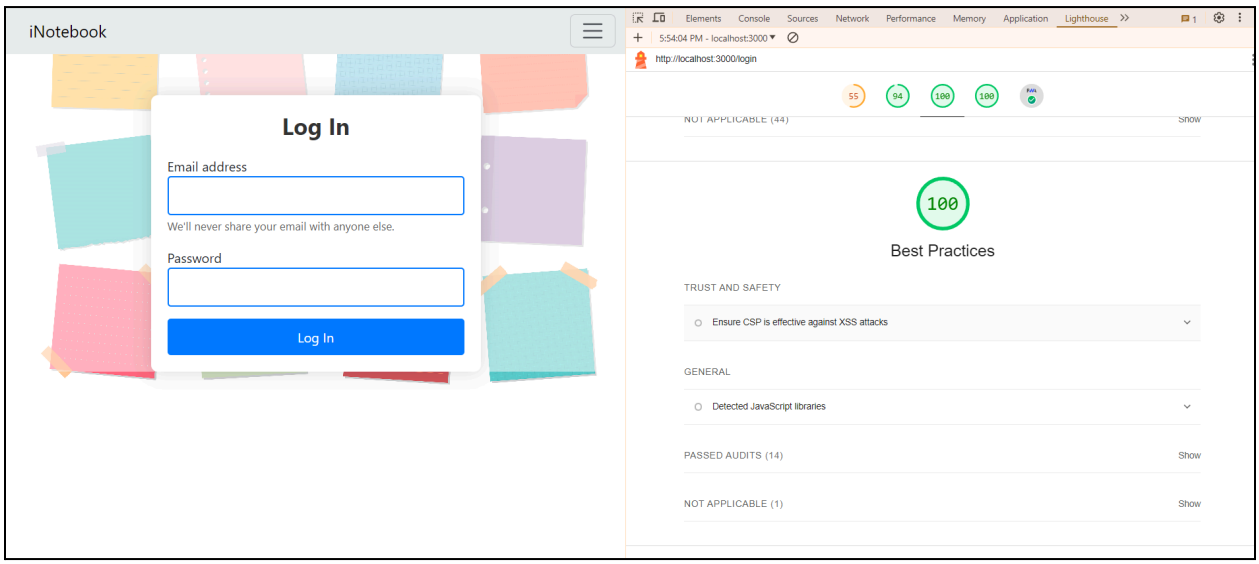
Project Title: Discord Flutter Clone/ PWA : To Do List **Roll No. 52**

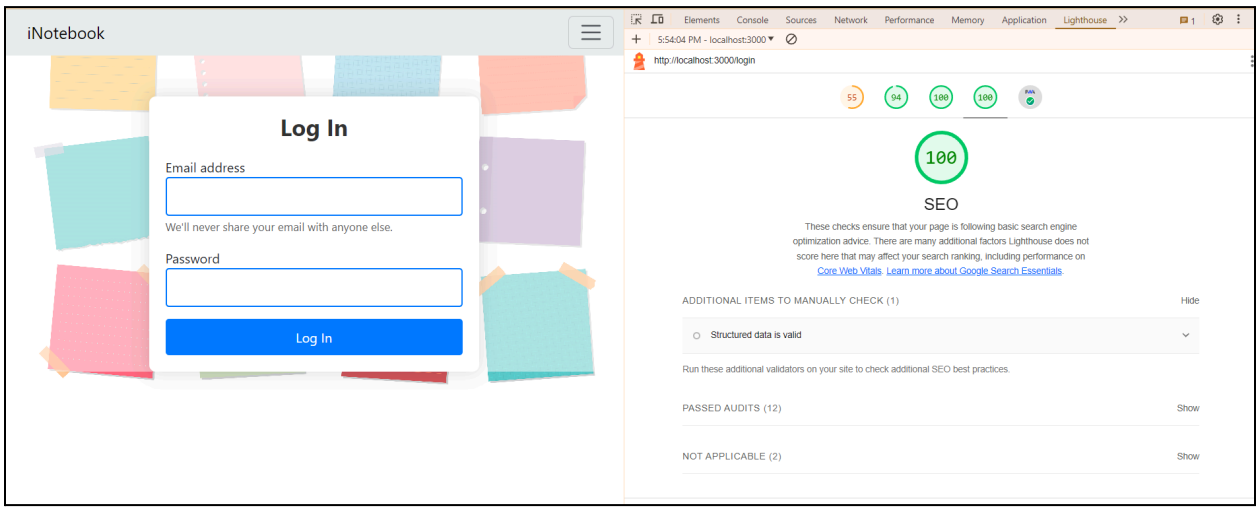


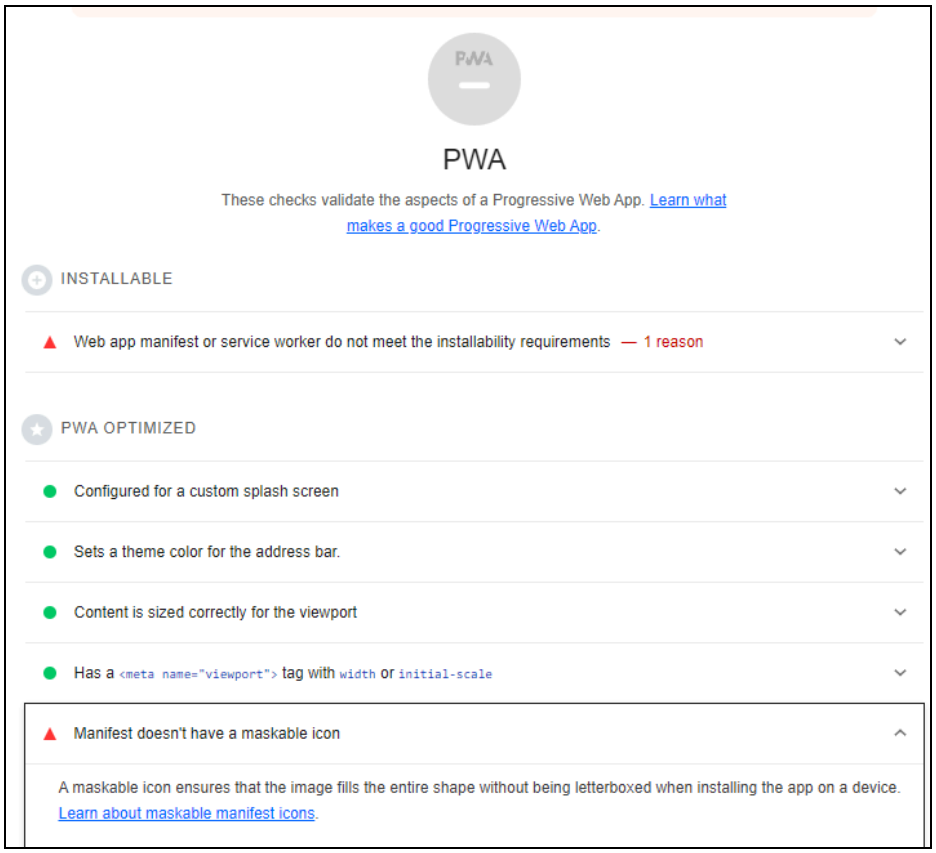


Project Title: Discord Flutter Clone/ PWA : To Do List **Roll No. 52**





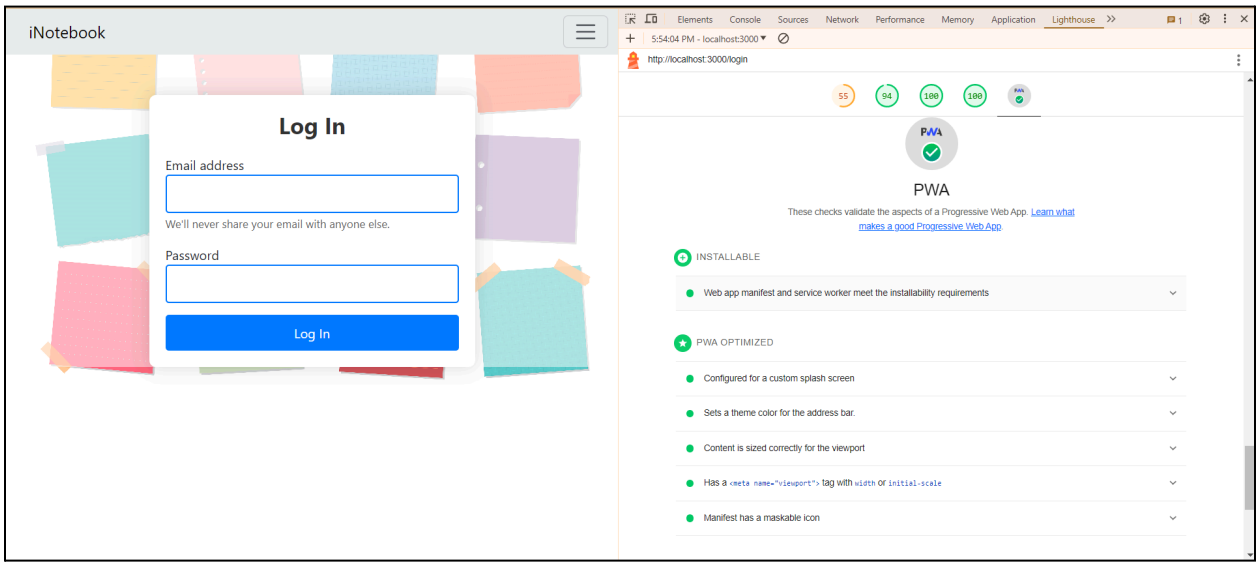




Made the following changes in **manifest.json**

```
{
  "short_name": "React App",
  "name": "Create React App Sample",
  "icons": [
    {
      "src": "favicon.ico",
      "sizes": "64x64 32x32 24x24 16x16",
      "type": "image/x-icon"
    },
    {
      "src": "logo192.png",
      "type": "image/png",
      "sizes": "192x192"
    },
    {
      "src": "logo512.png",
      "type": "image/png",
      "sizes": "512x512",
      "purpose":"maskable"
    },
```

```
    {
      "src": "download.png",
      "type": "image/png",
      "sizes": "144x144",
      "purpose": "any"
    }
  ],
  "start_url": ".",
  "display": "standalone",
  "theme_color": "#000000",
  "background_color": "#ffffff"
}
```



Conclusion: Thus we successfully used google Lighthouse PWA Analysis Tool for testing the PWA functioning.

MAD & PWA Lab
Journal

Experiment No.	Assignment-1
Assignment 1 Questions	1. Flutter Overview:Explain the key features and advantages of using Flutter for mobile app development. Discuss how the Flutter framework differs from traditional approaches and why it has gained popularity in the developer community. 2. Widget Tree and Composition: Describe the concept of the widget tree in Flutter. Explain how widget composition is used to build complex user interfaces. Provide examples of commonly used widgets and their roles in creating a widget tree. 3. State Management in Flutter: Discuss the importance of state management in Flutter applications. Compare and contrast the different state management approaches available in Flutter, such as setState, Provider, and Riverpod. Provide scenarios where each approach is suitable. 4. Firebase Integration in Flutter: Explain the process of integrating Firebase with a Flutter application. Discuss the benefits of using Firebase as a backend solution. Highlight the Firebase services commonly used in Flutter development and provide a brief overview of how data synchronization is achieved.
Roll No.	52
Name	Susmita Santi
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	05

MAD- PWA LAB

SUSMITA SANTI DIGA 52

ASSIGNMENT - I :

Q.1] Flutter Overview - Explain the key features and advantages of using flutter for mobile app development. Discuss how the flutter framework differs from traditional approaches and why it has gained popularity in the developer community.

→

- Key features of Flutter:
  - a] Single codebase for multiple platforms
    - Flutter allows developers to write code and deploy it on both iOS and Android Platforms.
  - b] Hot Reload
    - This enables developers to instantly see the results of the code changes they make.
  - c] Expressive UI
    - Developers have the flexibility to create expressive and flexible UIs.
  - d] Integration with other tools
    - Flutter can easily integrate with other popular development tools and frameworks.

- Advantages of Flutter:
  - a] Faster Development
    - Uses single ~~code~~ codebase for multiple platform.
  - b] Consistent UI across platforms
    - Widgets provide a consistent look and feel across different platforms.
  - c] Cost efficiency
    - Developing and maintaining single codebase for both iOS and Android reduces development

P.T.O

cost and resources.

- Differ from Traditional Approach
  a] Traditional approach uses a hierarchical structure for UI components whereas flutter uses a widget-based approach.
  b] Flutter compiles to native ARM code, providing performance comparable to native applications.
  c] Hot Reload allow to see changes made instantly.

- Flutter's popularity is driven by increased productivity, a growing community, flexibility in UI design, cross-platform development capabilities & adoption by major companies.

Q.2) Widget Tree and composition:
Describe the concept of widget tree in flutter.
Explain how widget composition is used to build complex user interfaces.
Provide examples of commonly used widgets and their roles in creating a widget tree.

→ Widget Tree
- The widget tree is a hierarchical structure of widgets that defines the user interface of an application.
- Every visual element, from simple components to complex layouts, is represented by a widget.

- Widget can be categorized in 2 types
  a] Stateless widget
  - It is immutable and cannot change over time
    Eg.: Images, text.
  b] Stateful widget
  - A widget that can change its state over time.
    Eg.: Buttons, forms.

Widget composition

- Widget composition in flutter involves combining multiple simple widgets to create more complex and compound widgets.
- This composability is a powerful concept that allows developers to build sophisticated user interfaces by nesting widgets within each other.

- Commonly used widgets:
  a] Container
  - A box model for padding, margin & decoration.
  b] Column and Row
  - Layout widgets for arranging children vertically or horizontally.
  c] Stack
  - Overlapping widgets, allowing them to be layered on top of each other.
  d] List View
  - A scrollable list of widgets
  e] Grid View
  - A scrollable ~~list~~ grid of widgets.

f] AppBar
- A material design app bar typically at top of screen.

g] TextField.
- An input field for users to enter text.

h] Button widgets.
- Interactive buttons for user actions.

Q.3] State Management in Flutter: Discuss the importance of state management in Flutter applications. Compare and contrast the different state management approaches available in Flutter, such as setState, Provider, and Riverpod. Provide scenario where each approach is suitable.

→
- State management is crucible in Flutter because it determines how your application responds to user input, network requests, and changes in data.
- Efficient state management ensures your application stays responsive and maintains a smooth user experience.
- Comparison of the 3 popular ~~state~~ management approaches in Flutter:

Aspect	setState	Provider	Riverpod
Complexity	Simple	Moderate	Advanced
Scope	Localized to the widget	Can be global or localized	Global or localized, supports both.

Aspect	setState	Provider	Riverpod
Boilerplate	Minimal	Moderate	Moderate, less than Provider
Dependencies	No external dependencies required	Requires the Provider Package	Requires both Provider & Riverpod.
Flexibility	Limited	Good	Excellent
Performance	Suitable for small-scale apps	Efficient for many use cases.	Efficient and optimized for large-scale apps.
Dependency injection	Not applicable	Not built-in, but can be added.	Built-in support for dependency injection.
Community Support	Widely used, well-documented.	Widely adopted.	Growing community, gaining popularity.

- Scenarios:

i] Use 'setState' when:
- Dealing with small, localized state changes.
- Working on simple apps or prototypes.

ii] Use 'Provider' when:
- Managing states across multiple widgets.
- Needing a straightforward & efficient solution for state management.
- Avoiding excessive boilerplate code.

iii] Use 'Riverpod' when:
- Implementing a more advanced state management system.
- Incorporating dependency injection.
- Preferring a flexible solution that can ~~be~~ handle both global and local states.

- The best choice depends on your specific needs & the complexity of your flutter application.
- Each approach has ~~its~~ its strength, & it's crucial ~~to~~ to choose the one that aligns with your project requirements.

Q.4] Firebase Integration in Flutter: Explain the process of integrating Firebase with a Flutter application. Discuss the benefits of using Firebase as a backend solution. Highlight the Firebase services commonly used in Flutter development & provide a brief overview of how data synchronization is achieved.

→ • Set up Firebase Project
- Create a new Project on Firebase Console.
- Follow the setup wizard to add your Flutter app to the project.

• Configure Flutter Project
- Add the necessary Firebase dependencies in your 'pubspec.yaml' file.
- Download the 'google-services.json' file for Android and 'GoogleService-Info.plist' for iOS from the ~~file~~ firebase console. Place them in the respective project folders.

• Initial Firebase
- In your main.dart file, initialize firebase using 'Firebase.initilizeApp()'.

• Benefits of using Firebase
i] Real-time Database.
- It provides a NoSQL Cloud database that syncs data in real-time across clients.
ii] Authentication.
- ~~Easy~~ Easily implement authentication with Firebase Authentication, ~~providing various pr~~ supporting various providers like email/password, Google, Facebook, etc.
iii] Cloud Firestore.
- Firebase's Firestore is a scalable NoSQL database, offering powerful querying & data modeling capabilities.
iv] Cloud Functions.
- Extend your backend with serverless functions that automatically respond to events triggered by Firebase features & HTTPS requests.
v] Cloud Storage
- Store & ~~retrieve~~ retrieve user-generated content like images, videos & audio-files.

vi] Hosting.
- Deploy your Flutter web app effortlessly with Firebase hosting.

vii] Cloud Messaging
- Implement push notifications to engage users ~~not~~ even when app is not in use.

• Data Synchronization in Firebase.
- Firebase achieves real-time data synchronization through its Websocket-based protocol.
- When data changes on the server, it is intentionally pushed to all the connected clients, triggering UI updates.
- This is crucial for Flutter, as it ensures a seamless & reactive user experience.
- In Firestore, you can listen to changes in your data using stream controllers.
- The 'StreamBuilder' widget in Flutter is a handy tool to automatically rebuild UI components whenever the underlying data changes.

MAD & PWA Lab
Journal

Experiment No.	Assignment-2
Assignment 2 Questions	1. Define Progressive Web App (PWA) and explain its significance in modern web development.Discuss the key characteristics that differentiate PWAs from traditional mobile apps 2. Define responsive web design and explain its importance in the context of Progressive Web Apps.Compare and contrast responsive, fluid, and adaptive web design approaches. 3. Describe the lifecycle of Service Workers, including registration, installation, and activation phases. 4. Explain the use of IndexedDB in the Service Worker for data storage.
Roll No.	52
Name	Susmita Santi
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO4:Understand various PWA frameworks and their requirements LO5: Design and Develop a responsive User Interface by applying PWA Design techniques LO6:Develop and Analyze PWA Features and deploy it over app hosting solutions
Grade:	04

SUSMITA SANTI
DIS A
52

ASSIGNMENT – 2

Q.1] Define Progressive Web App (PWA) & explain its significance in modern web development. Discuss the key characteristics that differentiate PWA from traditional web apps.

→ • A Progressive Web App (PWA) is a type of web application that uses modern web technologies to provide a native app.-like experience to users.
• They are designed to work on any platform that uses a standard compliant browser, such as desktop & mobile devices.
• Key characteristics that differentiate PWAs.

i] Cross-platform compatibility.
PWAs work across different devices & platforms.

ii] Responsive Design
It is built with responsive design principle.

iii] Offline functionality
It can work offline or with poor internet connection.

iv] App-like experience
It offers features such as push notification, homescreen installation, full-screen mode making them feel like native-app.

v] Fast Performance.

vi] Discoverability
It is discoverable through search engines & can be shared easily.

• Overall, PWAs combine the best feature of web & mobile apps, offering a compelling alternative to traditional native app development.

Q.2] Define responsive web design & explain its importance in the context of progressive web app. Compare & contrast responsive, fluid & adaptive web design approaches.

→ • Responsive Web Design is a web design approach that ensures a website adapt its layout & content to seamlessly function & display well across various screen sizes, from desktop to smartphones.

• Importance of PWA

i] Foundation for App-like Experience:

• RWD ensures proper layout & navigation across devices, mimicking the adaptability of native apps.

ii] Accessibility

• A RWD make PWA accessible wider audience using diverse devices.

iii] SEO Benefits.

• Responsive v/s Fluid v/s Adaptive

FEATURE	RESPONSIVE	FLUID	ADAPTIVE
• Layout	Flexible, adapt to any screen size.	Continuously adjusts based on relative units	User multiple fixed width layouts.
• Development & effort	Moderate	Less effort for basic layout	More effort to create multiple layo
• Control	Good control over layout	Less control over element appearance	High control over layout for each device category.

FEATURE	RESPONSIVE	FLUID	ADAPTIVE
• Suitability for PWAs.	Ideal foundation due to its versatality.	Can be used but may require adjustment for optimal experience	Less common for PWAs due to development overhead.

Q.3] Describe the lifecycle of service workers, including registration, installation & activation phases.

→ • The lifecycle of service workers involves three main phases :

i] Registration

- Service workers are first registered by the web application through Javascript code.
- This registration typically occurs in the main Javascript file of the application & initiated using the 'navigator.serviceWorker.register()' method.

ii] Installation

- After registration, the browser begins the installation phase. During installation, the service worker script is downloaded & cached by the browser.
- Once script is downloaded, the 'install' event is triggered in the service worker script.
- This event allows the service worker to perform tasks such as caching static assets or setting up other resources needed for offline functionality.

iii] Activation

- After the installation phase, the service worker enters the activation phase. During activation, the new version of the service worker takes control of the web page & any open instances of previous versions are terminated.
- The 'activate' event is triggered during this phase, allowing it to clean up old caches.

- Overall, the lifecycle of service workers involves registration, installation & activation phases, allowing them to control network requests, cache resources & provide offline functionality for web applications.

Q.4] Explain the use of Indexed DB in the service worker for data storage.

→ • Indexed DB is a low-level API for client-side storage of significant amounts of structured data, including files/blobs.

- It is particularly useful in Service workers for several reasons:

i] Asynchronous & non-blocking

- Service Workers execute in a separate thread from the main browser thread, & Indexed DB's asynchronous nature allows Service workers to perform data storage operations without blocking the main thread, ensuring smooth user experience.

ii] Persistant storage
- It provides persistant storage, meaning data stored in indexedDB persists even when the browser is closed or device is restarted.
- This is crucial for applications that require offline functionality.

iii] Large storage capacity
- IndexedDB supports storing large amounts of data locally, making it suitable for applications that need to store substantial datasets.

iv] Structured data storage
- IndexedDB stores data in a structured manner, allowing developers to organize & query data using indexes.

- In the context of service workers, indexedDB can be used to cache resources, such as, HTML, CSS, Javascript files, etc.
- It uses indexedDB to cache resources, enhancing performance by serving cached content & reducing reliance on the network.