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 **Siddhant Sanjay Sathe** 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 **2024-2025.**

Lab Assistant

Subject Teacher

Mrs. Kajal Joseph

Principal

Head of Department

Dr. Mrs. Shalu Chopra

Project Title: Stack Overflow Clone **Roll No. 50**

Name of the Course : MAD & PWA Lab **Course Code :** ITL604

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

Faculty Incharge : Mrs. Kajal Joseph.

Lab Teachers : Mrs. Kajal Joseph.

Email : kajal.jewani@ves.ac.in

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

Program specific Outcomes

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

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

Project Title: Stack Overflow Clone **Roll No. 50**

Lab Objectives:

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

Lab Outcomes:

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

Index

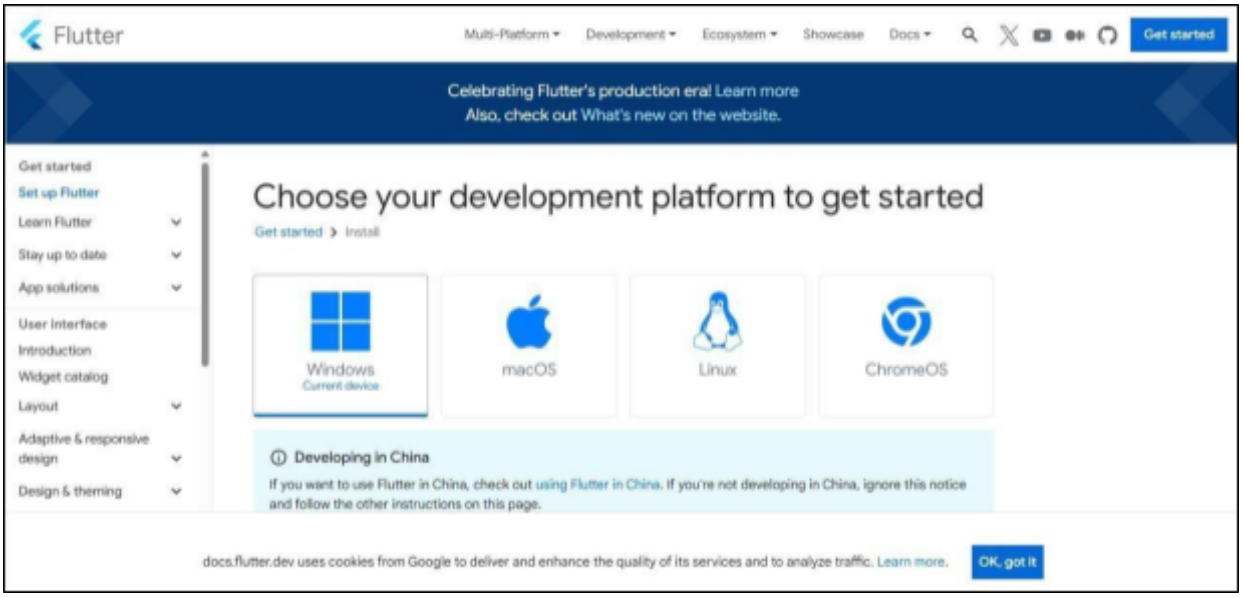
Sr. No	Experiment Title	LO	Grade
1.	To install and configure the Flutter Environment	LO1	
2.	To design Flutter UI by including common widgets.	LO2	
3.	To include icons, images, fonts in Flutter app	LO2	
4.	To create an interactive Form using form widget	LO2	
5.	To apply navigation, routing and gestures 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

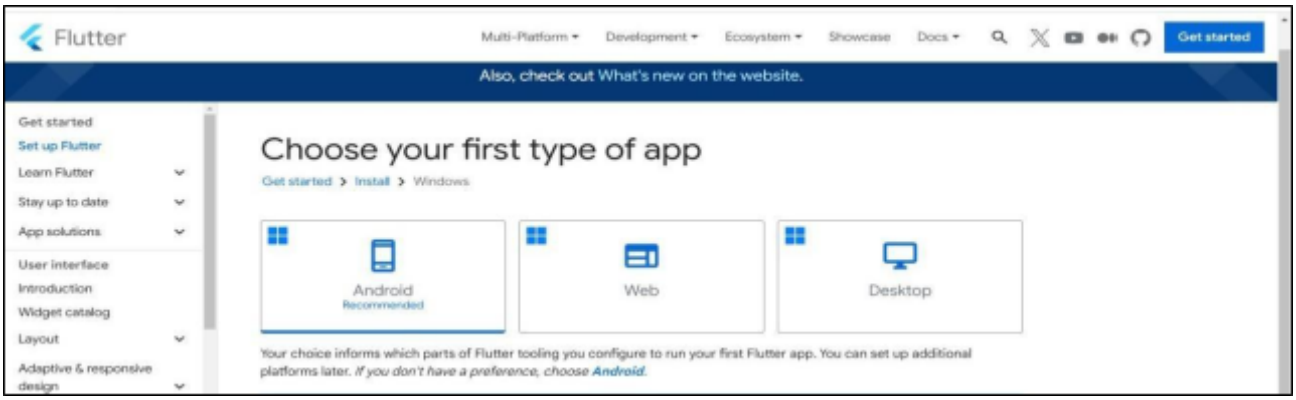
Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	50
Name	Siddhant Sathe
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	

AIM: - Installation and Configuration of Flutter Environment.

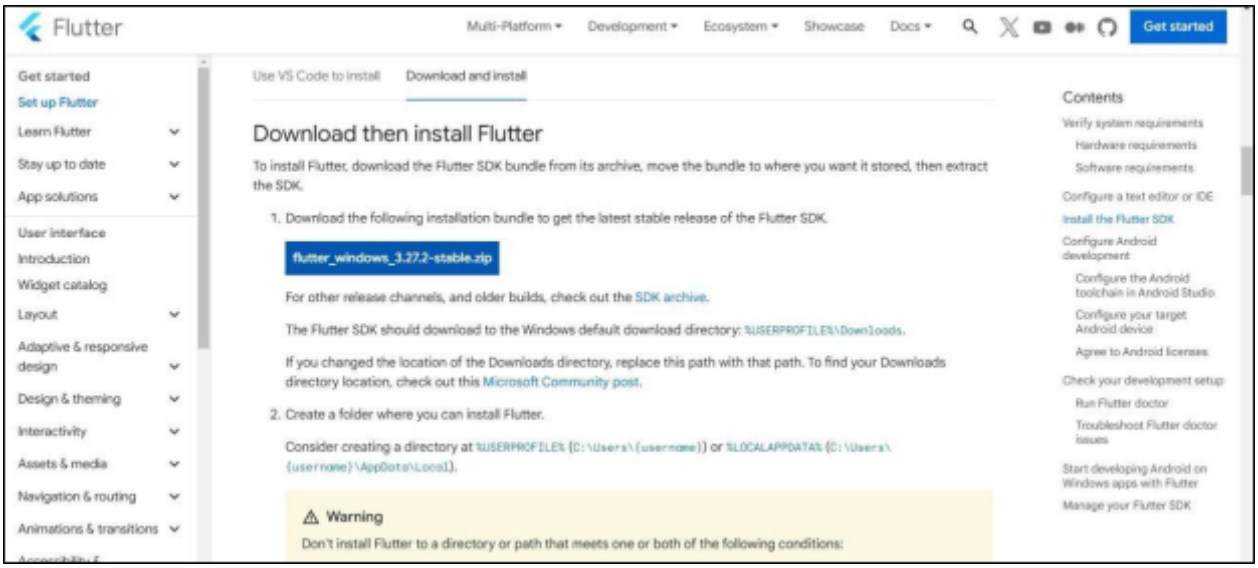
Step 1: Go to the official Flutter website: https://docs.flutter.dev/get-started/install



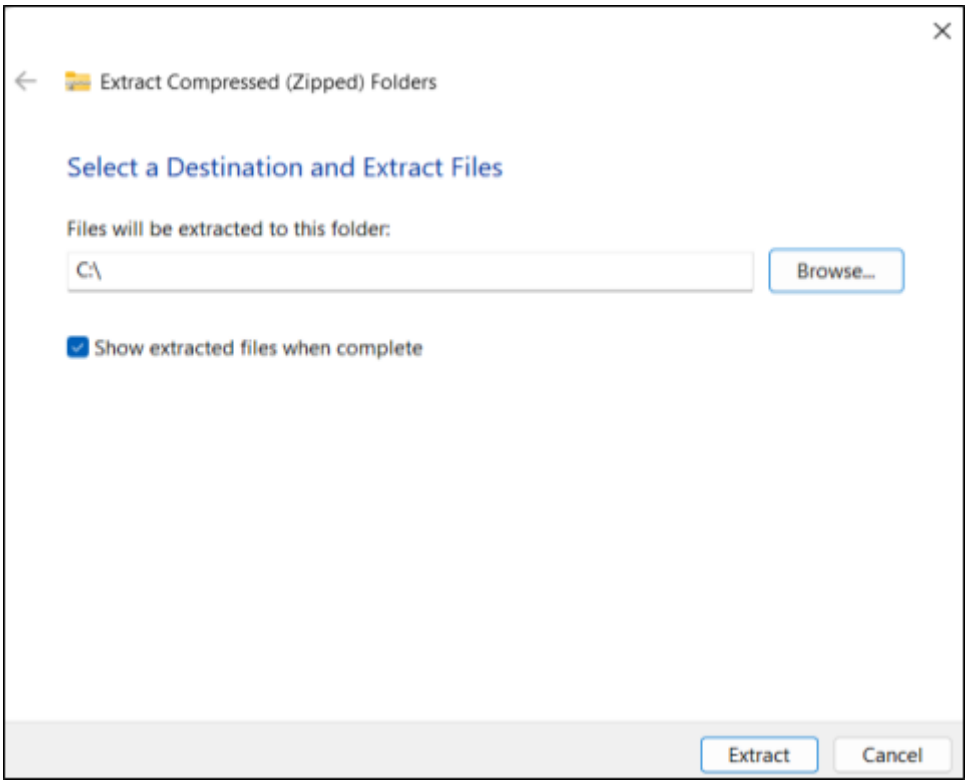
Step 2: To download the latest Flutter SDK, click on the Windows icon > Android



Step 3: For Windows, download the stable release (a .zip file).



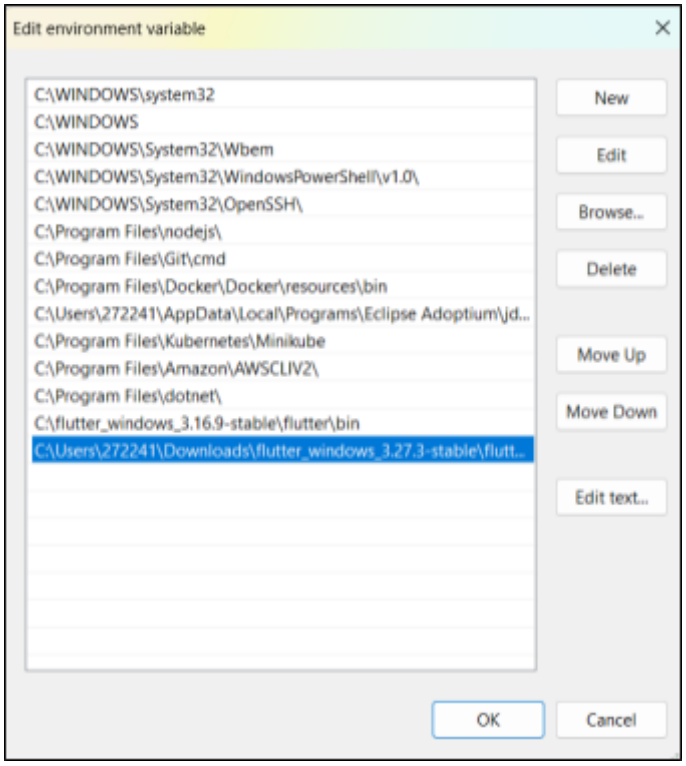
Step 4: Extract the ZIP file to a folder (e.g., C:\flutter).



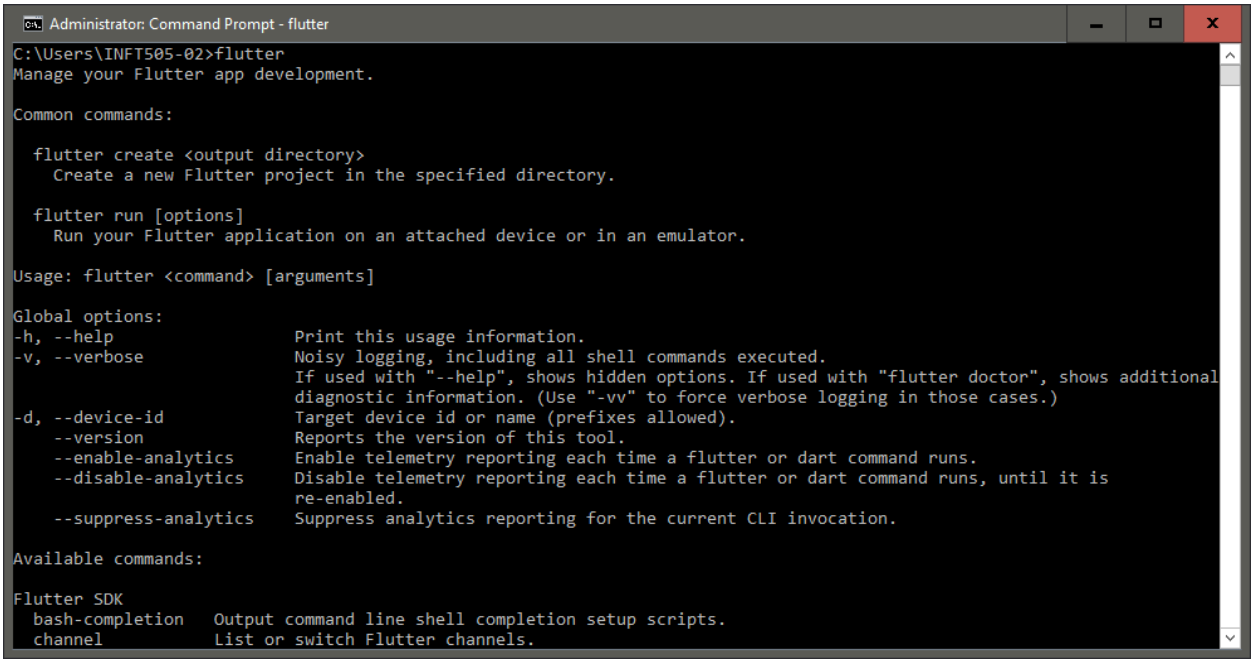
Step 5 :- Add Flutter to System PATH

Right-click on the Start Menu > System > Advanced system settings > Environment Variables. Under System Variables, find Path and click Edit.

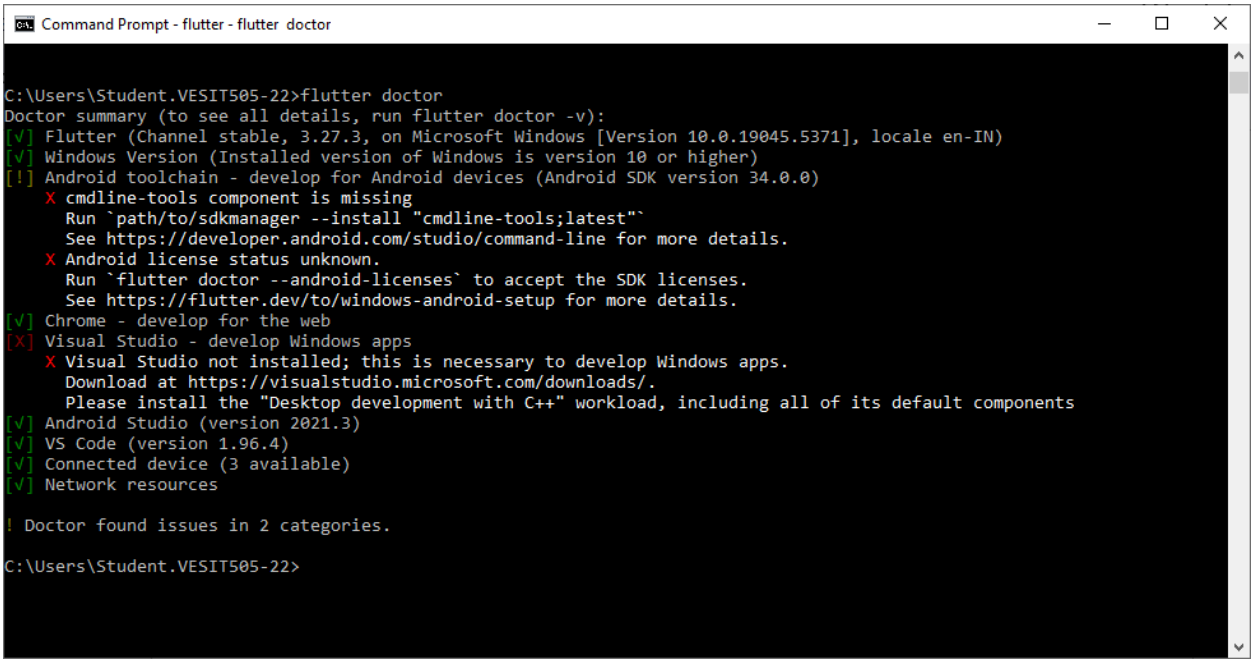
Add the full path to the flutter/bin directory (e.g., C:\flutter\bin).



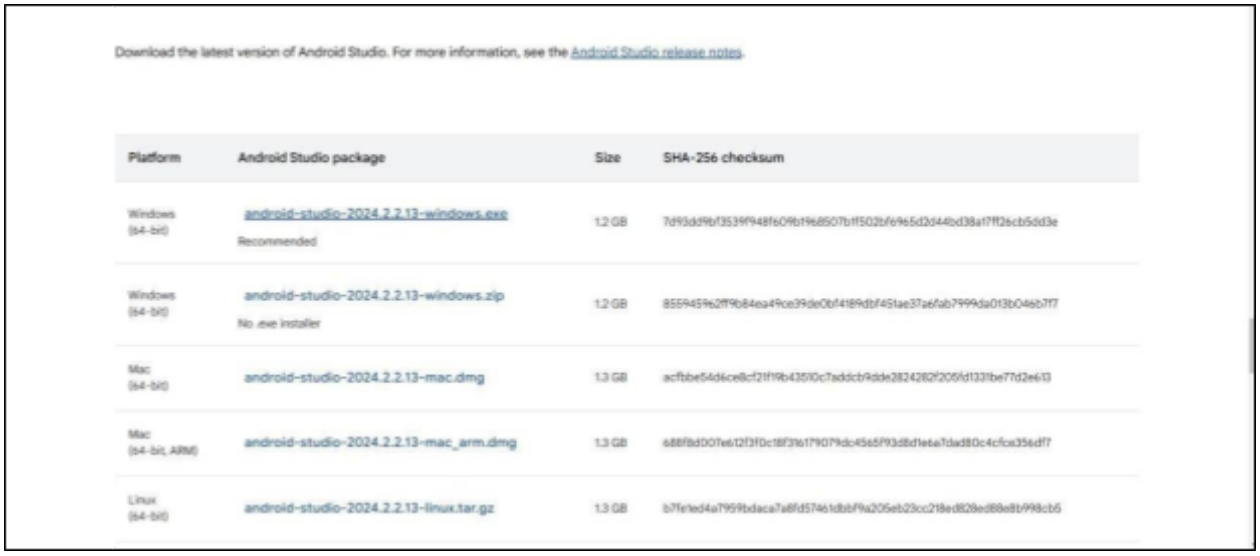
Step 6 : - Now, run the $ flutter command in command prompt.



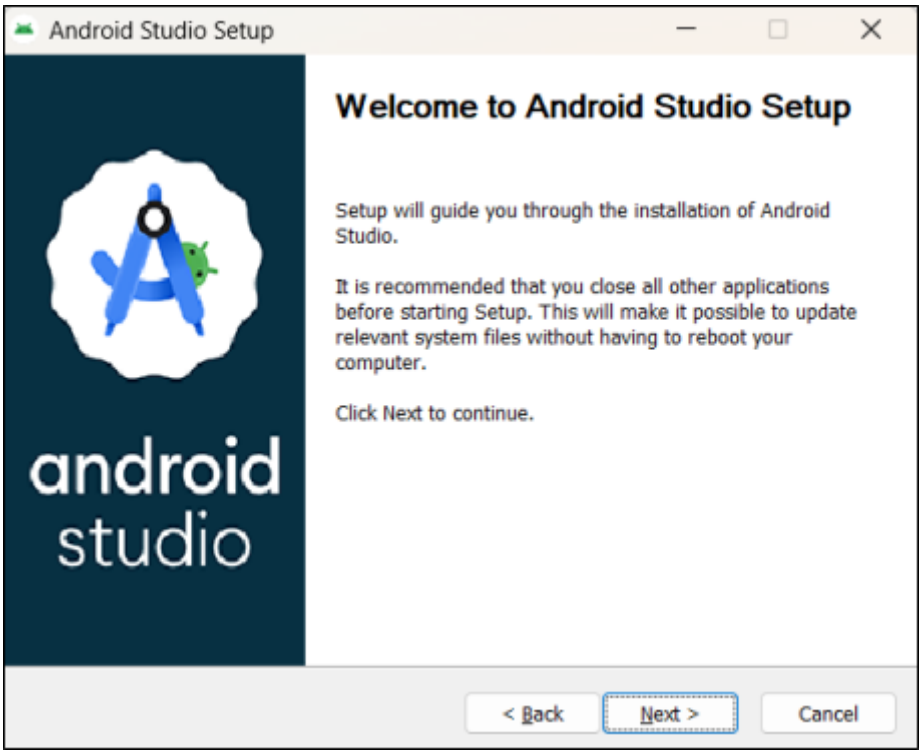
Step 7:- Run the $ flutter doctor command. This command checks for all the requirements of Flutter app development and displays a report of the status of your Flutter installation



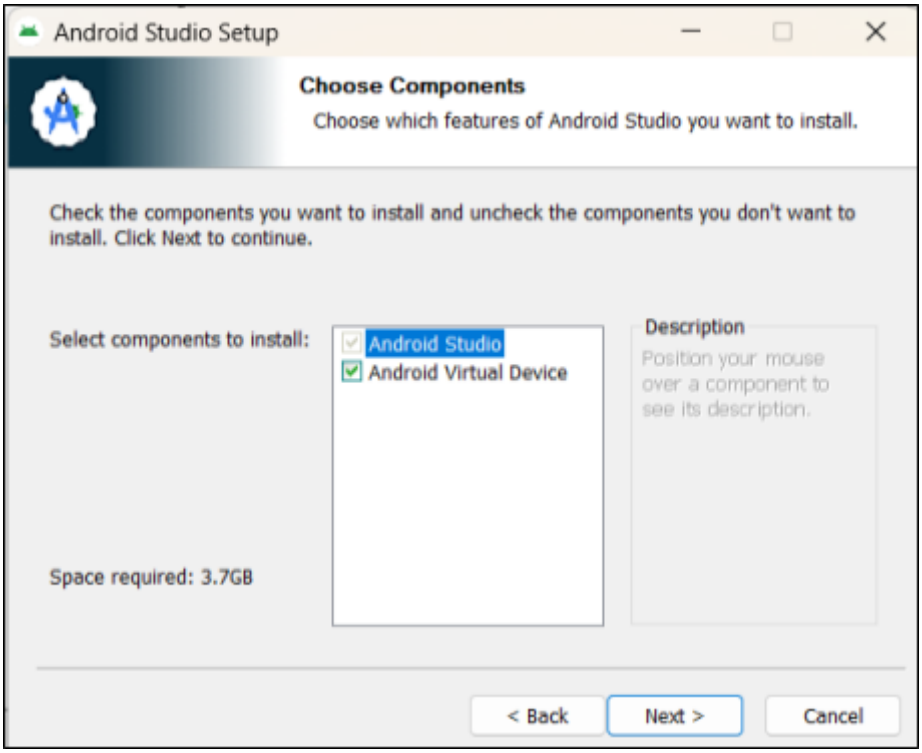
Step 8 : - Go to Android Studio and download the installer.



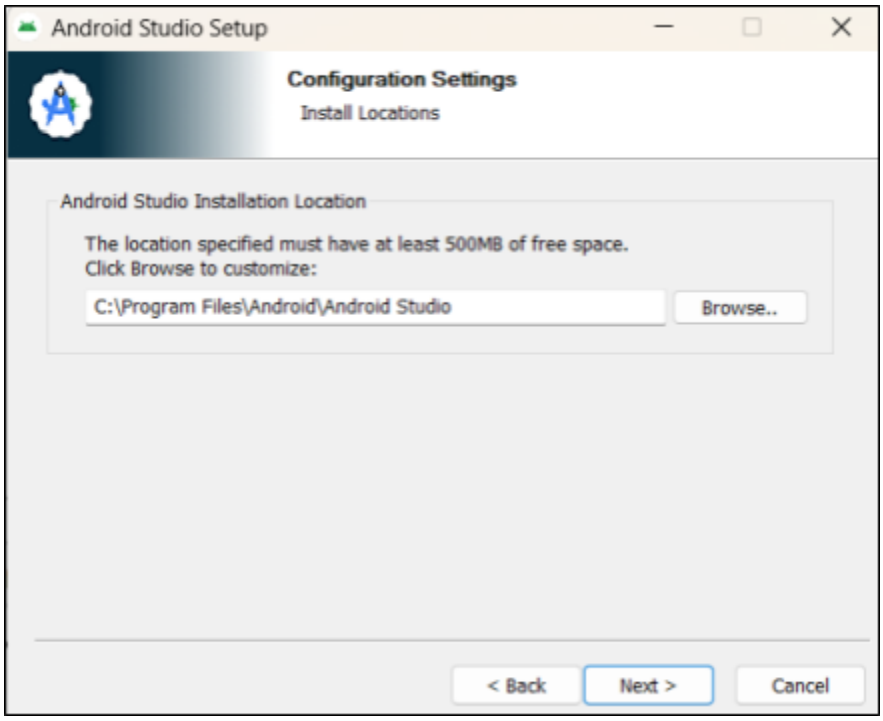
Step 8.1: - When the download is complete, open the .exe file and run it. You will get the following dialog box



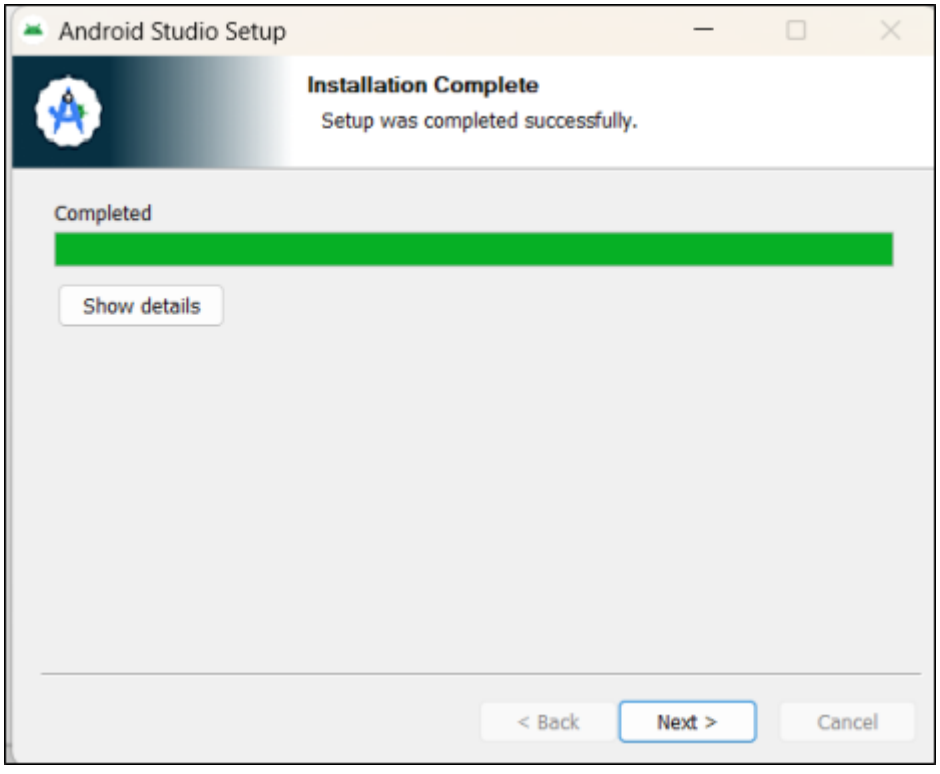
Step 8.2: - Select all the Checkboxes and Click on 'Next' Button.

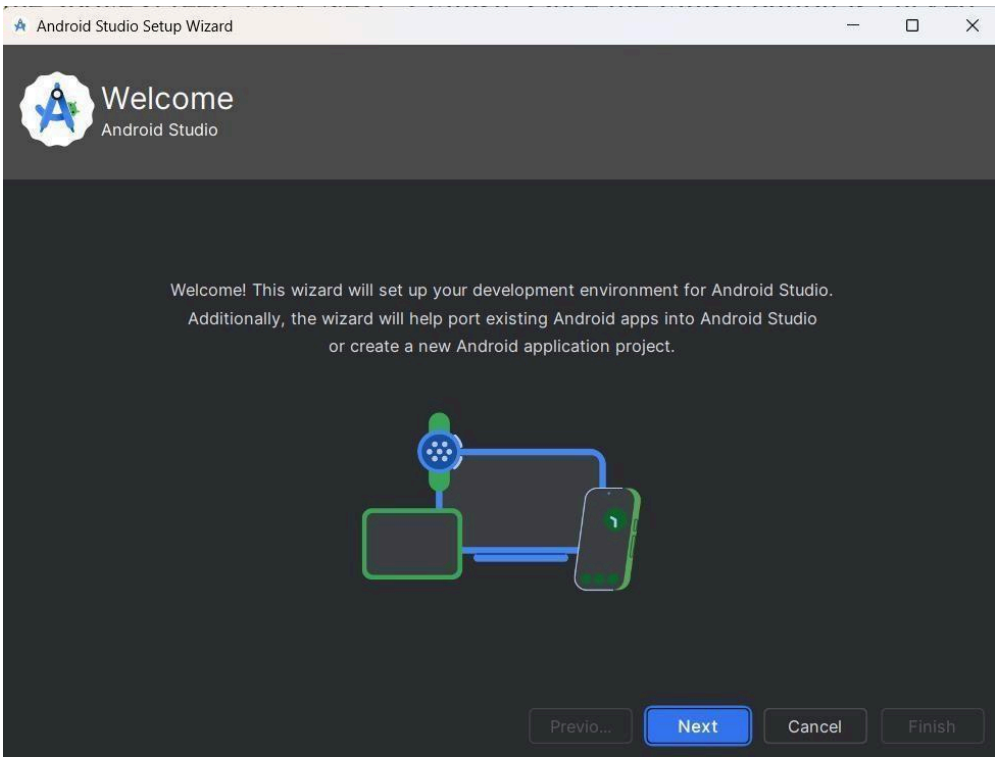


Step 8.3: - Change the destination as per your convenience and click on 'Next' Button.

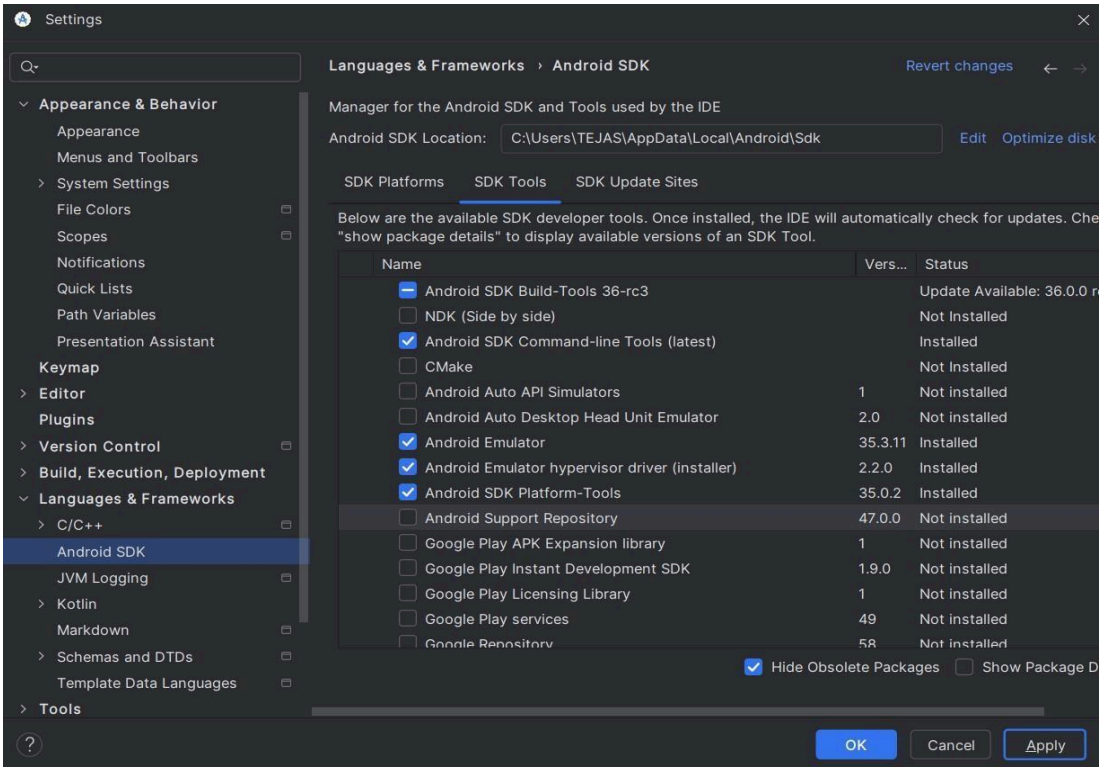


Step 8.4: - Follow the steps of the installation wizard. Once the installation wizard completes, you will get the following screen.



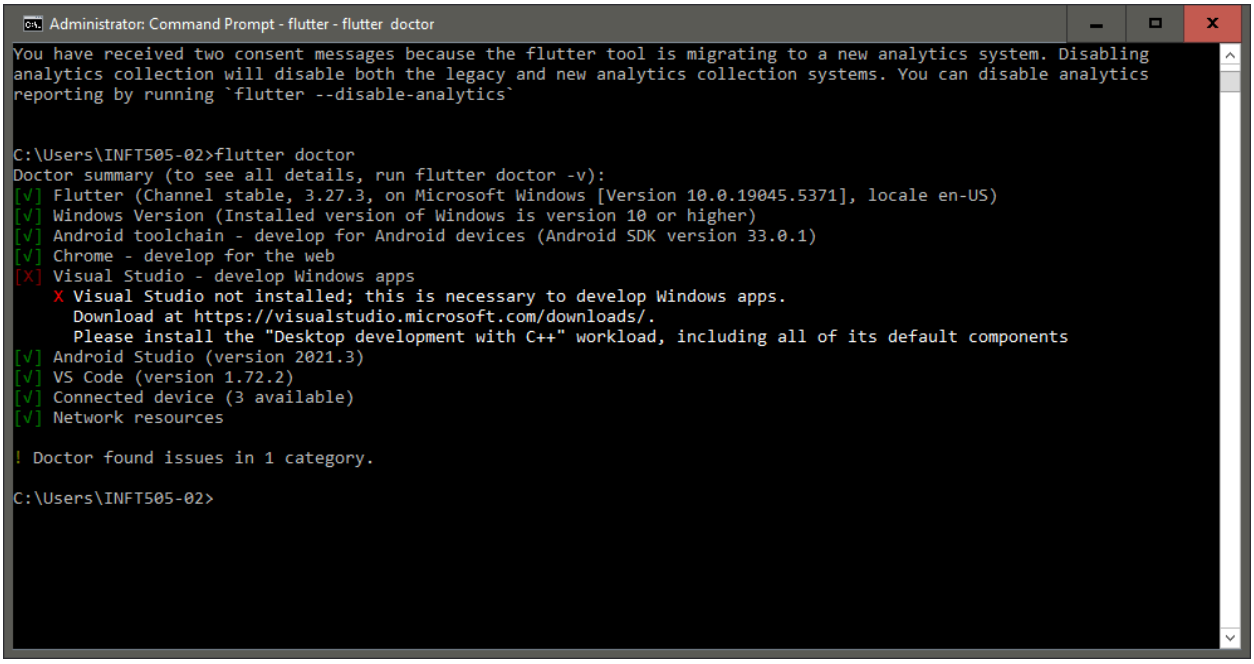


Step 8.5: - Go to Preferences > Appearance & Behavior > System Settings > Android SDK. Select the SDK Tools tab and check Android SDK Command-line Tools and Install it.

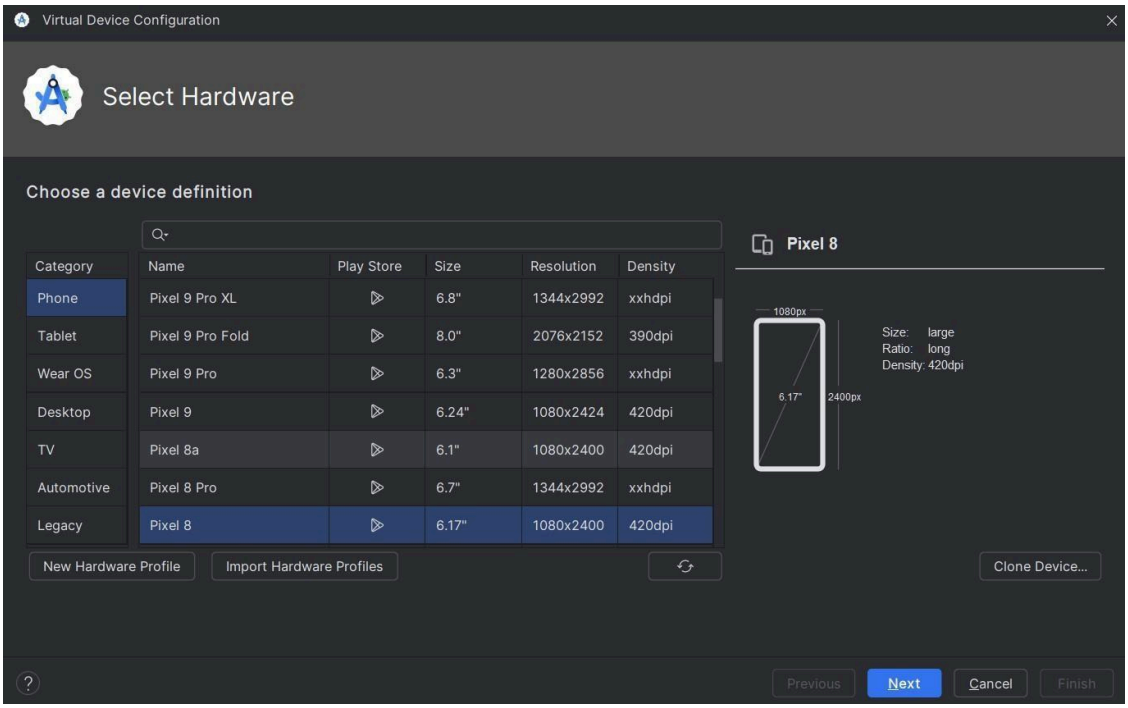


Step 9: - Open a terminal and run the following command

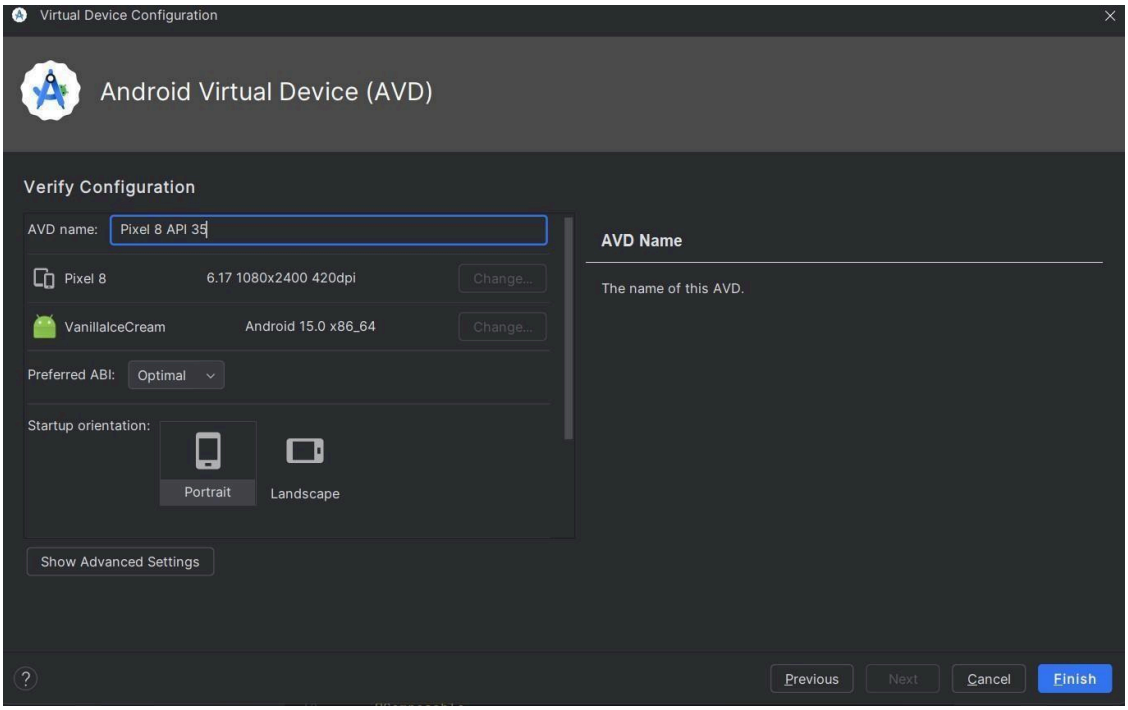
```
C:\Users\INFT505-02>flutter doctor --android-licenses
[===============                          ] 40% Fetch remote repository...        etch remote repository...
Warning: Additionally, the fallback loader failed to parse the XML.
[============================             ] 73% Fetch remote repository...        etch remote repository...
Warning: Additionally, the fallback loader failed to parse the XML.
[======================================] 100% Computing updates...
All SDK package licenses accepted.
```



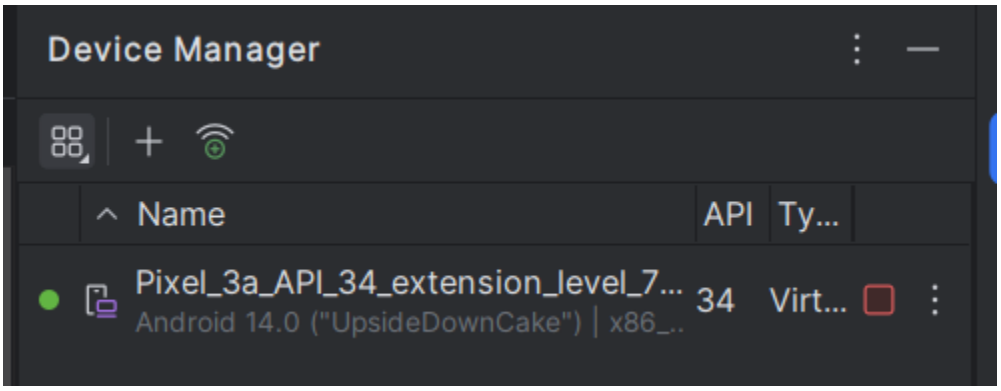
Step 10: - Next, you need to set up an Android emulator. It is responsible for running and testing the Flutter application



Step 10.1: - Open Android Studio and go to Tools > AVD Manager. Create a new virtual device.

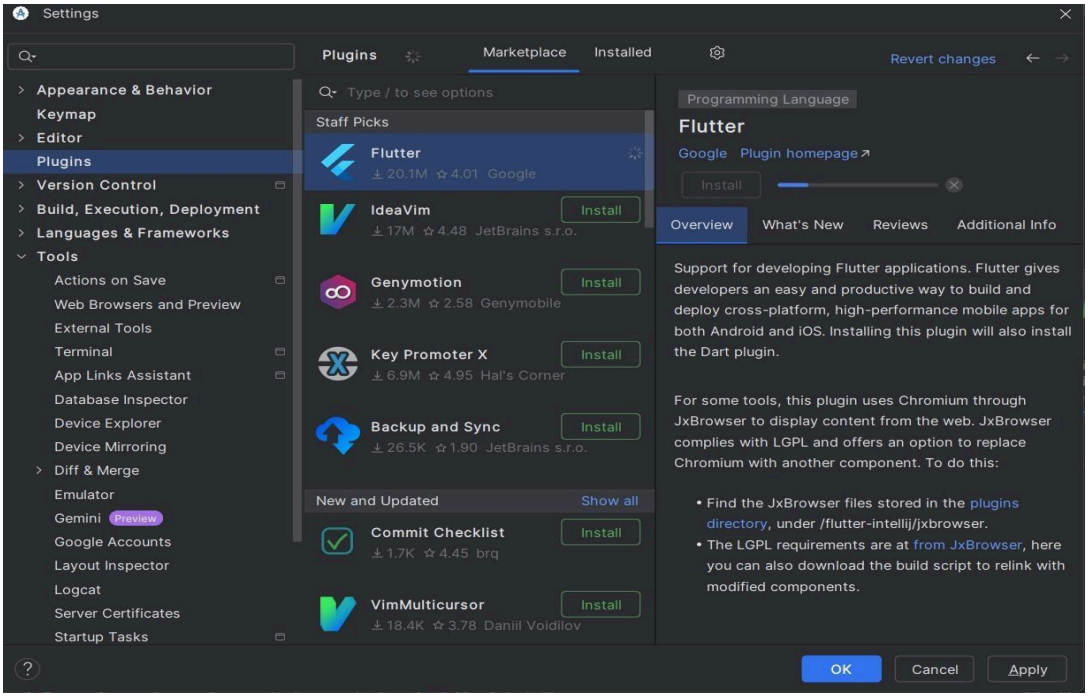


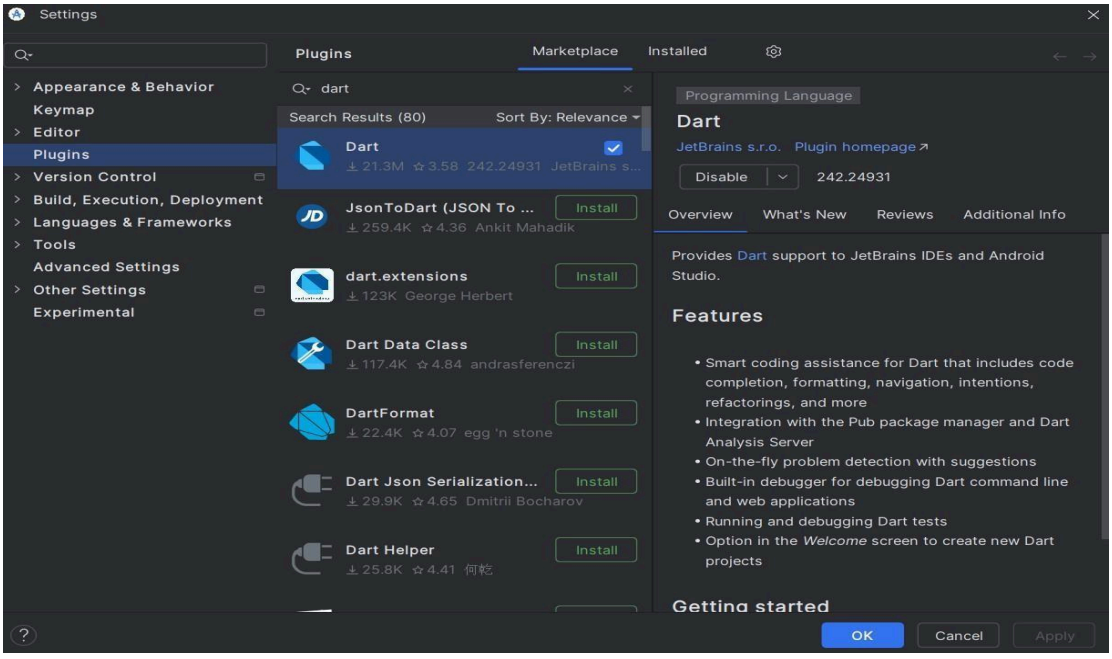
Step 10.2: - Click on the icon pointed into the red color rectangle. The Android emulator displayed as below screen



Step 11: - Now, install Flutter and Dart plugin for building Flutter application 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

Step 11.1: - Open the Android Studio and then go to File->Settings->Plugins. Now, search the Flutter plugin. If found, select Flutter plugin and click install





Step 11.2: - Restart the Android Studio

Step 12: - Go to File > New Project > Create Flutter Project, then select the project name and location, and click Next to proceed.

Code:

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

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

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

 // This widget is the root of your application.
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     title: 'Flutter Demo',
     theme: ThemeData(
       // This is the theme of your application.
       //
       // TRY THIS: Try running your application with "flutter run". You'll see
       // the application has a purple toolbar. Then, without quitting the app,
       // try changing the seedColor in the colorScheme below to Colors.green
       // and then invoke "hot reload" (save your changes or press the "hot
       // reload" button in a Flutter-supported IDE, or press "r" if you used
       // the command line to start the app).
       //
       // Notice that the counter didn't reset back to zero; the application
       // state is not lost during the reload. To reset the state, use hot
       // restart instead.
       //
       // This works for code too, not just values: Most code changes can be
       // tested with just a hot reload.
       colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
       useMaterial3: true,
     ),
     home: const MyHomePage(title: 'Flutter Demo Home Page'),
```

```
    );
  }
}

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

  // This widget is the home page of your application. It is stateful, meaning
  // that it has a State object (defined below) that contains fields that affect
  // how it looks.

  // This class is the configuration for the state. It holds the values (in this
  // case the title) provided by the parent (in this case the App widget) and
  // used by the build method of the State. Fields in a Widget subclass are
  // always marked "final".

  final String title;

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      // This call to setState tells the Flutter framework that something has
      // changed in this State, which causes it to rerun the build method below
      // so that the display can reflect the updated values. If we changed
      // _counter without calling setState(), then the build method would not be
      // called again, and so nothing would appear to happen.
      _counter++;
    });
  }
```

```
@override
Widget build(BuildContext context) {
  // This method is rerun every time setState is called, for instance as done
  // by the _incrementCounter method above.
  //
  // The Flutter framework has been optimized to make rerunning build methods
  // fast, so that you can just rebuild anything that needs updating rather
  // than having to individually change instances of widgets.
  return Scaffold(
    appBar: AppBar(
      // TRY THIS: Try changing the color here to a specific color (to
      // Colors.amber, perhaps?) and trigger a hot reload to see the AppBar
      // change color while the other colors stay the same.
      backgroundColor: Theme.of(context).colorScheme.inversePrimary,
      // Here we take the value from the MyHomePage object that was created by
      // the App.build method, and use it to set our appbar title.
      title: Text(widget.title),
    ),
    body: Center(
      // Center is a layout widget. It takes a single child and positions it
      // in the middle of the parent.
      child: Column(
        // Column is also a layout widget. It takes a list of children and
        // arranges them vertically. By default, it sizes itself to fit its
        // children horizontally, and tries to be as tall as its parent.
        //
        // Column has various properties to control how it sizes itself and
        // how it positions its children. Here we use mainAxisAlignment to
        // center the children vertically; the main axis here is the vertical
        // axis because Columns are vertical (the cross axis would be
        // horizontal).
        //
        // TRY THIS: Invoke "debug painting" (choose the "Toggle Debug Paint"
        // action in the IDE, or press "p" in the console), to see the
```

```
          // wireframe for each widget.
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            const Text(
              'Hello Siddhant',
            ),
            Text(
              '$_counter',
              style: Theme.of(context).textTheme.headlineMedium,
            ),
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: _incrementCounter,
        tooltip: 'Increment',
        child: const Icon(Icons.add),
      ), // This trailing comma makes auto-formatting nicer for build methods.
    );
  }
}
```

Output:

Project Title: Stack Overflow Clone **Roll No. 50**

MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	50
Name	Siddhant Sathe
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Common Widgets:

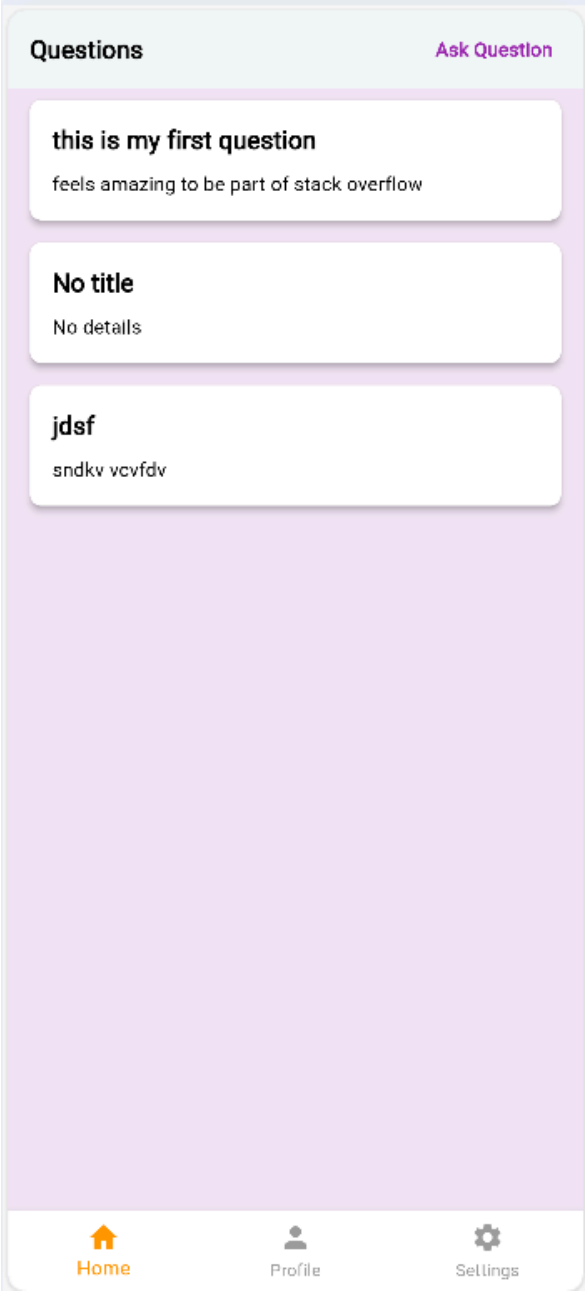
- **Scaffold:** Provides a basic screen structure with AppBar, Body,BottomNavigationBar.
- **AppBar:** Displays the page title and action buttons.
- **BottomNavigationBar:** Adds a navigation bar at the bottom.
- **SingleChildScrollView:** Allows scrolling when content overflows the screen.
- **Column:** Arranges widgets vertically.
- **Row:** Arranges widgets horizontally.
- **Center :**Centers content on the screen.
- **Padding :**Adds space around widgets.
- **SizedBox :**Adds empty space between widgets.
- **Container :**Used for styling elements (e.g., cards, backgrounds).
- **Form :**Wraps input fields to enable validation.
- **GlobalKey<FormState> :**Manages form state for validation.
- **TextFormField :**Input fields for email and password.
- **InputDecoration :**Adds styling like icons, labels, and borders inside TextFormField.
- **ElevatedButton :**For the Log in button.
- **TextButton :**For Forgot Password and Sign Up links.
- **GestureDetector :**Used to make tappable areas**.**
- **Text :** Displays plain text (e.g., "Questions", "The questions has not asked yet!").
- **Icon :** Displays icons (e.g., Icons.email, Icons.lock, Icons.home).
- **Image.asset :**Displays the Stack Overflow logo from assets.
- **SvgPicture.asset :**Displays an SVG image (useful for logos).
- **BoxDecoration :**Used inside Container for background color, border radius, and shadows.
- **BoxShadow :**Adds shadows to create a floating card effect.
- **Divider :**Adds a horizontal line to separate sections.
- **Navigator.push() :**Navigates from LoginPage to HomePage.
- **MaterialPageRoute :**Defines the transition animation when navigating to a new page.
- **StatefulWidget (if needed):** Used when the UI needs to update dynamically.
- **StatelessWidget:** Used for static pages like login and home screens.

Code:

```
import
'package:flutter/material.dart';
class HomePage extends
StatelessWidget {
  @override
  Widget build(BuildContext
context) {
      return Scaffold(
      backgroundColor:
Colors.purple.shade50,
      appBar: AppBar(
      backgroundColor:
Colors.white,
      elevation: 1,
      title: const Text(
            "Questions",
            style:
TextStyle(color: Colors.black,
fontSize: 18, fontWeight:
FontWeight.bold),
      ),
      actions: [
            TextButton(
            onPressed: () {
            // Navigate to the Ask
Question page
            },
            child: const Text(
            "Ask Question",
            style:
TextStyle(color: Colors.purple,
fontWeight: FontWeight.bold),
            ),
            ),
            const SizedBox(width:
10),
      ],
      ),
      body: Center(
      child: Container(
            padding: const
EdgeInsets.all(20),
            margin: const
EdgeInsets.symmetric(horizontal:
24),
            decoration:
BoxDecoration(
            color: Colors.white,
            borderRadius:
BorderRadius.circular(12),
            boxShadow: [
            BoxShadow(
                  color:
Colors.black12,
                  blurRadius: 10,
                  spreadRadius: 2,
            ),
            ],
            ),
            child: const Text(
            "The questions has not
asked yet!",
            style:
TextStyle(fontSize: 16,
fontWeight: FontWeight.bold),
            ),
      ),
      ),
      bottomNavigationBar:
BottomNavigationBar(
      backgroundColor:
Colors.white,
      selectedItemColor:
Colors.orange,
      unselectedItemColor:
Colors.grey,
      showUnselectedLabels: true,
      items: const [

BottomNavigationBarItem(
            icon:
Icon(Icons.home),
            label: "Home",
            ),

BottomNavigationBarItem(
            icon:
Icon(Icons.person),
            label: "Profile",
            ),
```

```
BottomNavigationBarItem(                    ),
            icon:                           );
Icon(Icons.settings), // Added          }
Settings Icon                       }
            label: "Settings",
            ),
      ],
```

Output:



MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	50
Name	Siddhant Sathe
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Theory:

Flutter is a versatile open-source UI framework , which allows developers to build natively compiled applications for mobile, web, and desktop platforms from a single codebase. One of the key strengths of Flutter is its flexibility in creating highly customizable UIs.
This practical focuses on incorporating essential visual elements—icons, images, and custom fonts—into a Flutter application. These elements enhance the visual appeal and usability of the app, providing an engaging experience for users.
A flutter app when built has both assets (resources) and code. Assets are available and deployed during runtime. The asset is a file that can include static data, configuration files, icons, and images. The Flutter app supports many image formats, such as JPEG, WebP, PNG, GIF, animated WebP/GIF, BMP, and WBMP. Visual elements play a significant role in app development.

• **Enhanced User Experience**: Images and icons make your app visually appealing and user-

friendly.

• **Information Conveyance**: They convey information quickly and intuitively. A well-chosen icon can replace lengthy text.

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

➢ Adding Icons in Flutter

Flutter provides built-in material design icons through the Icons class. Custom icons can also

be added using third-party packages such as flutter_launcher_icons and font_awesome_flutter.

```
Icon(
Icons.home,
size: 40,
);
```

➢ Adding Images in Flutter

Flutter supports images from three sources:

1. Assets (Stored locally in the project)

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

• Declare the image in pubspec.yaml

```
flutter:
assets:
- assets/images/sample.png
```

• Display the image in the app

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

2. Network (Fetched from the internet)

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

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

3. Memory or File (Stored on the device)

➢ Adding Custom Fonts in Flutter

By default, Flutter uses the Roboto font, but custom fonts can be added for a unique UI.

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

• Declare the font in pubspec.yaml

• Use the font in the app

Text(

'Custom Font Example',

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

);

Code:login_page.dart

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

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

  @override
  Widget build(BuildContext context) {
        return Scaffold(
        backgroundColor:
Colors.purple.shade50,
        body: Center(
        child: SingleChildScrollView(
        child: Column(
        mainAxisAlignment:
MainAxisAlignment.center,
        children: [

SvgPicture.asset('assets/images/logo.svg',
height: 60),
        //
Image.asset('assets/images/logo.svg', height:
60), // Add your logo in assets
        const SizedBox(height: 20),
        Padding(
        padding: const
EdgeInsets.symmetric(horizontal: 24),
        child: Form(
                key: _formKey,
                child: Container(
                padding: const
EdgeInsets.all(20),
                decoration: BoxDecoration(
                color: Colors.white,
                borderRadius:
BorderRadius.circular(12),
                boxShadow: [
                BoxShadow(
                color: Colors.black12,
                blurRadius: 10,
                spreadRadius: 2,
                ),
```

```
                ],
                ),
                child: Column(
                children: [
                TextFormField(
                decoration: InputDecoration(
                prefixIcon: Icon(Icons.email),
                labelText: "Email",
                border: OutlineInputBorder(
                        borderRadius:
BorderRadius.circular(8),
                ),
                ),
                keyboardType:
TextInputType.emailAddress,
                validator: (value) {
                if (value == null ||
value.isEmpty) {
                        return "Please enter
your email";
                }
                return null;
                },
                ),
                const SizedBox(height: 15),
                TextFormField(
                decoration: InputDecoration(
                prefixIcon: Icon(Icons.lock),
                labelText: "Password",
                border: OutlineInputBorder(
                        borderRadius:
BorderRadius.circular(8),
                ),
                ),
                obscureText: true,
                validator: (value) {
                if (value == null ||
value.isEmpty) {
                        return "Please enter
your password";
                }
                return null;
                },
                ),
                const SizedBox(height: 20),
                SizedBox(
                        width: double.infinity,
                        child: ElevatedButton(
                        style:
ElevatedButton.styleFrom(
                                backgroundColor:
Colors.purple.shade100,
                                shape:
RoundedRectangleBorder(
                                borderRadius:
BorderRadius.circular(8),
                                ),
                                padding: const
EdgeInsets.symmetric(vertical: 12),
                        ),
                        onPressed: () {
                                if
(_formKey.currentState!.validate()) {
                                Navigator.push(
                                context,

MaterialPageRoute(builder: (context) =>
HomePage()),
                                );


                                }
                        },
                        child: const Text(
                                "Log in",
                                style: TextStyle(color:
Colors.black),
                        ),
                        ),
                        ),
                        const SizedBox(height: 10),
                        TextButton(
                        onPressed: () {},
                        child: const Text(
                        "Forgot Password",
                        style: TextStyle(color:
Colors.blue),
                        ),
                        ),
                        const Divider(),
                        Row(
                        mainAxisAlignment:
MainAxisAlignment.center,
```

```
                children: [
                const Text("Don't have an
account?"),
                TextButton(
                        onPressed: () {},
                        child: const Text(
                        "Sign up",
                        style: TextStyle(color:
Colors.blue),
                        ),
                ),
                ],
                )
                ],
                ),
                ),
        ),
        )
        ],
        ),
        ),
        ),
        );
  }
```

Code:profile_page.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_svg/flutter_svg.dart';
import
‘package:google_fonts/google_fonts.dart';

class ProfilePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
        return Scaffold(
        backgroundColor:
Colors.purple.shade50,
        body: SingleChildScrollView(
        child: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Column(
        crossAxisAlignment:
CrossAxisAlignment.start,
        children: [
        Row(
        children: [
                CircleAvatar(
                radius: 40,
                child:
SvgPicture.asset('project/assets/images/icon.s
vg'),
                ),
                const SizedBox(width: 16),
                Column(
                crossAxisAlignment:
CrossAxisAlignment.start,
                children: const [
                Text(
                'Siddhant Sathe',
                style: GoogleFonts.montserrat(
                fontSize: 20,
                fontWeight: FontWeight.bold,
                ),
                ),
                SizedBox(height: 4),
                Text(
                '📅 Member since 1st Feb
2025',
```

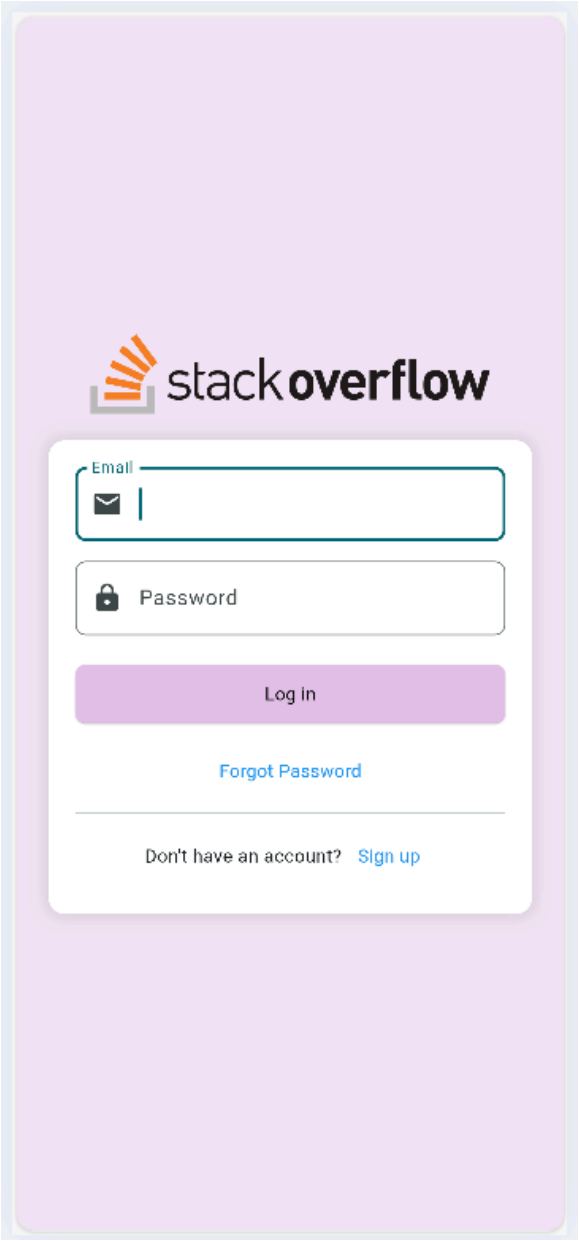
```
                style: TextStyle(
                color: Colors.grey,
                fontSize: 14,
                ),
                ),
                ],
                ),
        ],
        ),
        const SizedBox(height: 20),
        Row(
        mainAxisAlignment:
MainAxisAlignment.spaceBetween,
        children: const [
                Text(
                'About',
                style: TextStyle(
                fontSize: 18,
                fontWeight: FontWeight.bold,
                ),
                ),
                Text(
                'Log out',
                style: TextStyle(
                color: Colors.blue,
                fontWeight: FontWeight.bold,
                ),
                ),
        ],
        ),
        const SizedBox(height: 10),
        Container(
        width: double.infinity,
        padding: const EdgeInsets.all(16),
        decoration: BoxDecoration(
                color: Colors.white,
                borderRadius:
BorderRadius.circular(12),
        ),
        child: const Text(
                'No about me has been yet',
                style: TextStyle(
                color: Colors.grey,
                ),
        ),
        ),
        const SizedBox(height: 20),
        Row(
        mainAxisAlignment:
MainAxisAlignment.spaceBetween,
        children: const [
                Text(
                'Answers',
                style: TextStyle(
                fontSize: 18,
                fontWeight: FontWeight.bold,
                ),
                ),
                Text(
                'See All',
                style: TextStyle(
                color: Colors.blue,
                fontWeight: FontWeight.bold,
```

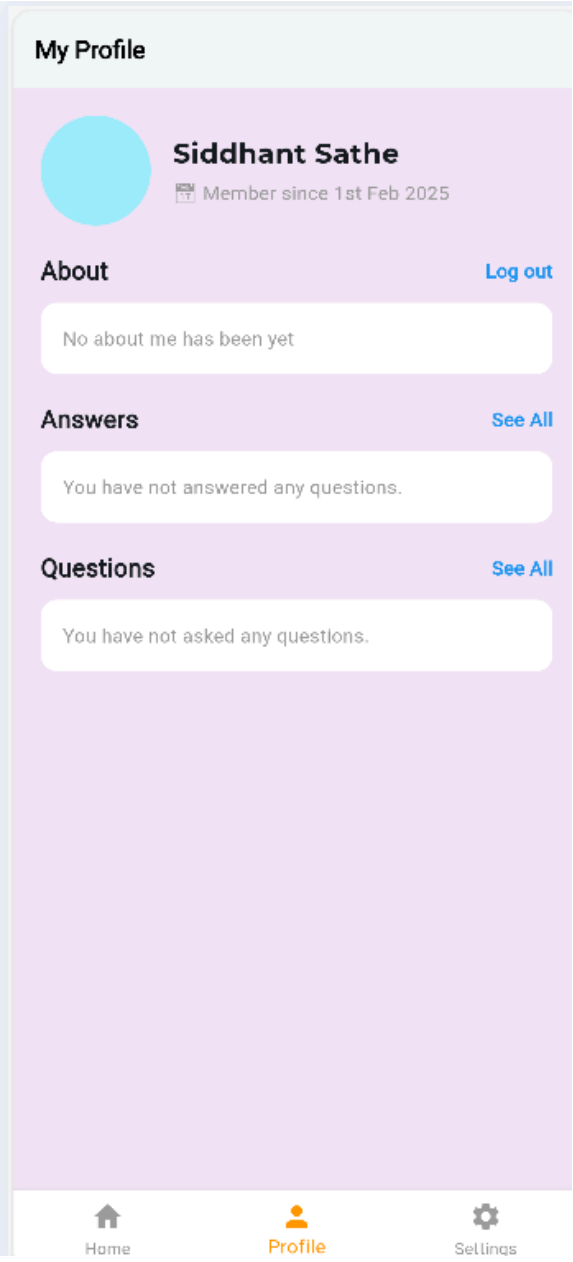
```
                ),
                ),
            ],
            ),
            const SizedBox(height: 10),
            Container(
            width: double.infinity,
            padding: const EdgeInsets.all(16),
            decoration: BoxDecoration(
                color: Colors.white,
                borderRadius:
BorderRadius.circular(12),
            ),
            child: const Text(
                'You have not answered any
questions.',
                style: TextStyle(
                color: Colors.grey,
                ),
            ),
            ),
            const SizedBox(height: 20),
            Row(
            mainAxisAlignment:
MainAxisAlignment.spaceBetween,
            children: const [
                Text(
                'Questions',
                style: TextStyle(
                fontSize: 18,
                fontWeight: FontWeight.bold,
                ),
                ),
                Text(
                'See All',
                style: TextStyle(
                color: Colors.blue,
                fontWeight: FontWeight.bold,
                ),
                ),
            ],
            ),
            const SizedBox(height: 10),
            Container(
            width: double.infinity,
            padding: const EdgeInsets.all(16),
            decoration: BoxDecoration(
                color: Colors.white,
                borderRadius:
BorderRadius.circular(12),
            ),
            child: const Text(
                'You have not asked any
questions.',
                style: TextStyle(
                color: Colors.grey,
                ),
            ),
            ),
            ],
            ),
            ),
            ),
            );
```

}

}

Output:


stackoverflow
Email
Password
Log in
Forgot Password
Don't have an account? Sign up


My Profile
Siddhant Sathe
Member since 1st Feb 2025
About
Log out
No about me has been yet
Answers
See All
You have not answered any questions.
Questions
See All
You have not asked any questions.
Home
Profile
Settings

MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	50
Name	Siddhant Sathe
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Theory:

A form in Flutter is a structured container that collects user input through various fields like text fields, dropdowns, checkboxes, and buttons. It plays a crucial role in applications that require user data entry, such as login pages, registration forms, and feedback submissions. Flutter provides the Form widget, which works alongside TextFormField and other input elements to manage validation, state handling, and error messages efficiently. By using form validation techniques, developers can ensure data accuracy and enhance user experience.

When you create a form, it is necessary to provide the GlobalKey. This key uniquely identifies the form and allows you to do any validation in the form fields. The form widget uses child widget TextFormField to provide the users to enter the text field. This widget renders a material design text field and also allows us to display validation errors when they occur.

Creation of a Form

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

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

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

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

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

➢ Different types of input require appropriate keyboard types, such as TextInputType.number for numeric fields or TextInputType.emailAddress for email fields.

➢ Proper state management is needed to store and retrieve user input, ensuring the form data is processed correctly.

➢ A submit button is necessary to trigger form validation and submit the collected data for further processing**.**

Some Properties of Form Widget

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

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

• **autovalidateMode:** An enum that specifies when the form should automatically validate its fields.

Some Methods of Form Widget

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

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

• **reset():** Resets the form to its initial state, clearing any user-entered data. • currentState: A getter that returns the current FormState associated with the form.

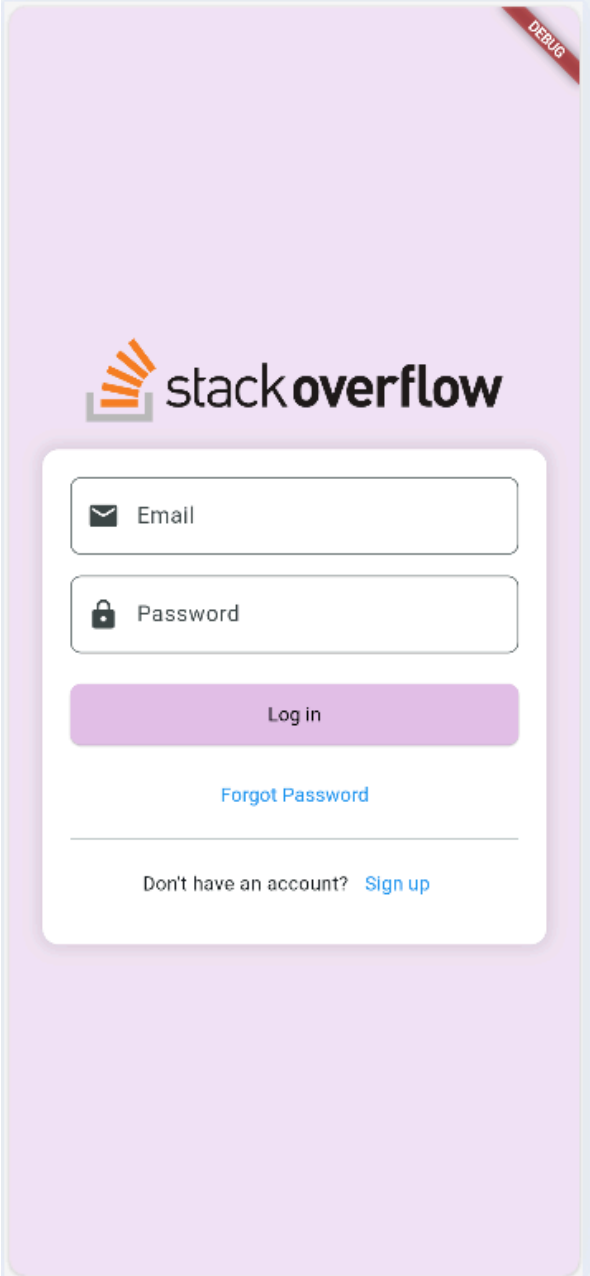
Code:

```
import
'package:flutter/material.dart'
;
import
'package:flutter_svg/flutter_sv
g.dart';
import 'home_page.dart';

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

  @override
  Widget build(BuildContext
context) {
     return Scaffold(
     backgroundColor:
Colors.purple.shade50,
     body: Center(
     child:
SingleChildScrollView(
          child: Column(
          mainAxisAlignment:
MainAxisAlignment.center,
          children: [

SvgPicture.asset('assets/images
/logo.svg', height: 60),
          //
Image.asset('assets/images/logo
.svg', height: 60), // Add your
logo in assets
          const
SizedBox(height: 20),
          Padding(
               padding: const
EdgeInsets.symmetric(horizontal
: 24),
               child: Form(
               key: _formKey,
               child:
Container(
                    padding:
const EdgeInsets.all(20),

decoration: BoxDecoration(
                    color:
Colors.white,

borderRadius:
BorderRadius.circular(12),
                    boxShadow:
[
                    BoxShadow(

color: Colors.black12,

blurRadius: 10,

spreadRadius: 2,
                    ),
                    ],
                    ),
                    child:
Column(
```

```
                    children:
[

TextFormField(

decoration: InputDecoration(

prefixIcon: Icon(Icons.email),

labelText: "Email",

border: OutlineInputBorder(

borderRadius:
BorderRadius.circular(8),
                            ),
                            ),

keyboardType:
TextInputType.emailAddress,

validator: (value) {
                            if
(value == null ||
value.isEmpty) {

return "Please enter your
email";
                            }

return null;
                            },
                        ),
                        const
SizedBox(height: 15),

TextFormField(

decoration: InputDecoration(

prefixIcon: Icon(Icons.lock),

labelText: "Password",

border: OutlineInputBorder(

borderRadius:
BorderRadius.circular(8),
                            ),
                            ),

obscureText: true,

validator: (value) {
                            if
(value == null ||
value.isEmpty) {

return "Please enter your
password";
                            }

return null;
                            },
                        ),
                        const
SizedBox(height: 20),
                        SizedBox(

width: double.infinity,

child: ElevatedButton(

style:
ElevatedButton.styleFrom(

backgroundColor:
Colors.purple.shade100,

shape: RoundedRectangleBorder(

borderRadius:
BorderRadius.circular(8),
                            ),

padding: const
EdgeInsets.symmetric(vertical:
12),
                            ),
```

```
onPressed: () {
                            if
(_formKey.currentState!.validat
e()) {

Navigator.push(

     context,

     MaterialPageRoute(builder:
(context) => HomePage()),

);

                            }
                            },

child: const Text(
                            "Log
in",

style: TextStyle(color:
Colors.black),
                            ),
                            ),
                      ),
                      const
SizedBox(height: 10),

TextButton(

onPressed: () {},

child: const Text(

"Forgot Password",

style: TextStyle(color:
Colors.blue),
                                  ),
                            ),
                            const
Divider(),
                            Row(

mainAxisAlignment:
MainAxisAlignment.center,

children: [
                                  const
Text("Don't have an account?"),

TextButton(

onPressed: () {},

child: const Text(

"Sign up",

style: TextStyle(color:
Colors.blue),
                                  ),
                                  ),
                                  ],
                            )
                            ],
                      ),
                      ),
                )
                ],
                ),
          ),
          ),
          );
    }
}
```

Output:


DEBUG
stackoverflow
Email
Password
Log in
Forgot Password
Don't have an account? Sign up

MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	50
Name	Siddhant Sathe
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Theory:

In Flutter, the screens and pages are known as routes, and these routes are just a widget. In Android, a route is similar to an Activity.

In any mobile app, navigating to different pages defines the workflow of the application, and the way to handle the navigation is known as routing. Flutter provides a basic routing class MaterialPageRoute and two methods Navigator.push() and Navigator.pop() that shows how to navigate between two routes. The following steps are required to start navigation in your application.

Gestures enable the app to respond to user interactions, making the application more dynamic and responsive.

Navigation and Routing in Flutter

1. Using Navigator Widget

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

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

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

2. Named Routes

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

Handling Gestures in Flutter

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

Tap Gestures

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

Long Press Gesture

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

Swipe and Drag Gestures

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

Code: home_page.dart

```
import 'package:flutter/material.dart';
import 'ask_question.dart';
import 'profile_page.dart';
import 'answer_page.dart';
import
'package:firebase_database/firebase_databas
e.dart';
import
'package:cloud_firestore/cloud_firestore.dart';

class HomePage extends StatefulWidget {
  @override
  State<HomePage> createState() =>
_HomePageState();
}

class _HomePageState extends
State<HomePage> {
  int _selectedIndex = 0;
  final List<String> _appBarTitles =
["Questions", "My Profile", "Settings"];

  // Firebase reference
  final DatabaseReference _database =
FirebaseDatabase.instance.ref('question_ans
wers');

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

  void addQuestion(String title, String details) {
        _database.push().set({
        'title': title,
        'details': details,
        'timestamp':
DateTime.now().toIso8601String(),
        });
  }

  List<Widget> get _pages => [
        StreamBuilder<QuerySnapshot>(
        stream:
FirebaseFirestore.instance.collection('questio
n_answers').snapshots(),
        builder: (context, snapshot) {
        if (snapshot.connectionState ==
ConnectionState.waiting) {
        return const Center(child:
CircularProgressIndicator());
        }

        if (!snapshot.hasData ||
snapshot.data!.docs.isEmpty) {
        return const Center(child: Text("No
questions asked yet"));
        }

        final questions = snapshot.data!.docs;

        return ListView.builder(
        itemCount: questions.length,
        itemBuilder: (context, index) {
        final question =
questions[index].data() as Map<String,
dynamic>;
        return Padding(
                padding: const
EdgeInsets.symmetric(vertical: 8.0, horizontal:
16.0),
                child: ElevatedButton(
                onPressed: () {
                Navigator.push(
                context,
                MaterialPageRoute(
                builder: (context) =>
QuestionDetailPage(
                title: question['title'] ?? 'No
title',
                details: question['details'] ??
'No details',
```

```
                timestamp:
(question['timestamp'] as
Timestamp).toDate().toString(),
                ),
                ),
                );
                },
                child: Column(
                crossAxisAlignment:
CrossAxisAlignment.start,
                children: [
                Align(
                alignment:
Alignment.centerLeft,
                child: Text(
                question['title'] ?? 'No title',
                style: const TextStyle(
                        fontSize: 18,
                        fontWeight:
FontWeight.bold,
                ),
                ),
                ),
                const SizedBox(height: 8),
                Align(
                alignment:
Alignment.centerLeft,
                child: Text(
                question['details'] ?? 'No
details',
                style: const TextStyle(fontSize:
14),
                ),
                ),
                ],
                ),
                style:
ElevatedButton.styleFrom(
                backgroundColor:
Colors.white,
                foregroundColor: Colors.black,
                elevation: 4,
                padding: const
EdgeInsets.all(16),
                shape:
RoundedRectangleBorder(
                        borderRadius:
BorderRadius.circular(8),
                ),
                ),
                ),
        );
        },
        );
        },
        ),
        ProfilePage(),
        const Center(
        child: Text("Settings Page", style:
TextStyle(fontSize: 20, fontWeight:
FontWeight.bold)),
        ),
        ];

  @override
  Widget build(BuildContext context) {
        return Scaffold(
        backgroundColor:
Colors.purple.shade50,
        appBar: AppBar(
        backgroundColor: Colors.white,
        elevation: 1,
        title: Text(
        _appBarTitles[_selectedIndex],
        style: const TextStyle(color:
Colors.black, fontSize: 18, fontWeight:
FontWeight.bold),
        ),
        actions: _selectedIndex == 0
        ? [
        TextButton(
                onPressed: () {
                Navigator.push(
                context,
                MaterialPageRoute(
                builder: (context) =>
AskQuestionPage(onSubmit: addQuestion),
                ),
                );
                },
                child: const Text(
                "Ask Question",
```

```
                    style: TextStyle(color:
Colors.purple, fontWeight: FontWeight.bold),
                    ),
          ),
          const SizedBox(width: 10),
          ]
          : [],
          ),
          body: _pages[_selectedIndex],
          bottomNavigationBar:
BottomNavigationBar(
          backgroundColor: Colors.white,
          selectedItemColor: Colors.orange,
          unselectedItemColor: Colors.grey,
          currentIndex: _selectedIndex,
          onTap: _onItemTapped,
          showUnselectedLabels: true,
          items: const [
          BottomNavigationBarItem(icon:
Icon(Icons.home), label: "Home"),
          BottomNavigationBarItem(icon:
Icon(Icons.person), label: "Profile"),
          BottomNavigationBarItem(icon:
Icon(Icons.settings), label: "Settings"),
          ],
          ),
          );
  }
}

class QuestionDetailPage extends
StatelessWidget {
  final String title;
  final String details;
  final String timestamp;

  const QuestionDetailPage({
          Key? key,
          required this.title,
          required this.details,
          required this.timestamp,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
          return Scaffold(
          appBar: AppBar(
          title: const Text("Question Details"),
          ),
          body: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Column(
          crossAxisAlignment:
CrossAxisAlignment.start,
          children: [
          Text(
          title,
          style: const TextStyle(fontSize: 24,
fontWeight: FontWeight.bold),
          ),
          const SizedBox(height: 16),
          Text(
          details,
          style: const TextStyle(fontSize: 18),
          ),
          const SizedBox(height: 16),
          Text(
          "Posted on:
${DateTime.parse(timestamp).toString()}",
          style: const TextStyle(fontSize: 14,
color: Colors.grey),
          ),
          const SizedBox(height: 16),
          ElevatedButton(
          onPressed: () {
          Navigator.push(
                    context,
                    MaterialPageRoute(
                    builder: (context) =>
AnswerQuestionPage(title: title),
                    ),
          );
          },
          child: const Text("Answer Question"),
          ),
          const SizedBox(height: 16),
          const Text(
          "Answers:",
          style: TextStyle(fontSize: 18,
fontWeight: FontWeight.bold),
          ),
          const SizedBox(height: 8),
```

```
        Expanded(
        child:
StreamBuilder<QuerySnapshot>(
        stream: FirebaseFirestore.instance
                .collection('question_answers')
                .doc(title)
                .collection('answers')
                .orderBy('timestamp',
descending: true)
                .snapshots(),
        builder: (context, snapshot) {
                if (snapshot.connectionState
== ConnectionState.waiting) {
                return const Center(child:
CircularProgressIndicator());
                }

                if (!snapshot.hasData ||
snapshot.data!.docs.isEmpty) {
                return const Center(child:
Text("No answers yet"));
                }

                final answers =
snapshot.data!.docs;

                return ListView.builder(
                itemCount: answers.length,
                itemBuilder: (context, index) {
                final answer =
answers[index].data() as Map<String,
dynamic>;
                return Padding(
                padding: const
EdgeInsets.symmetric(vertical: 8.0, horizontal:
16.0),
                child: Container(
                padding: const
EdgeInsets.all(16),
                decoration: BoxDecoration(
                color: Colors.white,
                borderRadius:
BorderRadius.circular(8),
                boxShadow: [
                        BoxShadow(
                        color:
Colors.grey.withOpacity(0.2),
                        spreadRadius: 1,
                        blurRadius: 4,
                        offset: const Offset(0,
2),
                        ),
                ],
                ),
                child: Column(
                crossAxisAlignment:
CrossAxisAlignment.start,
                children: [
                        Text(
                        answer['answer'] ??
'No answer',
                        style: const
TextStyle(fontSize: 16),
                        ),
                        const SizedBox(height:
8),
                        Text(
                        "Answered on:
${DateTime.parse(answer['timestamp']).toStri
ng()}",
                        style: const
TextStyle(fontSize: 12, color: Colors.grey),
                        ),
                ],
                ),
                ),
                );
                },
                );
        },
        ),
        ),
        ],
        ),
        ),
        );
  }
}
```

Code: ask_question.dart

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

class AskQuestionPage extends
StatefulWidget {
  final Function(String, String) onSubmit;

  AskQuestionPage({required this.onSubmit});

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

class _AskQuestionPageState extends
State<AskQuestionPage> {
  final _formKey = GlobalKey<FormState>();
  final _titleController = TextEditingController();
  final _detailsController =
TextEditingController();

  Future<void> _submitQuestion() async {
        if (_formKey.currentState!.validate()) {
        try {
        await
FirebaseFirestore.instance.collection('questio
n_answers').add({
        'title': _titleController.text,
        'details': _detailsController.text,
        'timestamp':
FieldValue.serverTimestamp(),
        });


ScaffoldMessenger.of(context).showSnackBar
(
        const SnackBar(content:
Text('Question submitted successfully!')),
        );
        widget.onSubmit(_titleController.text,
_detailsController.text);
        Navigator.pop(context);
        } catch (e) {
ScaffoldMessenger.of(context).showSnackBar
(
        SnackBar(content: Text('Failed to
submit question: $e')),
        );
        }
        }
  }

  @override
  Widget build(BuildContext context) {
        return Scaffold(
        backgroundColor: const
Color(0xFFF7EBFB),
        appBar: AppBar(
        title: const Text("Ask Question"),
        backgroundColor: Colors.white,
        ),
        body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Form(
        key: _formKey,
        child: Column(
        children: [
        TextFormField(
        controller: _titleController,
        decoration: const
InputDecoration(labelText: 'Title'),
        validator: (value) => value!.isEmpty ?
'Enter a title' : null,
        ),
        const SizedBox(height: 16),
        TextFormField(
        controller: _detailsController,
        maxLines: 5,
        decoration: const
InputDecoration(labelText: 'Details'),
        validator: (value) => value!.isEmpty ?
'Enter details' : null,
        ),
        const SizedBox(height: 24),
        ElevatedButton(
        onPressed: _submitQuestion,
```

```
        child: const Text("Review Your
Question"),
        ),
        ],
        ),
```

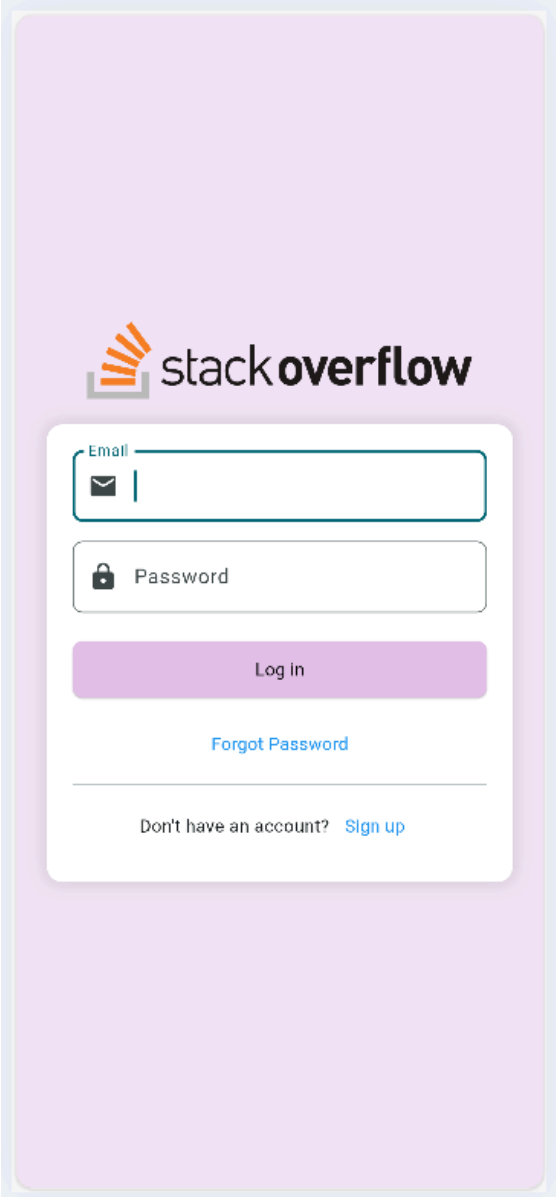
Code:profile_page.dart

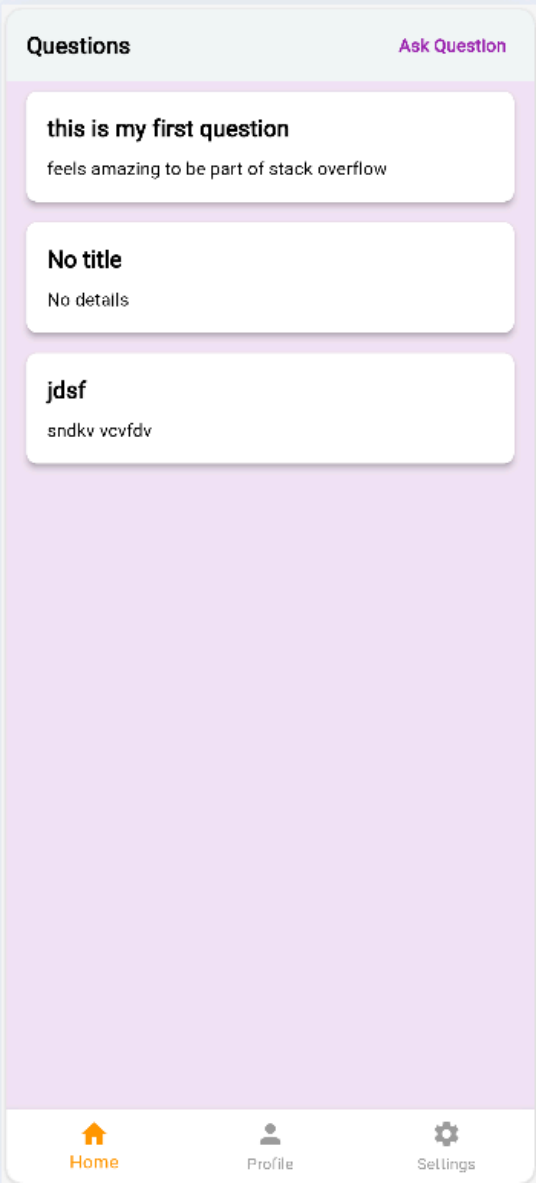
```
import 'package:flutter/material.dart';
import 'package:flutter_svg/flutter_svg.dart';

class ProfilePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
        return Scaffold(
        backgroundColor:
Colors.purple.shade50,
        body: SingleChildScrollView(
        child: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Column(
        crossAxisAlignment:
CrossAxisAlignment.start,
        children: [
        Row(
        children: [
                CircleAvatar(
                radius: 40,
                child:
SvgPicture.asset('project/assets/images/icon.
svg'),
                ),
                const SizedBox(width: 16),
                Column(
                crossAxisAlignment:
CrossAxisAlignment.start,
                children: const [
                Text(
                'Siddhant Sathe',
                style: TextStyle(
                fontSize: 20,
                fontWeight: FontWeight.bold,
                ),
                ),
                SizedBox(height: 4),
                Text(
                '📅 Member since 1st Feb
2025',
                style: TextStyle(
                color: Colors.grey,
                fontSize: 14,
                ),
                ),
                ],
                ),
        ],
        ),
        const SizedBox(height: 20),
        Row(
        mainAxisAlignment:
MainAxisAlignment.spaceBetween,
        children: const [
                Text(
                'About',
                style: TextStyle(
                fontSize: 18,
                fontWeight: FontWeight.bold,
                ),
                ),
                Text(
                'Log out',
                style: TextStyle(
                color: Colors.blue,
                fontWeight: FontWeight.bold,
                ),
                ),
        ],
        ),
        const SizedBox(height: 10),
        Container(
        width: double.infinity,
        padding: const EdgeInsets.all(16),
        decoration: BoxDecoration(
                color: Colors.white,
        ),
        ),
        );
  }
}
```

```
                borderRadius:
BorderRadius.circular(12),
        ),
        child: const Text(
                'No about me has been yet',
                style: TextStyle(
                color: Colors.grey,
                ),
        ),
        ),
        const SizedBox(height: 20),
        Row(
        mainAxisAlignment:
MainAxisAlignment.spaceBetween,
        children: const [
                Text(
                'Answers',
                style: TextStyle(
                fontSize: 18,
                fontWeight: FontWeight.bold,
                ),
                ),
                Text(
                'See All',
                style: TextStyle(
                color: Colors.blue,
                fontWeight: FontWeight.bold,
                ),
                ),
        ],
        ),
        const SizedBox(height: 10),
        Container(
        width: double.infinity,
        padding: const EdgeInsets.all(16),
        decoration: BoxDecoration(
                color: Colors.white,
                borderRadius:
BorderRadius.circular(12),
        ),
        child: const Text(
                'You have not answered any
questions.',
                style: TextStyle(
                color: Colors.grey,
                ),
        ),
        ),
        const SizedBox(height: 20),
        Row(
        mainAxisAlignment:
MainAxisAlignment.spaceBetween,
        children: const [
                Text(
                'Questions',
                style: TextStyle(
                fontSize: 18,
                fontWeight: FontWeight.bold,
                ),
                ),
                Text(
                'See All',
                style: TextStyle(
                color: Colors.blue,
                fontWeight: FontWeight.bold,
                ),
                ),
        ],
        ),
        const SizedBox(height: 10),
        Container(
        width: double.infinity,
        padding: const EdgeInsets.all(16),
        decoration: BoxDecoration(
                color: Colors.white,
                borderRadius:
BorderRadius.circular(12),
        ),
        child: const Text(
                'You have not asked any
questions.',
                style: TextStyle(
                color: Colors.grey,
                ),
        ),
        ),
        ],
        ),
        ),
        ),
        );
 }
```

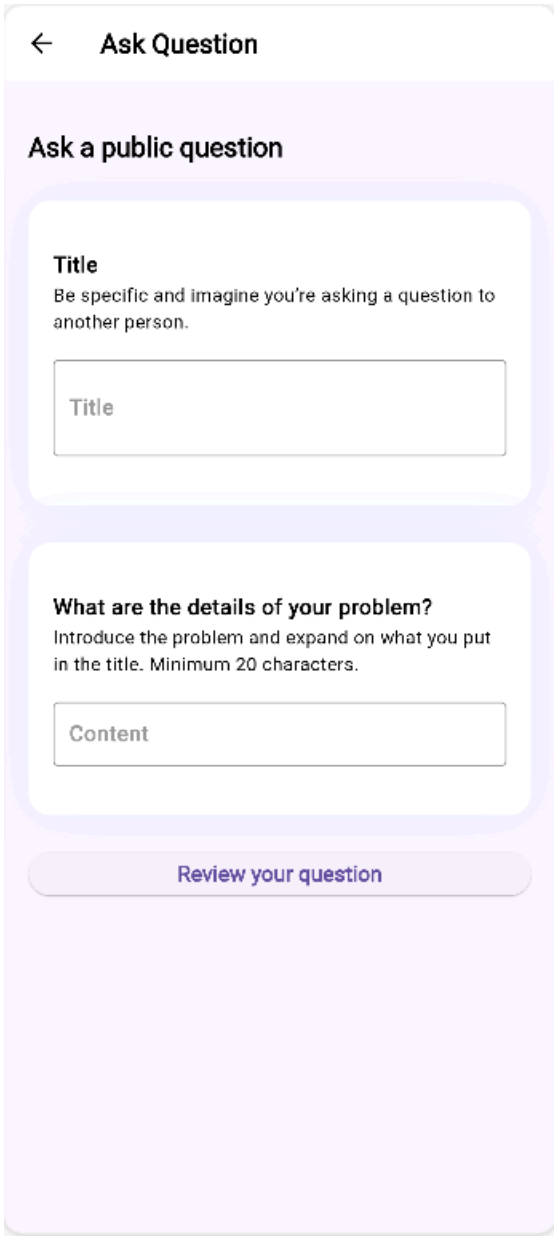
Project Title: Stack Overflow Clone **Roll No. 50**

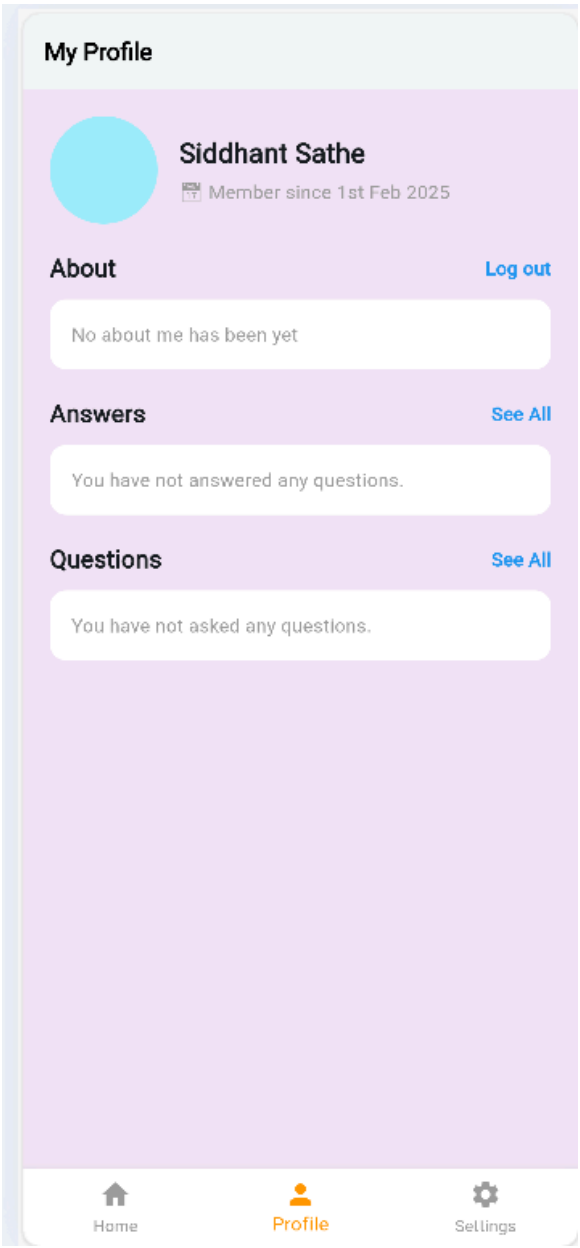
Output:





Project Title: Stack Overflow Clone **Roll No. 50**





MAD & PWA Lab
Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	50
Name	Siddhant Sathe
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	

Theory:

Introduction to Firebase and Flutter Integration

Firebase is a comprehensive platform developed by Google, designed to help developers build high-quality applications for both mobile and web. It provides essential services such as real-time databases, authentication, cloud storage, hosting, and much more.

One of the most widely used Firebase services is the Firebase Realtime Database, which is a NoSQL cloud database that allows data to be stored and synced in real-time across all connected devices. Flutter, on the other hand, is an open-source UI software development kit created by Google, which allows developers to build natively compiled applications for mobile, web, and desktop from a single codebase. Its rich set of pre-designed widgets and powerful tools makes Flutter an attractive option for developing visually appealing and performant applications. Integrating Firebase with Flutter allows developers to leverage the full potential of Firebase services in their applications.

By using Firebase's Realtime Database, Flutter apps can achieve features such as real-time data synchronization, secure authentication, and cloud-based storage. This combination enables developers to create powerful, scalable, and feature-rich mobile and web applications**.**

Firebase Realtime Database Overview

Firebase Realtime Database is a cloud-hosted NoSQL database that stores data in a JSON-like format. The key characteristic of this database is its real-time synchronization feature, meaning that any changes made to the database are instantly reflected on all clients (i.e., devices) connected to it.

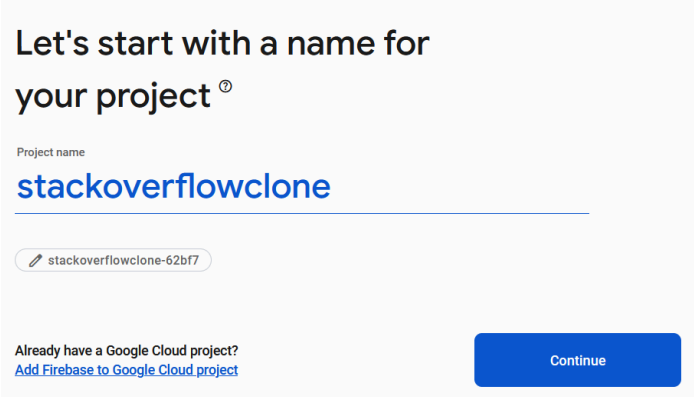
This makes it an ideal solution for applications that require frequent updates and need to maintain synchronized data across multiple users or devices, such as messaging apps, social media platforms, or collaborative tools. The Firebase Realtime Database is structured as a tree of data, where each node in the tree can contain key value pairs.

This structure allows for easy data retrieval and modification. Firebase's real-time capabilities enable apps to immediately receive updates to the data whenever it changes, without the need to refresh or reload the page. Additionally, the database supports offline data persistence, meaning that even if the user's device loses its internet connection, the app can still function by using the locally cached data**.**

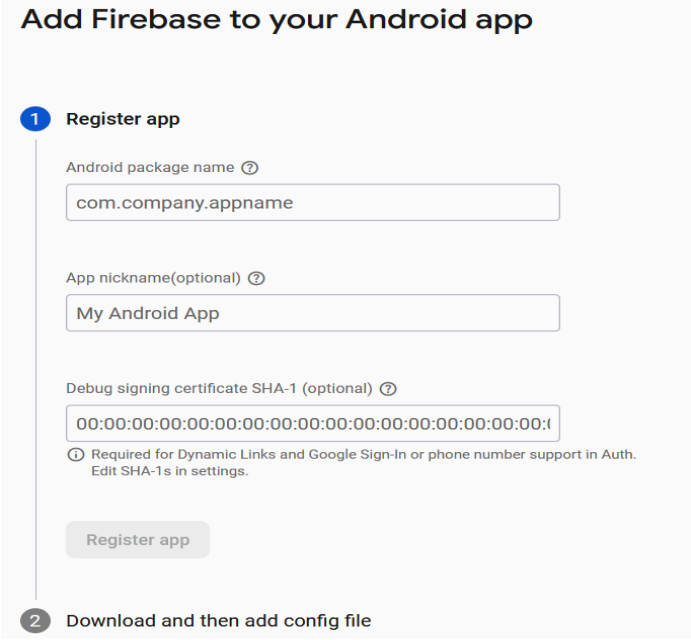
Creating a New Firebase Project

First, log in with your Google account to manage your Firebase projects. From within the

Firebase dashboard, select the Create new project button and give it a name



In order to add Android support to our Flutter application, select the Android logo from the dashboard. This brings us to the following screen:



The most important thing here is to match up the Android package name that you choose here with the one inside of our application.

Downloading the Config File

The next step is to add the Firebase configuration file into our Flutter project. This is important as it contains the API keys and other critical information for Firebase to use.Select Download google-services.json from this page

In android/build.gradle add-

dependencies {

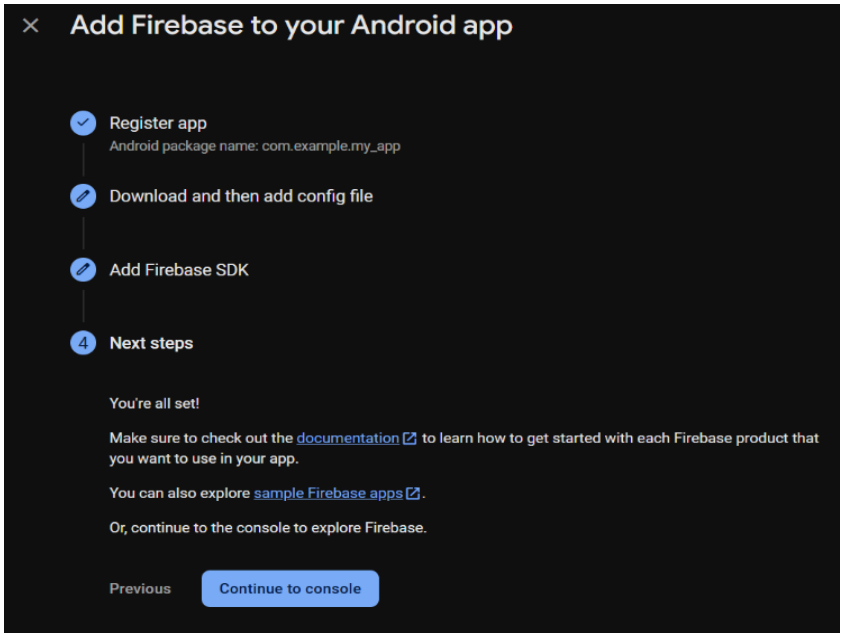
classpath 'com.google.gms.google-services:4.4.2' }

In app/build.gradle add-

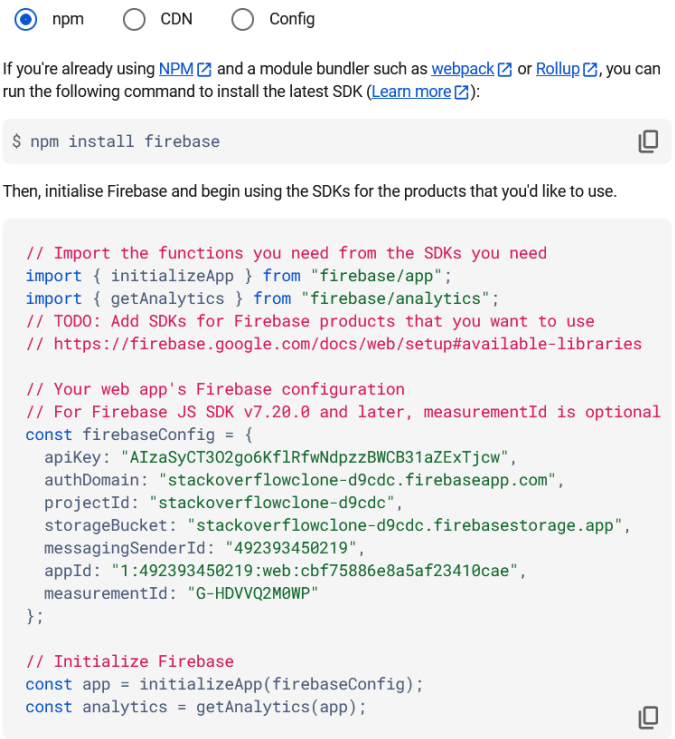
dependencies{

implementation platform('com.google.firebase:firebase-bom:33.9.0')

}



Let's setup web app-



In pubspec.yaml:

dependencies:

flutter:

sdk: flutter

The following adds the Cupertino Icons font to your application.

Use with the CupertinoIcons class for iOS style icons.

cupertino_icons: ^1.0.8

firebase_core: ^3.11.0

firebase_auth: ^5.4.2

cloud_firestore: ^5.6.3

firebase_storage: ^12.4.2

firebase_messaging: ^15.2.2

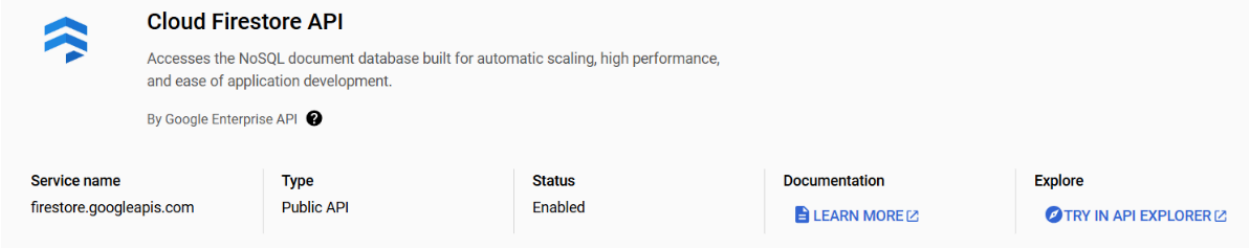
Firebase connection in code:

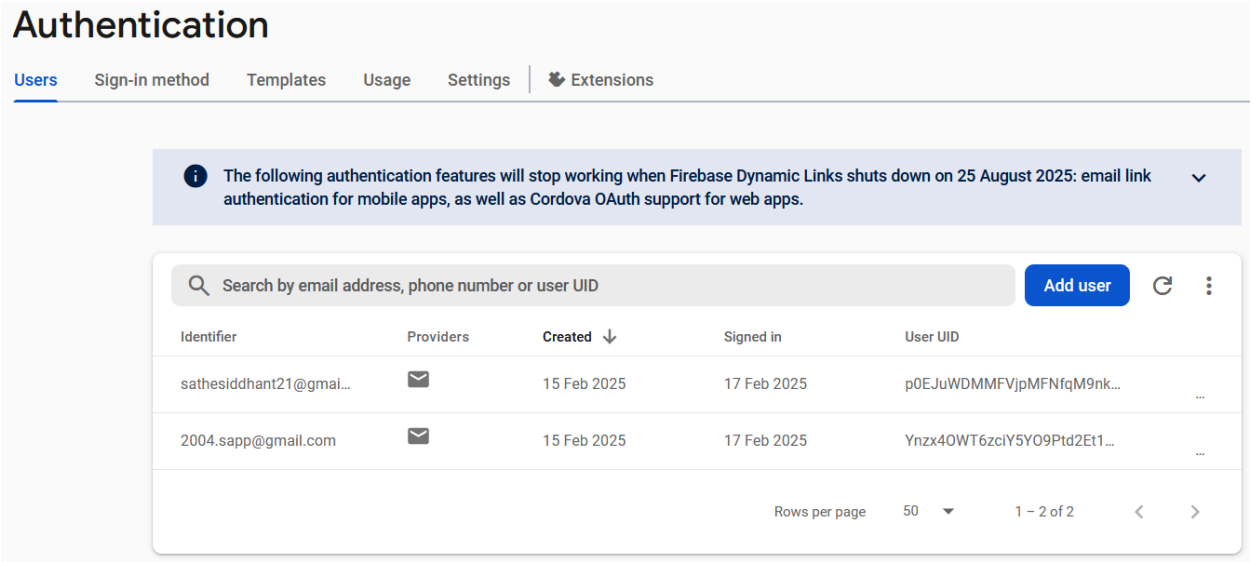
```
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'screens/login_page.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  try {
        await Firebase.initializeApp(
        options: FirebaseOptions(
        apiKey: "AIzaSyCT3O2go6KflRfwNdpzzBWCB31aZExTjcw",
        authDomain: "stackoverflowclone-d9cdc.firebaseapp.com",
        projectId: "stackoverflowclone-d9cdc",
        storageBucket: "stackoverflowclone-d9cdc.appspot.com", // Fixed typo
        messagingSenderId: "492393450219",
        appId: "1:492393450219:web:cbf75886e8a5af23410cae",
        measurementId: "G-HDVVQ2M0WP",
```

```
        databaseURL: "https://stackoverflowclone-d9cdc.firebaseio.com",
        ),
        );
        debugPrint('🔥 Firebase initialized successfully!');
  } catch (e) {
        debugPrint('❌ Firebase initialization failed: $e');
  }

  runApp(const MyApp());
}

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

  @override
  Widget build(BuildContext context) {
        return MaterialApp(
        title: 'Stack Overflow Sign In',
        theme: ThemeData(
        fontFamily: 'Arial',
        colorScheme: ColorScheme.fromSeed(seedColor: const Color(0xFFF2F2F2)),
        ),
        home: LoginPage(),
        debugShowCheckedModeBanner: false,
        );
  }
}
```

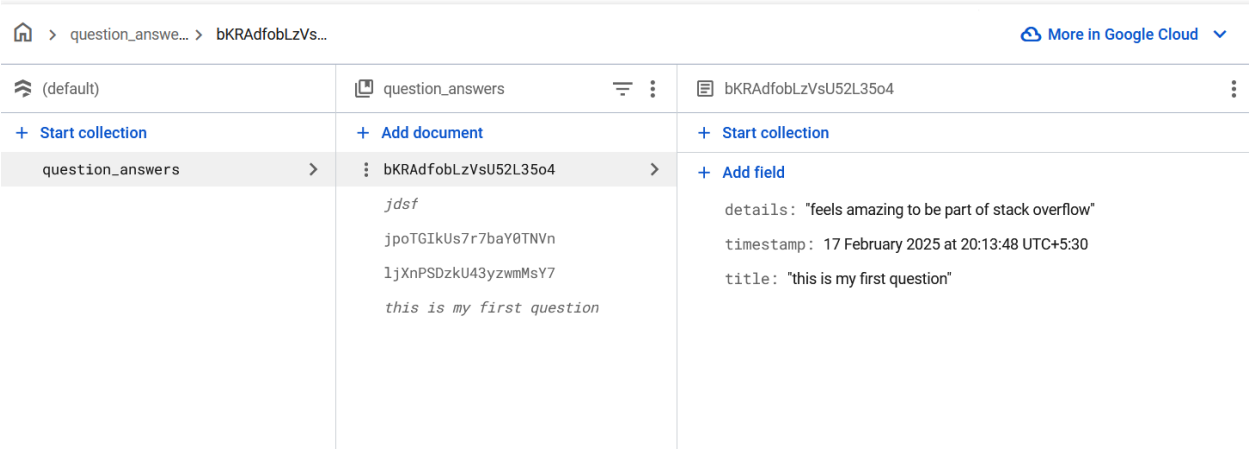
Project Title: Stack Overflow Clone **Roll No. 50**

Enable the firestore API-





Connecting Realtime Database: Firestore



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.	50
Name	Siddhant Sathe
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	

Project Title: Stack Overflow Clone **Roll No. 50**

Theory:-

Regular Web App

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

Progressive Web App

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

Difference between PWAs vs. Regular Web Apps:

A Progressive Web is different and better than a Regular Web app with features like:

1. Native Experience

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

2. Ease of Access

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

3. Faster Services

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

4. Engaging Approach

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

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

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

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

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

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

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

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

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

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

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

Linkable — Easily shared via URL without complex installations.

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

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

Code:-
manifest.json:-

```
{
        "name":"GlobeNews",
        "short_name":"News",
        "start_url":"index.html",
        "display":"standalone",
        "background_color":"#5900b3",
        "theme_color":"black",
        "scope": ".",
        "description":"Global news at one stop.",
        "icons":[
        {
        "src":"assets/newspaperlogo192.png",
        "sizes":"192x192",
        "type":"image/png"
        },
        {
        "src":"assets/newspaperlogo512.png",
        "sizes":"512x512",
        "type":"image/png"
        }
```

]
}

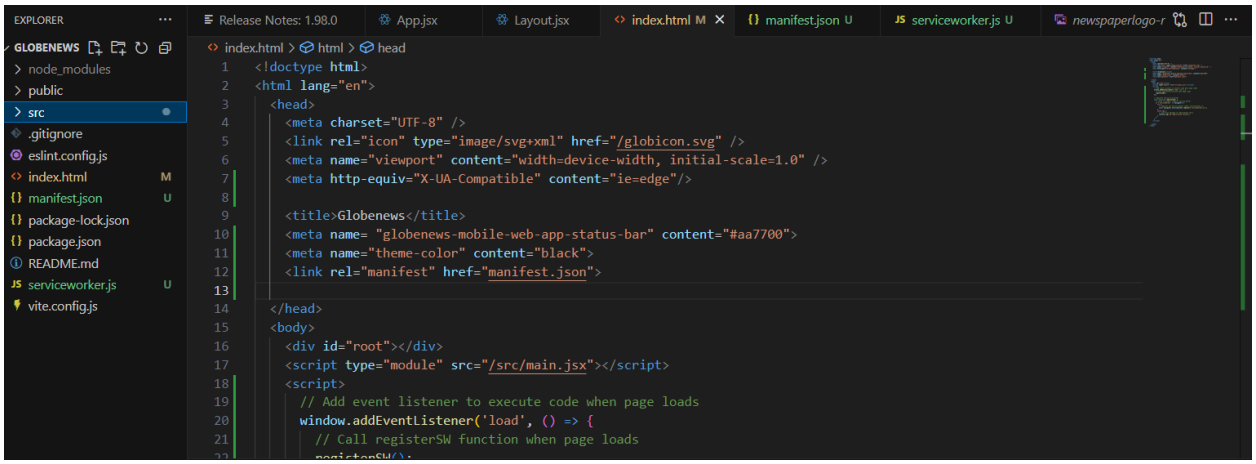
Add the link tag to link to the manifest.json file

```
<html>
   <head>
      <link rel="manifest" href="manifest.json">
      <title>Globenews</title>
      <style>
```

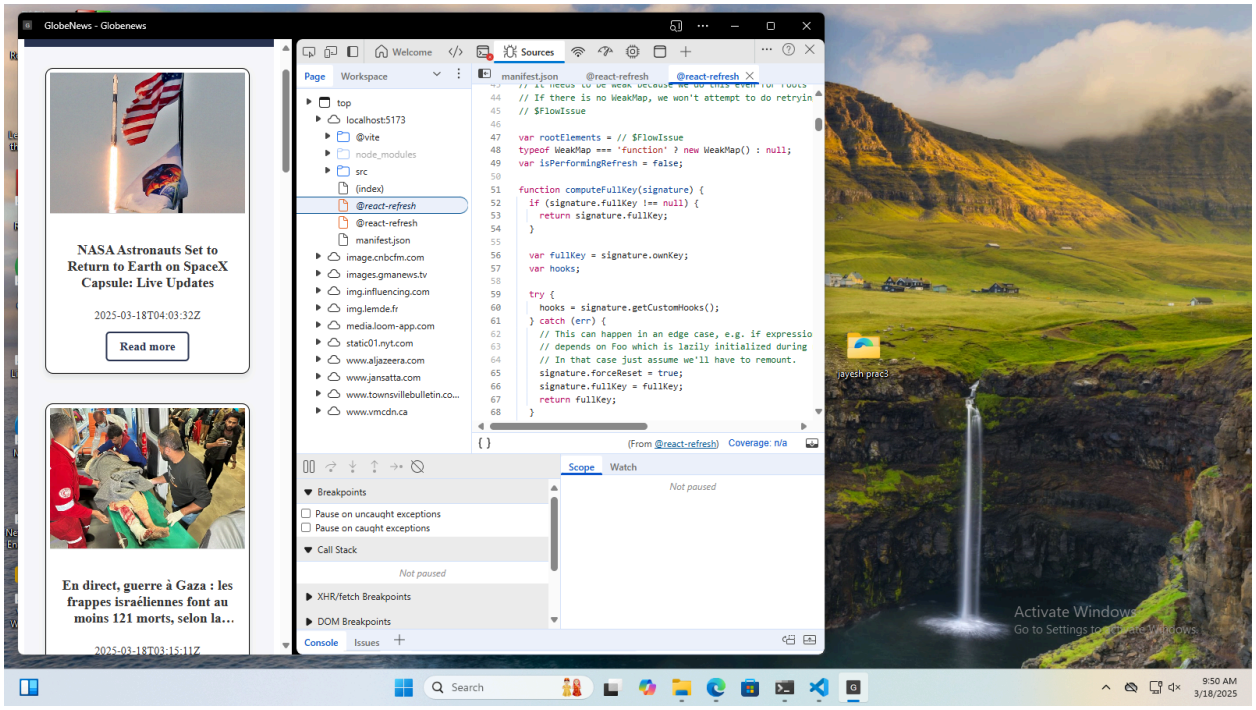
Output:-
Install nodejs
https://nodejs.org/en/download/
In cmd
npm -v
npm install -g browser-sync

```
npm install -g live-server
```

Open vs code
Select proper folder where all files are available
Install extensions node js
In vs code terminal type following commands
node -v
Npx .
Reference https://developer.mozilla.org/en-US/docs/Web/Manifest
https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps



Project Title: Stack Overflow Clone **Roll No. 50**



Conclusion:-
Hence, we learnt how to write a metadata of our E-commerce website PWA in a Web App Manifest File to enable add to homescreen feature.

https://medium.com/@svinkle/start-a-local-live-reload-web-server-with-one-command-72f99bc6e855

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.	50
Name	Siddhant Sathe
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

Theory:

Service Worker

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

Things to note about Service Worker:

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

What can we do with Service Workers?

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

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

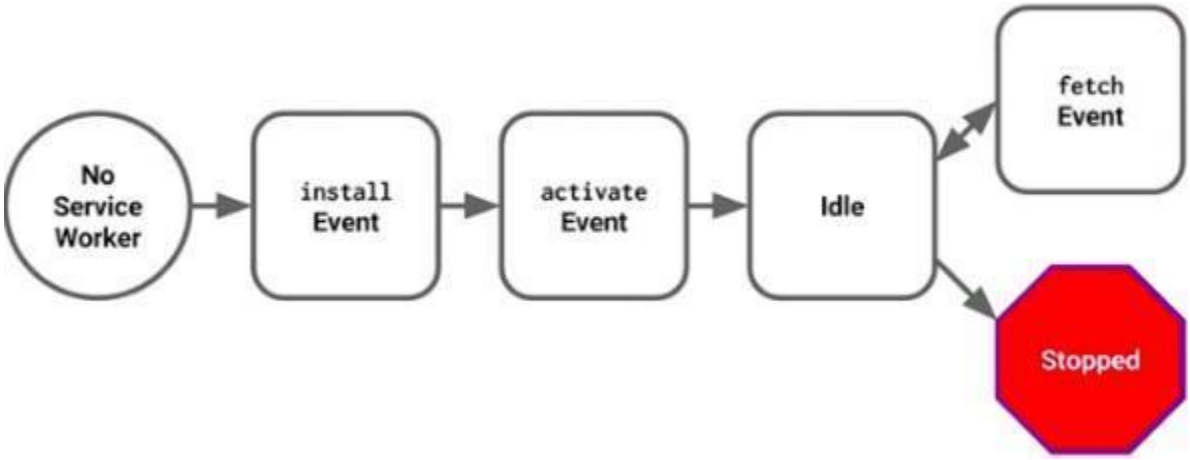
- You can manage **Push Notifications**
  You can manage push notifications with Service Worker and show any information message to the user.
- You can **Continue**
  Although Internet connection is broken, you can start any process with Background Sync of Service Worker.

What can’t we do with Service Workers?

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

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

Service Worker Cycle



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

- Registration
- Installation
- Activation

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

main.js

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

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

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

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

example: main.js

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

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

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

main.js

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

Installation

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

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

service-worker.js

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

Activation

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

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

service-worker.js

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

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

Code:

Index.html

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/globicon.svg" />
    <link rel="manifest" href="manifest.json">
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Globenews</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```

main.jsx

```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import App from './App.jsx'
import './index.css'

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <App />
  </StrictMode>,
)
// Service Worker Registration
if ('serviceWorker' in navigator) {
  window.addEventListener('load', () => {
    navigator.serviceWorker.register('public/sw.js') // Path to your service worker
file
      .then((registration) => {
        console.log('[Service Worker] Registered with scope:', registration.scope);

        // Check if the service worker is active
        if (registration.active) {
```

```
      console.log('[Service Worker] Active');
    }

    // Check if the service worker is installing
    if (registration.installing) {
      console.log('[Service Worker] Installing');
    }

    // Check if the service worker is waiting
    if (registration.waiting) {
      console.log('[Service Worker] Waiting');
    }

    // Listen for state changes
    registration.addEventListener('updatefound', () => {
      console.log('[Service Worker] Update found');
      const installingWorker = registration.installing;
      if (installingWorker) {
        installingWorker.addEventListener('statechange', () => {
          if (installingWorker.state === 'installed') {
            if (navigator.serviceWorker.controller) {
              console.log('[Service Worker] New content is available and will be used when all tabs for this
page are closed');
              // You can prompt the user to reload here if needed
            } else {
              console.log('[Service Worker] Content is cached for offline use.');
            }
          }
        });
      }
    });
  })
  .catch((error) => {
    console.error('[Service Worker] Registration failed:', error);
  });
 });
} else {
  console.warn('[Service Worker] Not supported in this browser.');
}
```

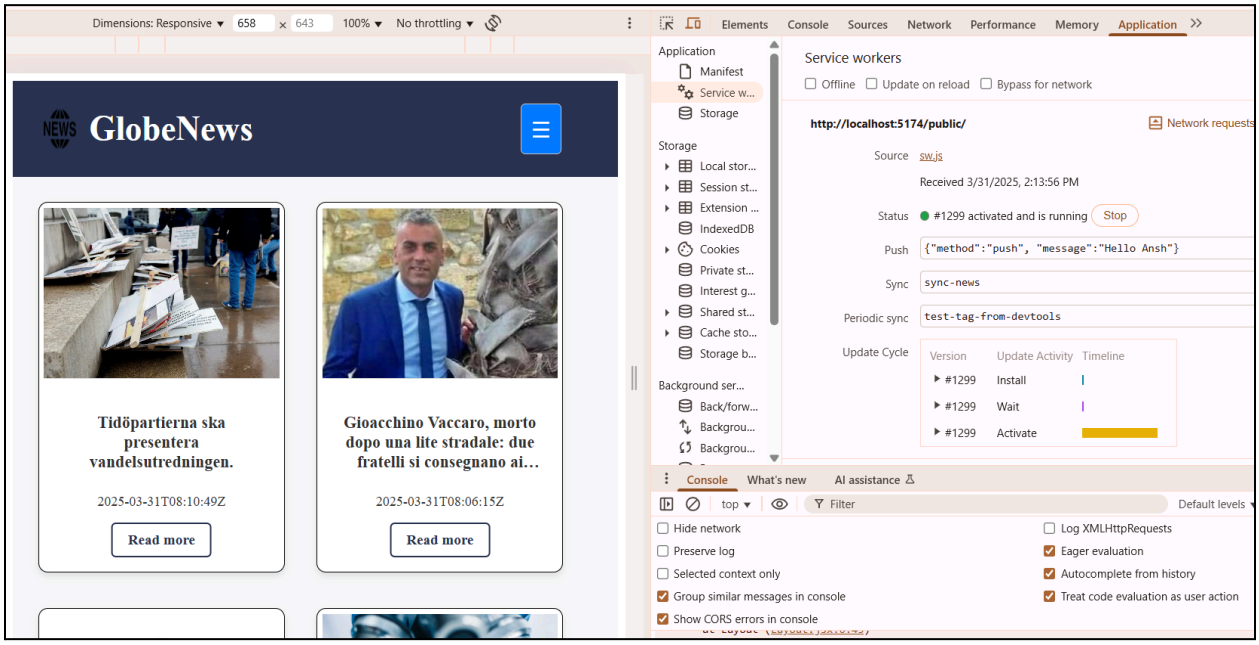
sw.js

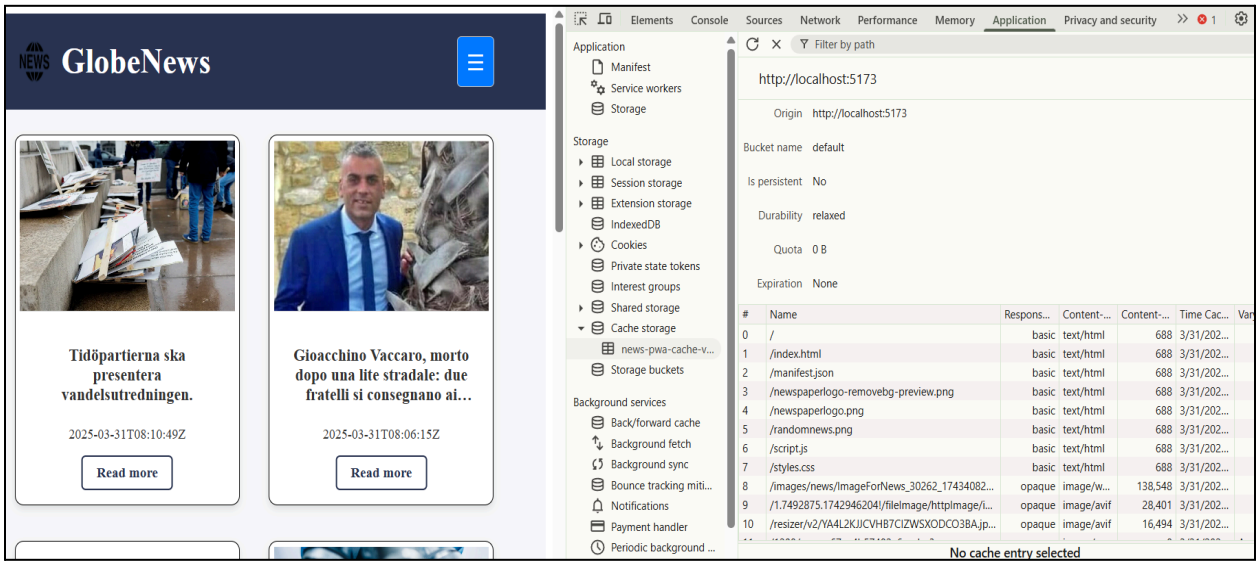
```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import App from './App.jsx'
import './index.css'

createRoot(document.getElementById('root')).render(
```

```
  <StrictMode>
    <App />
  </StrictMode>,
)
// Service Worker Registration
if ('serviceWorker' in navigator) {
  window.addEventListener('load', () => {
    navigator.serviceWorker.register('public/sw.js') // Path to your service worker
file
      .then((registration) => {
        console.log('[Service Worker] Registered with scope:', registration.scope);
        // Check if the service worker is active
        if (registration.active) {
          console.log('[Service Worker] Active');
        }
        // Check if the service worker is installing
        if (registration.installing) {
          console.log('[Service Worker] Installing');
        }
        // Check if the service worker is waiting
        if (registration.waiting) {
          console.log('[Service Worker] Waiting');
        }
        // Listen for state changes
        registration.addEventListener('updatefound', () => {
          console.log('[Service Worker] Update found');
          const installingWorker = registration.installing;
          if (installingWorker) {
            installingWorker.addEventListener('statechange', () => {
              if (installingWorker.state === 'installed') {
                if (navigator.serviceWorker.controller) {
                  console.log('[Service Worker] New content is available and will be used when all tabs for this
page are closed');
                  // You can prompt the user to reload here if needed
                } else {
                  console.log('[Service Worker] Content is cached for offline use.');
                }
              }
            });
          }
        });
      })
      .catch((error) => {
        console.error('[Service Worker] Registration failed:', error);
      });
  });
} else {
  console.warn('[Service Worker] Not supported in this browser.');
```

```
}
```

Output:


GlobeNews
Tidöpartierna ska presentera vandelsutredningen.
2025-03-31T08:10:49Z
Read more
Gioacchino Vaccaro, morto dopo una lite stradale: due fratelli si consegnano ai…
2025-03-31T08:06:15Z
Read more
Elements Console Sources Network Performance Memory Application
Service workers
Offline Update on reload Bypass for network
http://localhost:5174/public/
Source sw.js
Received 3/31/2025, 2:13:56 PM
Status #1299 activated and is running Stop
Push {"method":"push", "message":"Hello Ansh"}
Sync sync-news
Periodic sync test-tag-from-devtools
Update Cycle
Version Update Activity Timeline
#1299 Install
#1299 Wait
#1299 Activate
Console What's new AI assistance
Hide network
Preserve log
Selected context only
Group similar messages in console
Show CORS errors in console
Log XMLHttpRequests
Eager evaluation
Autocomplete from history
Treat code evaluation as user action


GlobeNews
Tidöpartierna ska presentera vandelsutredningen.
2025-03-31T08:10:49Z
Read more
Gioacchino Vaccaro, morto dopo una lite stradale: due fratelli si consegnano ai…
2025-03-31T08:06:15Z
Read more
Elements Console Sources Network Performance Memory Application Privacy and security
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
news-pwa-cache-v...
Storage buckets
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking miti...
Notifications
Payment handler
Periodic background ...
Filter by path
http://localhost:5173
Origin http://localhost:5173
Bucket name default
Is persistent No
Durability relaxed
Quota 0 B
Expiration None
Name Respons... Content-... Content-... Time Cac...
0 / basic text/html 688 3/31/202...
1 /index.html basic text/html 688 3/31/202...
2 /manifest.json basic text/html 688 3/31/202...
3 /newspaperlogo-removebg-preview.png basic text/html 688 3/31/202...
4 /newspaperlogo.png basic text/html 688 3/31/202...
5 /randomnews.png basic text/html 688 3/31/202...
6 /script.js basic text/html 688 3/31/202...
7 /styles.css basic text/html 688 3/31/202...
8 /images/news/ImageForNews_30262_17434082... opaque image/w... 138,548 3/31/202...
9 /1.7492875.1742946204!/fileImage/httpImage/i... opaque image/avif 28,401 3/31/202...
10 /resizer/v2/YA4L2KJJCVHB7CIZWSXODCO3BA.jp... opaque image/avif 16,494 3/31/202...
No cache entry selected

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.	50
Name	Siddhant Sathe
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

Theory:

Service Worker

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

Things to note about Service Worker:

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

Fetch Event

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

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

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

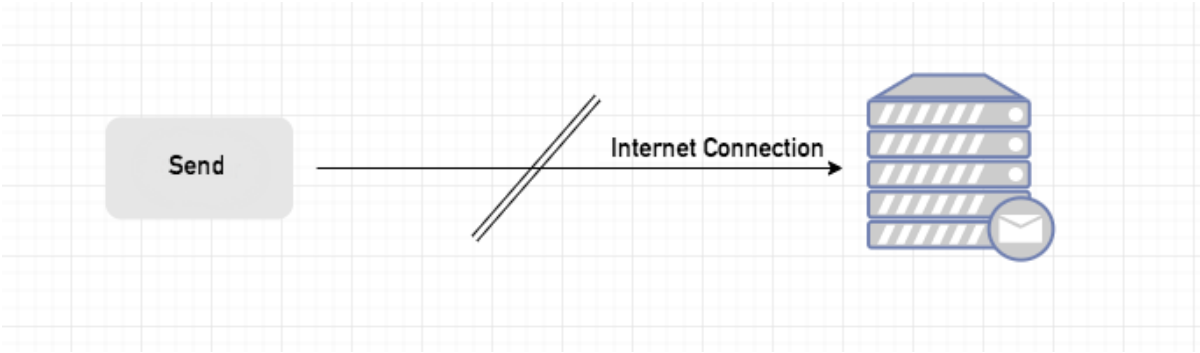
```
self.addEventListener("fetch", function (event) {
    const req = event.request;
    const url = new URL(req.url);

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

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

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

Sync Event

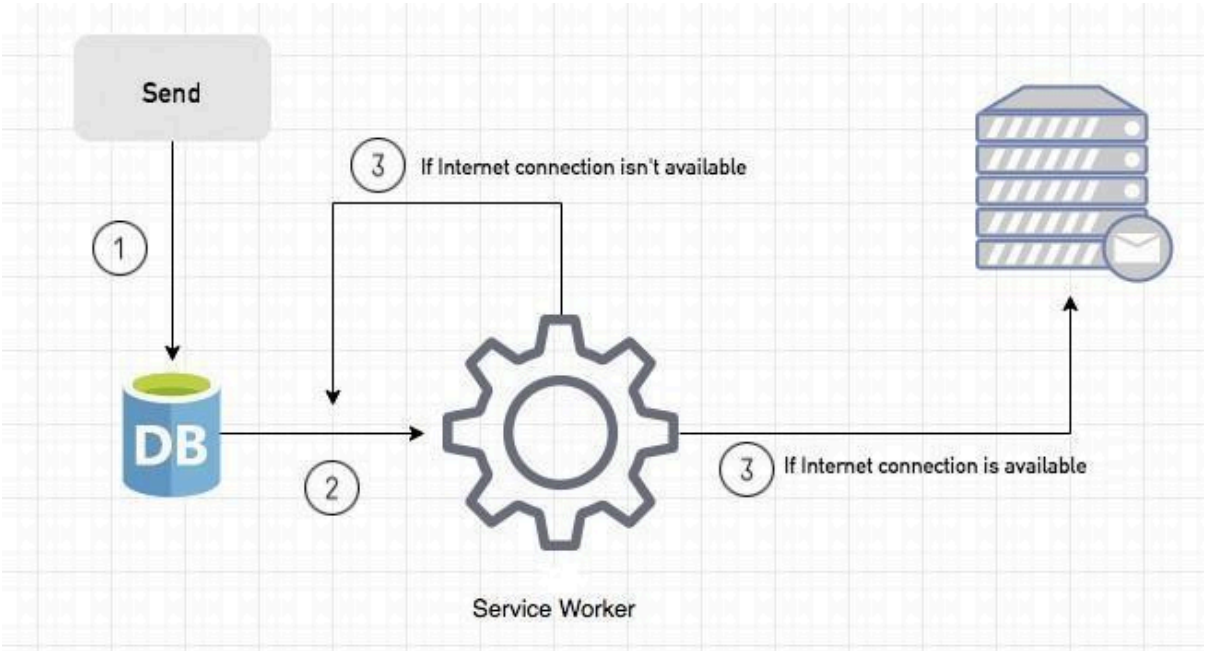
Background Sync is a Web API that is used to delay a process until the Internet connection is stable. We can adapt this definition to the real world; there is an e-mail client application that works on the browser and we want to send an email with this tool. Internet connection is broken while we are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

Here is a job for the Background Sync.

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



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



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

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

Event Listener for Background Sync Registration

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

Event Listener for sw.js

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

Push Event

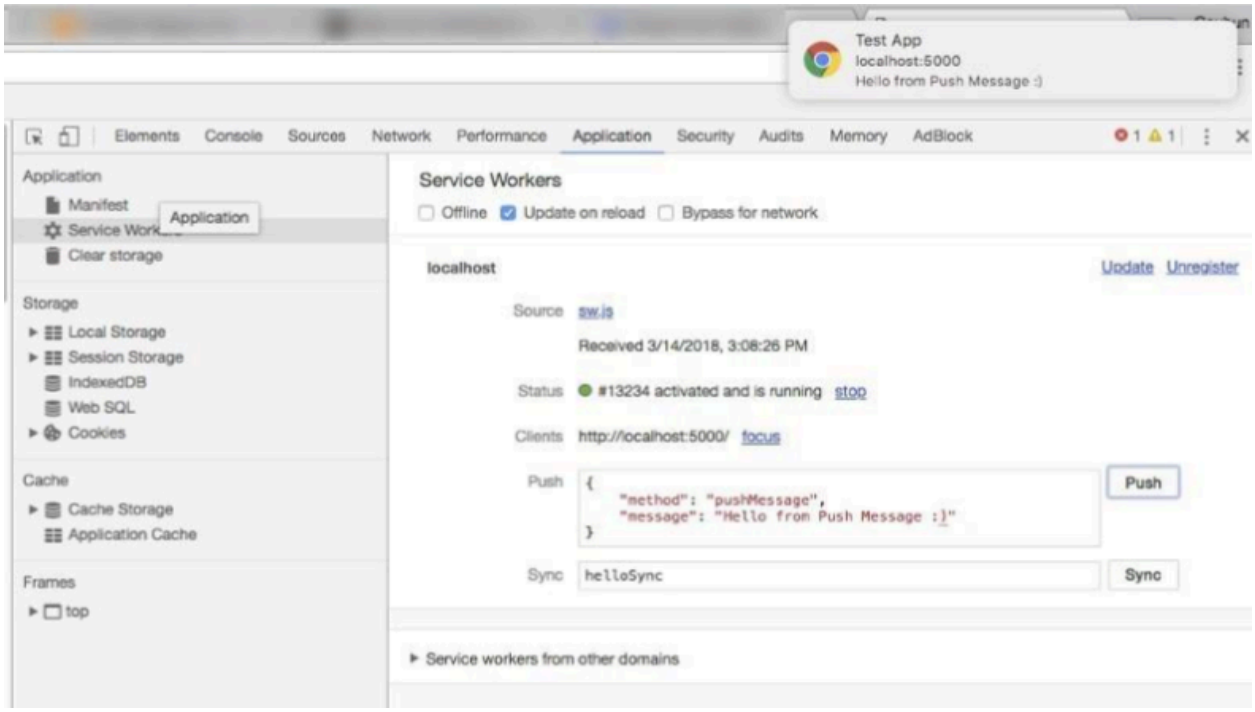
This is the event that handles push notifications that are received from the server. You can apply any method with received data.

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

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

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

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



Code:

service-worker.js const CACHE_NAME = 'news-pwa-cache-v1';
const urlsToCache = [
'/',
'/index.html',
'/styles.css',
'/script.js',
'/manifest.json',
'/newspaperlogo.png',
'/randomnews.png',
'/newspaperlogo-removebg-preview.png'
];

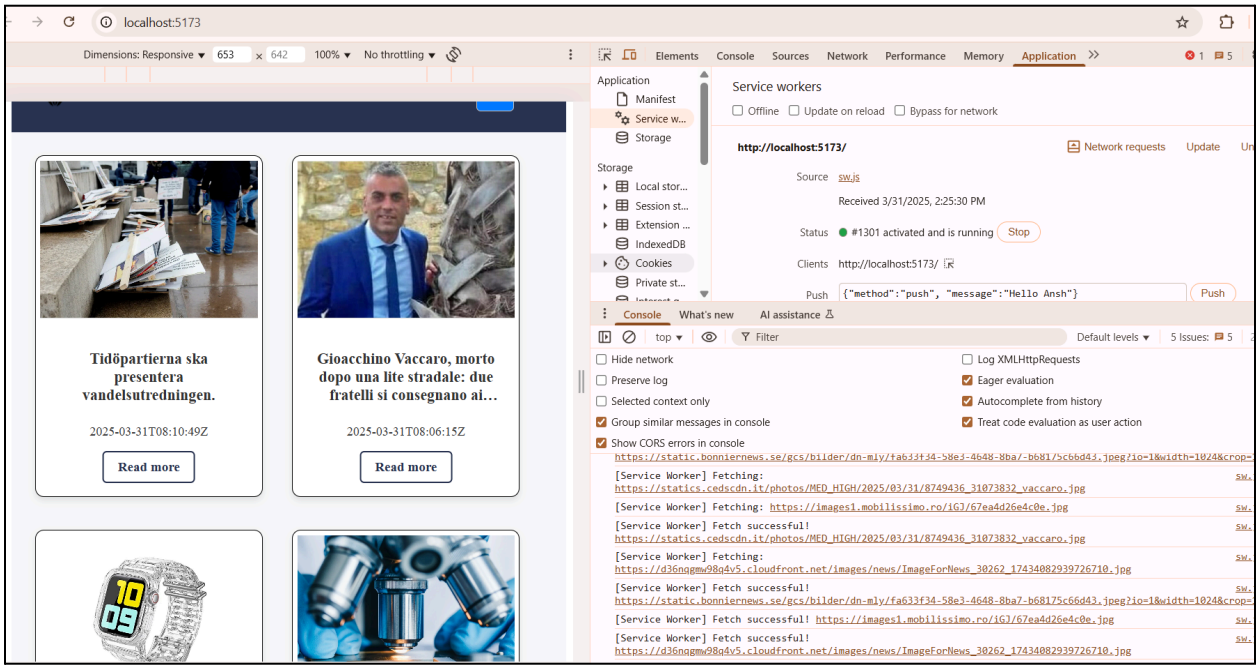
```
// Install event: Cache static assets
self.addEventListener('install', (event) => {
  console.log('[Service Worker] Installing...');
  event.waitUntil(
    caches.open(CACHE_NAME).then((cache) => {
      console.log('[Service Worker] Caching assets...');
      return cache.addAll(urlsToCache);
    })
  );
  self.skipWaiting();
});

// Activate event: Cleanup old caches
self.addEventListener('activate', (event) => {
  console.log('[Service Worker] Activating...');
```

```
  event.waitUntil(
    caches.keys().then((cacheNames) => {
      return Promise.all(
        cacheNames.map((cacheName) => {
          if (cacheName !== CACHE_NAME) {
            console.log('[Service Worker] Deleting old cache:', cacheName);
            return caches.delete(cacheName);
          }
        })
      );
    })
  );
  self.clients.claim();
});
// Fetch event: Cache-first strategy
// Fetch event: Cache-first strategy with logging
self.addEventListener('fetch', (event) => {
    console.log('[Service Worker] Fetching:', event.request.url);

    event.respondWith(
      caches.match(event.request).then((cachedResponse) => {
        if (cachedResponse) {
          console.log('[Service Worker] Serving from cache:', event.request.url);
          return cachedResponse;
        }

        return fetch(event.request)
          .then((networkResponse) => {
            console.log('[Service Worker] Fetch successful!', event.request.url);
            return caches.open(CACHE_NAME).then((cache) => {
              cache.put(event.request, networkResponse.clone());
              return networkResponse;
            });
          })
          .catch((error) => {
            console.error('[Service Worker] Fetch failed:', error);
            return new Response('Network error!', { status: 408 });
          });
      })
    );
  });

// Handle Push Notifications
self.addEventListener('push', (event) => {
    if (!event.data) {
      console.error('[Service Worker] Push event has NO data!');
      return;
    }

    const data = event.data.json();
    console.log('[Service Worker] Push received:', data); // Log full push data
```

```
    console.log('[Service Worker] Notification Message:', data.message); // Log message only

    const options = {
      body: data.message || 'Breaking News!',
      icon: '/newspaperlogo.png',
      badge: '/newspaperlogo-removebg-preview.png'
    };

    event.waitUntil(
      self.registration.showNotification(data.title || 'News Alert', options)
    );
  });
// Handle Notification Click
self.addEventListener('notificationclick', (event) => {
  event.notification.close();
  event.waitUntil(clients.openWindow('/'));
});
// Background Sync
self.addEventListener('sync', (event) => {
  if (event.tag === 'syncNews') {
    console.log('[Service Worker] Sync successful!');
  }
});
```

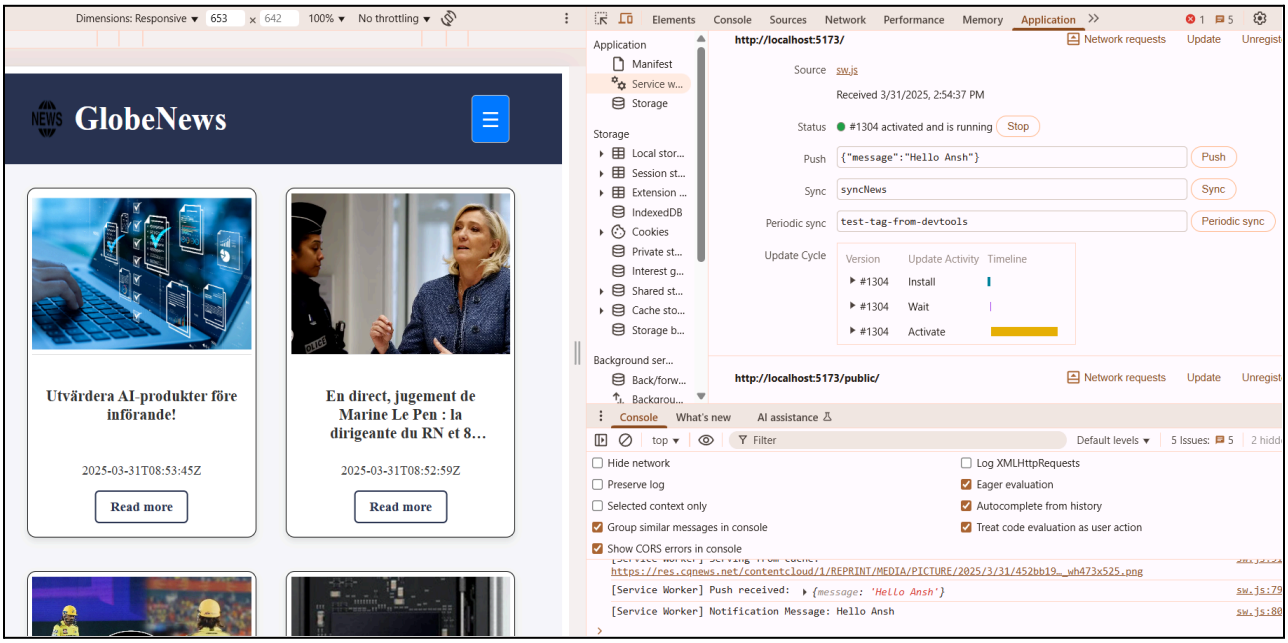
Output:

Fetch event

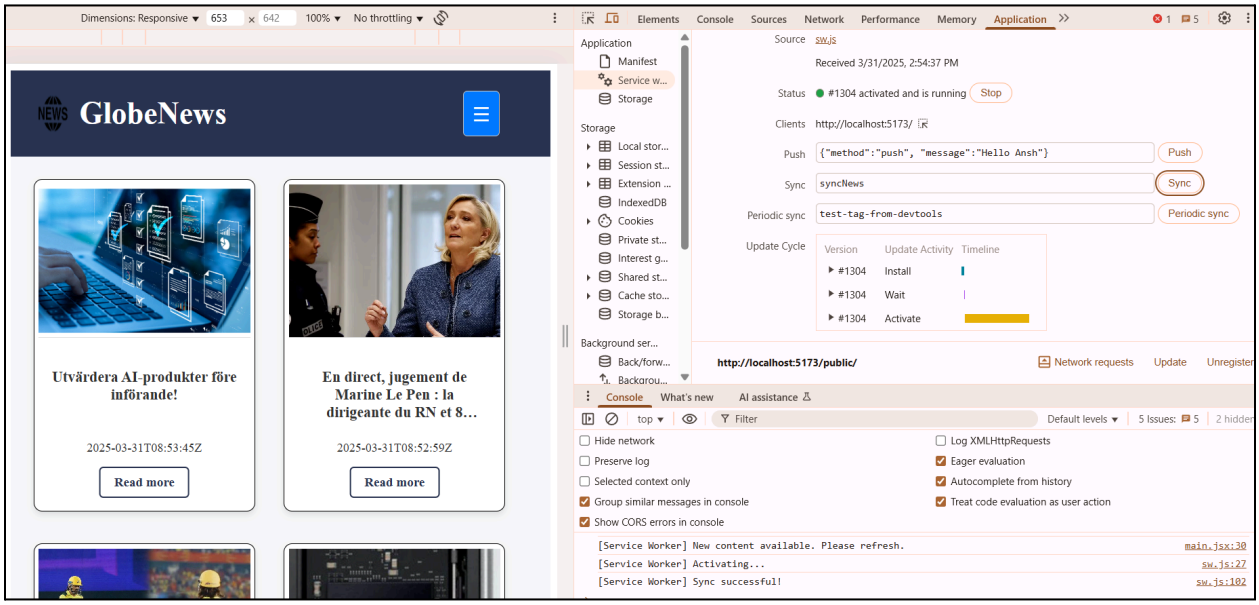


Project Title: Stack Overflow Clone **Roll No. 50**

Push event



Sync event



MAD & PWA Lab
Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	50
Name	Siddhant Sathe
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

Theory:

GitHub Pages

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

GitHub Pages provides the following key features:

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

Reasons for favoring this over Firebase:

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

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

Pros

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

Cons

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

Firebase

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

Some of the features offered by Firebase are:

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

Reasons for favoring over GitHub Pages:

1. Realtime backend made easy
2. Fast and responsive

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

Pros

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

Cons

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

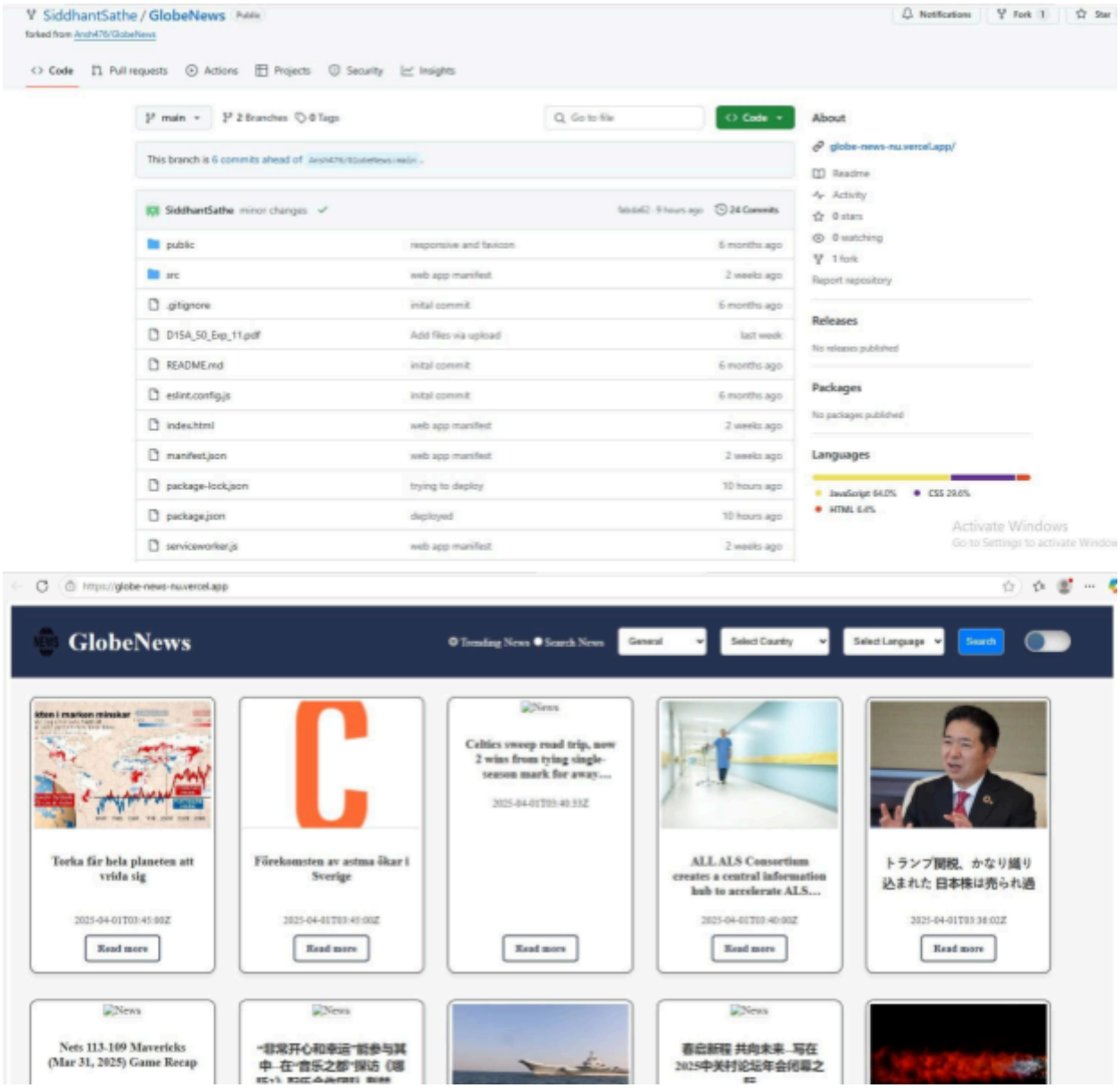
Link to our GitHub repository:

Github List: [SiddhantSathe/Portfolio_bootstrap]

Github Screenshot:

Github Hosted Website: [Siddhant Sathe]

PWA Website:



Github Website: [GitHub - SiddhantSathe/GlobeNews]
Hosted Website: [globe-news-nu.vercel.app/]

MAD & PWA Lab
Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	50
Name	Siddhant Sathe
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	

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:

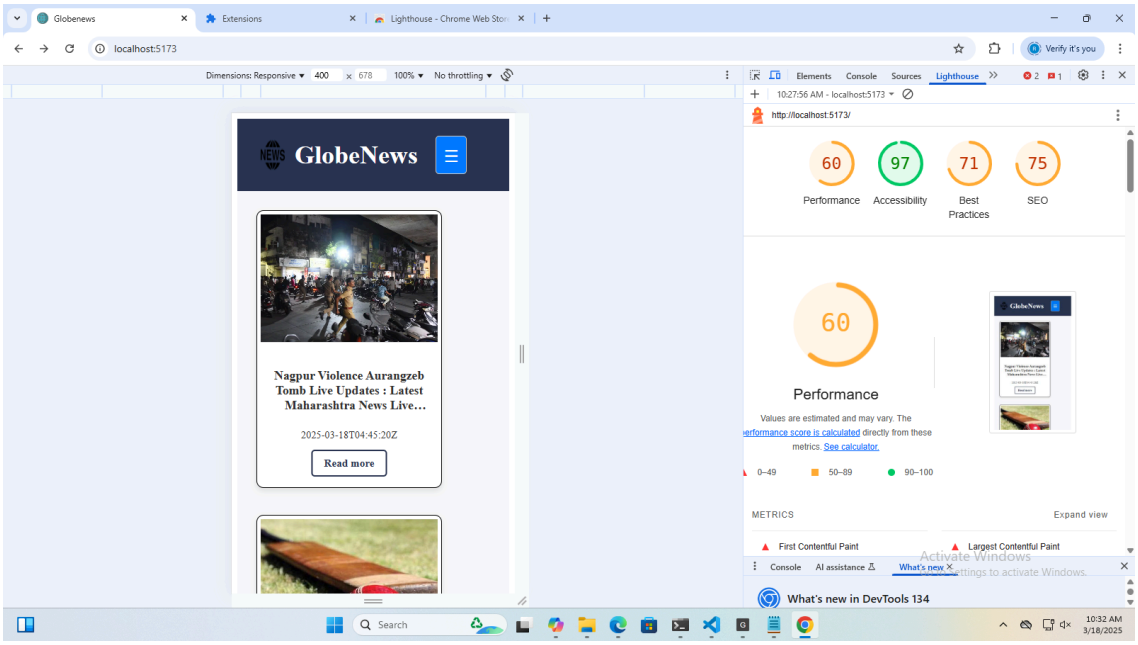
Performance: This score is an aggregation of how the page fared in aspects such as (but not limited to) loading speed, time taken for loading for basic frame(s), displaying meaningful content to the user, etc. To a layman, this score is indicative of how decently the site performs, with a score of 100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

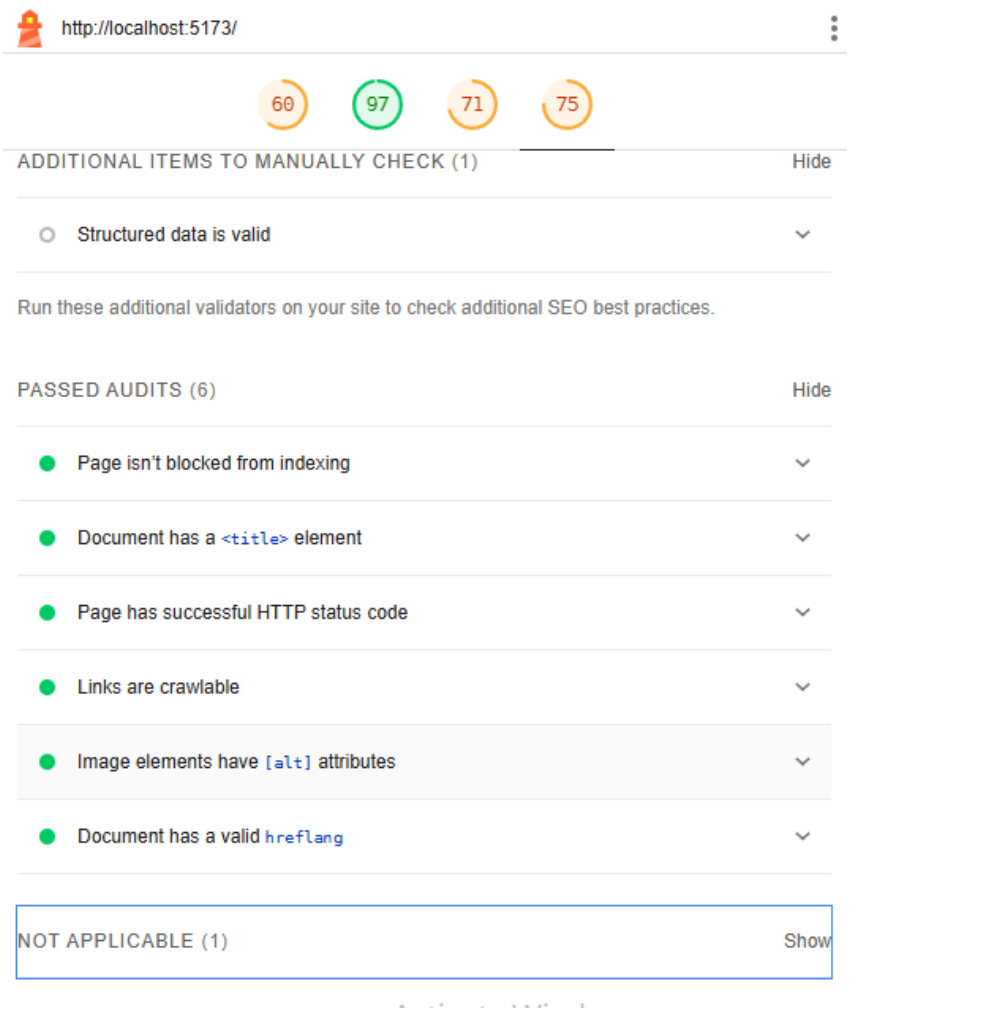
PWA Score (Mobile): Thanks to the rise of Service Workers, app manifests, etc., a lot of modern web applications are moving towards the PWA paradigm, where the objective is to make the application behave as close as possible to native mobile applications. Scoring points are based on the Baseline PWA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

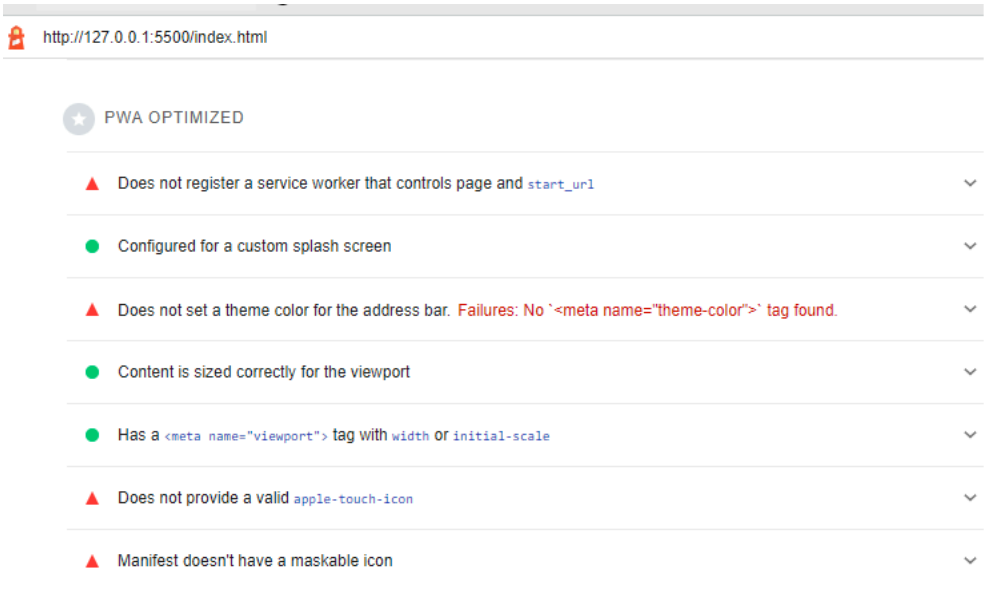
Accessibility: As you might have guessed, this metric is a measure of how accessible your website is, across a plethora of accessibility features that can be implemented in your page (such as the 'aria-' attributes like aria-required, audio captions, button names, etc.). Unlike the other metrics though, 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.

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.

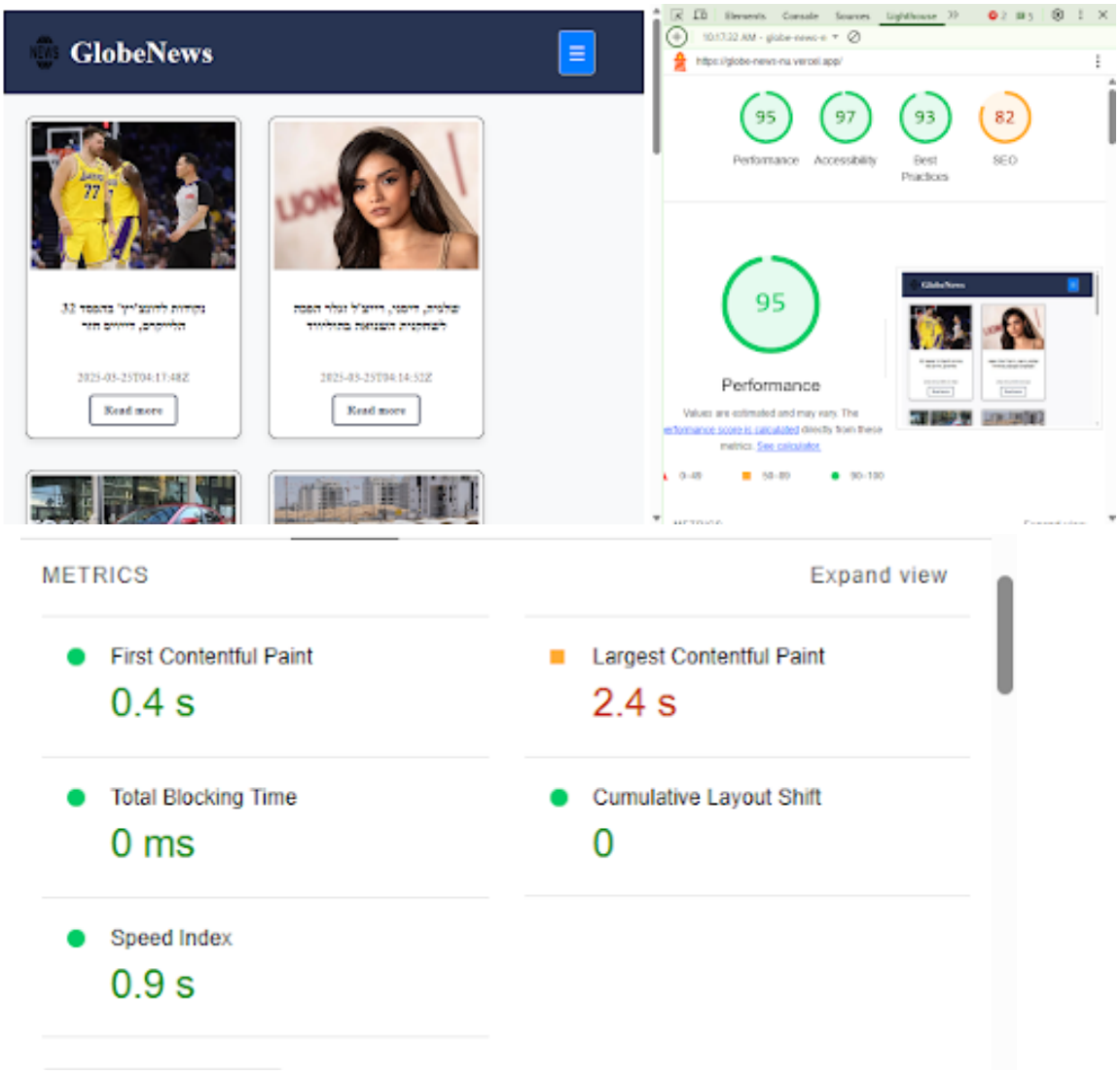
Before :




http://localhost:5173/
60
97
71
75
ADDITIONAL ITEMS TO MANUALLY CHECK (1)
Hide
Structured data is valid
Run these additional validators on your site to check additional SEO best practices.
PASSED AUDITS (6)
Hide
Page isn't blocked from indexing
Document has a <title> element
Page has successful HTTP status code
Links are crawlable
Image elements have [alt] attributes
Document has a valid hreflang
NOT APPLICABLE (1)
Show


http://127.0.0.1:5500/index.html
PWA OPTIMIZED
Does not register a service worker that controls page and start_url
Configured for a custom splash screen
Does not set a theme color for the address bar. Failures: No `<meta name="theme-color">` tag found.
Content is sized correctly for the viewport
Has a <meta name="viewport"> tag with width or initial-scale
Does not provide a valid apple-touch-icon
Manifest doesn't have a maskable icon

After:



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