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 **Anushka Nilesh Shahane** 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: **Roll No.**

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

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

Faculty Incharge : Mrs. Kajal Joseph.

Lab Teachers : Mrs. Kajal Joseph.

Email : kajal.jewani@ves.ac.in

Programme Outcomes: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

Program specific Outcomes

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

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

Project Title: **Roll No.**

Lab Objectives:

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

Lab Outcomes:

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

Project Title: **Roll No.**

Index

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

MAD & PWA Lab

Journal

Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	54
Name	Anushka Shahane
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	

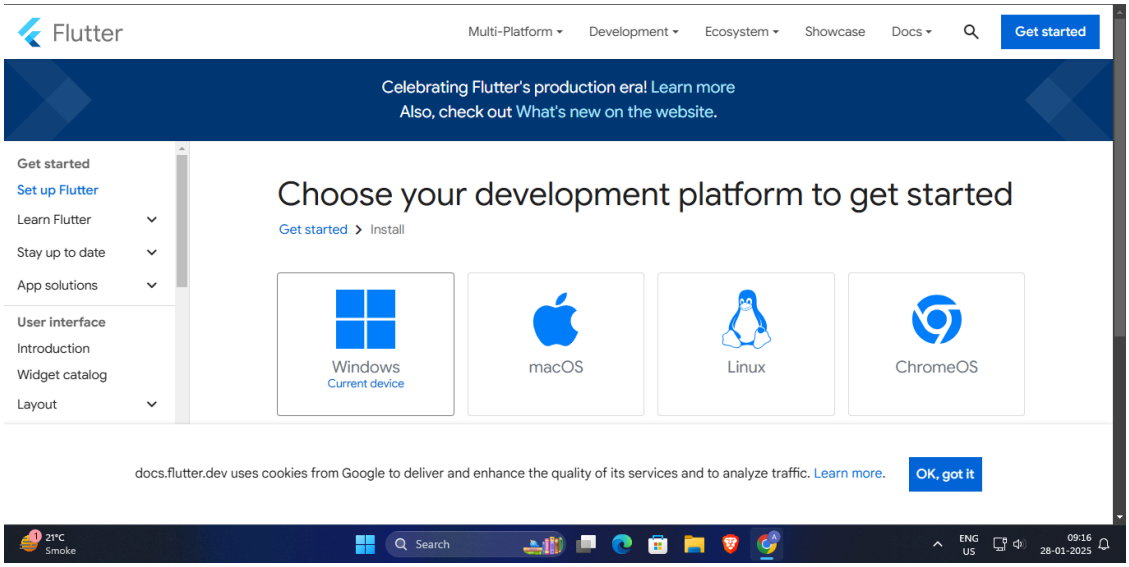
Anushka Shahane D15A 54

EXPERIMENT 1

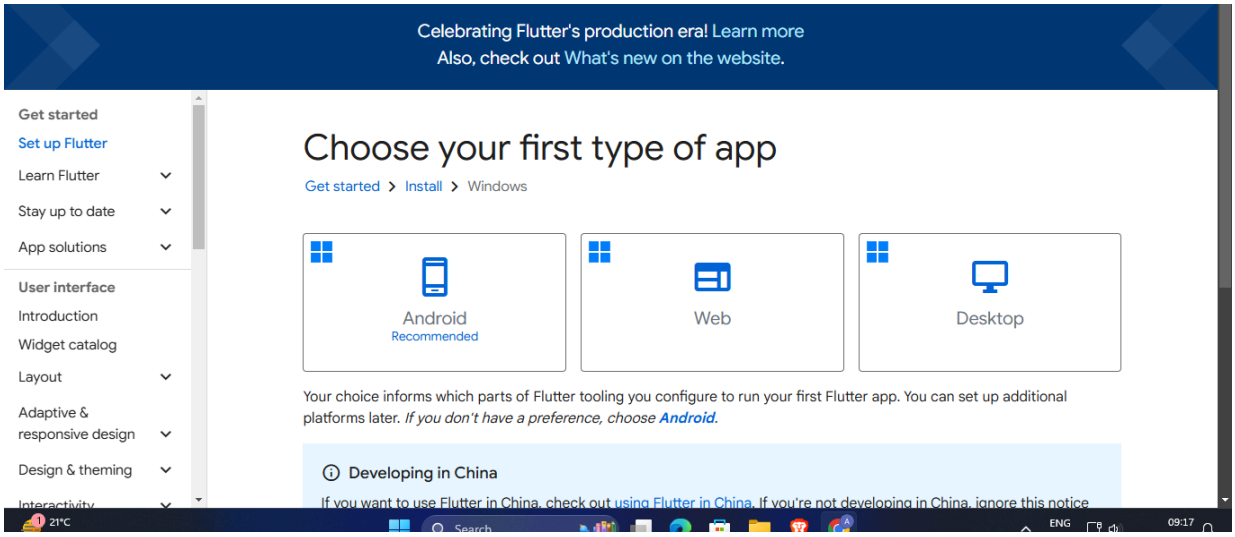
Aim : Installation and Configuration of Flutter Environment.

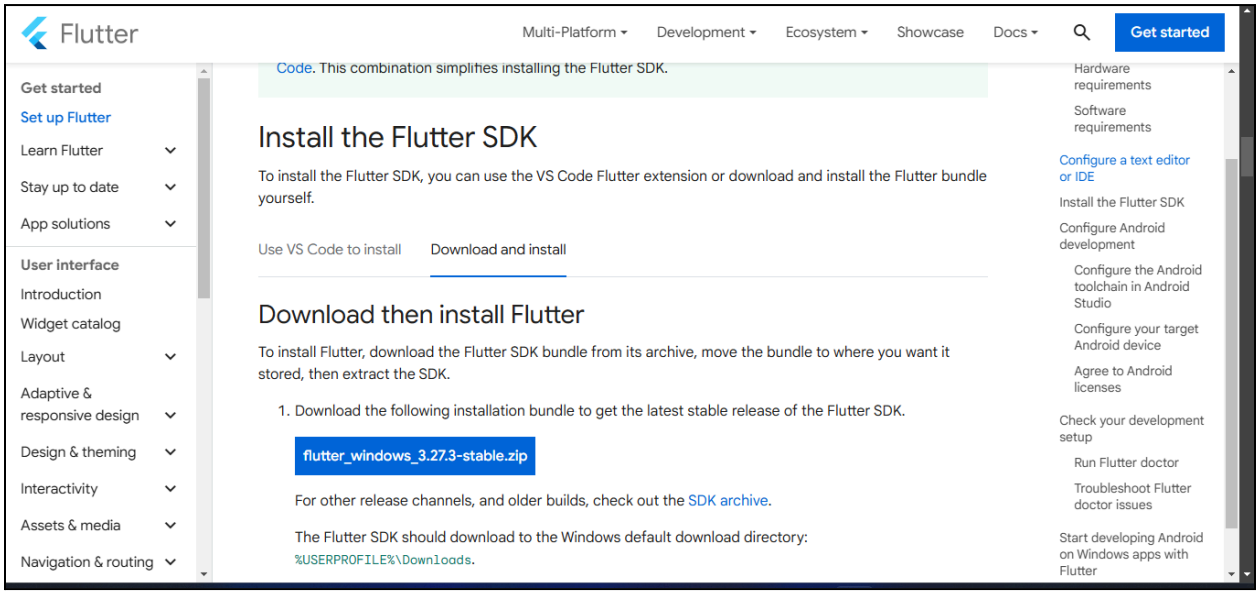
Steps :

Step 1: Download the installation bundle of the Flutter Software Development Kit for windows. To download Flutter SDK, Go to its official website https://docs.flutter.dev/get-started/install , you will get the following screen.



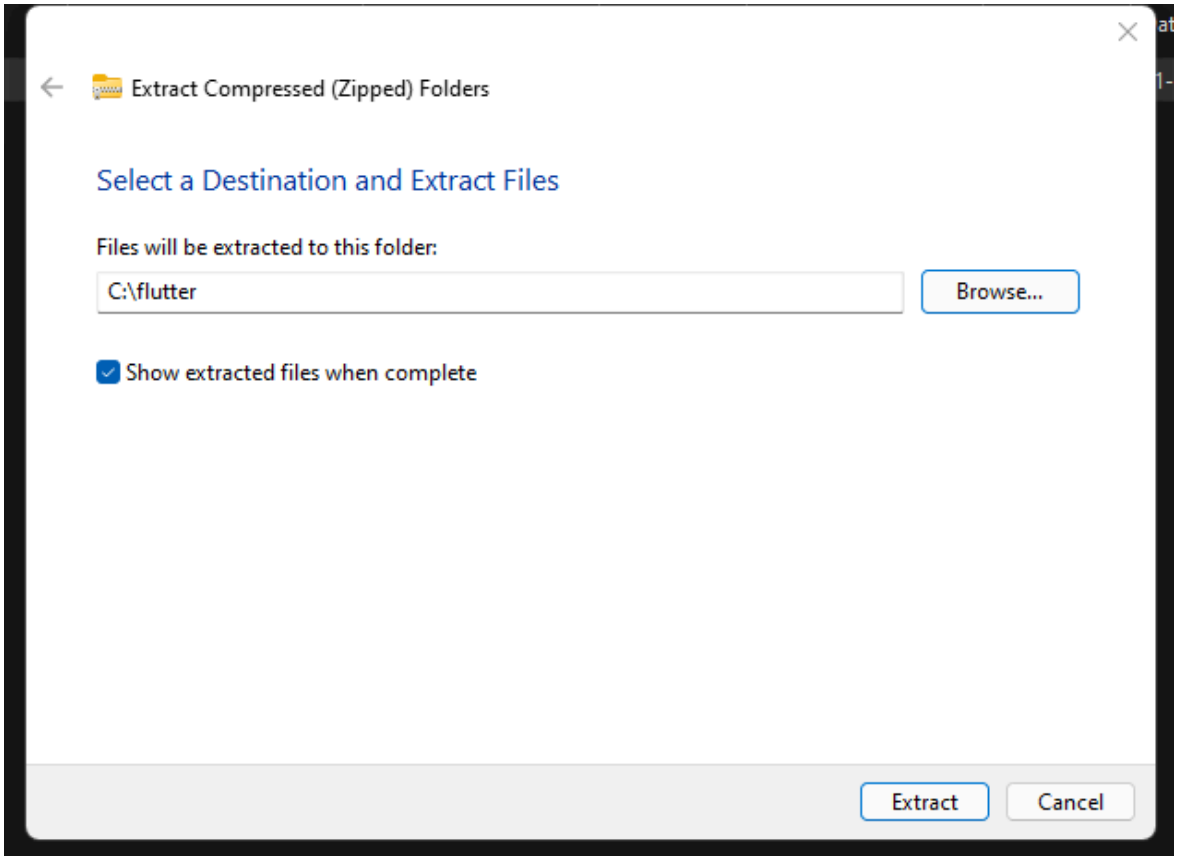
Step 2: Next, to download the latest Flutter SDK, click on the Windows icon and then select Android. Here, you will find system requirements and the download link for SDK.





Step 3: When your download is complete, extract the zip file and place it in the desired installation folder or location, for example, C: /Flutter. (Here I have created Flutter folder in C drive and inside that created src folder and extracted in it.)

Anushka Shahane D15A 54



Step 4 : Now, run the $ flutter command in command prompt.

```
Microsoft Windows [Version 10.0.22631.4751]
(c) Microsoft Corporation. All rights reserved.

C:\Users\anush>flutter
Manage your Flutter app development.

Common commands:

  flutter create <output directory>
    Create a new Flutter project in the specified directory.

  flutter run [options]
    Run your Flutter application on an attached device or in an emulator.

Usage: flutter <command> [arguments]

Global options:
-h, --help                  Print this usage information.
-v, --verbose               Noisy logging, including all shell commands executed.
                            If used with "--help", shows hidden options. If used with "flutter doctor", shows additional
                            diagnostic information. (Use "-vv" to force verbose logging in those cases.)
-d, --device-id             Target device id or name (prefixes allowed).
    --version               Reports the version of this tool.
    --enable-analytics      Enable telemetry reporting each time a flutter or dart command runs.
    --disable-analytics     Disable telemetry reporting each time a flutter or dart command runs, until it is
                            re-enabled.
    --suppress-analytics    Suppress analytics reporting for the current CLI invocation.

Available commands:

Flutter SDK
  bash-completion   Output command line shell completion setup scripts.
  channel           List or switch Flutter channels.
  config            Configure Flutter settings.
  doctor            Show information about the installed tooling.
  downgrade         Downgrade Flutter to the last active version for the current channel.
  precache          Populate the Flutter tool's cache of binary artifacts.
```

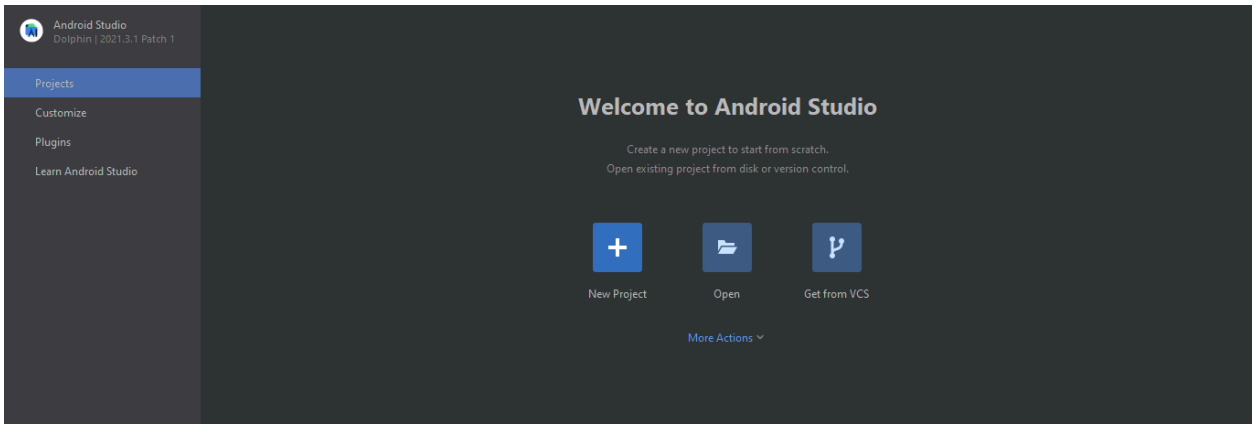
Step 5 : Now, run the $ flutter doctor command. This command checks for all the requirements of Flutter app development and displays a report of the status of your Flutter installation. Step 8: When you run the above command, it will analyze the system and show its report, as shown in the below image. Here, you will find the details of all missing tools, which required to run Flutter as well as the development tools that are available but not connected with the device.

```
C:\Users\anush>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.27.3, on Microsoft Windows [Version 10.0.22631.4751], locale en-IN)
[✓] Windows Version (Installed version of Windows is version 10 or higher)
[✓] Android toolchain - develop for Android devices (Android SDK version 34.0.0)
[✓] Chrome - develop for the web
[✓] Visual Studio - develop Windows apps (Visual Studio Build Tools 2022 17.12.3)
[!] Android Studio (not installed)
[✓] VS Code (version 1.96.4)
[✓] Connected device (4 available)
[✓] Network resources

! Doctor found issues in 1 category.

C:\Users\anush>
```

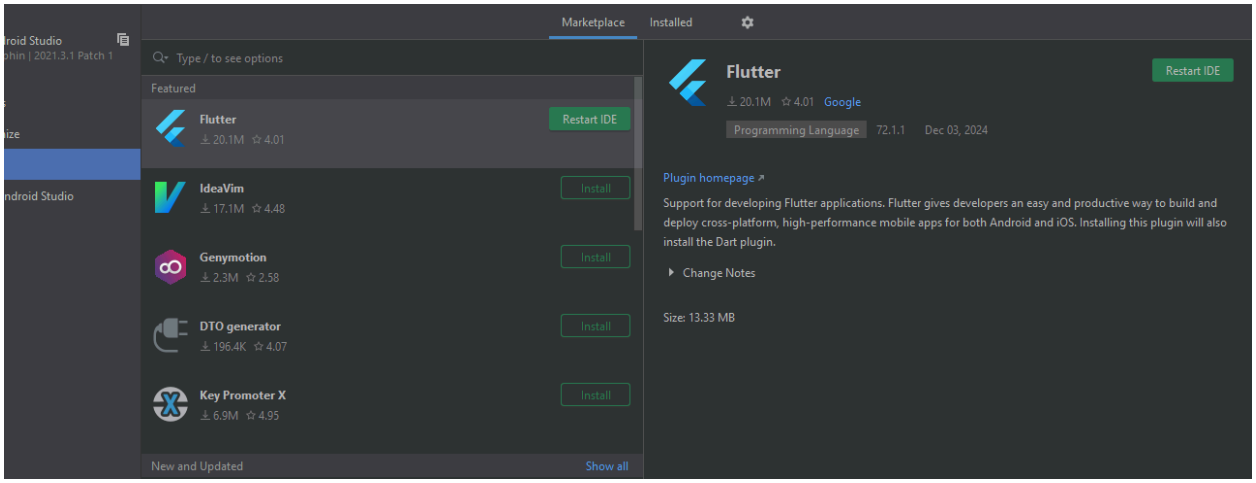
Anushka Shahane D15A 54

Step 6: Install the Android SDK. If the flutter doctor command does not find the Android SDK tool in your system, then you need first to install the Android Studio IDE. To install Android Studio IDE, do the following steps.

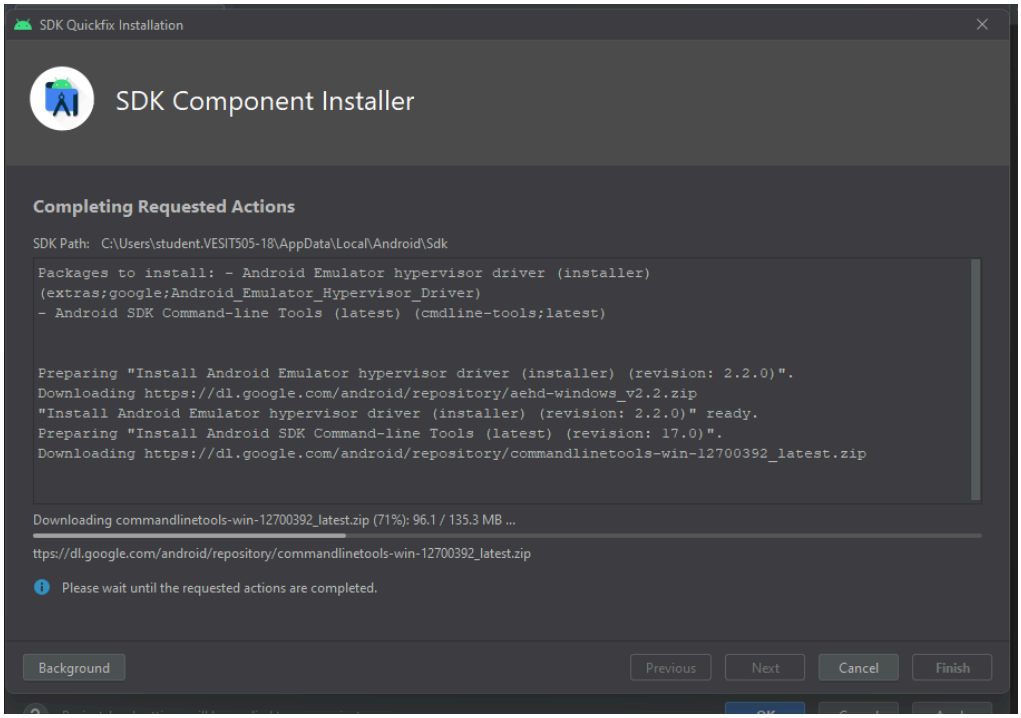
Step 7 : Download the latest Android Studio executable or zip file from the official site by accepting terms and conditions



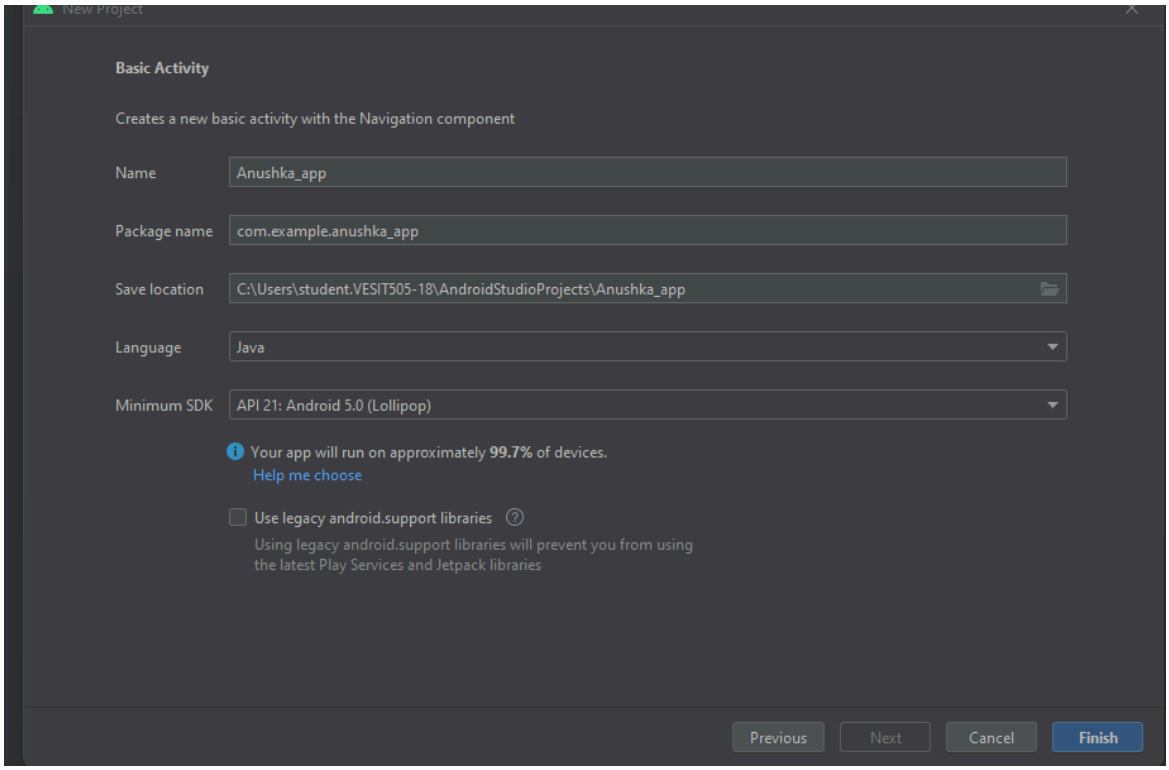
Now open android studio you will see the following window. Click on more actions-> Import an android code Sample -> select Android SDK command-line tools (latest) this will download command-line tools.



Anushka Shahane D15A 54



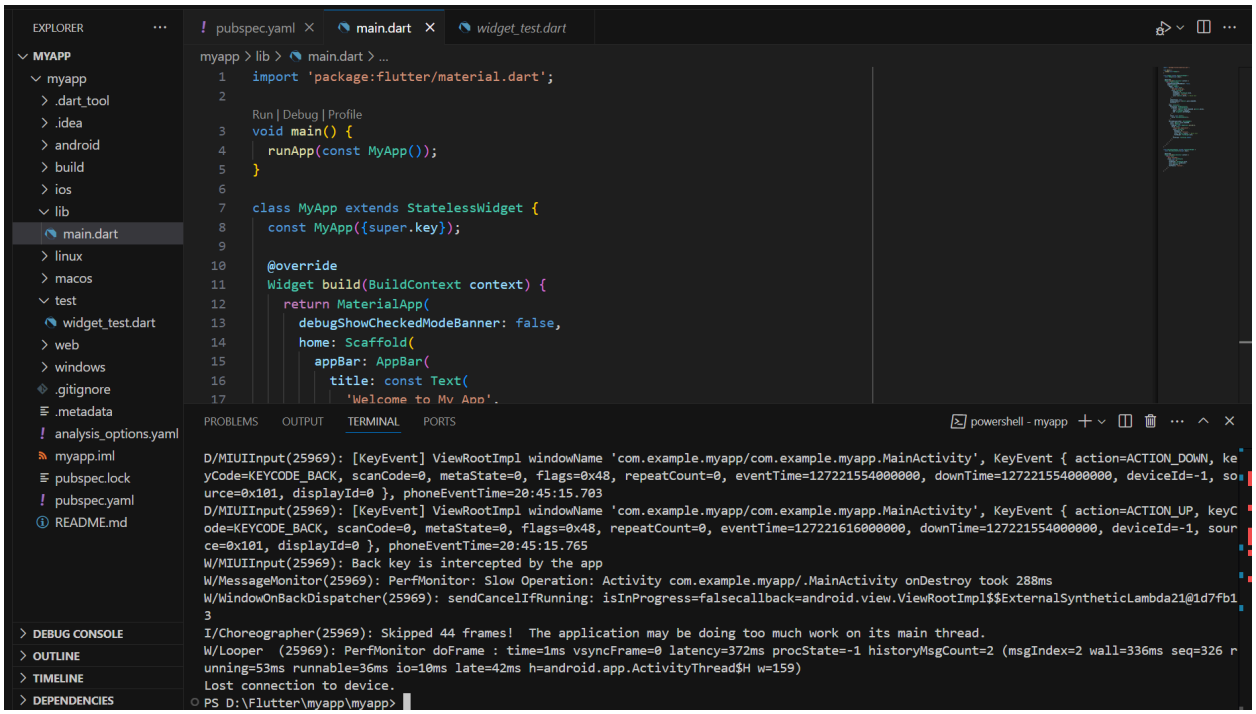
If you want you can create the project in Android Studio or Visual Studio code To create the project in Android studio:



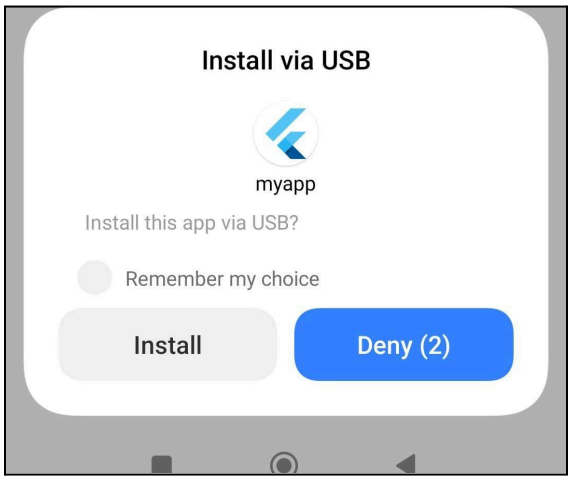
Anushka Shahane D15A 54

Creating project in visual studio code :

>flutter create myapp
>cd myapp
>flutter run



You can allow by giving necessary permissions for installation :



Anushka Shahane D15A 54

You can view the project as follow:

MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	54
Name	Anushka Shahane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

EXPERIMENT NO: - 02

Name : Anushka Shahane

Class : D15A **Roll:No :** 54

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

Theory: -

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

When you made any alteration in the code, the widget rebuilds its description by calculating the difference of previous and current widget to determine the minimal changes for rendering in UI of the app.Widgets are nested with each other to build the app. It means the root of your app is itself a widget, and all the way down is a widget also. For example, a widget can display something, can define design, can handle interaction, etc.

The single child layout widget is a type of widget, which can have only **one child widget** inside the parent layout widget. These widgets can also contain special layout functionality. Flutter provides us many single child widgets to make the app UI attractive. If we use these widgets appropriately, it can save our time and makes the app code more readable.

The multiple child widgets are a type of widget, which contains **more than one child widget**, and the layout of these widgets are **unique**. For example, Row widget laying out of its child widget in a horizontal direction, and Column widget laying out of its child widget in a vertical direction. If we combine the Row and Column widget, then it can build any level of the complex widget.

Type of Widgetss

- **StatefulWidget**

  A StatefulWidget has state information. It contains mainly two classes: the state object and the widget. It is dynamic because it can change the inner data during the widget lifetime. This widget does not have a build() method. It has createState() method, which returns a class that extends the Flutters State Class. The examples of the StatefulWidget are Checkbox, Radio, Slider, InkWell, Form, and TextField.

- **StatelessWidget**

  The StatelessWidget does not have any state information. It remains static throughout its lifecycle. The examples of the StatelessWidget are Text, Row, Column, Container, etc.

Some of the commonly used widgets

Container – A box widget used for styling with padding, margins, colors, borders, and constraints. It helps in layout structuring and positioning.

Row & Column – Used to arrange widgets in horizontal (Row) or vertical (Column) orientation. They manage spacing, alignment, and distribution of child widgets.

Stack – Overlaps widgets on top of each other, useful for creating layered UIs like banners, tooltips, or floating elements.

Text – Displays text on the screen with customizable font size, color, alignment, and styling options

Image – Loads and displays images from assets, network, or memory with scaling, fit, properties.

Scaffold – Provides a basic layout structure with an app bar, body, floating action button, and bottom navigation.

ListView – A scrollable list widget that efficiently renders large amounts of dynamic content. Supports both vertical and horizontal scrolling.

GridView – Displays widgets in a grid format, useful for galleries, product listings, or dashboards. It supports dynamic column adjustments.

SizedBox – Used to create space between widgets or define fixed width and height for layout adjustments.

ElevatedButton – A button with elevation that provides a raised effect, customizable with color, shape, and click actions.

TextField – A user input field that supports text entry, keyboard configurations, validation.

AppBar – A top navigation bar that includes a title, actions, and menu icons, commonly used in Scaffold.

BottomNavigationBar – A bar at the bottom of the screen used for navigation between different app sections with icons and labels.

Drawer – A side navigation panel that slides out from the left, typically used for app menus and quick navigation.

Card – A material design component that displays content inside a box with elevation.

Code: - home_screen.dart

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

class HomeScreen extends StatefulWidget {
  @override
  _HomeScreenState createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  int _selectedIndex = 0;
  String _selectedDate = "Select date";

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

  Future<void> _selectDate(BuildContext context) async {
    DateTimeRange? picked = await showDateRangePicker(
    context: context,
      firstDate: DateTime.now(),
      lastDate: DateTime(2030),
    );

    if (picked != null) {
      setState(() {
        _selectedDate =
            "${DateFormat('E dd MMM').format(picked.start)} –
 ${DateFormat('E dd MMM').format(picked.end)}";
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.grey[100],
      appBar: AppBar(
        backgroundColor: Colors.blue[900],
        title: Text(
          'Booking.com',
          style: TextStyle(
              color: Colors.white, fontWeight: FontWeight.bold,
fontSize: 18),
        ),
        actions: [
          IconButton(
            icon: Icon(Icons.notifications, color: Colors.white),
            onPressed: () {},
          ),
          IconButton(
            icon: Icon(Icons.account_circle, color:
Colors.white),
            onPressed: () {},
          ),
        ],
      ),
      body: SingleChildScrollView(
        child: Padding(
          padding: const EdgeInsets.all(12),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              // **Category Title**
              Row(
              mainAxisAlignment:
MainAxisAlignment.center,
                children: [
                  Icon(Icons.hotel, color: Colors.blue[900],
size: 24),
                  SizedBox(width: 8),
                  Text(
                    "Find your stays",
                    style: TextStyle(
                      color: Colors.blue[900],
                      fontSize: 14,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                ],
              ),

              SizedBox(height: 15),

              // **Search Section**
              Container(
                padding: EdgeInsets.all(12),
                decoration: BoxDecoration(
                  borderRadius: BorderRadius.circular(10),
                  border: Border.all(color: Colors.orange,
width: 2),
                  color: Colors.white,
                ),
                child: Column(
                  children: [
                    _buildSearchField(Icons.search, "Enter your
destination"),
                    GestureDetector(
                      onTap: () => _selectDate(context),
                      child: _buildSearchField(
                      Icons.calendar_today,
                        _selectedDate,
```

```
SizedBox(height: 10),
                ElevatedButton(
                  style: ElevatedButton.styleFrom(
                    backgroundColor: Colors.blue[900],
                    padding:
                        EdgeInsets.symmetric(vertical: 12, horizontal:
80),
                  ),
                  onPressed: () {
                    Navigator.push(
                    context,
                    MaterialPageRoute(
                         builder: (context) => DetailsFormScreen()),
                    );
                  },
                  child: Text("Search",
                      style: TextStyle(color: Colors.white, fontSize:
14)),
                ),
              ],
            ),
          ),

          SizedBox(height: 15),

          // **Discount Section**
          Text(
            "Exclusive Discounts",
            style: TextStyle(fontSize: 14, fontWeight: FontWeight
.bold),
          ),
          SizedBox(height: 8),
          SizedBox(
            height: 110,
            child: ListView(
              scrollDirection: Axis.horizontal,
              children: [
                _buildDiscountCard(
                    "10% discounts on stays", "Enjoy discounts
worldwide"),
                _buildDiscountCard(
                    "10% off rental cars", "Save on select rental cars"),
                _buildDiscountCard(
                    "Exclusive flight deals", "Best fares for
members"),
              ],
            ),
          ),

          SizedBox(height: 15),

          // **Deals Section**
          Text(
            "Deals for the weekend",
style: TextStyle(fontSize: 14, fontWeight:
 FontWeight.bold),
          ),
          Text(
            "Save on stays for 31 January - 2 February",
            style: TextStyle(fontSize: 12),
          ),

 SizedBox(height: 8),
          SizedBox(
            height: 160,
            child: ListView(
              scrollDirection: Axis.horizontal,
              children: [
                _buildDealCard(
                  "Paris, France",
                  "Starting from \$120/night",
                  "https://thumbs.dreamstime.com/b/paris-
eiffel-tower-river-seine-sunset-france-one-most-iconic-
landmarks-107376702.jpg",
                ),
                _buildDealCard(
                  "New York, USA",
                  "Starting from \$150/night",
                  "https://t3.ftcdn.net/jpg/02/09/70/56/360_
F_209705645_b78HGJI1i1mxqLwMYA7z1m3VvCxgx
JFO.jpg",
                ),
                _buildDealCard(
                  "Tokyo, Japan",
                  "Starting from \$130/night",
                  "https://media.istockphoto.com/id/484915
982/photo/akihabara-
tokyo.jpg?s=612x612&w=0&k=20&c=kbCRYJS5vZu
F4jLB3y4-apNebcCEkWnDbKPpxXdf9Cg=",
                ),
              ],
            ),
          ),
        ],
      ),
    ),

    // **Bottom Navigation Bar**
    bottomNavigationBar: BottomNavigationBar(
    backgroundColor: Colors.white, // Set background
color to white

      currentIndex: _selectedIndex,
      onTap: _onBottomNavTapped,
      selectedItemColor: Colors.blue[900],
      unselectedItemColor: Colors.grey,
      type: BottomNavigationBarType.fixed,
```

```
            icon: Icon(Icons.person, size: 22), label: "My account"),
        ],
      ),
    );
  }
Widget _buildSearchField(IconData icon, String hintText,
      {bool isReadOnly = false}) {
    return Padding(
      padding: EdgeInsets.symmetric(vertical: 4),
      child: TextField(
        readOnly: isReadOnly,
        decoration: InputDecoration(
          prefixIcon: Icon(icon, color: Colors.black54, size: 20),
          hintText: hintText,
          hintStyle: TextStyle(fontSize: 12),
          border: OutlineInputBorder(borderRadius:
BorderRadius.circular(5)),
        ),
      ),
    );
  }

  Widget _buildDiscountCard(String title, String subtitle) {
    return Container(
      width: 150,
      margin: EdgeInsets.only(right: 8),
      padding: EdgeInsets.all(8),
      decoration: BoxDecoration(
      color: Colors.white,
        border: Border.all(color: Colors.blue),
        borderRadius: BorderRadius.circular(8),
      ),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Text(title,
              style: TextStyle(fontWeight: FontWeight.bold,
 fontSize: 12)),
          SizedBox(height: 4),
          Text(subtitle, style: TextStyle(fontSize: 10)),
        ],
      ),
    );
  }

  Widget _buildDealCard(String location, String price,
 String imageUrl) {
    return Container(
      width: 180,
      margin: EdgeInsets.only(right: 8),
padding: EdgeInsets.all(8),
      decoration: BoxDecoration(
      color: Colors.white,
        borderRadius: BorderRadius.circular(8),



  boxShadow: [


  BoxShadow(color: Colors.grey.withOpacity(0.2),
blurRadius: 4)
        ],
      ),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          ClipRRect(



borderRadius: BorderRadius.circular(8),

 child: Image.network(
              imageUrl,
              height: 80,
              width: double.infinity,
              fit: BoxFit.cover,
            ),
          ),
          SizedBox(height: 4),
          Text(location,
              style: TextStyle(fontWeight: FontWeight.bold,
fontSize: 12)),
          SizedBox(height: 4),
          Text(price, style: TextStyle(fontSize: 11, color:
Colors.black54)),
        ],
      ),
    );
  }
}
```

Code : welcome_screen.dart

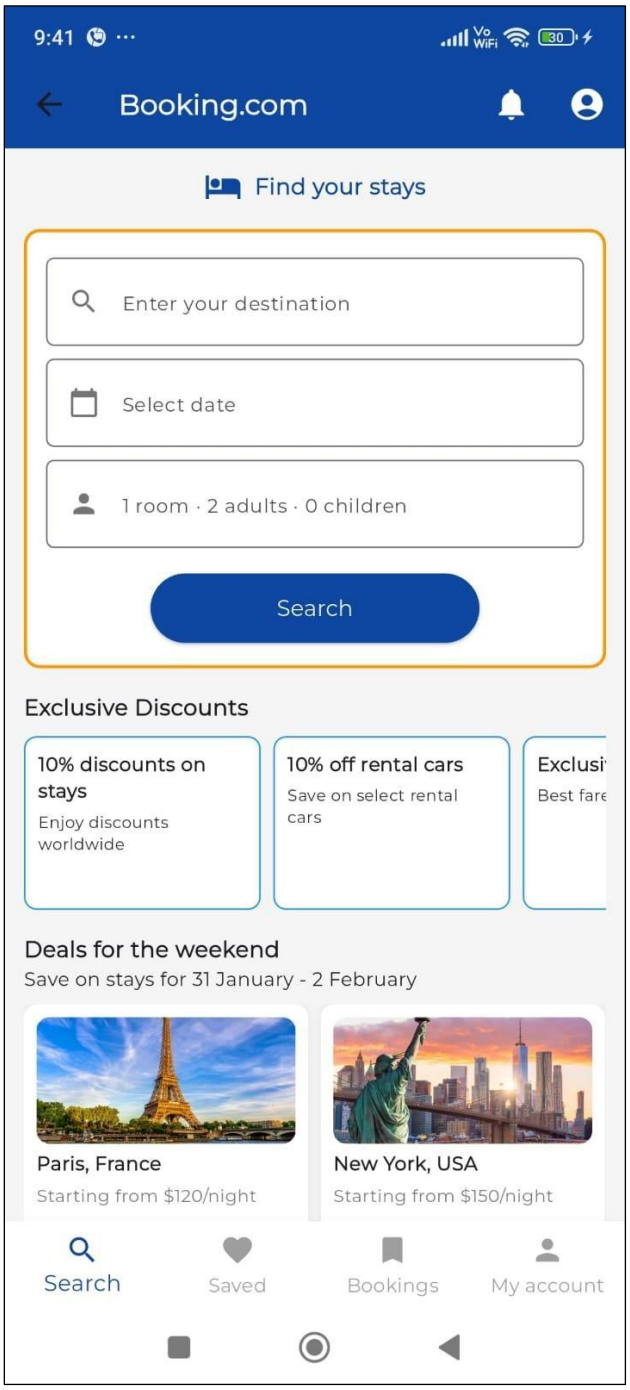
```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import '../core/routes.dart';
import '../widgets/custom_button.dart';

class WelcomeScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      appBar: AppBar(
        backgroundColor: const Color(0xFF003580),
// Booking.com Blue
        elevation: 0,
        title: Text(
        "Booking.com",
          style: GoogleFonts.montserrat(
            fontSize: 20,
            fontWeight: FontWeight.w600,
            color: Colors.white,
          ),
        ),
        centerTitle: true,
        leading: IconButton(
          icon: const Icon(Icons.arrow_back,
              color: Colors.white), // Back arrow in
white
          onPressed: () => Navigator.pop(context), //
Navigate back
        ),
      ),
      body: SingleChildScrollView(
        child: Padding(
          padding: const
EdgeInsets.symmetric(horizontal: 25, vertical: 40),
          child: Column(
            crossAxisAlignment:
CrossAxisAlignment.start,
            children: [
              // Title
              Text(
                "Register here",
              ),
              const SizedBox(height: 30),

                  border: Border.all(color:
Colors.grey.shade400),
                ),
                child: TextField(
                  decoration: InputDecoration(
                    labelText: "Enter your name",
                    labelStyle: GoogleFonts.montserrat(
 const SizedBox(height: 5),
              Text(
                "We'll use this to create an account if you
don't have one yet.",
                style: GoogleFonts.montserrat(
                  fontSize: 14,
                  color: Colors.grey[600],
                ),
              ),
              const SizedBox(height: 30),

              // Email Entry Box
              Container(
              padding: const
EdgeInsets.symmetric(horizontal: 12),
                decoration: BoxDecoration(
                  borderRadius: BorderRadius.circular(8),
                  border: Border.all(color:
Colors.grey.shade400),
                ),
                child: TextField(
                  decoration: InputDecoration(
                    labelText: "Enter your name",
                    labelStyle: GoogleFonts.montserrat(
                        fontSize: 14, color: Colors.grey[600]),
                    border: InputBorder.none,
                  ),
                ),
              ),
              const SizedBox(height: 30),

              // Continue Button
              CustomButton(
              text: "Continue",
                onTap: () => Navigator.pushNamed(context,
Routes.signUp),
                backgroundColor: const Color(0xFF003580),
                textColor: Colors.white,
              ),
              const SizedBox(height: 40),
```

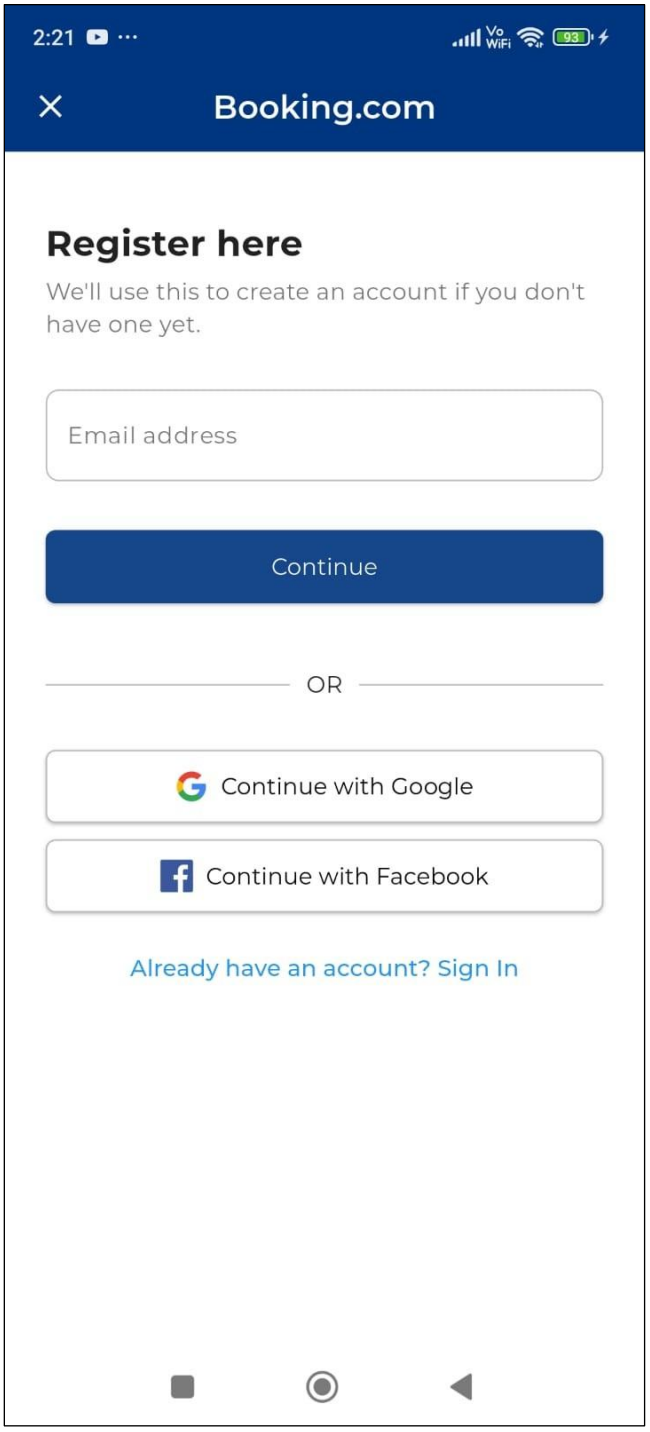
```
            Row(
              children: [
                Expanded(child: Divider(color: Colors.grey.shade400)),
                Padding(
                  padding: const EdgeInsets.symmetric(horizontal: 10),
                  child: Text("OR",
                      style: GoogleFonts.montserrat(
                          fontSize: 14, color: Colors.black87)),
                ),
                Expanded(child: Divider(color: Colors.grey.shade400)),
              ],
            ),
            const SizedBox(height: 30),

            // Continue with Google
            CustomButton(
              text: "Continue with Google",
              onTap: () {},
              isOutlined: true,
              textColor: Colors.black,
              borderColor: Colors.grey.shade400,
              iconPath: "assets/images/google.png", // Add Google icon
            ),
            const SizedBox(height: 10),

            // Continue with Facebook
            CustomButton(
              text: "Continue with Facebook",
              iconPath: "assets/images/facebook.png", // Add Facebook icon
            ),
            const SizedBox(height: 10),
            Center(
              child: TextButton(
                onPressed: () => Navigator.pushNamed(context, Routes.signIn),
                child: Text(
                  "Already have an account? Sign In",
                  style: GoogleFonts.montserrat(
                  fontSize: 14,
                    fontWeight: FontWeight.w500,
                    color: Colors.blue,
),
),
}
}
```

Code: splash_screen.dart

```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:booking/core/theme.dart';
import 'package:booking/core/routes.dart';

class SplashScreen extends StatefulWidget {
  @override
  _SplashScreenState createState() => _SplashScreenState();
}

class _SplashScreenState extends State<SplashScreen>
    with SingleTickerProviderStateMixin {
  late AnimationController _controller;
  late Animation<double> _fadeInAnimation;

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

    _controller = AnimationController(
      vsync: this,
      duration: const Duration(milliseconds: 1200),
    );

    _fadeInAnimation = Tween<double>(begin: 0, end: 1).animate(_controller);

    _controller.forward();

    Future.delayed(const Duration(seconds: 3), () {
      Navigator.pushReplacementNamed(context, Routes.welcome);
    });
  }
@override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: AppColors.primaryColor,
      body: SafeArea(
        child: Center(
          child: FadeTransition(
            opacity: _fadeInAnimation,
            child: Text(
            "Booking.com",
              style: GoogleFonts.openSans(
                fontSize: 27, // Font size for visibility
                fontWeight: FontWeight.bold, // Using Open Sans Bold
                color: Colors.white,
                letterSpacing: 1.2, // Improved spacing for readability
              )
    );
  } @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }
}
```

OUTPUT :


9:41
Booking.com
Find your stays
Enter your destination
Select date
1 room · 2 adults · 0 children
Search
Exclusive Discounts
10% discounts on stays
Enjoy discounts worldwide
10% off rental cars
Save on select rental cars
Deals for the weekend
Save on stays for 31 January - 2 February
Paris, France
Starting from $120/night
New York, USA
Starting from $150/night
Search
Saved
Bookings
My account


2:21
Booking.com
Register here
We'll use this to create an account if you don't have one yet.
Email address
Continue
OR
Continue with Google
Continue with Facebook
Already have an account? Sign In


9:41
Booking.com

MAD & PWA Lab

Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	54
Name	Anushka Shahane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

EXPERIMENT NO : 03

Name : Anushka Shahane

Class:- D15A **Roll:No : 54**

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

Theory: -

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

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

Visual elements play a significant role in app development.

- **Enhanced User Experience**: Images and icons make your app visually appealing and 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: - home_screen.dart

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

class HomeScreen extends StatefulWidget {
  @override
  _HomeScreenState createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  int _selectedIndex = 0;
  String _selectedDate = "Select date";

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

  Future<void> _selectDate(BuildContext context) async {
    DateTimeRange? picked = await showDateRangePicker(
    context: context,
      firstDate: DateTime.now(),
      lastDate: DateTime(2030),
    );

    if (picked != null) {
      setState(() {
        _selectedDate =
            "${DateFormat('E dd MMM').format(picked.start)} –
${DateFormat('E dd MMM').format(picked.end)}";
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.grey[100],
      appBar: AppBar(
        backgroundColor: Colors.blue[900],
        title: Text(
          'Booking.com',
          style: TextStyle(
              color: Colors.white, fontWeight: FontWeight.bold,
fontSize: 18),
        ),
        actions: [
          IconButton(
            icon: Icon(Icons.notifications, color: Colors.white),
            onPressed: () {},
          ),
          IconButton(
            icon: Icon(Icons.account_circle, color:
Colors.white),
            onPressed: () {},
          ),
        ],
      ),
      body: SingleChildScrollView(
        child: Padding(
          padding: const EdgeInsets.all(12),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              // **Category Title**
              Row(
              mainAxisAlignment:
MainAxisAlignment.center,
                children: [
                  Icon(Icons.hotel, color: Colors.blue[900],
size: 24),
                  SizedBox(width: 8),
                  Text(
                    "Find your stays",
                    style: TextStyle(
                      color: Colors.blue[900],
                      fontSize: 14,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                ],
              ),

              SizedBox(height: 15),

              // **Search Section**
              Container(
                padding: EdgeInsets.all(12),
                decoration: BoxDecoration(
                  borderRadius: BorderRadius.circular(10),
                  border: Border.all(color: Colors.orange,
width: 2),
                  color: Colors.white,
                ),
                child: Column(
                  children: [
                    _buildSearchField(Icons.search, "Enter your
destination"),
                    GestureDetector(
                      onTap: () => _selectDate(context),
                      child: _buildSearchField(
                      Icons.calendar_today,
                        _selectedDate,
```

```
SizedBox(height: 10),
                ElevatedButton(
                  style: ElevatedButton.styleFrom(
                    backgroundColor: Colors.blue[900],
                    padding:
                        EdgeInsets.symmetric(vertical: 12, horizontal:
80),
                  ),
                  onPressed: () {
                    Navigator.push(
                    context,
                    MaterialPageRoute(
                         builder: (context) => DetailsFormScreen()),
                    );
                  },
                  child: Text("Search",
                      style: TextStyle(color: Colors.white, fontSize:
14)),
                ),
              ],
            ),
          ),

          SizedBox(height: 15),

          // **Discount Section**
          Text(
            "Exclusive Discounts",
            style: TextStyle(fontSize: 14, fontWeight: FontWeight
.bold),
          ),
          SizedBox(height: 8),
          SizedBox(
            height: 110,
            child: ListView(
              scrollDirection: Axis.horizontal,
              children: [
                _buildDiscountCard(
                    "10% discounts on stays", "Enjoy discounts
worldwide"),
                _buildDiscountCard(
                    "10% off rental cars", "Save on select rental cars"),
                _buildDiscountCard(
                    "Exclusive flight deals", "Best fares for
members"),
              ],
            ),
          ),

          SizedBox(height: 15),

          // **Deals Section**
          Text(
            "Deals for the weekend",
            style: TextStyle(fontSize: 14, fontWeight:
 FontWeight.bold),
          ),
          Text(
            "Save on stays for 31 January - 2 February",
            style: TextStyle(fontSize: 12),
          ),

 SizedBox(height: 8),
          SizedBox(
            height: 160,
            child: ListView(
              scrollDirection: Axis.horizontal,
              children: [
                _buildDealCard(
                  "Paris, France",
                  "Starting from \$120/night",
                  "https://thumbs.dreamstime.com/b/paris-
eiffel-tower-river-seine-sunset-france-one-most-iconic-
landmarks-107376702.jpg",
                ),
                _buildDealCard(
                  "New York, USA",
                  "Starting from \$150/night",
                  "https://t3.ftcdn.net/jpg/02/09/70/56/360_
F_209705645_b78HGJI1i1mxqLwMYA7z1m3VvCxgx
JFO.jpg",
                ),
                _buildDealCard(
                  "Tokyo, Japan",
                  "Starting from \$130/night",
                  "https://media.istockphoto.com/id/484915
982/photo/akihabara-
tokyo.jpg?s=612x612&w=0&k=20&c=kbCRYJS5vZu
F4jLB3y4-apNebcCEkWnDbKPpxXdf9Cg=",
                ),
              ],
            ),
          ),
        ],
      ),
    ),

    // **Bottom Navigation Bar**
    bottomNavigationBar: BottomNavigationBar(
    backgroundColor: Colors.white, // Set background
color to white

      currentIndex: _selectedIndex,
      onTap: _onBottomNavTapped,
      selectedItemColor: Colors.blue[900],
      unselectedItemColor: Colors.grey,
      type: BottomNavigationBarType.fixed,
```

```
            icon: Icon(Icons.person, size: 22), label: "My account"),
        ],
      ),
    );
  }
Widget _buildSearchField(IconData icon, String hintText,
      {bool isReadOnly = false}) {
    return Padding(
      padding: EdgeInsets.symmetric(vertical: 4),
      child: TextField(
        readOnly: isReadOnly,
        decoration: InputDecoration(
          prefixIcon: Icon(icon, color: Colors.black54, size: 20),
          hintText: hintText,
          hintStyle: TextStyle(fontSize: 12),
          border: OutlineInputBorder(borderRadius:
BorderRadius.circular(5)),
        ),
      ),
    );
  }

  Widget _buildDiscountCard(String title, String subtitle) {
    return Container(
      width: 150,
      margin: EdgeInsets.only(right: 8),
      padding: EdgeInsets.all(8),
      decoration: BoxDecoration(
      color: Colors.white,
        border: Border.all(color: Colors.blue),
        borderRadius: BorderRadius.circular(8),
      ),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Text(title,
              style: TextStyle(fontWeight: FontWeight.bold,
 fontSize: 12)),
          SizedBox(height: 4),
          Text(subtitle, style: TextStyle(fontSize: 10)),
        ],
      ),
    );
  }

  Widget _buildDealCard(String location, String price,
 String imageUrl) {
    return Container(
      width: 180,
      margin: EdgeInsets.only(right: 8),
      padding: EdgeInsets.all(8),
      decoration: BoxDecoration(
      color: Colors.white,
        borderRadius: BorderRadius.circular(8),


    boxShadow: [


    BoxShadow(color: Colors.grey.withOpacity(0.2),
blurRadius: 4)
        ],
      ),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          ClipRRect(


borderRadius: BorderRadius.circular(8),

  child: Image.network(
              imageUrl,
              height: 80,
              width: double.infinity,
              fit: BoxFit.cover,
            ),
          ),
          SizedBox(height: 4),
          Text(location,
              style: TextStyle(fontWeight: FontWeight.bold,
fontSize: 12)),
          SizedBox(height: 4),
          Text(price, style: TextStyle(fontSize: 11, color:
Colors.black54)),
        ],
      ),
    );
  }
}
```

Code : welcome_screen.dart

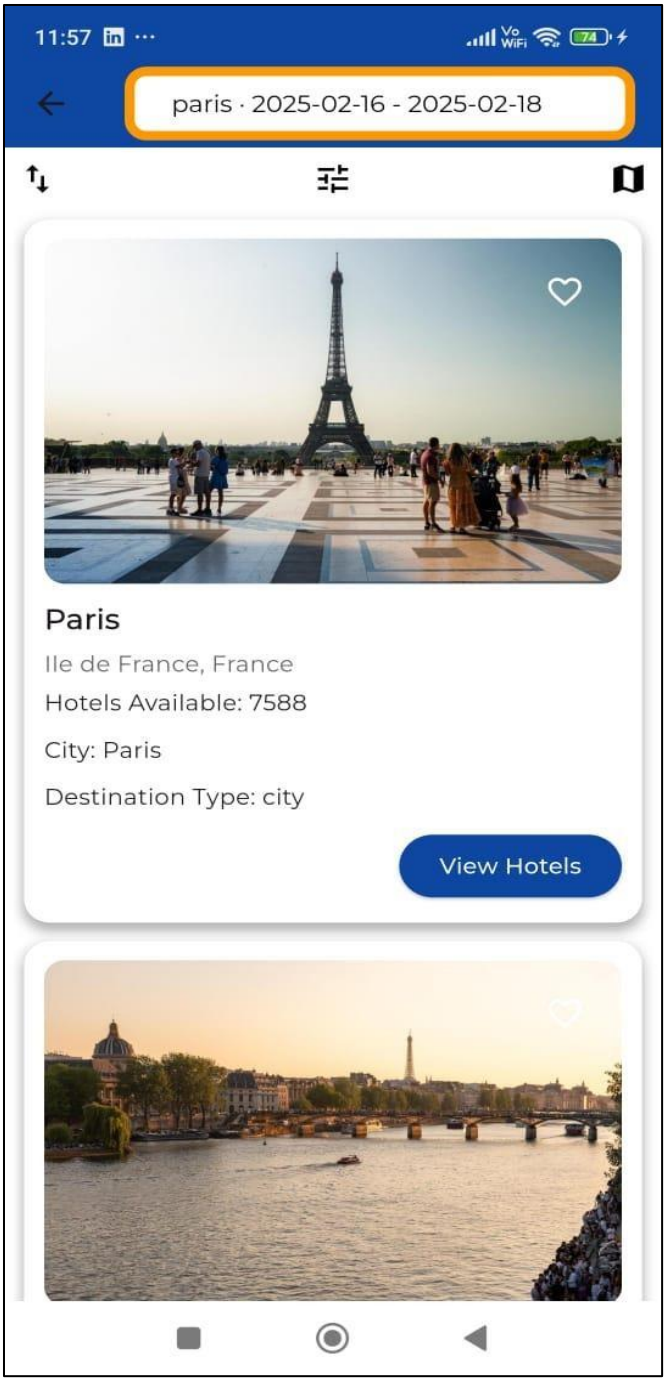
```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import '../core/routes.dart';
import '../widgets/custom_button.dart';

class WelcomeScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      appBar: AppBar(
        backgroundColor: const Color(0xFF003580),
// Booking.com Blue
        elevation: 0,
        title: Text(
        "Booking.com",
          style: GoogleFonts.montserrat(
            fontSize: 20,
            fontWeight: FontWeight.w600,
            color: Colors.white,
          ),
        ),
        centerTitle: true,
        leading: IconButton(
          icon: const Icon(Icons.arrow_back,
              color: Colors.white), // Back arrow in
white
          onPressed: () => Navigator.pop(context), //
Navigate back
        ),
      ),
      body: SingleChildScrollView(
        child: Padding(
          padding: const
EdgeInsets.symmetric(horizontal: 25, vertical: 40),
          child: Column(
            crossAxisAlignment:
CrossAxisAlignment.start,
            children: [
              // Title
              Text(
                "Register here",
              ),
              const SizedBox(height: 30),

                  border: Border.all(color:
Colors.grey.shade400),
                ),
                child: TextField(
                  decoration: InputDecoration(
                    labelText: "Enter your name",
                    labelStyle: GoogleFonts.montserrat(
 const SizedBox(height: 5),
              Text(
                "We'll use this to create an account if you
don't have one yet.",
                style: GoogleFonts.montserrat(
                  fontSize: 14,
                  color: Colors.grey[600],
                ),
              ),
              const SizedBox(height: 30),

              // Email Entry Box
              Container(
              padding: const
EdgeInsets.symmetric(horizontal: 12),
                decoration: BoxDecoration(
                  borderRadius: BorderRadius.circular(8),
                  border: Border.all(color:
Colors.grey.shade400),
                ),
                child: TextField(
                  decoration: InputDecoration(
                    labelText: "Enter your name",
                    labelStyle: GoogleFonts.montserrat(
                        fontSize: 14, color: Colors.grey[600]),
                    border: InputBorder.none,
                  ),
                ),
              ),
              const SizedBox(height: 30),

              // Continue Button
              CustomButton(
              text: "Continue",
                onTap: () => Navigator.pushNamed(context,
Routes.signUp),
                backgroundColor: const Color(0xFF003580),
                textColor: Colors.white,
              ),
              const SizedBox(height: 40),
```

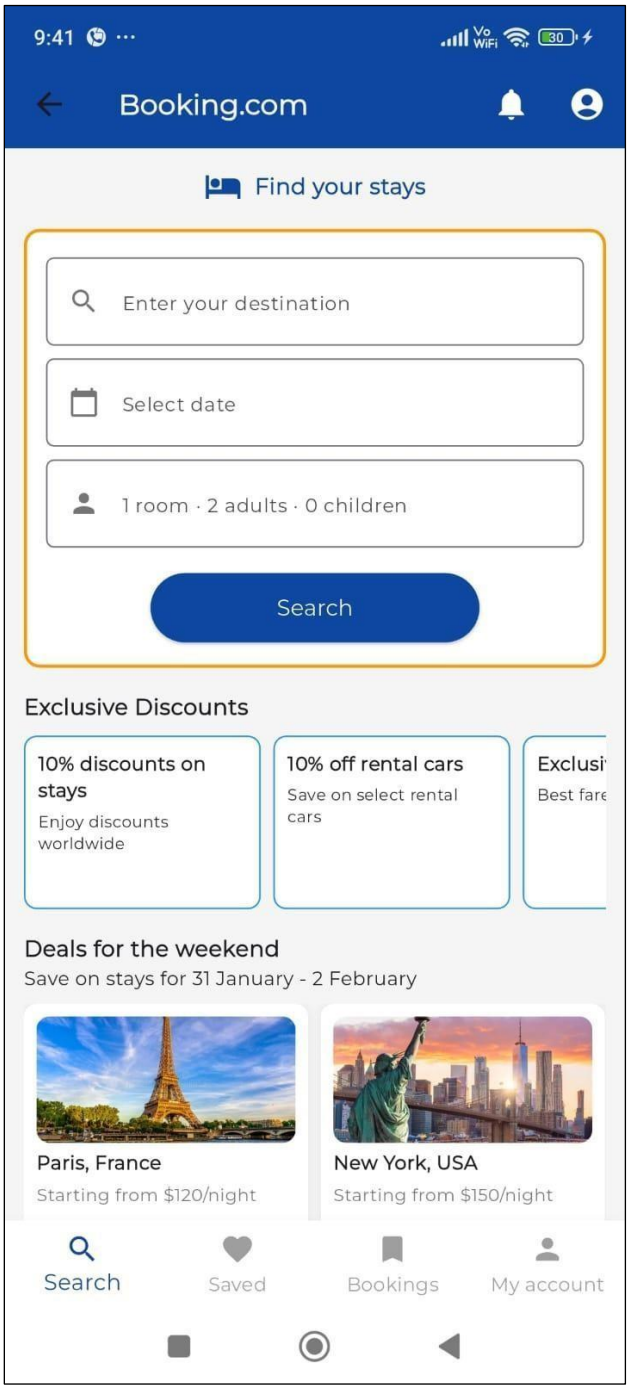
```
            Row(
              children: [
                Expanded(child: Divider(color: Colors.grey.shade400)),
                Padding(
                  padding: const EdgeInsets.symmetric(horizontal: 10),
                  child: Text("OR",
                      style: GoogleFonts.montserrat(
                          fontSize: 14, color: Colors.black87)),
                ),
                Expanded(child: Divider(color: Colors.grey.shade400)),
              ],
            ),
            const SizedBox(height: 30),

            // Continue with Google
            CustomButton(
              text: "Continue with Google",
              onTap: () {},
              isOutlined: true,
              textColor: Colors.black,
              borderColor: Colors.grey.shade400,
              iconPath: "assets/images/google.png", // Add Google icon
            ),
            const SizedBox(height: 10),

            // Continue with Facebook
            CustomButton(
              text: "Continue with Facebook",
              iconPath: "assets/images/facebook.png", // Add Facebook icon
            ),
            const SizedBox(height: 10),
            Center(
              child: TextButton(
                onPressed: () => Navigator.pushNamed(context, Routes.signIn),
                child: Text(
                  "Already have an account? Sign In",
                  style: GoogleFonts.montserrat(
                  fontSize: 14,
                    fontWeight: FontWeight.w500,
                    color: Colors.blue,
),
),
}
}
```

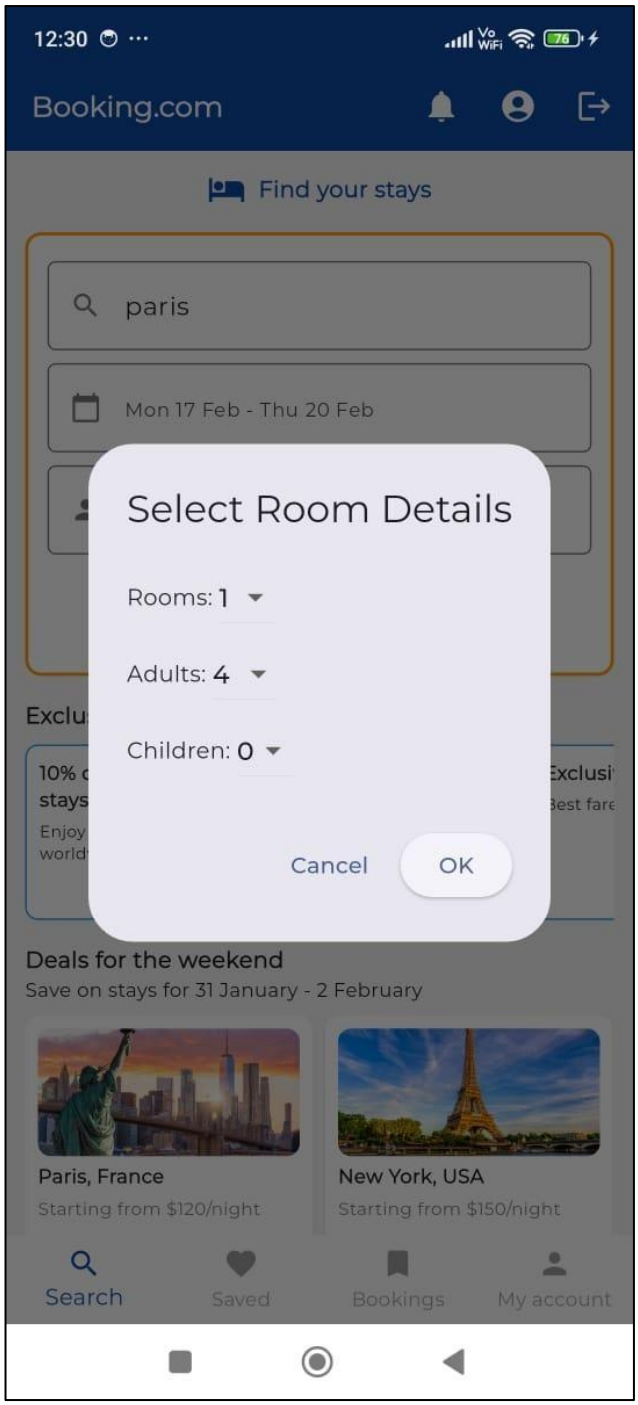
Code: hotel_screen.dart

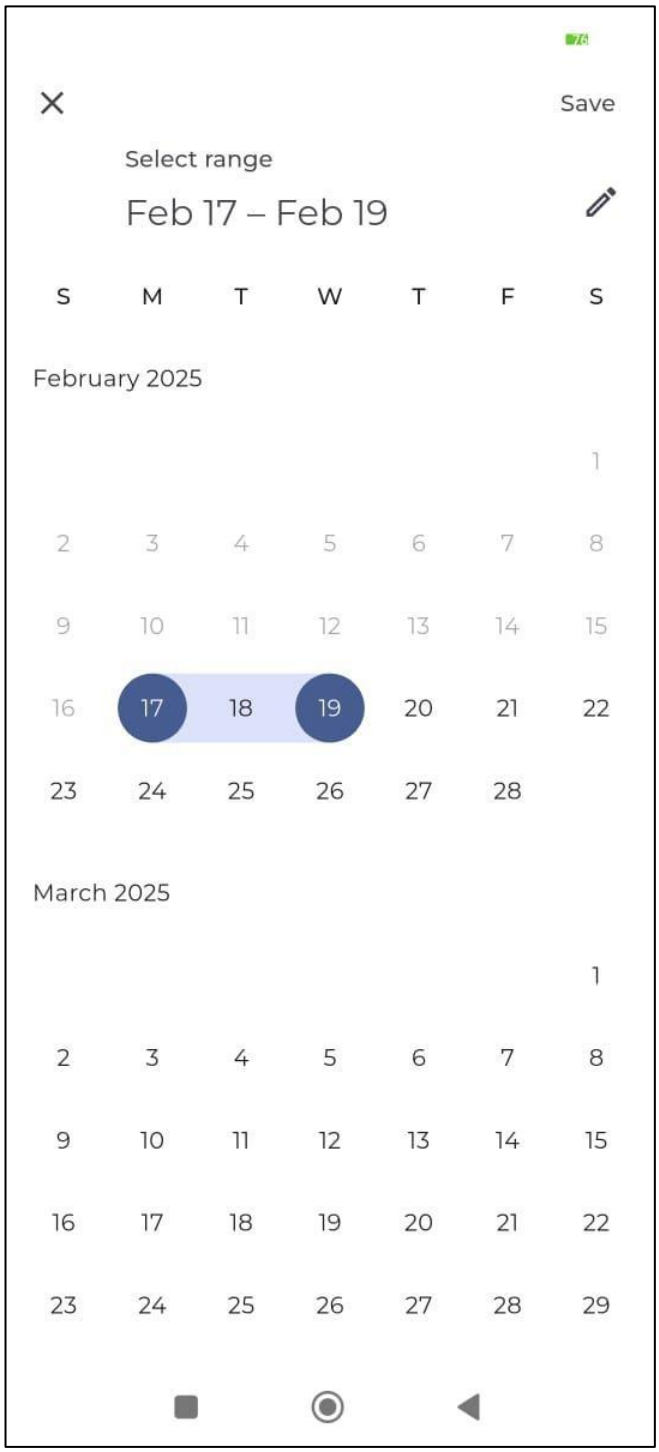
```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:booking/core/theme.dart';
import 'package:booking/core/routes.dart';

class SplashScreen extends StatefulWidget {
  @override
  _SplashScreenState createState() => _SplashScreenState();
}

class _SplashScreenState extends State<SplashScreen>
    with SingleTickerProviderStateMixin {
  late AnimationController _controller;
  late Animation<double> _fadeInAnimation;

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

    _controller = AnimationController(
      vsync: this,
      duration: const Duration(milliseconds: 1200),
    );

    _fadeInAnimation = Tween<double>(begin: 0, end: 1).animate(_controller);

    _controller.forward();

    Future.delayed(const Duration(seconds: 3), () {
      Navigator.pushReplacementNamed(context, Routes.welcome);
    });
  }
@override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: AppColors.primaryColor,
      body: SafeArea(
        child: Center(
          child: FadeTransition(
            opacity: _fadeInAnimation,
            child: Text(
            "Booking.com",
              style: GoogleFonts.openSans(
                fontSize: 27, // Font size for visibility
                fontWeight: FontWeight.bold, // Using Open Sans Bold
                color: Colors.white,
                letterSpacing: 1.2, // Improved spacing for readability
              )
    );
  } @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }
}
```

OUTPUT :


11:57
paris · 2025-02-16 - 2025-02-18
Paris
Ile de France, France
Hotels Available: 7588
City: Paris
Destination Type: city
View Hotels


9:41
Booking.com
Find your stays
Enter your destination
Select date
1 room · 2 adults · 0 children
Search
Exclusive Discounts
10% discounts on stays
Enjoy discounts worldwide
10% off rental cars
Save on select rental cars
Deals for the weekend
Save on stays for 31 January - 2 February
Paris, France
Starting from $120/night
New York, USA
Starting from $150/night
Search
Saved
Bookings
My account


12:30
Booking.com
Find your stays
paris
Mon 17 Feb - Thu 20 Feb
Select Room Details
Rooms: 1
Adults: 4
Children: 0
Cancel
OK
Deals for the weekend
Save on stays for 31 January - 2 February
Paris, France
Starting from $120/night
New York, USA
Starting from $150/night
Search
Saved
Bookings
My account


Save
Select range
Feb 17 – Feb 19
S M T W T F S
February 2025
1
2 3 4 5 6 7 8
9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28
March 2025
1
2 3 4 5 6 7 8
9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29

MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	54
Name	Anushka Shahane
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

EXPERIMENT NO: - 04

Name : Anushka Shahane

Class : D15A **Roll:No : 54**

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

Theory: -

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

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

Creation of a Form

- While creating a form in Flutter, the **Form widget** is essential as it acts as a container for grouping multiple form fields and managing validation.
- A **GlobalKey<FormState>** 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 :

Sign_in_screen.dart :

```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:booking/screens/home_screen.dart';
// Correct import
class SignInScreen extends StatefulWidget {
  @override
  _SignInScreenState createState() => _SignInScreenState();
}

class _SignInScreenState extends State<SignInScreen> {
final TextEditingController _emailController =
TextEditingController();
  final TextEditingController _passwordController =
TextEditingController();
  bool _isPasswordVisible = false;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      appBar: AppBar(
        backgroundColor: const Color(0xFF003580),
        elevation: 0,
        leading: IconButton(
          icon: const Icon(Icons.close, color: Colors.white),
          onPressed: () => Navigator.pop(context),
        ),
        centerTitle: true,
        title: Text(
          "Booking.com",
          style: GoogleFonts.montserrat(
            fontSize: 22,
            fontWeight: FontWeight.w600,
            color: Colors.white,
          ),
        ),
      ),
      body: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              "Sign in",
              style: GoogleFonts.montserrat(
                fontSize: 22,
                fontWeight: FontWeight.bold,
                color: Colors.black,
              ),
            ),
            const SizedBox(height: 20),

            // Email Field
            Text(
              "Email address",
              style: GoogleFonts.montserrat(
                fontSize: 14,
                fontWeight: FontWeight.w500,
                color: Colors.black,
```

```
controller: _emailController,
            keyboardType: TextInputType.emailAddress,
            decoration: InputDecoration(
              hintText: "Enter your email",
              hintStyle: GoogleFonts.montserrat(color:
Colors.grey[500]),
              border: OutlineInputBorder(
                borderRadius: BorderRadius.circular(8),
                borderSide: const BorderSide(color: Colors.blue),
              ),
              focusedBorder: OutlineInputBorder(
                borderRadius: BorderRadius.circular(8),
                borderSide: const BorderSide(color:
Colors.blue, width: 2),
              ),
              contentPadding: const EdgeInsets.symmetric(
                vertical: 12,
                horizontal: 15,
              ),
            ),
          ),
            // Sign In Button
          SizedBox(
            width: double.infinity,
            height: 50,
            child: ElevatedButton(
              onPressed: () {
                String email = _emailController.text.trim();
                String password = _passwordController.text.trim();
  if (email.isEmpty || !email.contains("@")) {
                ScaffoldMessenger.of(context).showSna
ckBar(
                  const SnackBar(
                    content: Text("Please enter a valid
email")),
                );
              } else if (password.isEmpty ||
password.length < 6) {
                ScaffoldMessenger.of(context).showSna
ckBar(
                  const SnackBar(
                    content:
                      Text("Password must be at least 6
characters")),
                  );
                } else {
                  // Navigate to Home Screen after
successful login
                  Navigator.pushReplacement(
                    context,
                    MaterialPageRoute(
                      builder: (context) => HomeScreen()),
// Corrected
                    );
                  }
                },
              style: ElevatedButton.styleFrom(
                backgroundColor: const
Color(0xFF003580), // Booking.com Blue
                shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.circular(8),
                ),
    );
  }
}
```

Sign_up_screen.dart :

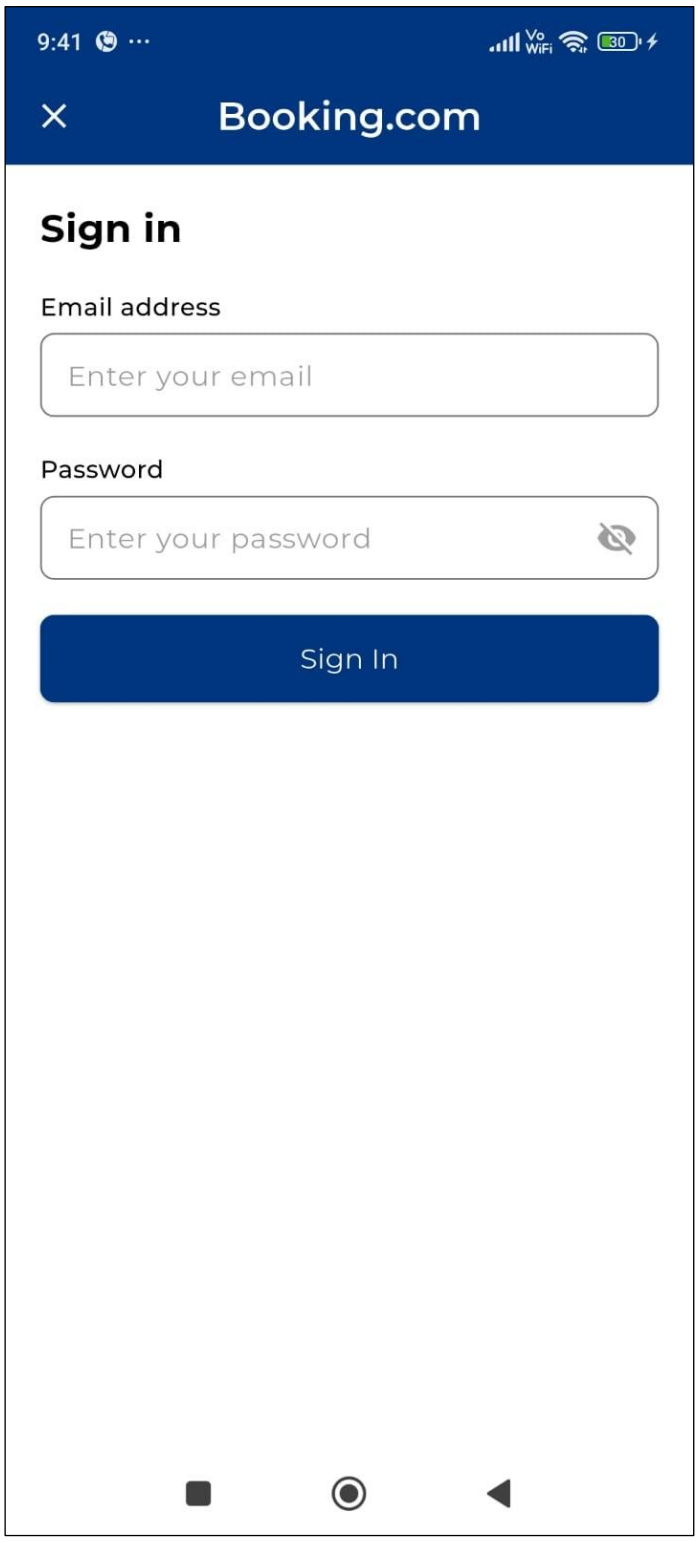
```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'home_screen.dart'; // Import your home
screen

class SignUpScreen extends StatefulWidget {
  @override
  _SignUpScreenState createState() =>
_SignUpScreenState();
}

class _SignUpScreenState extends
State<SignUpScreen> {
  final TextEditingController _emailController =
TextEditingController();
  final TextEditingController _passwordController =
TextEditingController();
  final TextEditingController
_confirmPasswordController =
      TextEditingController();
  bool _isPasswordVisible = false;
  bool _isConfirmPasswordVisible = false;

  void _register() {
    String email = _emailController.text.trim();
    String password =
_passwordController.text.trim();
    String confirmPassword =
_confirmPasswordController.text.trim();

    if (email.isEmpty || !email.contains("@")) {
      ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text("Please enter a
valid email")),
      );
      return;
    }
    if (password.isEmpty || password.length < 6) {
      ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text("Password must
be at least 6 characters")),
      );
      return;
    }
    if (confirmPassword.isEmpty) {
      ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text("Please confirm
your password")),
      );
      return;
    const SizedBox(height: 20),
          SizedBox(
            width: double.infinity,
            height: 50,
            child: ElevatedButton(
              onPressed: _register,
              style: ElevatedButton.styleFrom(
                backgroundColor: const
Color(0xFF003580),
                shape: RoundedRectangleBorder(
                  borderRadius:
BorderRadius.circular(8)),
              ),
              child: Text("Register",
                style: GoogleFonts.montserrat(
                  fontSize: 16,
                  fontWeight: FontWeight.w400,
                  color: Colors.white)),
            ),
          ),
        ],
      ),
    ),
  );
 }

  InputDecoration _inputDecoration(String
hintText) {
    return InputDecoration(
      hintText: hintText,
      hintStyle: GoogleFonts.montserrat(color:
Colors.grey[500]),
      border: OutlineInputBorder(borderRadius:
BorderRadius.circular(8)),
      focusedBorder: OutlineInputBorder(
        borderRadius: BorderRadius.circular(8),
        borderSide: const BorderSide(color:
Colors.blue, width: 2)),
      contentPadding: const
EdgeInsets.symmetric(vertical: 12, horizontal: 15),
    );
  }
EdgeInsets.symmetric(vertical
      contentPadding: const
EdgeInsets.symmetric(vertical: 12, horizontal: 15),
      suffixIcon: IconButton(
        icon: Icon(isVisible ? Icons.visibility :
Icons.visibility_off,
            color: Colors.grey),
        onPressed: toggleVisibility,
      ),
    ),
}
```

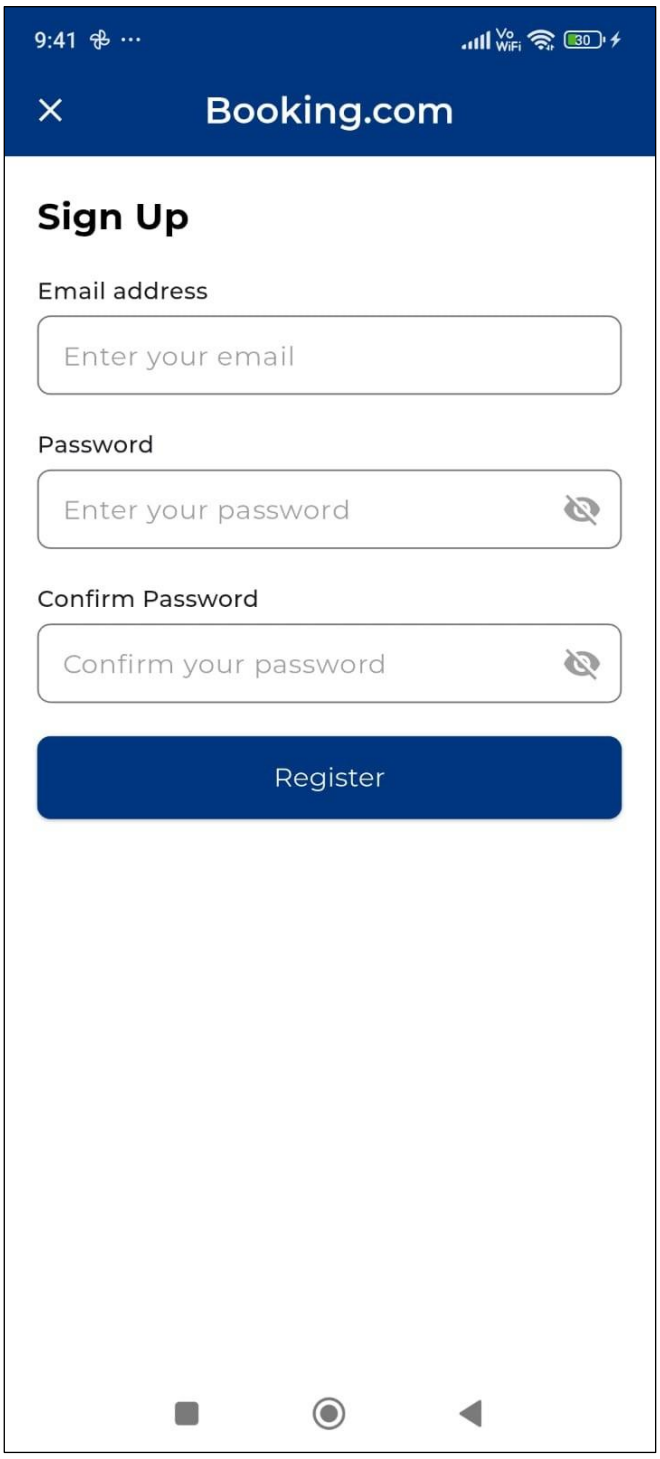
Details_form_screen.dart :

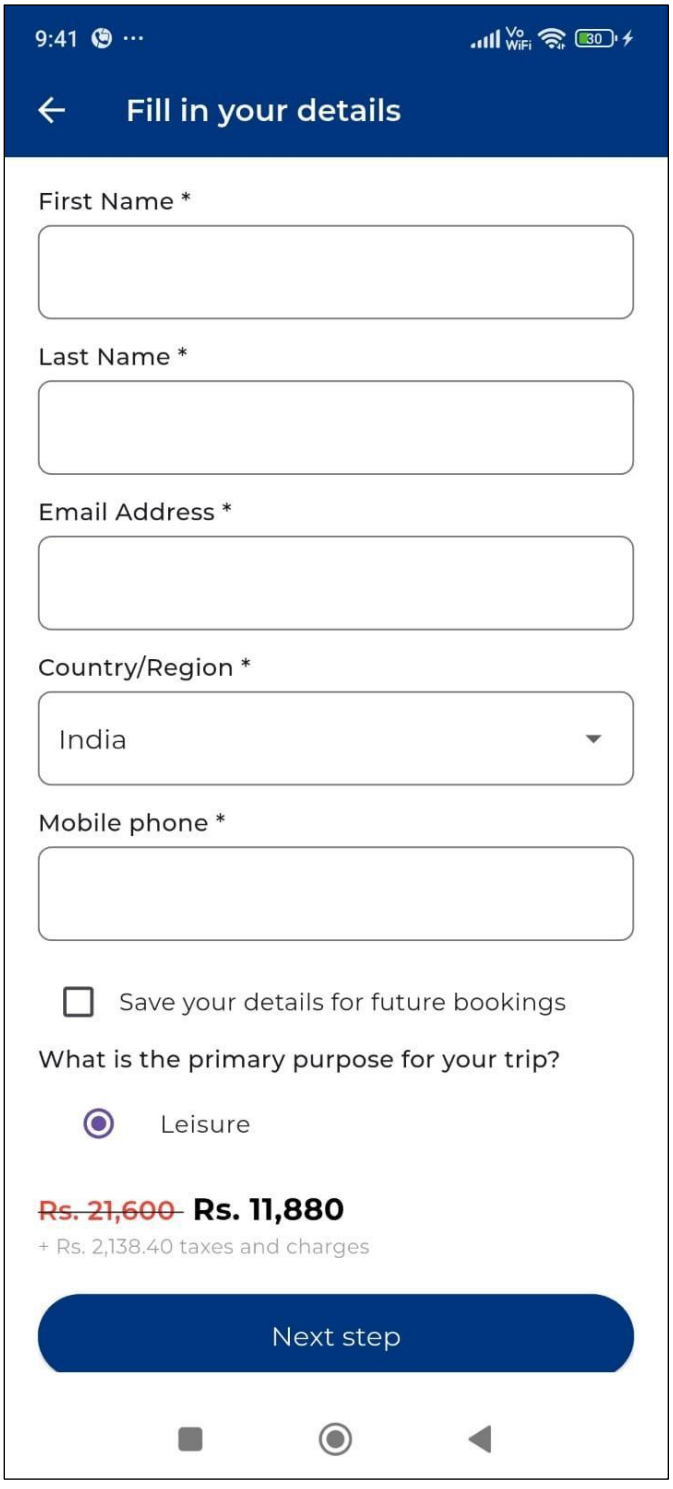
```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';

class DetailsFormScreen extends StatefulWidget {
  @override
  _DetailsFormScreenState createState() =>
_DetailsFormScreenState();
}

class _DetailsFormScreenState extends
State<DetailsFormScreen> {
  String? selectedCountry = "India";
  bool saveDetails = false;
  String tripPurpose = "Leisure";

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white, // Set
background color
      appBar: AppBar(
        backgroundColor: Color(0xFF003580), //
Booking.com blue
        title: Text(
          "Fill in your details",
          style: GoogleFonts.montserrat(
            fontSize: 18,
            fontWeight: FontWeight.w600,
            color: Colors.white,
          ),
        ),
        leading: IconButton(
          icon: Icon(Icons.arrow_back, color:
Colors.white),
          onPressed: () {
            Navigator.pop(context); // Navigate back
          },
        ),
      ),
      body: Padding(
        padding: const EdgeInsets.symmetric(
            horizontal: 20, vertical: 16), // Added
padding
        child: SingleChildScrollView(
          child: Column(
          crossAxisAlignment:
```

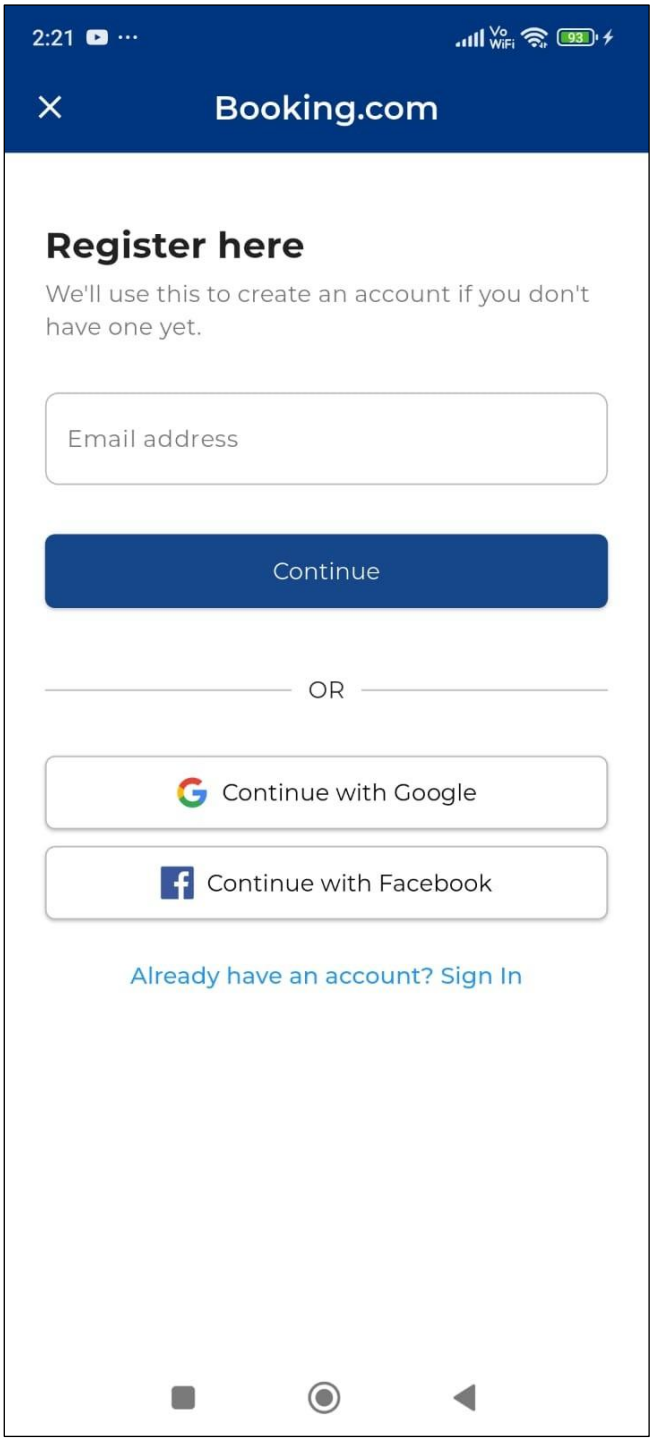
```
Widget buildTextField(String label) {
    return Column(
    crossAxisAlignment:
CrossAxisAlignment.start,
      children: [
        Text(
          label,
          style:
              GoogleFonts.montserrat(fontSize: 14,
fontWeight: FontWeight.w500),
        ),
        SizedBox(height: 4),
        TextFormField(
          decoration: InputDecoration(
            border: OutlineInputBorder(
                borderRadius:
BorderRadius.circular(8.0)), // Added border
          ),
        ),
        SizedBox(height: 12),
      ],
    );
  }

  Widget buildDropdown(String label,
List<String> options) {
    return Column(
      crossAxisAlignment:
CrossAxisAlignment.start,
      children: [
        Text(
          label,
          style:
              GoogleFonts.montserrat(fontSize: 14,
fontWeight: FontWeight.w500),
        ),
        SizedBox(height: 4),
        DropdownButtonFormField(
        value: selectedCountry,
          items: options.map((String country) {
            return DropdownMenuItem(value: country,
child: Text(country));
          }).toList(),
          onChanged: (String? newValue) {
            setState(() {
              selectedCountry = newValue;
            });
          },
          decoration: InputDecoration(
            border: OutlineInputBorder(
                borderRadius:
BorderRadius.circular(8.0)), // Added border
          ),
        ),
        SizedBox(height: 12),
      ],
    );
  }
```

OUTPUT :


9:41
Booking.com
Sign in
Email address
Enter your email
Password
Enter your password
Sign In


9:41
Booking.com
Sign Up
Email address
Enter your email
Password
Enter your password
Confirm Password
Confirm your password
Register


9:41
Fill in your details
First Name *
Last Name *
Email Address *
Country/Region *
India
Mobile phone *
Save your details for future bookings
What is the primary purpose for your trip?
Leisure
Rs. 21,600 Rs. 11,880
+ Rs. 2,138.40 taxes and charges
Next step


2:21
Booking.com
Register here
We'll use this to create an account if you don't have one yet.
Email address
Continue
OR
Continue with Google
Continue with Facebook
Already have an account? Sign In

MAD & PWA Lab

Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	54
Name	Anushka Shahane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

EXPERIMENT NO: - 05

Name : Anushka Shahane

Class : D15A **Roll No :** 54

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

Theory: -

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

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

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

- **Navigation and Routing in Flutter**

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

  1. Using Navigator Widget

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

  - **Pushing a Route**: To navigate to a new screen, use Navigator.push().
  - **Popping a Route**: To go back to the previous screen, use Navigator.pop().

```
ElevatedButton(
  onPressed: () {
  Navigator.push(
```

```
      context,
      MaterialPageRoute(builder: (context) => SecondScreen()),
    );},
);
```

2. Named Routes

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

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

Navigate to the route using Navigator.pushNamed() Navigator.pushNamed(context,

'/second');

Handling Gestures in Flutter

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

Tap Gestures

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

Long Press Gesture

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

Swipe and Drag Gestures

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

Code: - home_screen.dart

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

class HomeScreen extends StatefulWidget {
 @override
 _HomeScreenState createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
 int _selectedIndex = 0;
 String _selectedDate = "Select date";

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

 Future<void> _selectDate(BuildContext context) async {
  DateTimeRange? picked = await showDateRangePicker(
  context: context,
   firstDate: DateTime.now(),
   lastDate: DateTime(2030),
  );

  if (picked != null) {
   setState(() {
    _selectedDate =
      "${DateFormat('E dd MMM').format(picked.start)} –
${DateFormat('E dd MMM').format(picked.end)}";
   });
  }
 }

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   backgroundColor: Colors.grey[100],
   appBar: AppBar(
    backgroundColor: Colors.blue[900],
    title: Text(
     'Booking.com',
     style: TextStyle(
       color: Colors.white, fontWeight: FontWeight.bold,
fontSize: 18),
    ),
    actions: [
     IconButton(
      icon: Icon(Icons.notifications, color: Colors.white),
      onPressed: () {},
     ),
     IconButton(
      icon: Icon(Icons.account_circle, color:
Colors.white),
      onPressed: () {},
     ),
    ],
   ),
   body: SingleChildScrollView(
    child: Padding(
     padding: const EdgeInsets.all(12),
     child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
       // **Category Title**
       Row(
       mainAxisAlignment:
MainAxisAlignment.center,
        children: [
         Icon(Icons.hotel, color: Colors.blue[900],
size: 24),
         SizedBox(width: 8),
         Text(
          "Find your stays",
          style: TextStyle(
           color: Colors.blue[900],
           fontSize: 14,
           fontWeight: FontWeight.bold,
          ),
         ),
        ],
       ),

       SizedBox(height: 15),

       // **Search Section**
       Container(
        padding: EdgeInsets.all(12),
        decoration: BoxDecoration(
         borderRadius: BorderRadius.circular(10),
         border: Border.all(color: Colors.orange,
width: 2),
         color: Colors.white,
        ),
        child: Column(
         children: [
          _buildSearchField(Icons.search, "Enter your
destination"),
          GestureDetector(
           onTap: () => _selectDate(context),
           child: _buildSearchField(
           Icons.calendar_today,
            _selectedDate,
```

```
SizedBox(height: 10),
                ElevatedButton(
                  style: ElevatedButton.styleFrom(
                    backgroundColor: Colors.blue[900],
                    padding:
                        EdgeInsets.symmetric(vertical: 12, horizontal:
80),
                  ),
                  onPressed: () {
                    Navigator.push(
                    context,
                    MaterialPageRoute(
                        builder: (context) => DetailsFormScreen()),
                    );
                  },
                  child: Text("Search",
                      style: TextStyle(color: Colors.white, fontSize:
14)),
                ),
              ],
            ),
          ),

          SizedBox(height: 15),

          // **Discount Section**
          Text(
            "Exclusive Discounts",
            style: TextStyle(fontSize: 14, fontWeight: FontWeight
.bold),
          ),
          SizedBox(height: 8),
          SizedBox(
            height: 110,
            child: ListView(
              scrollDirection: Axis.horizontal,
              children: [
                _buildDiscountCard(
                    "10% discounts on stays", "Enjoy discounts

worldwide"),
                _buildDiscountCard(
                    "10% off rental cars", "Save on select rental cars"),
                _buildDiscountCard(
                    "Exclusive flight deals", "Best fares for
members"),
              ],
            ),
          ),

          SizedBox(height: 15),

          // **Deals Section**
          Text(
            "Deals for the weekend",
            style: TextStyle(fontSize: 14, fontWeight:
 FontWeight.bold),
          ),
          Text(
            "Save on stays for 31 January - 2 February",
            style: TextStyle(fontSize: 12),
          ),



 SizedBox(height: 8),
          SizedBox(
            height: 160,
            child: ListView(
              scrollDirection: Axis.horizontal,
              children: [
                _buildDealCard(
                  "Paris, France",
                  "Starting from \$120/night",
                  "https://thumbs.dreamstime.com/b/paris-
eiffel-tower-river-seine-sunset-france-one-most-iconic-
landmarks-107376702.jpg",
                ),
                _buildDealCard(
                  "New York, USA",
                  "Starting from \$150/night",
                  "https://t3.ftcdn.net/jpg/02/09/70/56/360_
F_209705645_b78HGJI1i1mxqLwMYA7z1m3VvCxgx
JFO.jpg",
                ),
                _buildDealCard(
                  "Tokyo, Japan",
                  "Starting from \$130/night",
                  "https://media.istockphoto.com/id/484915
982/photo/akihabara-
tokyo.jpg?s=612x612&w=0&k=20&c=kbCRYJS5vZu
F4jLB3y4-apNebcCEkWnDbKPpxXdf9Cg=",
                ),
              ],
            ),
          ),
        ],

      ),
    ),

    // **Bottom Navigation Bar**
    bottomNavigationBar: BottomNavigationBar(
    backgroundColor: Colors.white, // Set background
color to white

      currentIndex: _selectedIndex,
      onTap: _onBottomNavTapped,
      selectedItemColor: Colors.blue[900],
      unselectedItemColor: Colors.grey,
      type: BottomNavigationBarType.fixed,
```

```
            icon: Icon(Icons.person, size: 22), label: "My account"),
        ],
      ),
    );
  }
Widget _buildSearchField(IconData icon, String hintText,
      {bool isReadOnly = false}) {
    return Padding(
      padding: EdgeInsets.symmetric(vertical: 4),
      child: TextField(
        readOnly: isReadOnly,
        decoration: InputDecoration(
          prefixIcon: Icon(icon, color: Colors.black54, size: 20),
          hintText: hintText,
          hintStyle: TextStyle(fontSize: 12),
          border: OutlineInputBorder(borderRadius:
BorderRadius.circular(5)),
        ),
      ),
    );
  }

  Widget _buildDiscountCard(String title, String subtitle) {
    return Container(
      width: 150,
      margin: EdgeInsets.only(right: 8),
      padding: EdgeInsets.all(8),
      decoration: BoxDecoration(
      color: Colors.white,
        border: Border.all(color: Colors.blue),
        borderRadius: BorderRadius.circular(8),
      ),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Text(title,
              style: TextStyle(fontWeight: FontWeight.bold,
 fontSize: 12)),
          SizedBox(height: 4),
          Text(subtitle, style: TextStyle(fontSize: 10)),
        ],
      ),
    );

  }

  Widget _buildDealCard(String location, String price,
 String imageUrl) {
    return Container(
      width: 180,
      margin: EdgeInsets.only(right: 8),
padding: EdgeInsets.all(8),
      decoration: BoxDecoration(
      color: Colors.white,
        borderRadius: BorderRadius.circular(8),



  boxShadow: [


  BoxShadow(color: Colors.grey.withOpacity(0.2),
blurRadius: 4)
        ],
      ),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          ClipRRect(



borderRadius: BorderRadius.circular(8),

 child: Image.network(
              imageUrl,
              height: 80,
              width: double.infinity,
              fit: BoxFit.cover,
            ),
          ),
          SizedBox(height: 4),
          Text(location,
              style: TextStyle(fontWeight: FontWeight.bold,
fontSize: 12)),
          SizedBox(height: 4),
          Text(price, style: TextStyle(fontSize: 11, color:
Colors.black54)),
        ],
      ),
    );
  }
}
```

Code : hotel_list.dart

```
import 'package:flutter/material.dart';
import '../models/destination_model.dart';
import '../services/hotel_service.dart';

class HotelListScreen extends StatefulWidget {
  final String destination;
  final String checkIn;
  final String checkOut;
  final String rooms;

  const HotelListScreen({
    Key? key,
    required this.destination,
    required this.checkIn,
    required this.checkOut,
    required this.rooms,
  }) : super(key: key);

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

class _HotelListScreenState extends
State<HotelListScreen> {
  late Future<List<Destination>>
_futureDestinations;
  Set<String> favoriteHotels = {};

  @override
  void initState() {
    super.initState();
    _futureDestinations =
HotelService.fetchDestinations(
      destinationQuery: widget.destination,
      checkIn: widget.checkIn,
      checkOut: widget.checkOut,
      rooms: widget.rooms,
    );
  }

@override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      appBar: AppBar(
        backgroundColor: Colors.blue[900],
        title: Container(
          padding: EdgeInsets.symmetric(vertical: 8,
horizontal: 12),
          decoration: BoxDecoration(
            color: Colors.white,
            borderRadius: BorderRadius.circular(13),
            border: Border.all(color: Colors.orange, width: 6),
          ),
          child: Row(
            children: [
              // Icon(Icons.arrow_back, color: Colors.black),
              SizedBox(
                width: 10,
                height: 10,
              ),
              Text(
                "${widget.destination} · ${widget.checkIn} -
${widget.checkOut}",
                style: TextStyle(
                    color: Colors.black,
                    fontWeight: FontWeight.w100,
                    fontSize: 15),
              ),
            ],
          ),
        ),
      ),
      body: Column(
        children: [
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: Row(
              mainAxisAlignment:
MainAxisAlignment.spaceBetween,
              children: [
                Icon(Icons.swap_vert, color: Colors.black), //
Sort Icon
                Icon(Icons.tune, color: Colors.black), // Filter
Icon
                Icon(Icons.map, color: Colors.black), // Map
Icon
```

```
  ],
        ),
      ),
      Expanded(
        child: FutureBuilder<List<Destination>>(
          future: _futureDestinations,
          builder: (context, snapshot) {
            if (snapshot.connectionState ==
ConnectionState.waiting) {
              return Center(child:
CircularProgressIndicator());
            } else if (snapshot.hasError) {
              return Center(child: Text("Error:
${snapshot.error}"));
            } else if (!snapshot.hasData ||
snapshot.data!.isEmpty) {
              return Center(child: Text("No hotels
found"));
            } else {
              final destinations = snapshot.data!;
              return ListView.builder(
                itemCount: destinations.length,
                itemBuilder: (context, index) {
                  final dest = destinations[index];
                  return Card(
                    margin:
                        EdgeInsets.symmetric(horizontal:
12, vertical: 6),
                    shape: RoundedRectangleBorder(
                      borderRadius:
BorderRadius.circular(15),
                    ),
                    elevation: 5,
                    child: Container(
                      decoration: BoxDecoration(
                        color: Colors.white,
                        borderRadius:
BorderRadius.circular(15),
                        boxShadow: [
                          BoxShadow(
                            color:
Colors.grey.withOpacity(0.3),
                            spreadRadius: 2,
                            blurRadius: 5,
                            offset: Offset(0, 3),
                          ),
                        ],
                      ),
```

```
const SizedBox(height: 5),
            Text(
              "We'll use this to create an account if you
don't have one yet.",
              style: GoogleFonts.montserrat(
                fontSize: 14,
                color: Colors.grey[600],
              ),
            ),
            const SizedBox(height: 30),

            // Email Entry Box
            Container(
            padding: const
EdgeInsets.symmetric(horizontal: 12),
              decoration: BoxDecoration(
                borderRadius: BorderRadius.circular(8),
                border: Border.all(color:
Colors.grey.shade400),
              ),
              child: TextField(
                decoration: InputDecoration(
                  labelText: "Enter your name",
                  labelStyle: GoogleFonts.montserrat(
                      fontSize: 14, color: Colors.grey[600]),
                  border: InputBorder.none,
                ),
              ),
            ),
            const SizedBox(height: 30),

            // Continue Button
            CustomButton(
            text: "Continue",
              onTap: () => Navigator.pushNamed(context,
Routes.signUp),
              backgroundColor: const Color(0xFF003580),
              textColor: Colors.white,
            ),
            const SizedBox(height: 40),
```

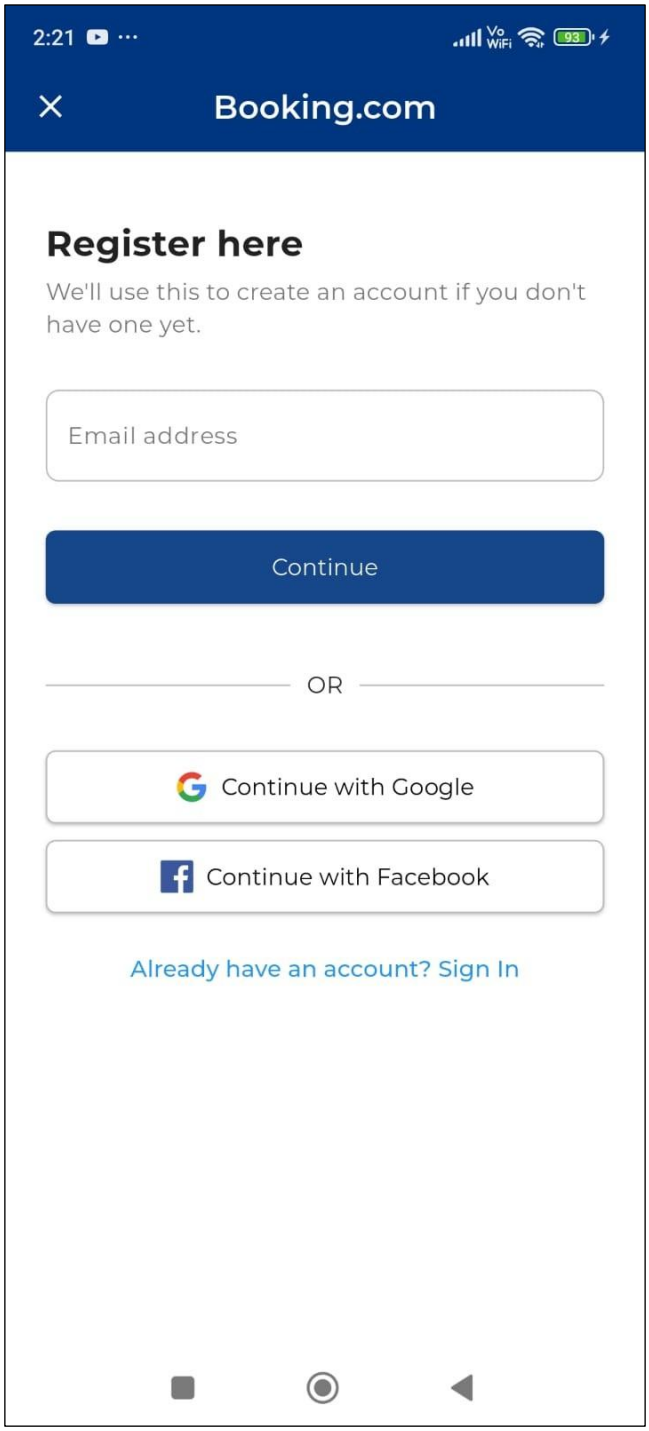
```
            Row(
              children: [
                Expanded(child: Divider(color: Colors.grey.shade400)),
                Padding(
                  padding: const EdgeInsets.symmetric(horizontal: 10),
                  child: Text("OR",
                      style: GoogleFonts.montserrat(
                          fontSize: 14, color: Colors.black87)),
                ),
                Expanded(child: Divider(color: Colors.grey.shade400)),
              ],
            ),
            const SizedBox(height: 30),

            // Continue with Google
            CustomButton(
              text: "Continue with Google",
              onTap: () {},
              isOutlined: true,
              textColor: Colors.black,
              borderColor: Colors.grey.shade400,
              iconPath: "assets/images/google.png", // Add Google icon
            ),
            const SizedBox(height: 10),

            // Continue with Facebook
            CustomButton(
              text: "Continue with Facebook",
              iconPath: "assets/images/facebook.png", // Add Facebook icon
            ),
            const SizedBox(height: 10),
            Center(
              child: TextButton(
                onPressed: () => Navigator.pushNamed(context, Routes.signIn),
                child: Text(
                  "Already have an account? Sign In",
                  style: GoogleFonts.montserrat(
                  fontSize: 14,
                    fontWeight: FontWeight.w500,
                    color: Colors.blue,
),
),
}
}
```

Code: hotel_service.dart

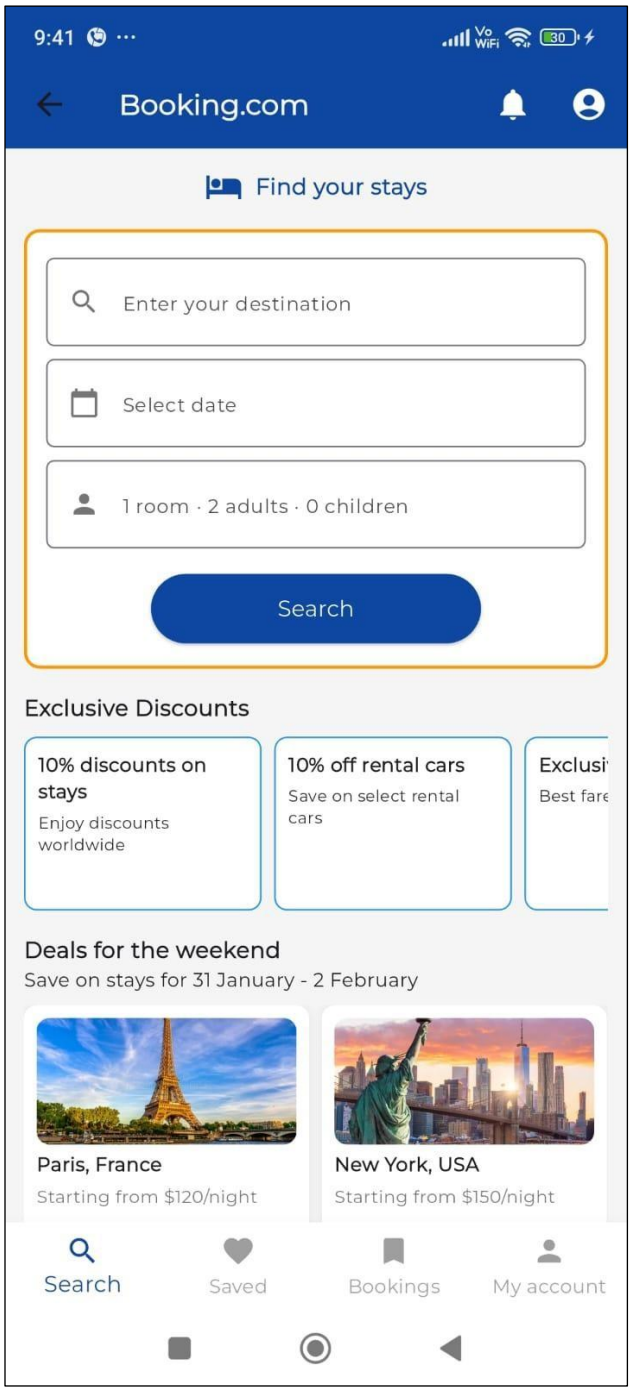
```
import 'dart:convert';
import 'package:http/http.dart' as http;
import '../models/destination_model.dart';

class HotelService {
  // Updated endpoint using your curl command
  static const String _endpoint =
      "https://booking-com15.p.rapidapi.com/api/v1/hotels/searchDestination";

  static Future<List<Destination>> fetchDestinations({
    required String destinationQuery,
    required String checkIn, // include if supported by your API
    required String checkOut, // include if supported by your API
    required String rooms, // include if supported by your API
  }) async {
    // Build the URL with query parameters.
    // The curl shows the API accepts a "query" parameter.
    // If your API supports additional parameters, they can be appended here.
    final Uri url = Uri.parse(_endpoint).replace(queryParameters: {
      'query': destinationQuery,
      'checkin': checkIn,
      'checkout': checkOut,
      'rooms': rooms,
    });

    // Headers required by the API. (Remember to secure your API key in production.)
    final headers = {
      'x-rapidapi-host': 'booking-com15.p.rapidapi.com',
      'x-rapidapi-key': '94e5999c47msh7a64d356faaa760p1b75eajsn392f227c4818',
      'Content-Type': 'application/json',
    };

    final response = await http.get(url, headers: headers);

    if (response.statusCode == 200) {
      final Map<String, dynamic> jsonResponse = json.decode(response.body);
      // Expecting a response similar to:
      // {
      //   "status": true,
      //   "message": "Success",
      //   "timestamp": 1739725005441,
      //   "data": [ { ... }, { ... }, ... ]
      // }
      if (jsonResponse['status'] == true) {
        final List<dynamic> data = jsonResponse['data'];
        return data.map((item) => Destination.fromJson(item)).toList();
      } else {
        throw Exception("API error: ${jsonResponse['message']}");
      }
    } else {
      throw Exception(
          "Failed to load destinations. HTTP Status: ${response.statusCode}");
    }
  }
}
```

OUTPUT :

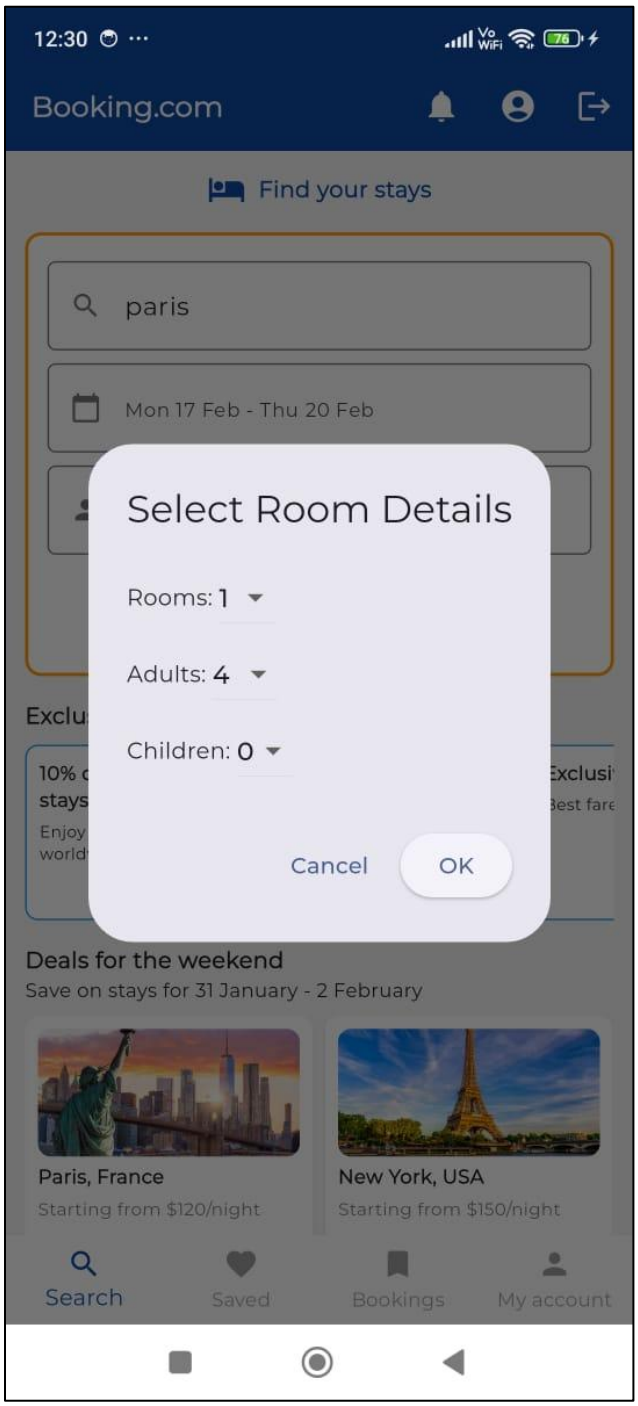
From the welcome page once registration and login is done it navigates to home page



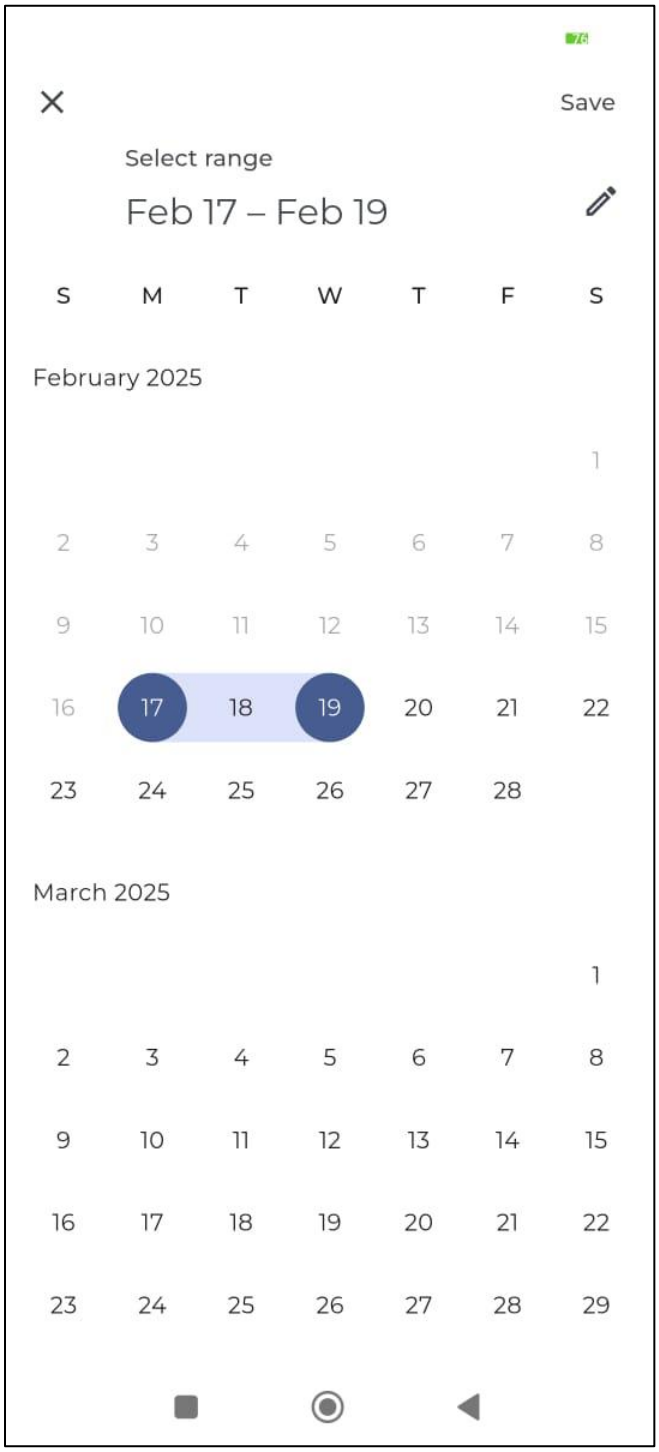
From the home page the user can add destination , date and rooms



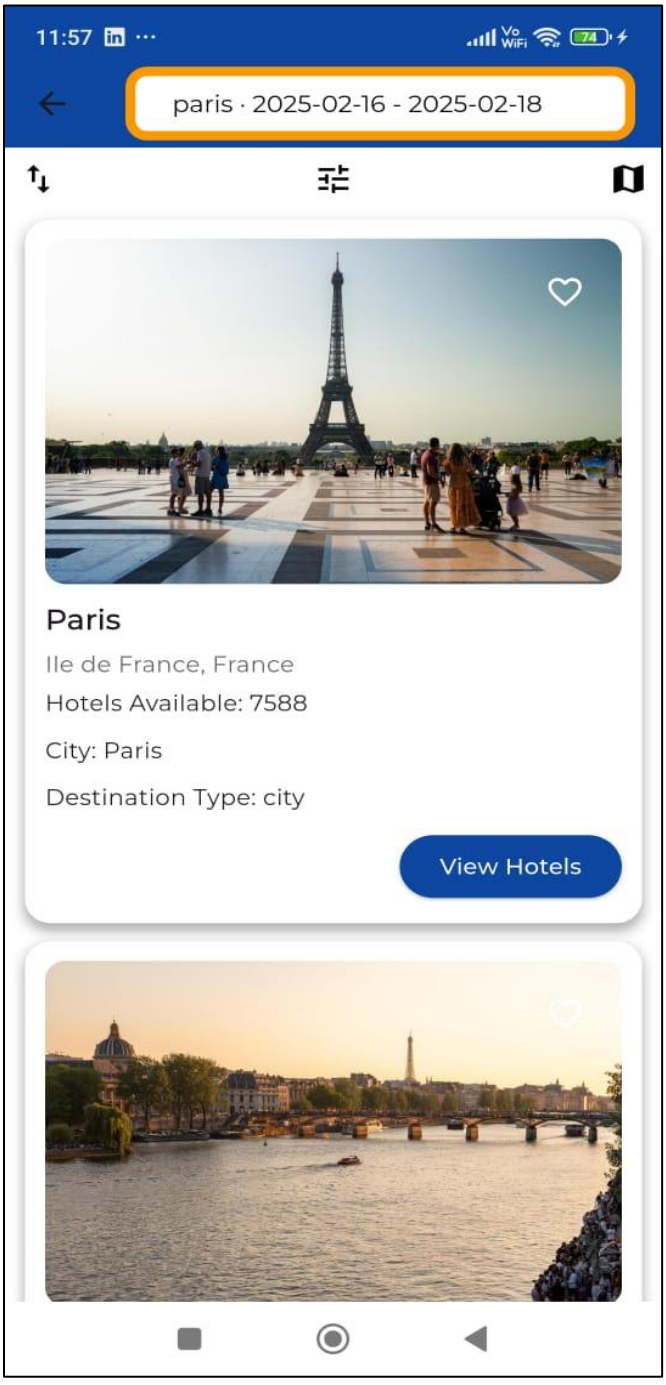
Gesture of rooms is enabled



Gesture of calender is enabled



Then after clicking on search the user navigates to hotels page



MAD & PWA Lab
Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	54
Name	Anushka Shahane
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:	

EXPERIMENT NO: - 06

Name : Anushka Shahane

Class : D15A Roll:No : 54

AIM: To connect flutter UI with firebase database.

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.

Setting Up Firebase in Flutter :

To connect a Flutter app with Firebase, the following steps are typically followed:

1. **Creating a Firebase Project:** To start using Firebase with Flutter, the first step is to create a Firebase project in the Firebase Console. Once the project is created, developers can associate their Flutter app with the Firebase project by following the platform-specific instructions for Android or iOS. This usually involves configuring API keys, downloading configuration files, and adding them to the Flutter project.

2. **Integrating Firebase SDK in Flutter**: After the Firebase project is set up, developers need to integrate Firebase's SDK into the Flutter app. This involves adding the necessary dependencies to the Flutter project's pubspec.yaml file. For Firebase's Realtime Database, the package firebase_database is used. Additionally, Firebase's core SDK (firebase_core) must also be included to initialize Firebase services.

3. **Initializing Firebase**: Before any Firebase functionality can be used, it is essential to initialize Firebase in the Flutter app. This is done by calling Firebase.initializeApp() in the main entry point of the app (usually in the main.dart file). Firebase needs to be initialized before interacting with any Firebase services, such as the Realtime Database, Cloud Firestore, or Authentication.

Connecting Firebase to a Flutter app enables developers to create robust, scalable, and real-time applications with ease. Firebase's Realtime Database offers a powerful, cloud-based solution for managing data in real-time, while Firebase Authentication ensures secure access control. By integrating Firebase with Flutter, developers can take advantage of real-time data synchronization, offline support, and a wide range of other Firebase features, allowing them to build feature-rich apps that meet modern user expectations.

Code: -authentication.dart

```
import
'package:cloud_firestore/clou
d_firestore.dart';
import
'package:firebase_auth/firebas
e_auth.dart';

class AuthMethod {
      final      FirebaseFirestore
_firestore                    =
FirebaseFirestore.instance;
  final FirebaseAuth _auth =
FirebaseAuth.instance;

  // Sign Up User using email
and password only.
                 Future<String>
signUpUser({
   required String email,
   required String password,
  }) async {
    String res = "Some error
occurred";
   try {
     // Check that both email
and password are provided.
     if (email.isNotEmpty &&
password.isNotEmpty) {
        // Register user with
Firebase Auth.
        UserCredential cred =
await
_auth.createUserWithEmailA
ndPassword(
        email: email,
        password: password,
      );
        // Add user details to
Firestore.
      await
_firestore.collection("users").
doc(cred.user!.uid).set({
        'uid': cred.user!.uid,
        'email': email,
       });
       res = "success";
      } else {
       res = "Please enter all the
fields";
      }
    } catch (err) {
     res = err.toString();
    }
    return res;
  }

   // Log In User using email
and password.
  Future<String> loginUser({
    required String email,
    required String password,
  }) async {
     String res = "Some error
occurred";
    try

  // Sign out the current user.
     Future<void>    signOut()
async {
    await _auth.signOut();
  }
}

Colors.grey[100],      appBar:
AppBar(
 backgroundColor:
 Colors.blue[900], title: Text(
 'Booking.co
  m',      style:
  TextStyle(
      color:       Colors.white,       fontWeight:
FontWeight.bold, fontSize: 18),
    ),
```

```
      actions:          [
       IconButton(
        icon:       Icon(Icons.notifications,       color:
        Colors.white), onPressed: () {},
       ),

         )         ,
worldwide"),
              _buildDiscountCard(
                "10% off rental cars", "Save on select
                rental cars"),
              _buildDiscountCard(
                "Exclusive flight deals", "Best fares for
members

          SizedBox(height: 15),

          // **Deals Section**
          Text(
           "Deals for the weekend",



       Text(subtitle, style: TextStyle(fontSize: 10)),
      ],
    ),
  );

  // Log In User using email and password.
   Future<String> loginUser({
    required String email,
    required String password,
   }) async {
    String res = "Some error occurred";
    try {
     // Ensure both fields are filled.
     if (email.isNotEmpty && password.isNotEmpt
  y) {
       await _auth.signInWithEmailAndPassword(
        email: email,
        password: password,
       );
```

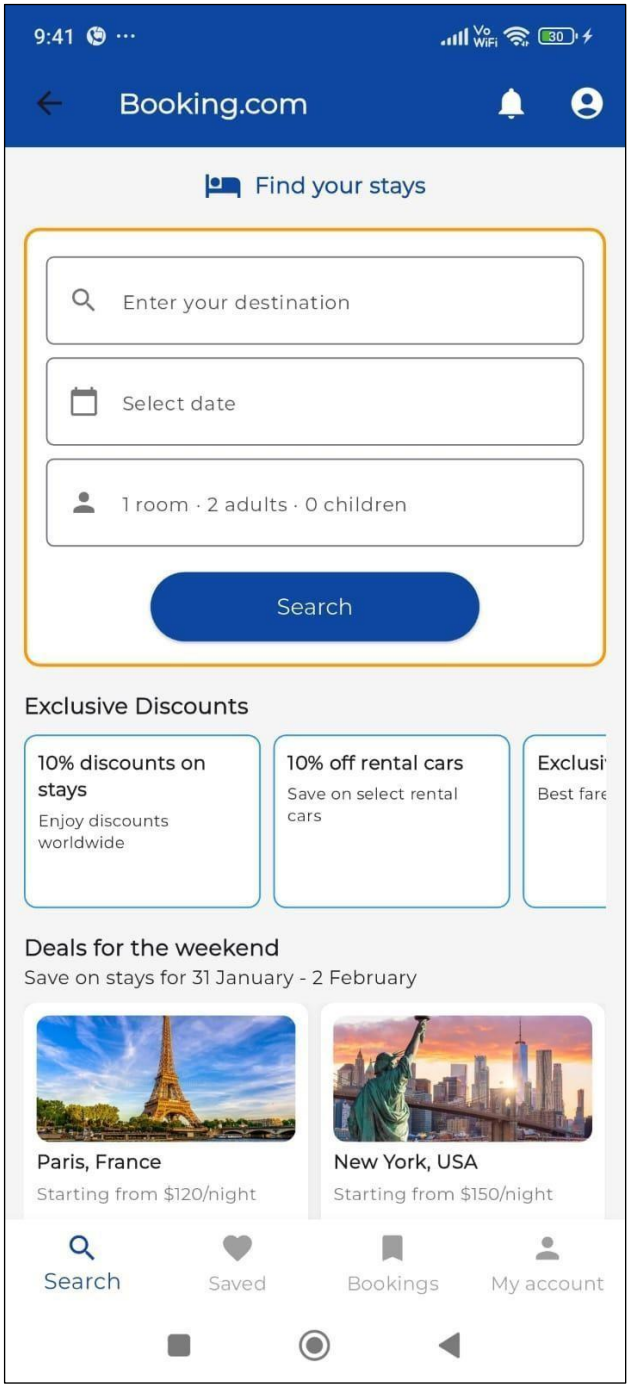
Hotel_model.dart

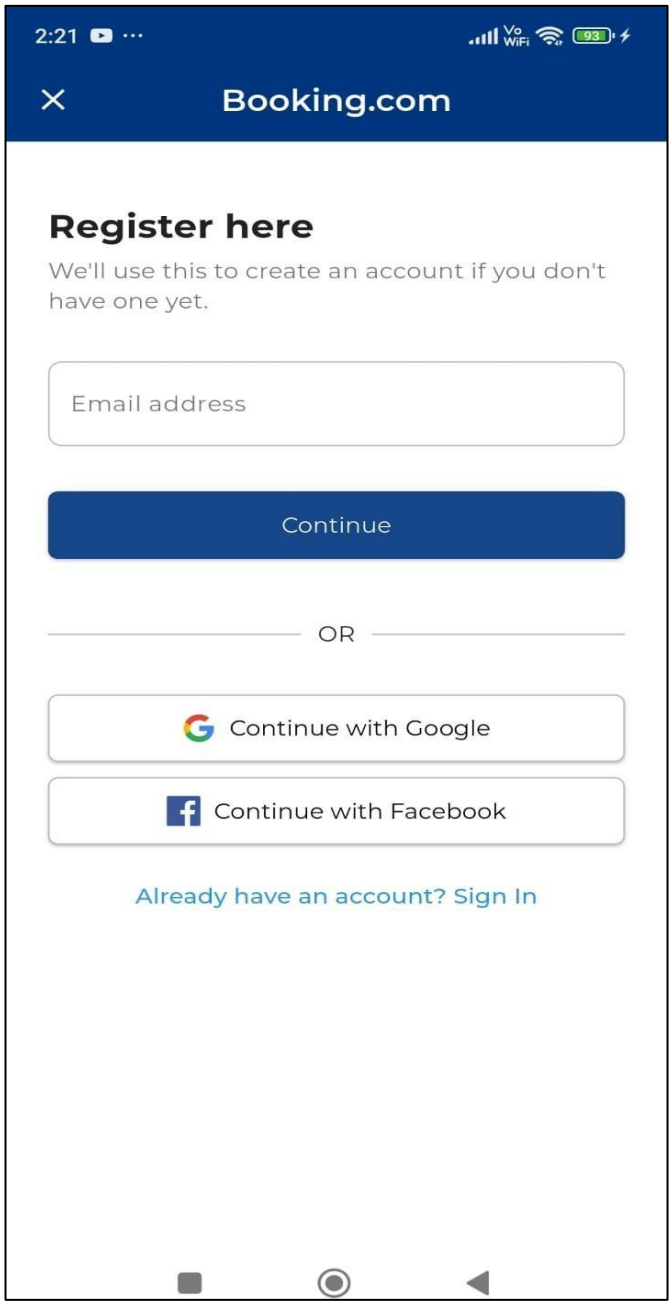
```
// lib/models/hotel_model.dart
class Hotel {
  final String name;
  final String address;
  final String price;
  final String imageUrl;

  Hotel({
    required this.name,
    required this.address,
    required this.price,
    required this.imageUrl,
  });

  factory Hotel.fromJson(Map<String, dynamic> json) {
    return Hotel(
      name: json['hotel_name'] ?? 'No Name',
      address: json['address'] ?? 'No Address',
      price: json['price']?.toString() ?? 'N/A',
      imageUrl: json['image_url'] ?? '',
    );
  }
}
```

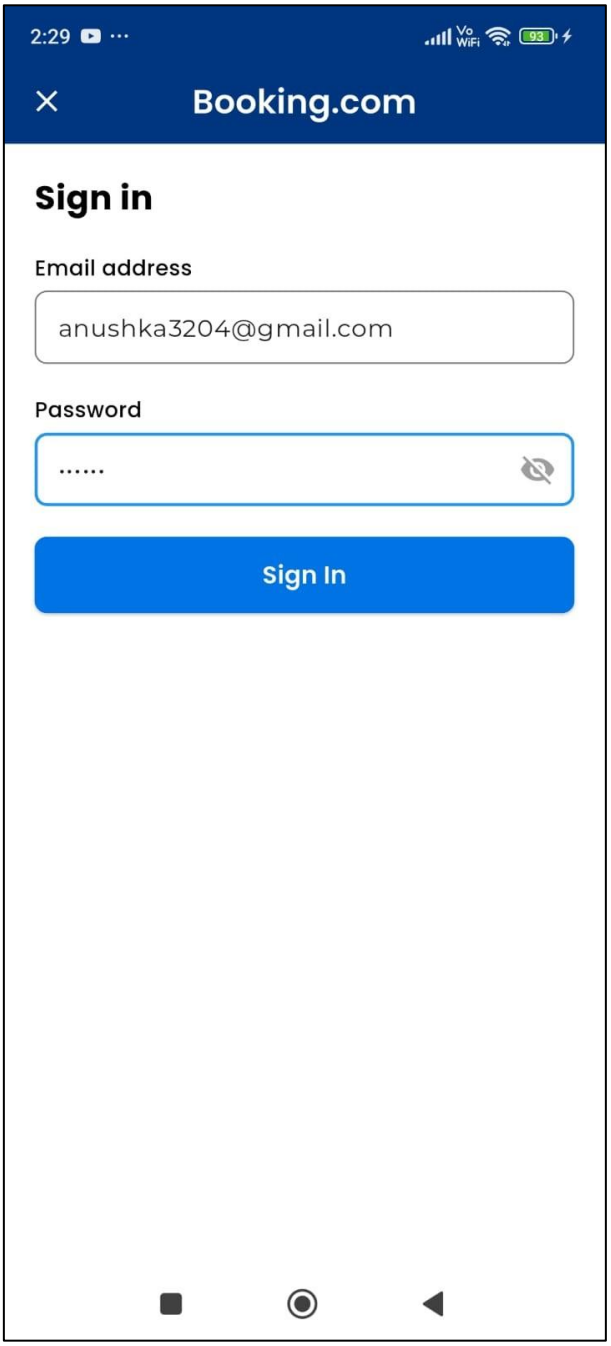
Hotel_service.dart

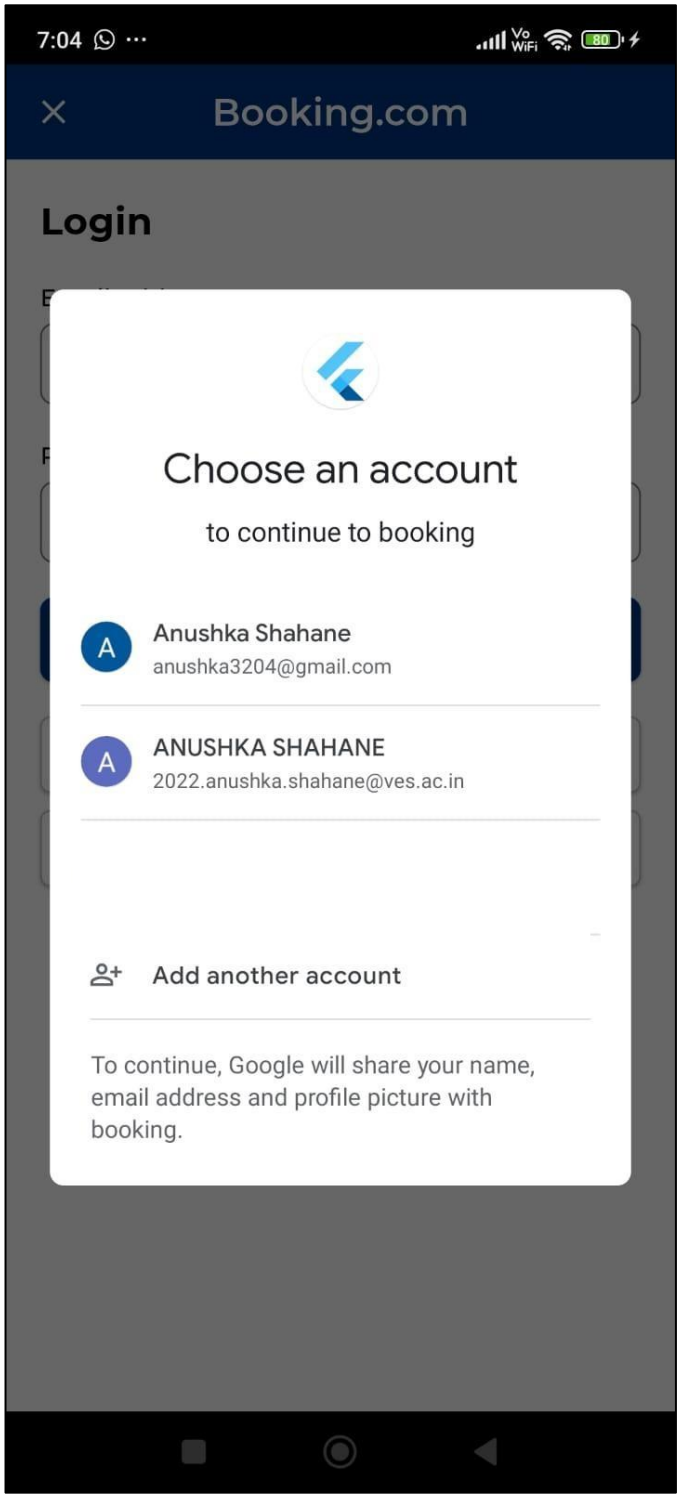
```
// lib/services/hotel_service.dart
import 'dart:convert';
import 'package:http/http.dart' as http;
import '../models/destination_model.dart';

class HotelService {
  // Updated endpoint using your curl command
  static const String _endpoint =
      "https://booking-com15.p.rapidapi.com/api/v1/hotels/searchDestination";

  static Future<List<Destination>> fetchDestinations({
    required String destinationQuery,
    required String checkIn, // include if supported by your API
    required String checkOut, // include if supported by your API
    required String rooms, // include if supported by your API
  }) async {
```

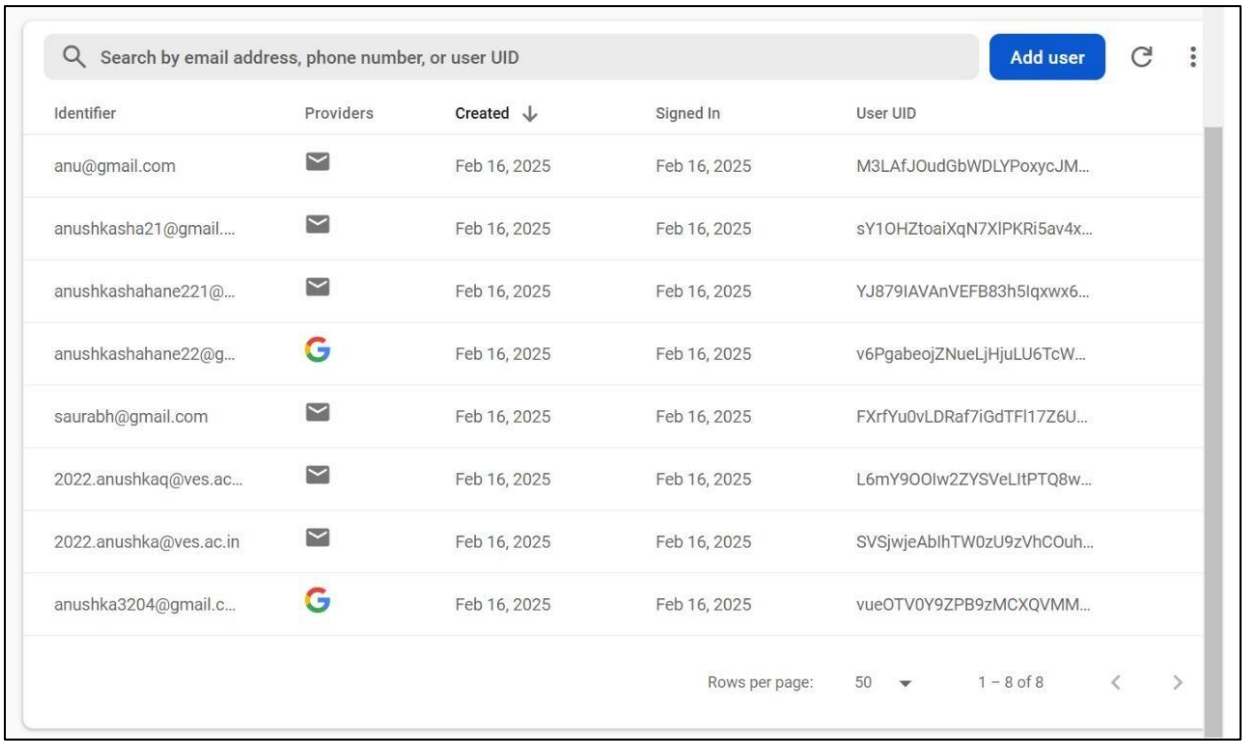
```
    final Uri url = Uri.parse(_endpoint).replace(queryParameters: {
      'query': destinationQuery,
      'checkin': checkIn,
      'checkout': checkOut,
      'rooms': rooms,
    });



  // Headers required by the API. (Remember to secure your API key in production.)
    final headers = {
      'x-rapidapi-host': 'booking-com15.p.rapidapi.com',
      'x-rapidapi-key': 'your-api-key',
      'Content-Type': 'application/json',
    };

    final response = await http.get(url, headers: headers);

    if (response.statusCode == 200) {
      f

inal Map<String, dynamic> jsonResponse = json.decode(response.body);
      if (jsonResponse['status'] == true) {



        final List<dynamic> data = jsonResponse['data'];
        return data.map((item) => Destination.fromJson(item)).toList();
      } else {
        throw Exception("API error: ${jsonResponse['message']}");
      }
    } else {
      throw Exception(
          "Failed to load destinations. HTTP Status: ${response.statusCode}");
    }
```

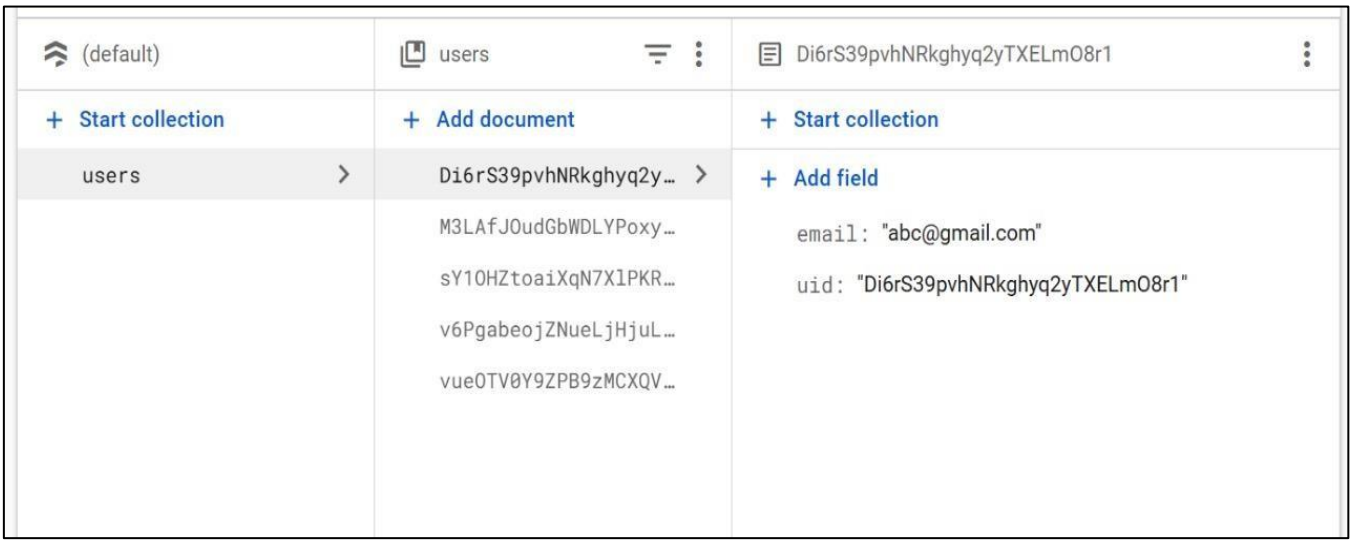

9:41
Booking.com
Find your stays
Enter your destination
Select date
1 room · 2 adults · 0 children
Search
Exclusive Discounts
10% discounts on stays
Enjoy discounts worldwide
10% off rental cars
Save on select rental cars
Deals for the weekend
Save on stays for 31 January - 2 February
Paris, France
Starting from $120/night
New York, USA
Starting from $150/night
Search
Saved
Bookings
My account

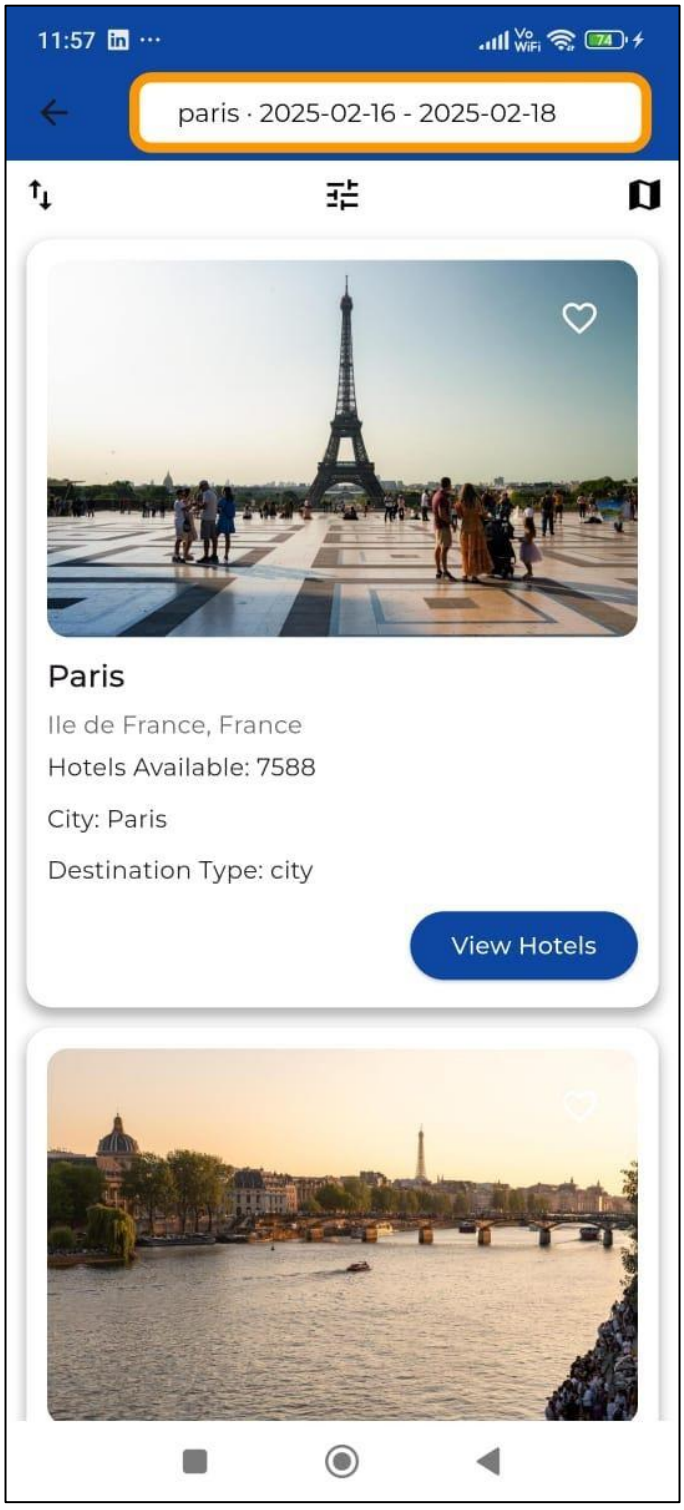

2:21
Booking.com
Register here
We'll use this to create an account if you don't have one yet.
Email address
Continue
OR
Continue with Google
Continue with Facebook
Already have an account? Sign In


2:29
Booking.com
Sign in
Email address
anushka3204@gmail.com
Password
.....
Sign In


7:04
Booking.com
Login
Choose an account
to continue to booking
Anushka Shahane
anushka3204@gmail.com
ANUSHKA SHAHANE
2022.anushka.shahane@ves.ac.in
Add another account
To continue, Google will share your name, email address and profile picture with booking.


Search by email address, phone number, or user UID
Add user
Identifier
Providers
Created
Signed In
User UID
anu@gmail.com
Feb 16, 2025
Feb 16, 2025
M3LAfJOudGbWDLYPoxycJM...
anushkasha21@gmail....
Feb 16, 2025
Feb 16, 2025
sY1OHZtoaiXqN7XlPKRi5av4x...
anushkashahane221@...
Feb 16, 2025
Feb 16, 2025
YJ879IAVAnVEFB83h5lqxwx6...
anushkashahane22@g...
Feb 16, 2025
Feb 16, 2025
v6PgabeojZNueLjHjuLU6TcW...
saurabh@gmail.com
Feb 16, 2025
Feb 16, 2025
FXrfYu0vLDRaf7iGdTFl17Z6U...
2022.anushkaq@ves.ac...
Feb 16, 2025
Feb 16, 2025
L6mY9OOlw2ZYSVeLltPTQ8w...
2022.anushka@ves.ac.in
Feb 16, 2025
Feb 16, 2025
SVSjwjeAblhTW0zU9zVhCOuh...
anushka3204@gmail.c...
Feb 16, 2025
Feb 16, 2025
vueOTV0Y9ZPB9zMCXQVMM...
Rows per page:
50
1 – 8 of 8


(default)
users
Di6rS39pvhNRkghyq2yTXELmO8r1
+ Start collection
+ Add document
+ Start collection
users
Di6rS39pvhNRkghyq2y…
M3LAfJOudGbWDLYPoxy…
sY1OHZtoaiXqN7XlPKR…
v6PgabeojZNueLjHjuL…
vueOTV0Y9ZPB9zMCXQV…
+ Add field
email: "abc@gmail.com"
uid: "Di6rS39pvhNRkghyq2yTXELmO8r1"


11:57
paris · 2025-02-16 - 2025-02-18
Paris
Ile de France, France
Hotels Available: 7588
City: Paris
Destination Type: city
View Hotels

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.	54
Name	Anushka Shahane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	

Experiment No. 7

Anushka Shahane : D15A - 54

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

Online Reference:
https://developer.mozilla.org/en-US/docs/Web/Manifest
https://www.geeksforgeeks.org/making-a-simple-pwa-under-5-minutes/
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.).

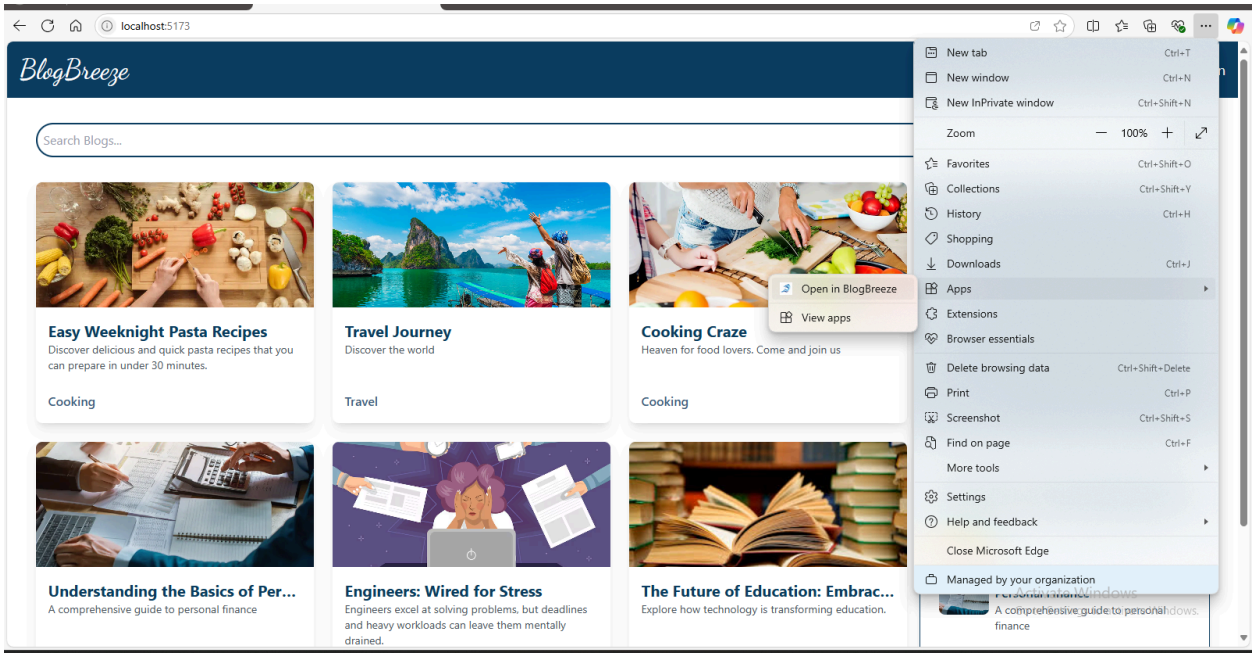
Code:-
manifest.json:-

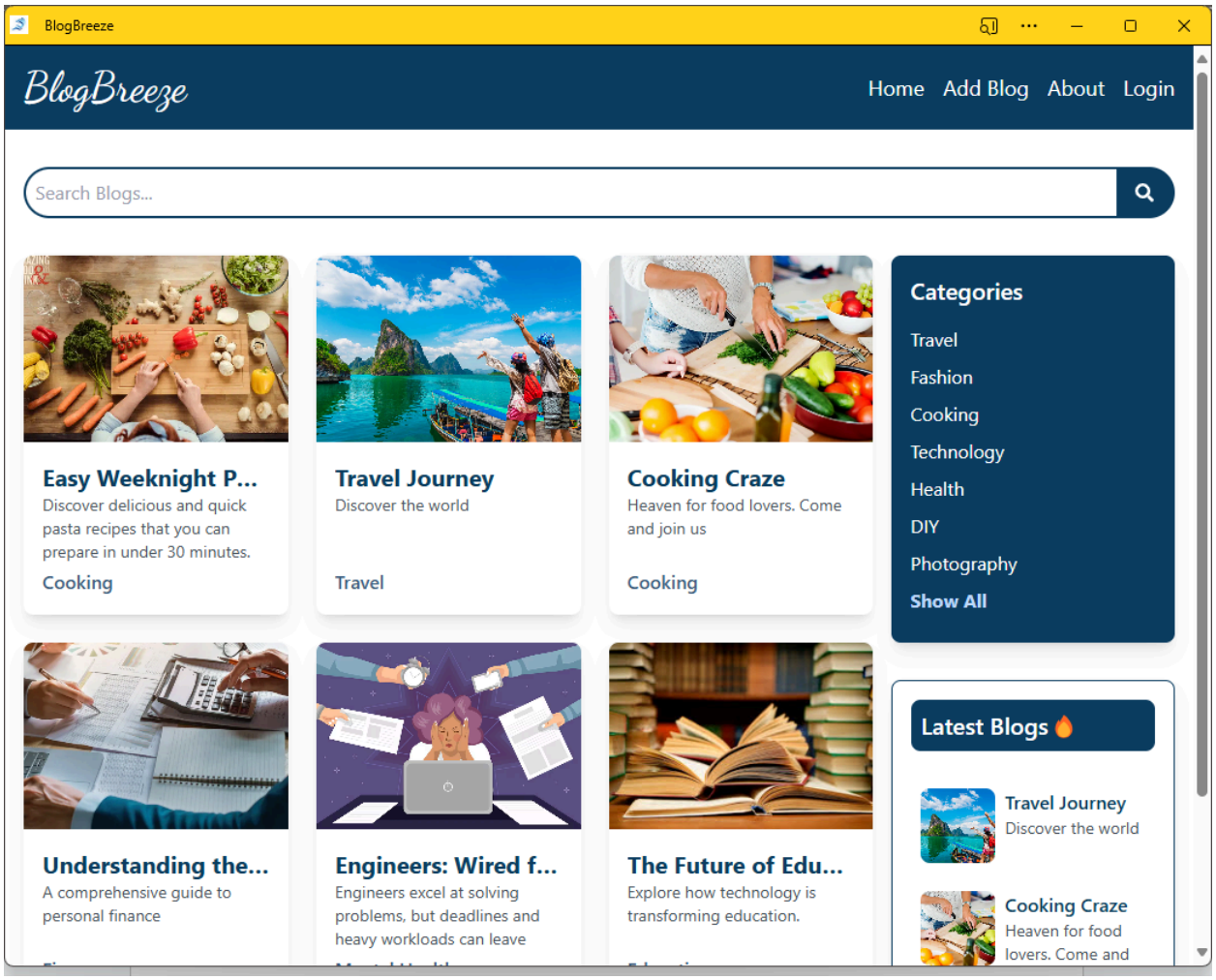
```
{
   "name": "BlogBreeze",
   "short_name": "BlogBreeze",
   "start_url": "ani.html",
   "scope": "./",
   "icons": [
      {
       "src": "/src/assets/react.svg",
       "sizes": "192x192",
       "type": "image/svg"
      }
    ],
    "theme_color": "#ffd31d",
    "background_color": "#333",
   "display": "standalone"
}
```

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

```html
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="manifest" href="manifest.json">
    <link rel="icon" type="image/svg+xml" href="./public/icon.png" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>BlogBreeze</title>

    <!-- Google Fonts Dancing Script -->
    <link href="https://fonts.googleapis.com/css2?family=Dancing+Script:wght@400;500;600;700&display=swap" rel="stylesheet">
<!-- Add this in the <head> section of your index.html -->
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css" />

  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
```

Output:-





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.

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.	54
Name	Anushka Shahane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

EXPERIMENT 8

Name	**Roll No.**
Anushka Shahane	**54 - D15A**

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

Theory:

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

Things to note about Service Worker:

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

What can we do with Service Workers?

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

response or when the page requests an HTML file, you can send a png file as a response. You can also send a true response too.

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

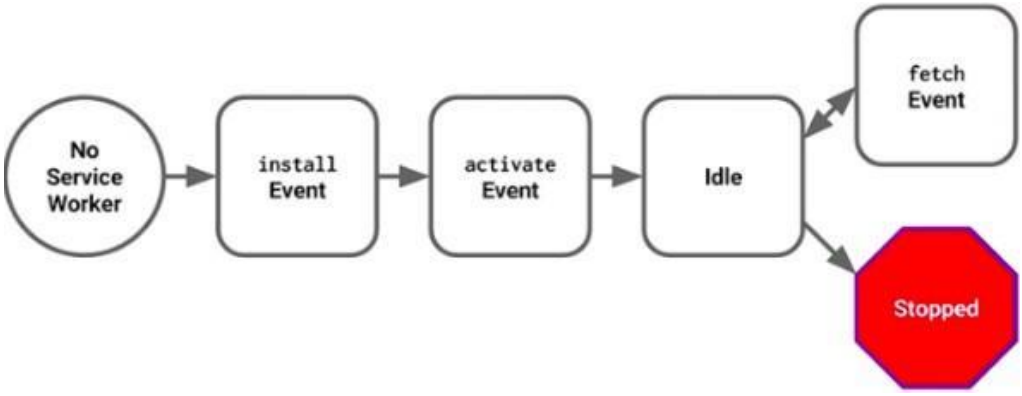
- You can manage **Push Notifications**
  You can manage push notifications with Service Worker and show any information message to the user.
- You can **Continue**
  Although Internet connection is broken, you can start any process with Background Sync of Service Worker.

What can't we do with Service Workers?

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

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

Service Worker Cycle



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

- Registration

- Installation
- Activation

Registration

To install a service worker, you need to register it in your main JavaScript code

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

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

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

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

example: main.js

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

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

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

main.js

```
// Service Worker Script

const CACHE_NAME = 'blogbreeze-v1'; // Cache name to identify the version of cached content
const urlsToCache = [
  '/', // Home page
  'index.html', // Main HTML file
  'ani.html', // Any additional pages you want to cache
  '/src/assets/react.svg', // App icon
  '/public/icon.png', // Favicon
  '/src/main.jsx', // Main JS file for app functionality
//   'https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css' // FontAwesome for icons
];

// Install Service Worker
self.addEventListener('install', (event) => {
  console.log('Service Worker: Installed');
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then((cache) => {
        return cache.addAll(urlsToCache);
      })
  );
});

// Activate Service Worker
self.addEventListener('activate', (event) => {
  console.log('Service Worker: Activated');
  // Remove old caches if there are any
  event.waitUntil(
    caches.keys().then((cacheNames) => {
      return Promise.all(
        cacheNames.map((cacheName) => {
          if (cacheName !== CACHE_NAME) {
            return caches.delete(cacheName);
          }
        })
      );
    })
  );
});

// Fetch event: Intercept network requests and serve cached content
self.addEventListener('fetch', (event) => {
  console.log('Service Worker: Fetching', event.request.url);
  event.respondWith(
    caches.match(event.request)
```

```
      .then((cachedResponse) => {
        // Return cached content if found, otherwise fetch from network
        return cachedResponse || fetch(event.request);
      })
  );
})

// Service Worker Script

const CACHE_NAME = 'blogbreeze-v1'; // Cache name to identify the version of cached content
const urlsToCache = [
  '/', // Home page
  'index.html', // Main HTML file
  'ani.html', // Any additional pages you want to cache
  '/src/assets/react.svg', // App icon
  '/public/icon.png', // Favicon
  '/src/main.jsx', // Main JS file for app functionality
//   'https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css' // FontAwesome for icons
];

// Install Service Worker
self.addEventListener('install', (event) => {
  console.log('Service Worker: Installed');
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then((cache) => {
        return cache.addAll(urlsToCache);
      })
  );
});

// Activate Service Worker
self.addEventListener('activate', (event) => {
  console.log('Service Worker: Activated');
  // Remove old caches if there are any
  event.waitUntil(
    caches.keys().then((cacheNames) => {
      return Promise.all(
        cacheNames.map((cacheName) => {
          if (cacheName !== CACHE_NAME) {
            return caches.delete(cacheName);
          }
        })
      );
    })
  );
```

```
});

// Fetch event: Intercept network requests and serve cached content
self.addEventListener('fetch', (event) => {
  console.log('Service Worker: Fetching', event.request.url);
  event.respondWith(
    caches.match(event.request)
      .then((cachedResponse) => {
        // Return cached content if found, otherwise fetch from network
        return cachedResponse || fetch(event.request);
      })
  );
});
```

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

Installation

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

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

service-worker.js

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

Activation

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

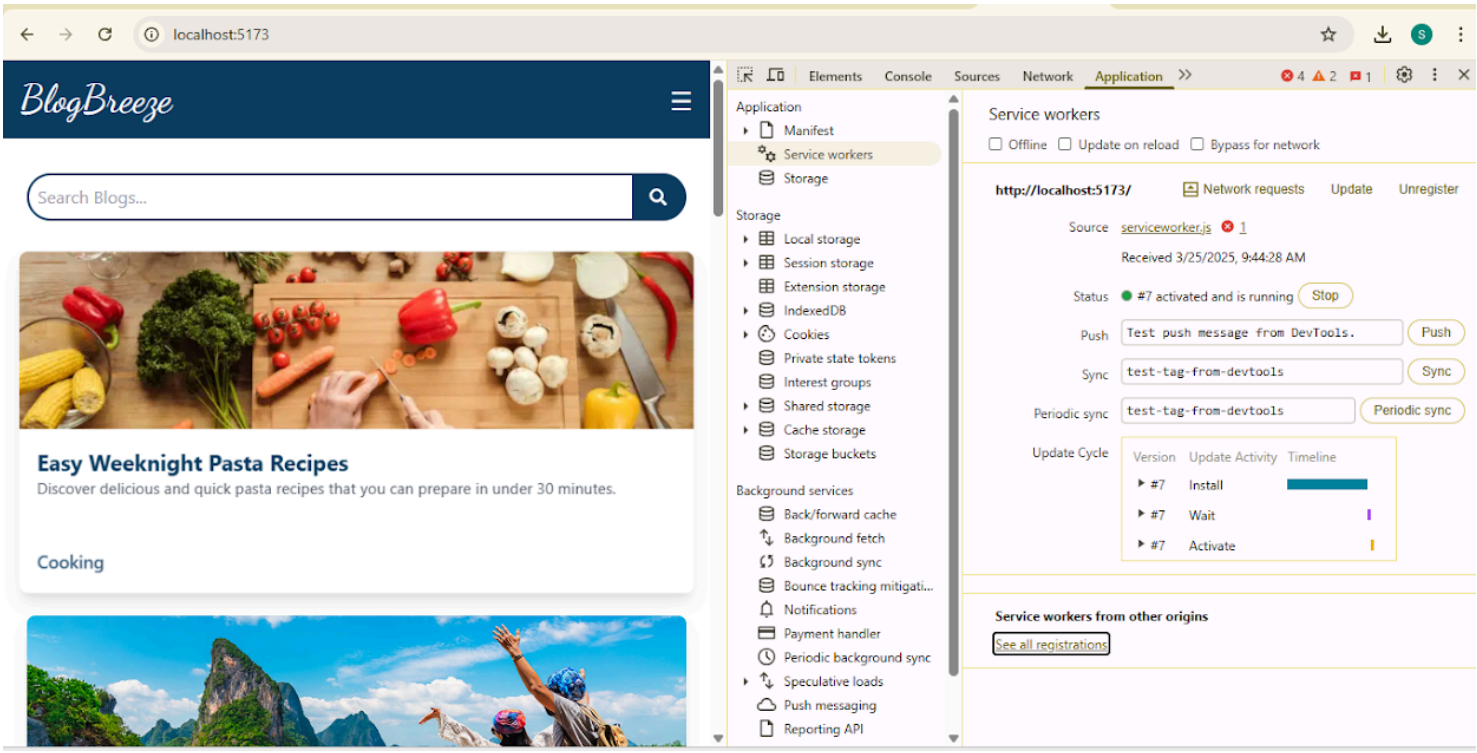
When the new service worker activates, an activate event is triggered in the activating service worker. This event listener is a good place to clean up outdated caches (see the Offline Cookbook for an example).

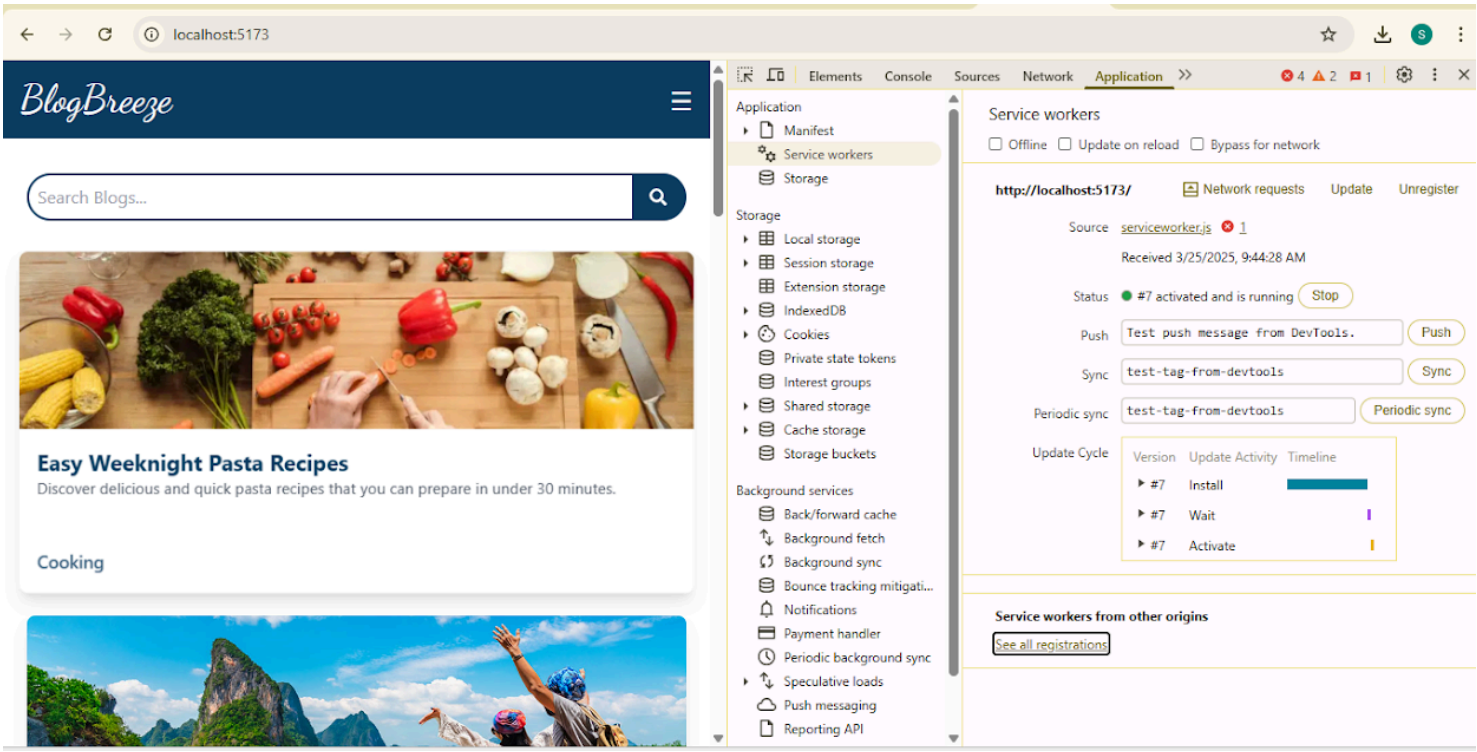
Once activated, the service worker controls all pages that load within its scope, and starts listening for events from those pages. However, pages in your app that were loaded before the service worker activation will not be under service worker control. The new service worker will only take over when 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.

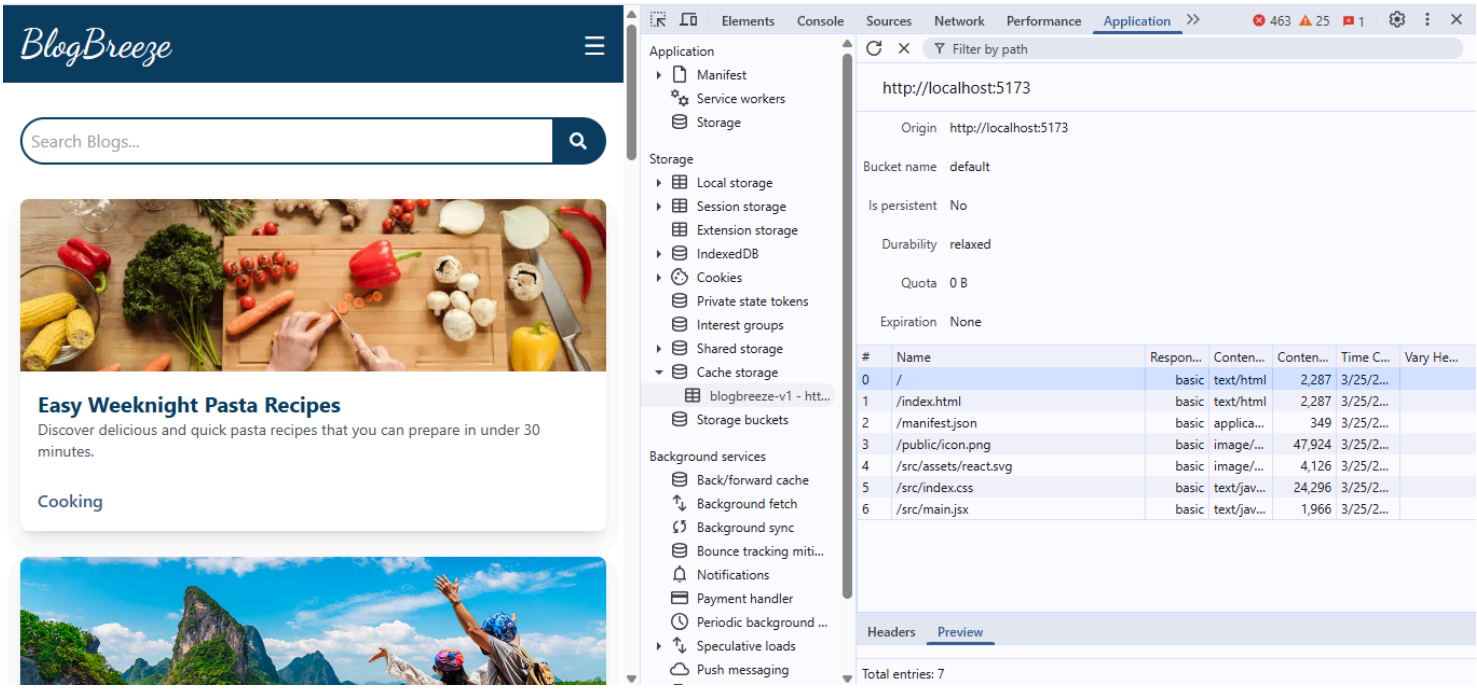
Code :

```
// Service Worker Script

const CACHE_NAME = 'blogbreeze-v1'; // Cache name to identify the version of cached content
const urlsToCache = [
  '/', // Home page
  'index.html', // Main HTML file
  'ani.html', // Any additional pages you want to cache
  '/src/assets/react.svg', // App icon
  '/public/icon.png', // Favicon
  '/src/main.jsx', // Main JS file for app functionality
//    'https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css' // FontAwesome for icons
];

// Install Service Worker
self.addEventListener('install', (event) => {
  console.log('Service Worker: Installed');
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then((cache) => {
        return cache.addAll(urlsToCache);
      })
  );
});

// Activate Service Worker
self.addEventListener('activate', (event) => {
```

```
  console.log('Service Worker: Activated');
  // Remove old caches if there are any
  event.waitUntil(
    caches.keys().then((cacheNames) => {
      return Promise.all(
        cacheNames.map((cacheName) => {
          if (cacheName !== CACHE_NAME) {
            return caches.delete(cacheName);
          }
        })
      );
    })
  );
});

// Fetch event: Intercept network requests and serve cached content
self.addEventListener('fetch', (event) => {
  console.log('Service Worker: Fetching', event.request.url);
  event.respondWith(
    caches.match(event.request)
      .then((cachedResponse) => {
        // Return cached content if found, otherwise fetch from network
        return cachedResponse || fetch(event.request);
      })
  );
});
```


localhost:5173
BlogBreeze
Search Blogs...
Easy Weeknight Pasta Recipes
Discover delicious and quick pasta recipes that you can prepare in under 30 minutes.
Cooking
Elements
Console
Sources
Network
Application
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Storage buckets
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking mitigati...
Notifications
Payment handler
Periodic background sync
Speculative loads
Push messaging
Reporting API
Service workers
Offline
Update on reload
Bypass for network
http://localhost:5173/
Network requests
Update
Unregister
Source
serviceworker.js
1
Received 3/25/2025, 9:44:28 AM
Status
#7 activated and is running
Stop
Push
Test push message from DevTools.
Push
Sync
test-tag-from-devtools
Sync
Periodic sync
test-tag-from-devtools
Periodic sync
Update Cycle
Version
Update Activity
Timeline
#7
Install
#7
Wait
#7
Activate
Service workers from other origins
See all registrations


localhost:5173
BlogBreeze
Search Blogs...
Easy Weeknight Pasta Recipes
Discover delicious and quick pasta recipes that you can prepare in under 30 minutes.
Cooking
Elements
Console
Sources
Network
Application
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Storage buckets
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking mitigati...
Notifications
Payment handler
Periodic background sync
Speculative loads
Push messaging
Reporting API
Service workers
Offline
Update on reload
Bypass for network
http://localhost:5173/
Network requests
Update
Unregister
Source
serviceworker.js
1
Received 3/25/2025, 9:44:28 AM
Status
#7 activated and is running
Stop
Push
Test push message from DevTools.
Push
Sync
test-tag-from-devtools
Sync
Periodic sync
test-tag-from-devtools
Periodic sync
Update Cycle
Version
Update Activity
Timeline
#7
Install
#7
Wait
#7
Activate
Service workers from other origins
See all registrations



Conclusion : Thus we have learnt to code and register a service worker, and complete the install and activation process for a new service worker for the PWA.

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.	54
Name	Anushka Shahane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

EXPERIMENT 9

Name	**Roll No.**
Anushka Shahane	**54 - D15A**

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

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

Things to note about Service Worker:

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

Fetch Event
You can track and manage page network traffic with this event. You can check existing

cache, manage “cache first” and “network first” requests and return a response that you want.

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

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

SERVICE WORKER FOR PUSH NOTIFICATIONS :

```
const CACHE_NAME = 'blogbreeze-v1';
const filesToCache = [
  '/',
  'index.html',
  'manifest.json',
  '/src/assets/react.svg',
  '/public/icon.png',
  '/src/index.css',
  '/src/main.jsx',
];

// Install event: Caching important files
```

```
self.addEventListener('install', (event) => {
  console.log('Service Worker: Installed');
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then((cache) => {
        console.log('Service Worker: Caching files');
        return cache.addAll(filesToCache);
      })
  );
});

// Activate event: Cleaning up old caches
self.addEventListener('activate', (event) => {
  console.log('Service Worker: Activated');
  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);
          }
        })
      );
    })
  );
});

// Fetch event: Intercepting requests and serving from cache if available
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: Returning cached response');
          return cachedResponse; // Return the cached response if available
        }

        return fetch(event.request)
          .then((networkResponse) => {
            caches.open(CACHE_NAME).then((cache) => {
              console.log('Service Worker: Caching new response for', event.request.url);
              cache.put(event.request, networkResponse.clone()); // Cache the new response
            });
            return networkResponse;
          });
      })
  );
});

self.addEventListener('push', (event) => {
    console.log('Push notification received:', event);

    if (event && event.data) {
      // Parse the push notification data
      const data = event.data.json();

      if (data.method === 'pushMessage') {
        console.log('Push notification sent with message:', data.message);

        // Set up notification options (like body, icon, etc.)
        const options = {
```

```
      body: data.message || 'Hello, this is a default message!', // Default or push message
      icon: '/src/assets/react.svg', // Icon for the notification
      badge: '/src/assets/react.svg', // Badge icon for the notification
    };

    // Check if the notification permission is granted before showing the notification
    if (Notification.permission === 'granted') {
      // Show the notification with a title
      event.waitUntil(
        self.registration.showNotification('Blog Breeze', options)
      );
    } else {
      console.log('Notification permission not granted yet.');
    }
  }
});
```

SERVICE WORKER SCRIPT FOR SYNC AND FETCH :

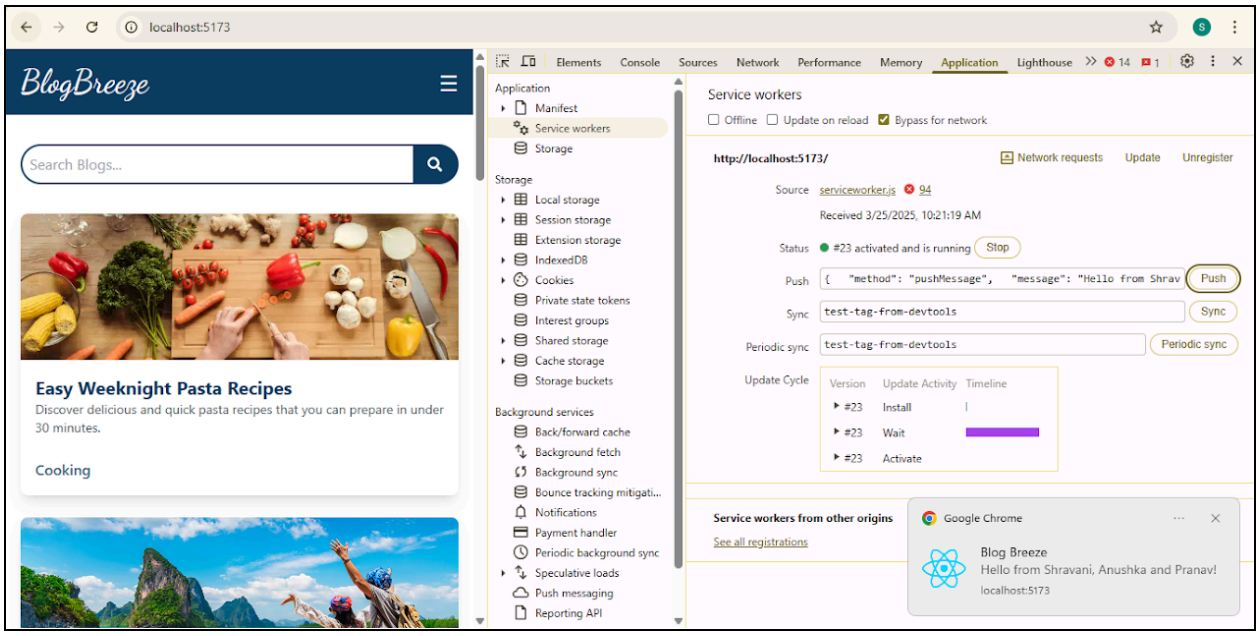
```
const CACHE_NAME = 'blogbreeze-v1';
const filesToCache = [
  '/',
  'index.html',
  'manifest.json',
  '/src/assets/react.svg',
  '/public/icon.png',
  '/src/index.css',
  '/src/main.jsx',

];
```

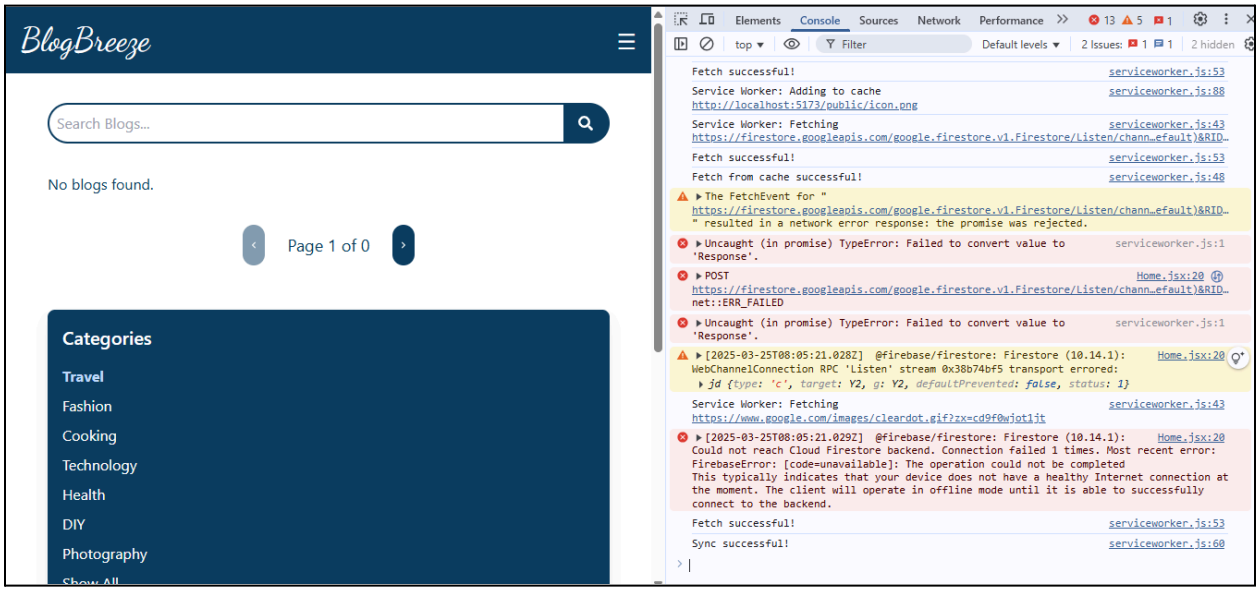
```
// Install event: Caching important files
self.addEventListener('install', (event) => {
  console.log('Service Worker: Installed');
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then((cache) => {
        console.log('Service Worker: Caching files');
        return cache.addAll(filesToCache);
      })
  );
});

// Activate event: Cleaning up old caches
self.addEventListener('activate', (event) => {
  console.log('Service Worker: Activated');
  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);
          }
        })
      );
    })
  );
});

// Fetch event: Intercepting requests and serving from cache if available
self.addEventListener('fetch', function (event) {
```

```
  console.log('Service Worker: Fetching', event.request.url);

  // Skip caching for Firebase API requests
  if (event.request.url.includes('firestore.googleapis.com')) {
    // Always fetch Firebase data from the network (no cache)
    event.respondWith(fetch(event.request));
    return;
  }

  event.respondWith(
    checkResponse(event.request)
      .catch(function () {
        console.log('Fetch from cache successful!');
        return returnFromCache(event.request);
      })
  );

  console.log('Fetch successful!');
  event.waitUntil(addToCache(event.request));
});

// Sync event: Handling background sync
self.addEventListener('sync', function (event) {
  if (event.tag === 'syncMessage') {
    console.log('Sync successful!');
    // You can add more background sync logic here
  }
});

// Placeholder functions for cache and response handling
function checkResponse(request) {
  return caches.match(request)
```

```
    .then(function (cachedResponse) {
      if (cachedResponse) {
        return cachedResponse;
      }
      return fetch(request);
    });
}

function returnFromCache(request) {
  return caches.match(request);
}

function addToCache(request) {
  return fetch(request)
    .then(function (response) {
      if (!response || response.status !== 200 || response.type !== 'basic') {
        return response;
      }
      return caches.open(CACHE_NAME)
        .then(function (cache) {
          console.log('Service Worker: Adding to cache', request.url);
          cache.put(request, response);
          return response;
        });
    });
}
```

OUTPUT :





Conclusion : Thus we learnt to implement service worker events like fetch, sync and push for PWA.

MAD & PWA Lab

Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	54
Name	Anushka Shahane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

Experiment No.10
MAD & PWA Lab

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

Theory:

GitHub Pages

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

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

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

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

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

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

Firebase

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

Some of the features offered by Firebase are:

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

Reasons for favoring over GitHub Pages:

1. Realtime backend made easy
2. Fast and responsive

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

Pros

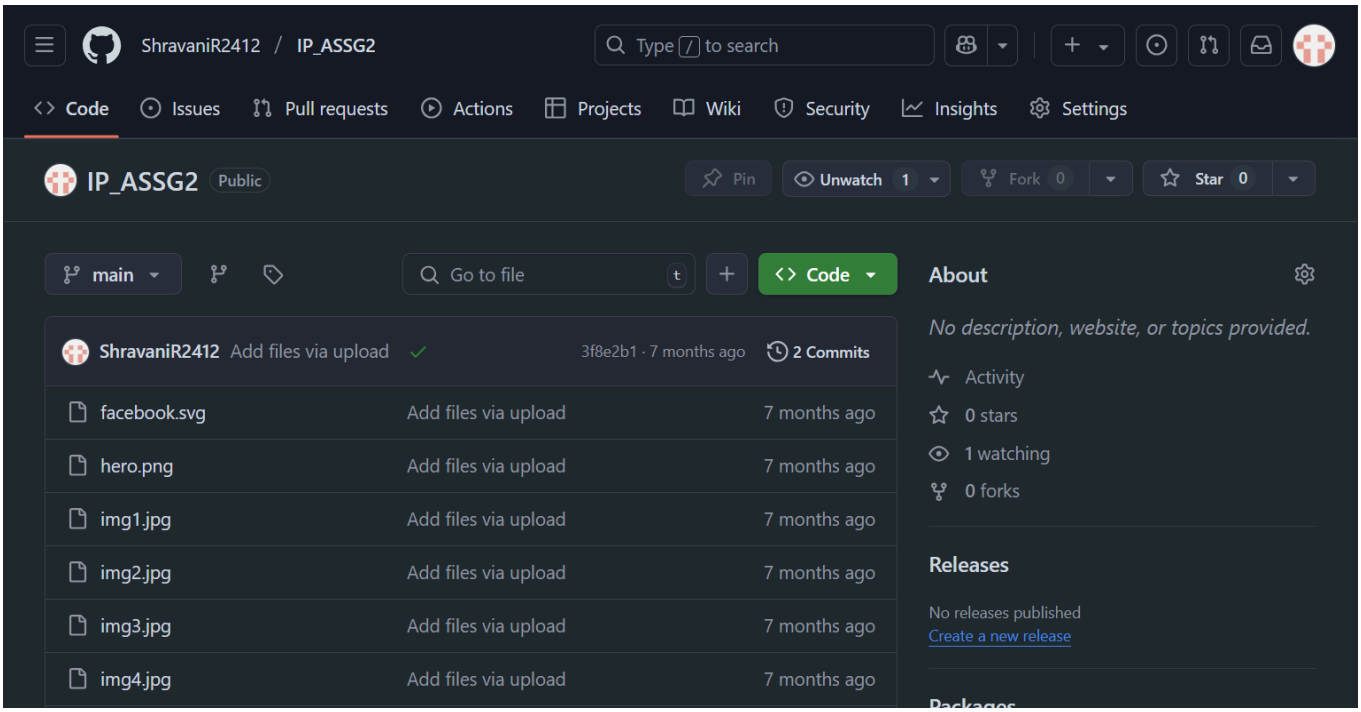
1. Hosted by Google. Enough said.
2. Authentication, Cloud Messaging, and a whole lot of other handy services will be available to you.
3. A real-time database will be available to you, which can store 1 GB of data.
4. You'll also have access to a blob store, which can store another 1 GB of data.
5.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

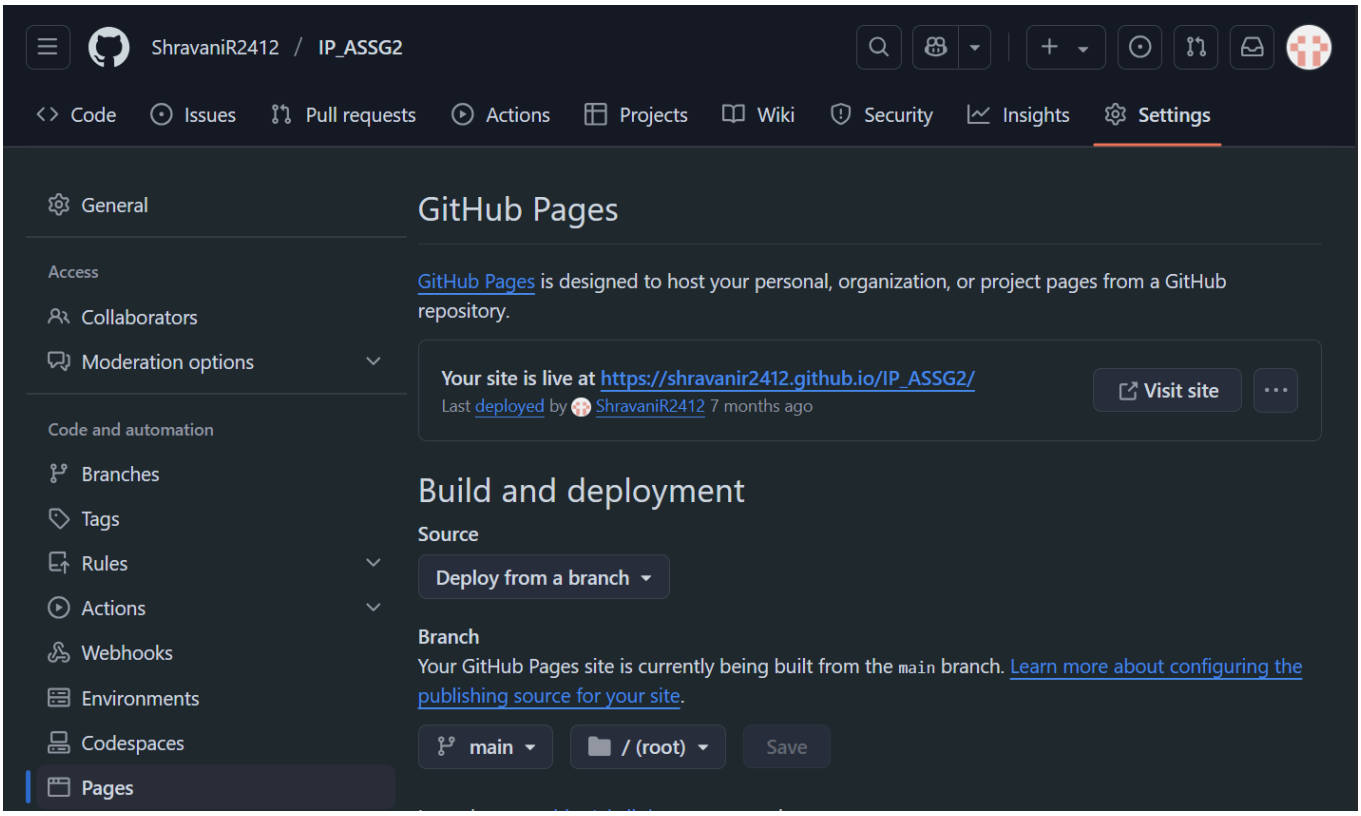
Cons

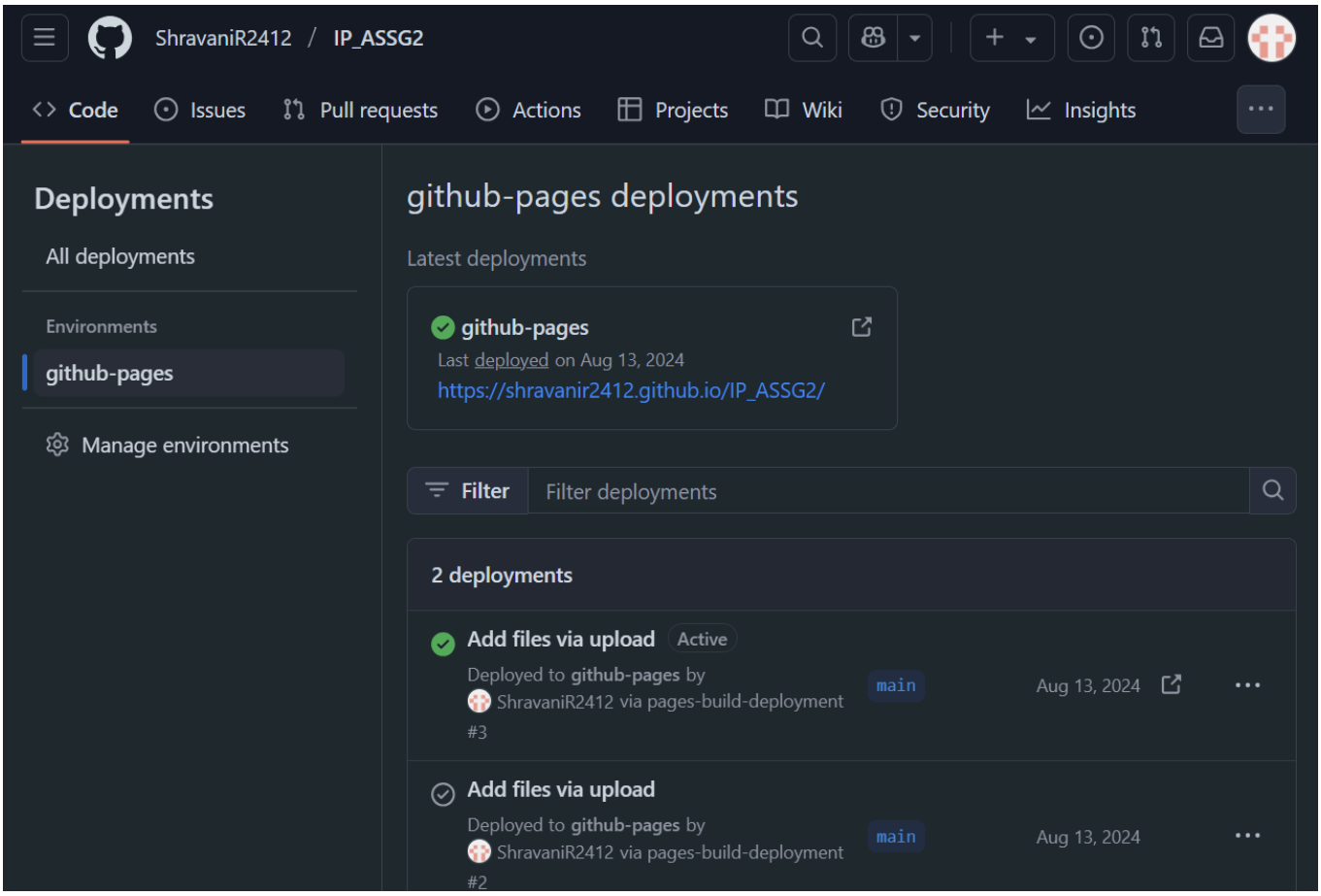
1. Only 10 GB of data transfer is allowed per month. But this is not really a big problem, if you use a CDN or AMP.
2. Command-line interface only.
3. No in-built support for any static site generator.

Link to our GitHub repository: https://github.com/ShravaniR2412/IP_ASSG2

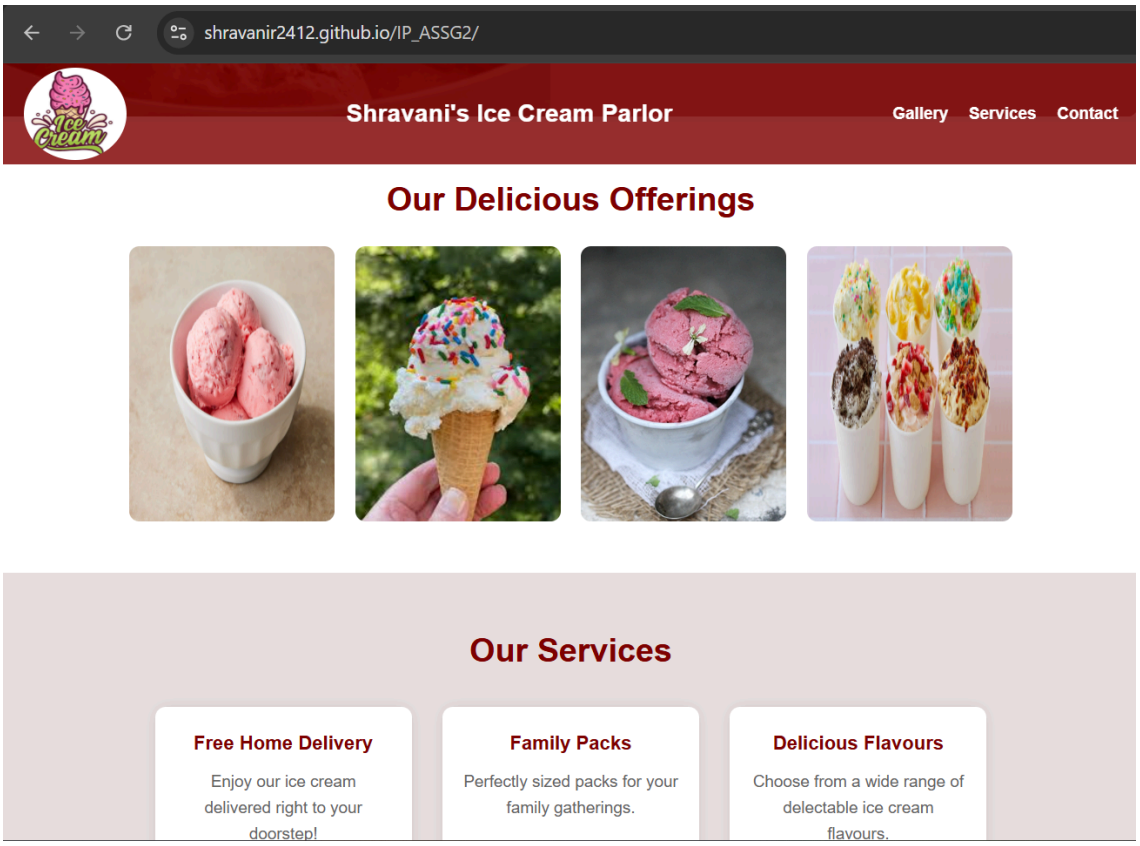
Github Screenshot:




ShravaniR2412 / IP_ASSG2
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
General
Access
Collaborators
Moderation options
Code and automation
Branches
Tags
Rules
Actions
Webhooks
Environments
Codespaces
Pages
GitHub Pages
GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.
Your site is live at https://shravanir2412.github.io/IP_ASSG2/
Last deployed by ShravaniR2412 7 months ago
Visit site
Build and deployment
Source
Deploy from a branch
Branch
Your GitHub Pages site is currently being built from the main branch. Learn more about configuring the publishing source for your site.
main
/ (root)
Save

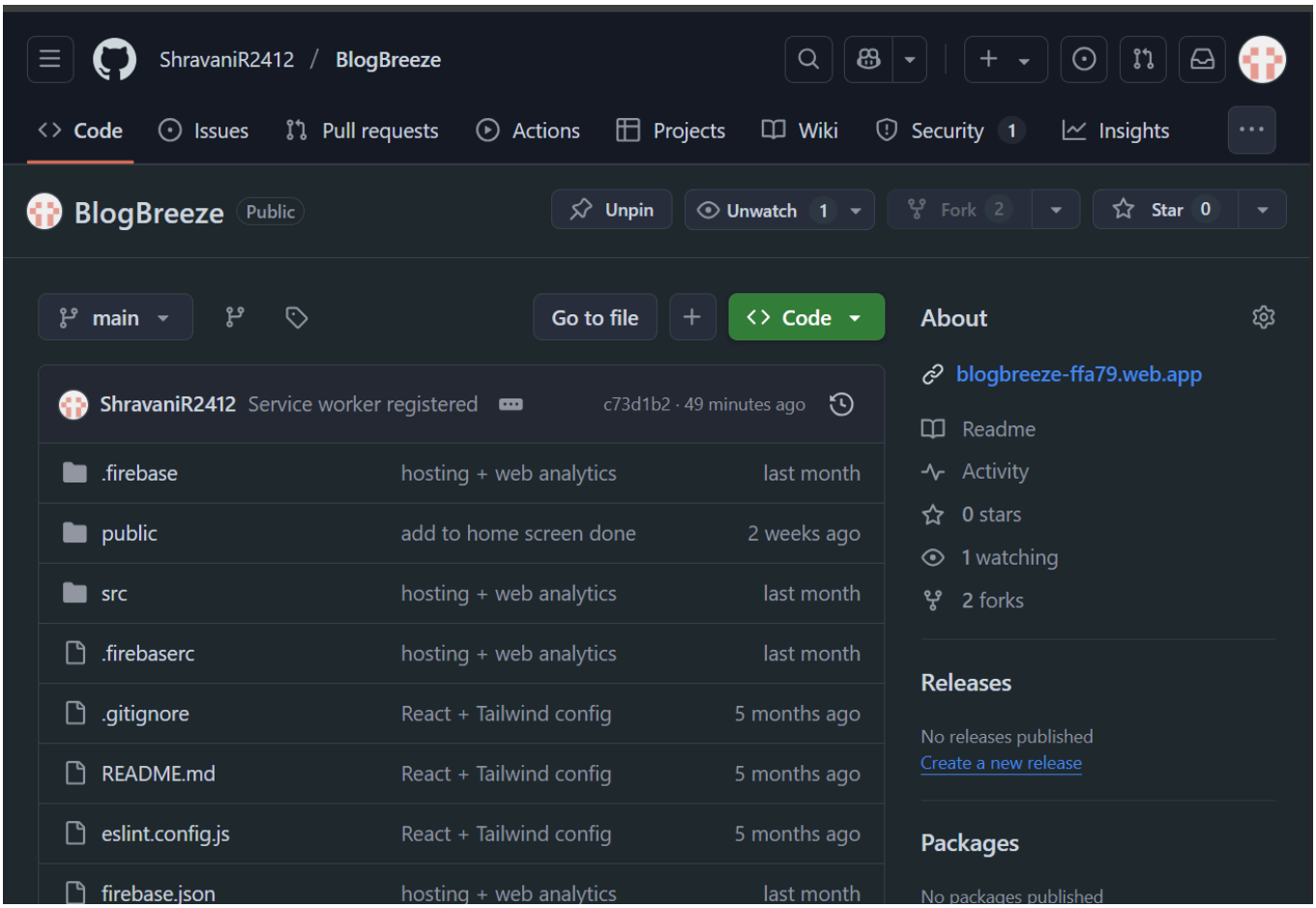

ShravaniR2412 / IP_ASSG2
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Deployments
All deployments
Environments
github-pages
Manage environments
github-pages deployments
Latest deployments
github-pages
Last deployed on Aug 13, 2024
https://shravanir2412.github.io/IP_ASSG2/
Filter
Filter deployments
2 deployments
Add files via upload
Active
Deployed to github-pages by
ShravaniR2412 via pages-build-deployment
#3
main
Aug 13, 2024
Add files via upload
Deployed to github-pages by
ShravaniR2412 via pages-build-deployment
#2
main
Aug 13, 2024

ANUSHKA SHAHANE : ROLL NO. 54



Deployed Link: https://shravanir2412.github.io/IP_ASSG2/

PWA Website:



Github Link: https://github.com/ShravaniR2412/BlogBreeze

Deployed Link: blogbreeze-ffa79.web.app

MAD & PWA Lab
Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	54
Name	Anushka Shahane
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	

Experiment 11

Anushka Shahane D15A 54

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

Theory :

Reference : https://www.semrush.com/blog/google-lighthouse/

Google Lighthouse :

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

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

Key Features and Audit Metrics

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

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

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

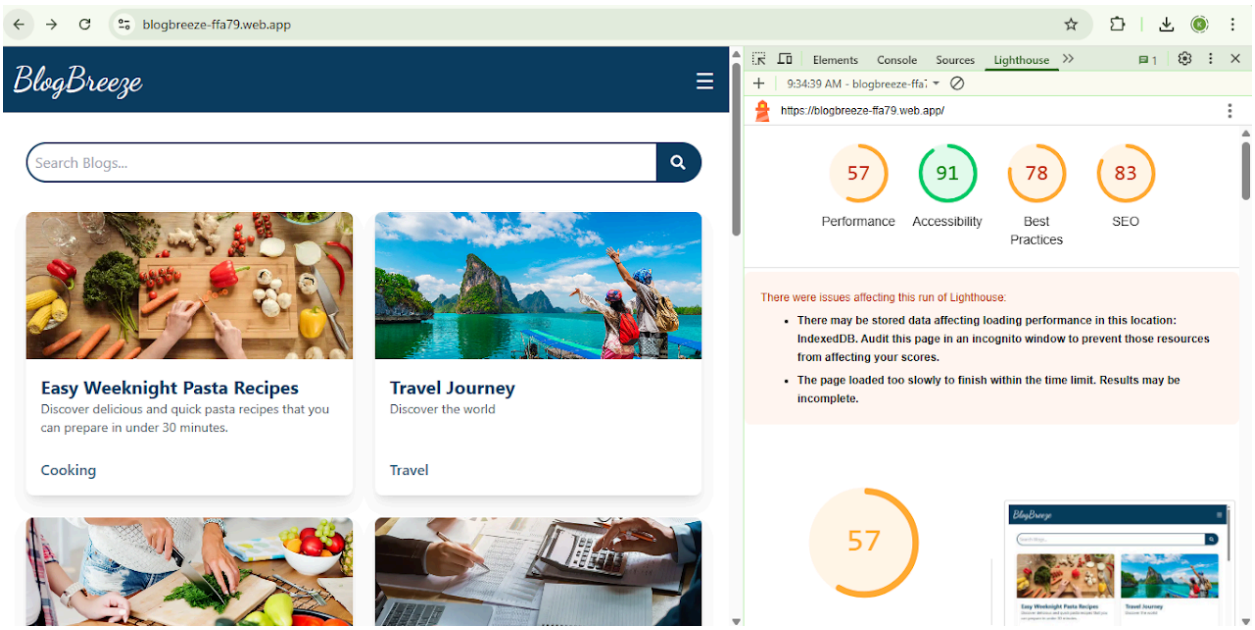
4. **Best Practices:** As any developer would know, there are a number of practices that have been deemed ‘best’ based on empirical data. This metric is an aggregation of many such points, including but not limited to:Use of HTTPS

   Avoiding the use of deprecated code elements like tags, directives, libraries, etc.

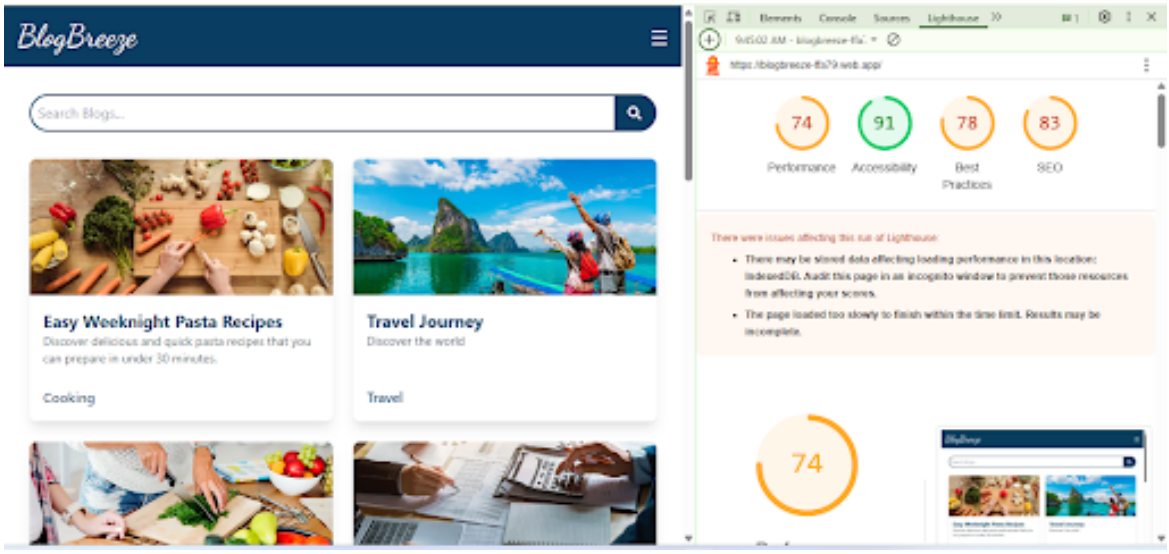
   Password input with paste-into disabled

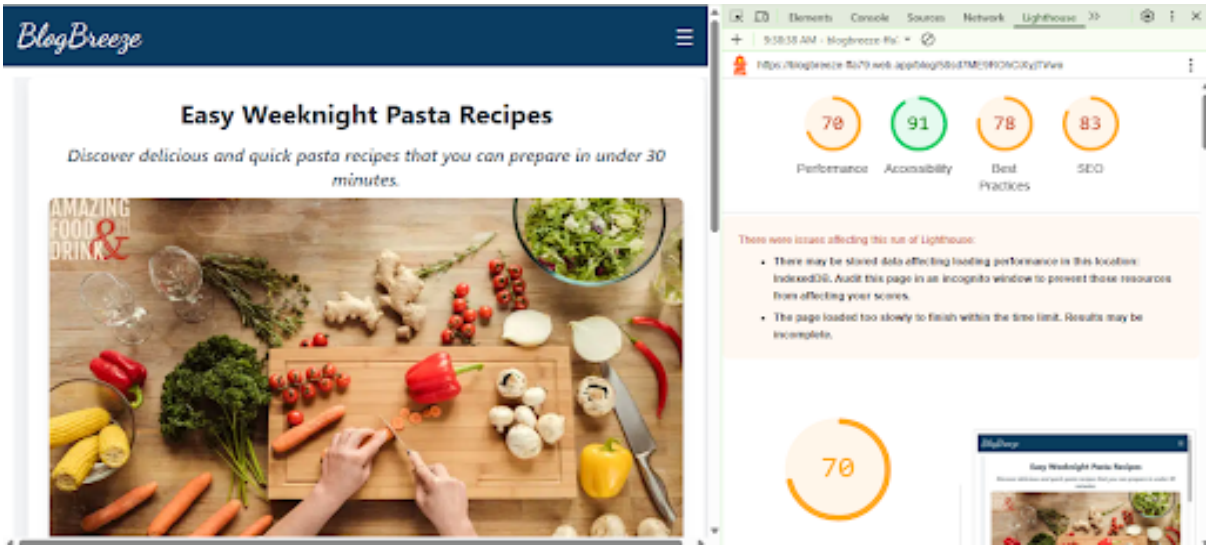
   Geo-Location and cookie usage alerts on load, etc.

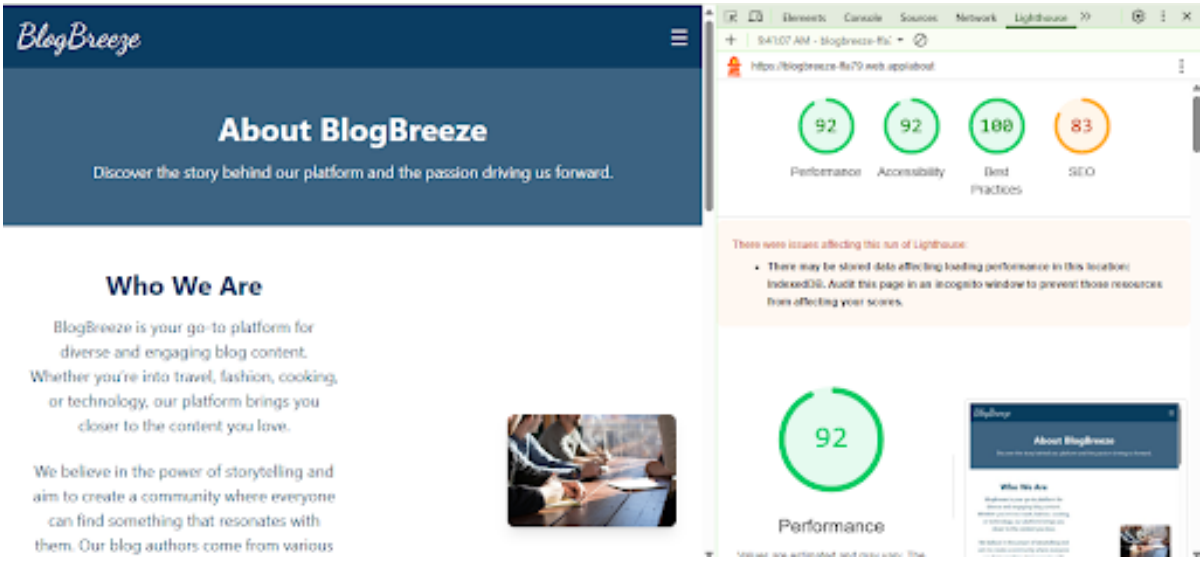
Performance before changes



Performance after changes







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