# 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 **Pranav Pramod Titambe** 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: DailyHunt** **Roll No. 60**

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

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

**Faculty Incharge** : Mrs. Kajal Joseph.

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

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

**Programme Outcomes**: The graduate will be able to:

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

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

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

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

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

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

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

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

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

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

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

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

**Program specific Outcomes**

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

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

**Project Title: DailyHunt** **Roll No. 60**

**Lab Objectives:**

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

**Lab Outcomes:**

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

# Index

# MAD & PWA Lab

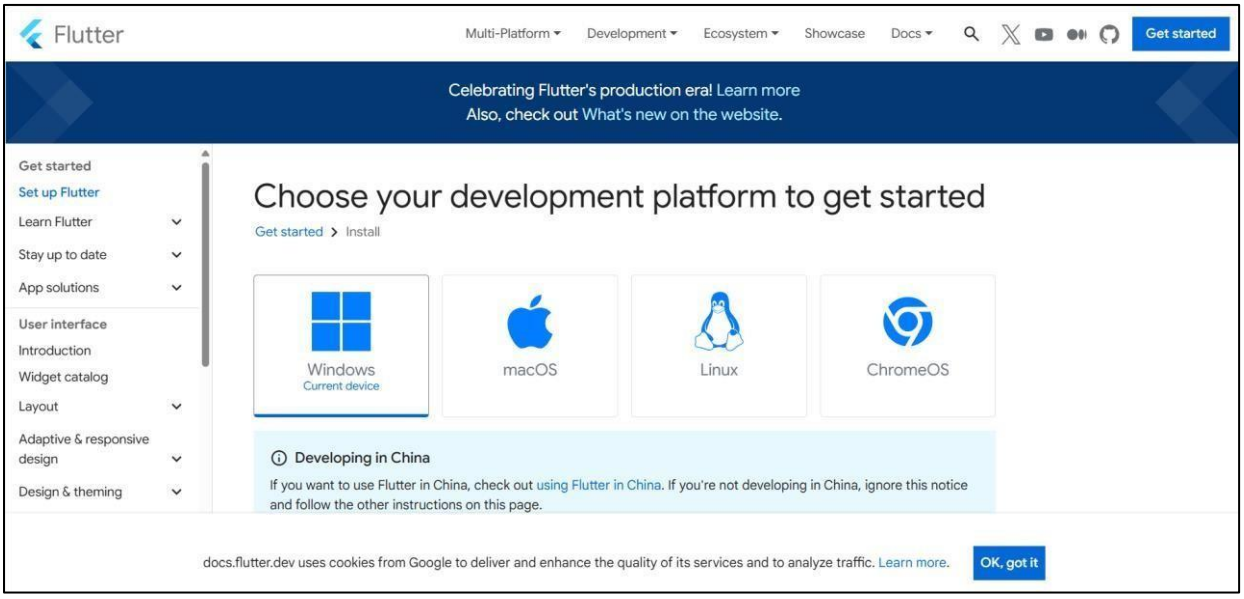
# Journal

| | |
|---|---|
| Experiment No. | 01 |
| Experiment Title. | To install and configure the Flutter Environment |
| Roll No. | 60 |
| Name | Pranav Pramod Titambe |
| Class | D15A |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO1: Understand cross platform mobile application development using Flutter framework |
| Grade: | |

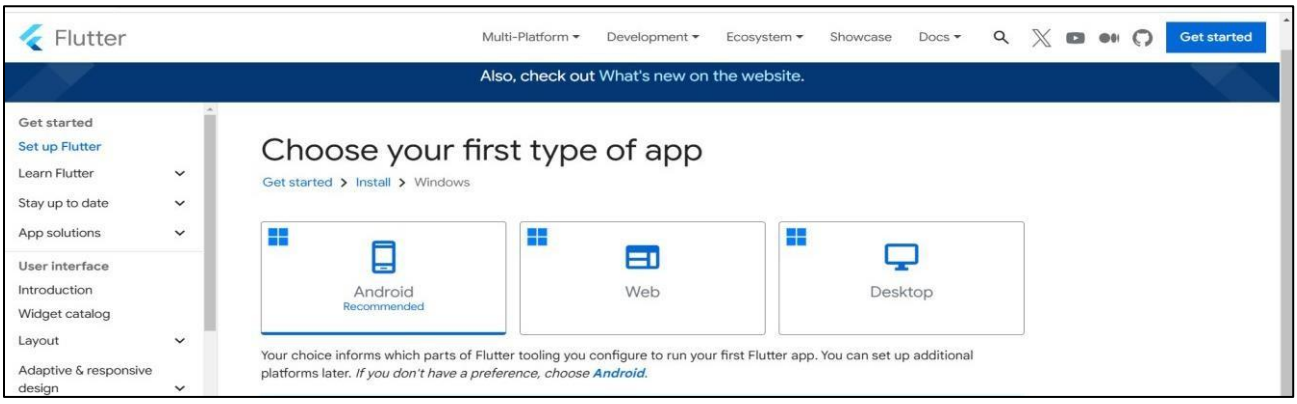
**Project Title: DailyHunt** **Roll No. 60**

**AIM: -** Installation and Configuration of Flutter Environment.

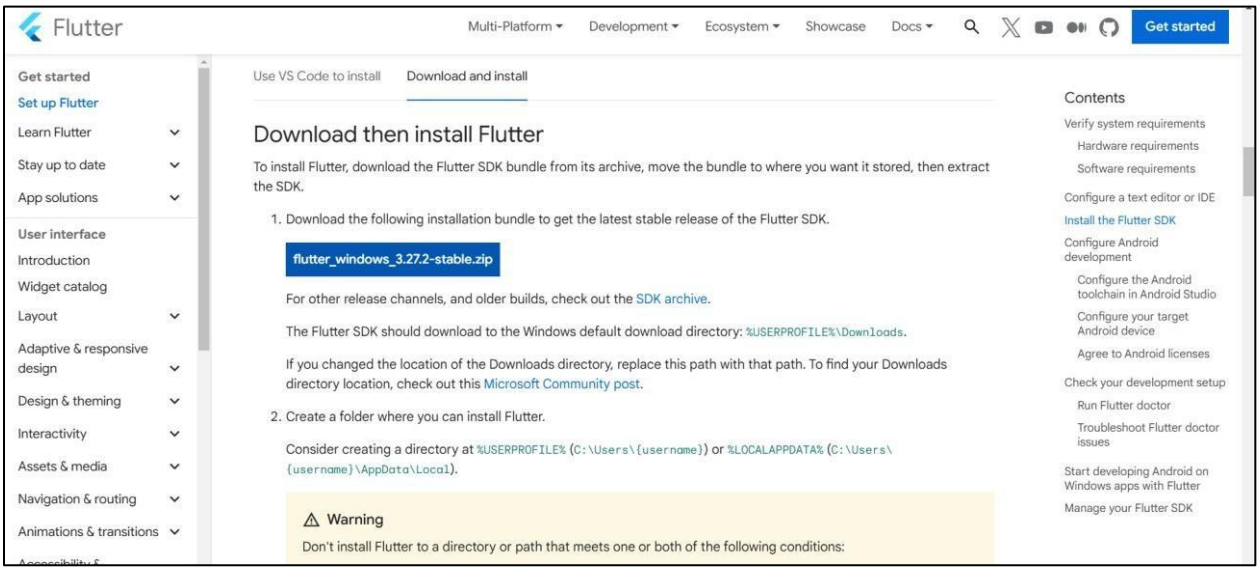
**Step 1:** Go to the official Flutter website: https://docs.flutter.dev/get-started/install



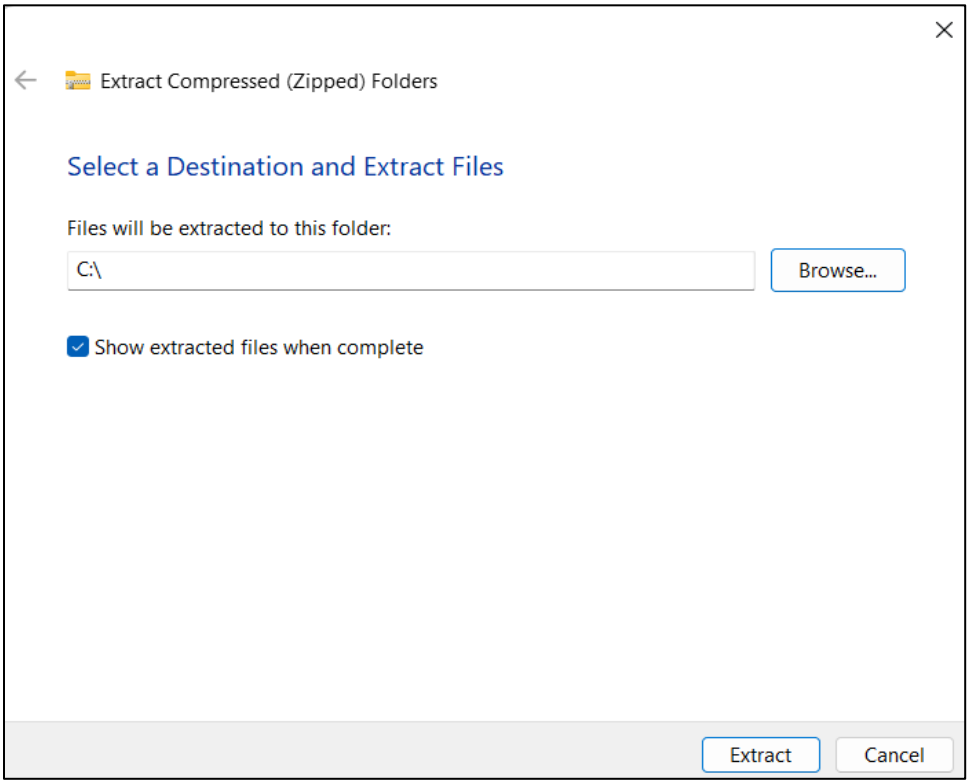

**Step 2:** To download the latest Flutter SDK, click on the Windows icon > Android



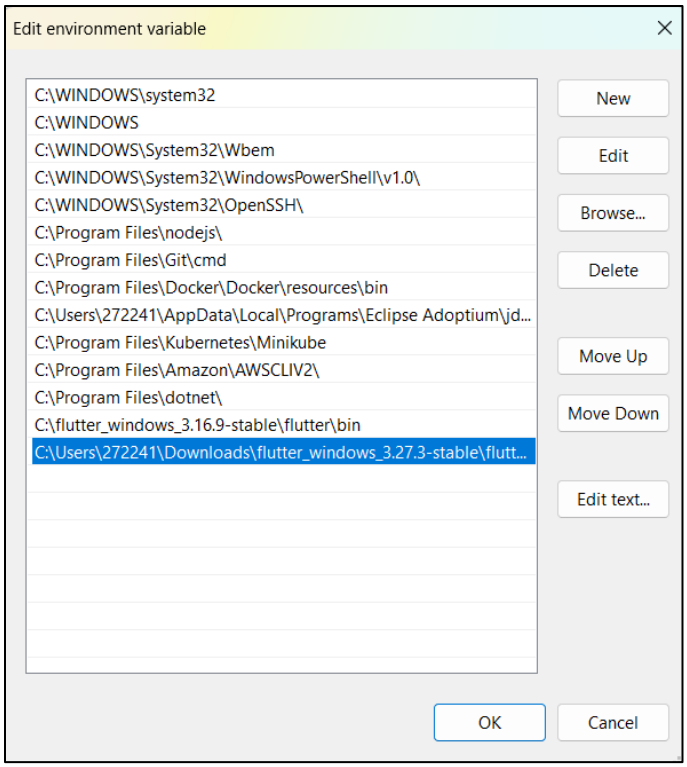
**Step 3:** For Windows, download the stable release (a .zip file).




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



**Step 5 :-** Add Flutter to System PATH
Right-click on the Start Menu > System > Advanced system settings > Environment Variables. Under System Variables, find Path and click Edit.
Add the full path to the flutter/bin directory (e.g., C:\flutter\bin).



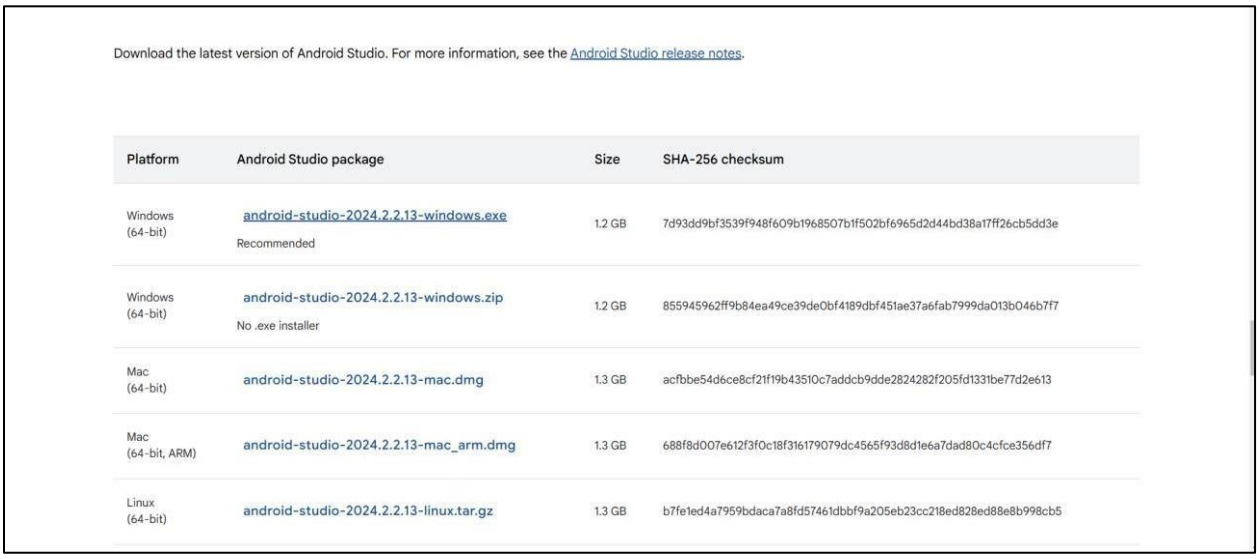

**Step 6 : -** Now, run the $ flutter command in command prompt.

```
(base) PS C:\Users\prana> 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.
```

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

```
(base) PS C:\Users\prana> flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.27.3, on Microsoft Windows [Version 10.0.26100.3037], locale en-IN)
[✓] Windows Version (Installed version of Windows is version 10 or higher)
[✓] Android toolchain - develop for Android devices (Android SDK version 35.0.1)
[✓] Chrome - develop for the web
[!] Visual Studio - develop Windows apps (Visual Studio Community 2022 17.12.4)
    ✗ Visual Studio is missing necessary components. Please re-run the Visual Studio installer for the "Desktop
      development with C++" workload, and include these components:
        MSVC v142 - VS 2019 C++ x64/x86 build tools
         - If there are multiple build tool versions available, install the latest
        C++ CMake tools for Windows
        Windows 10 SDK
[✓] Android Studio (version 2024.2)
[✓] IntelliJ IDEA Ultimate Edition (version 2024.2)
[✓] VS Code (version 1.96.4)
[✓] Connected device (4 available)
[✓] Network resources

! Doctor found issues in 1 category.
(base) PS C:\Users\prana>
```

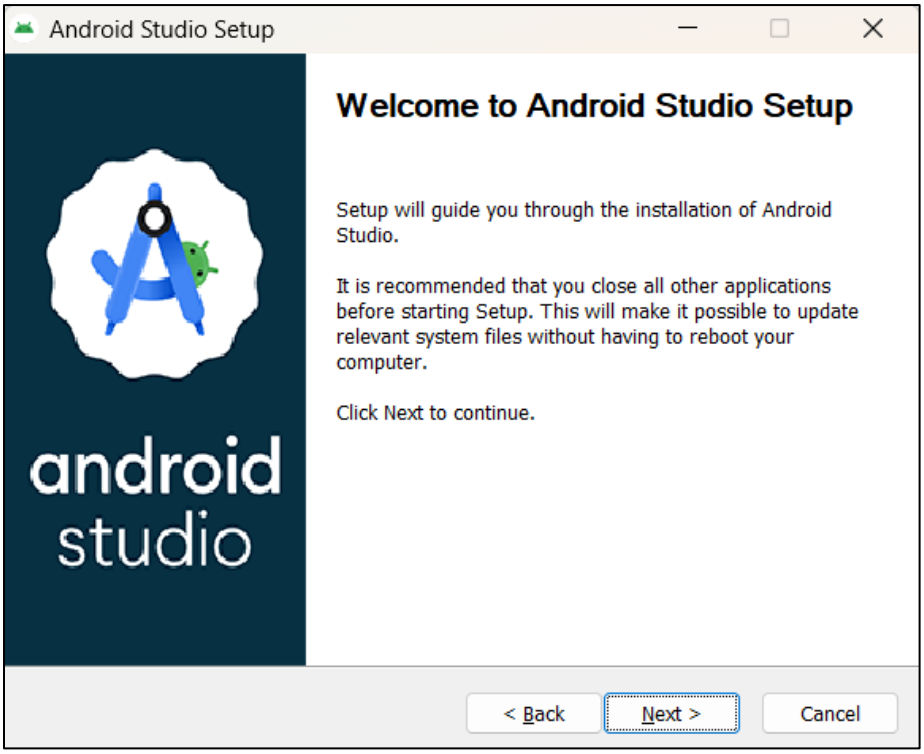
**Step 8 : -** Go to Android Studio and download the installer.



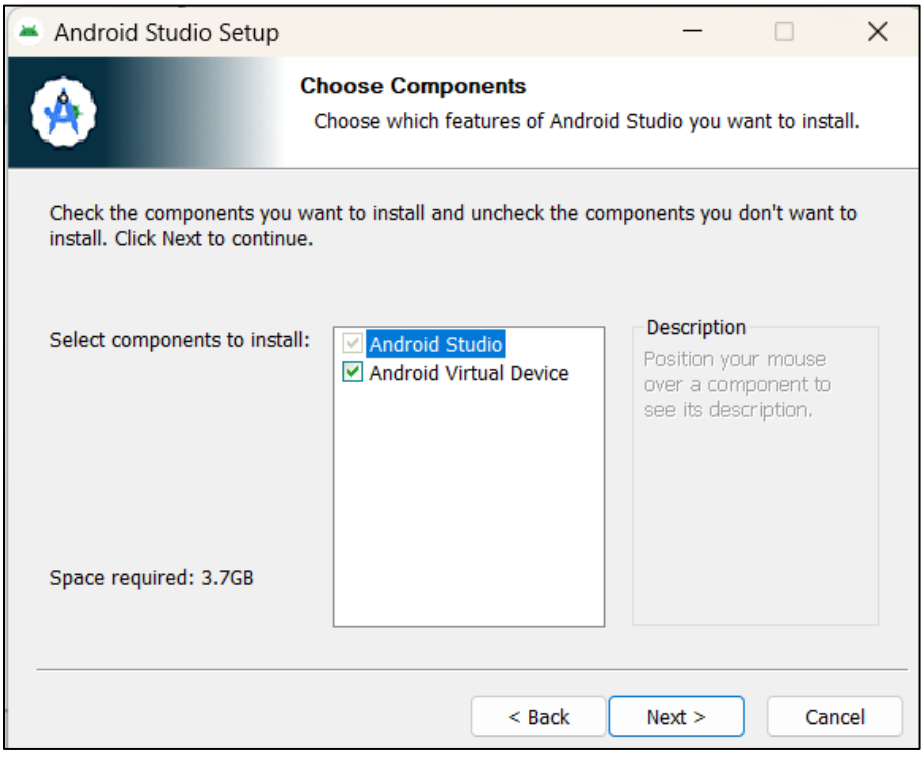
Download the latest version of Android Studio. For more information, see the Android Studio release notes.

| Platform | Android Studio package | Size | SHA-256 checksum |
|---|---|---|---|
| Windows (64-bit) | android-studio-2024.2.2.13-windows.exe Recommended | 1.2 GB | 7d93dd9bf3539f948f609b1968507b1f502bf6965d2d44bd38a17ff26cb5dd3e |
| Windows (64-bit) | android-studio-2024.2.2.13-windows.zip No .exe installer | 1.2 GB | 855945962ff9b84ea49ce39de0bf4189dbf451ae37a6fab7999da013b046b7f7 |
| Mac (64-bit) | android-studio-2024.2.2.13-mac.dmg | 1.3 GB | acfbbe54d6ce8cf21f19b43510c7addcb9dde2824282f205fd1331be77d2e613 |
| Mac (64-bit, ARM) | android-studio-2024.2.2.13-mac_arm.dmg | 1.3 GB | 688f8d007e612f3f0c18f316179079dc4565f93d8d1e6a7dad80c4cfce356df7 |
| Linux (64-bit) | android-studio-2024.2.2.13-linux.tar.gz | 1.3 GB | b7fe1ed4a7959bdaca7a8fd57461dbbf9a205eb23cc218ed828ed88e8b998cb5 |

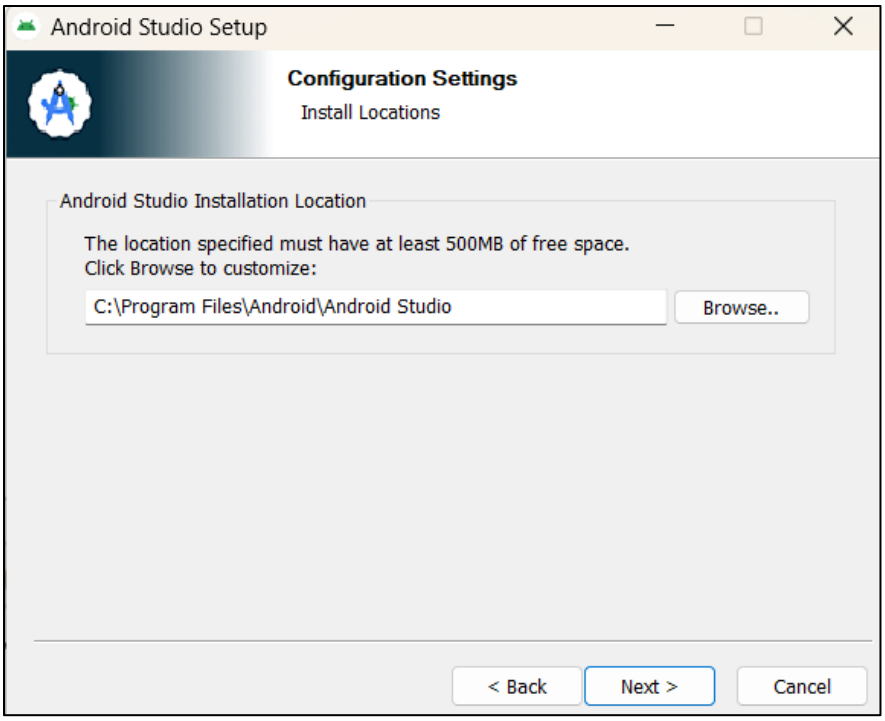
**Step 8.1: -** When the download is complete, open the .exe file and run it. You will get the following dialog box



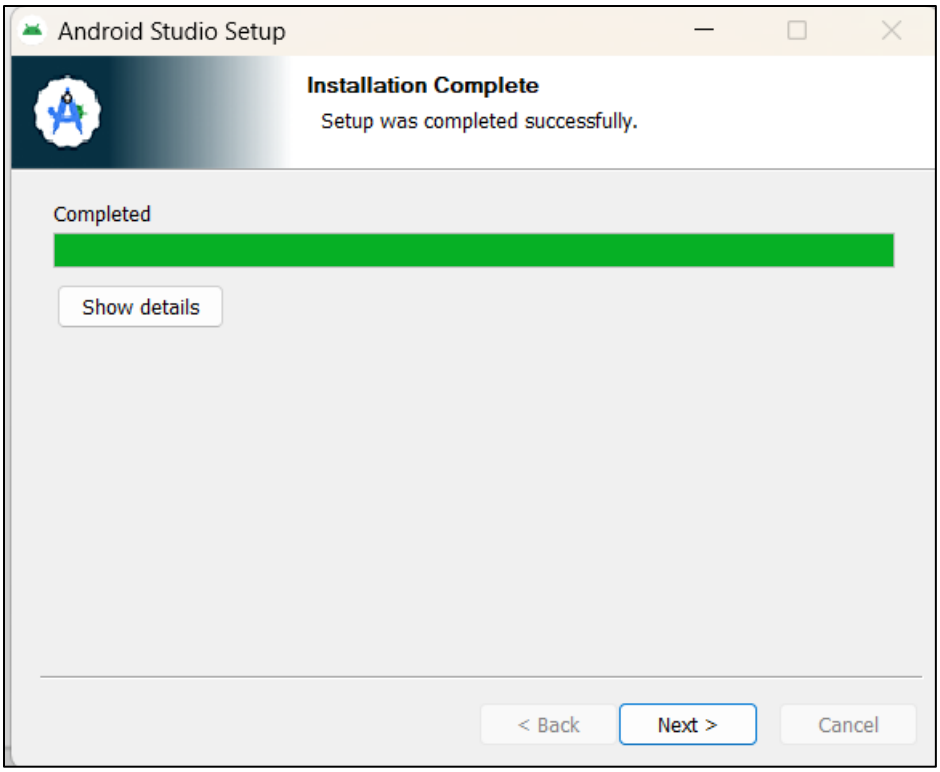

**Step 8.2: -** Select all the Checkboxes and Click on 'Next' Button.

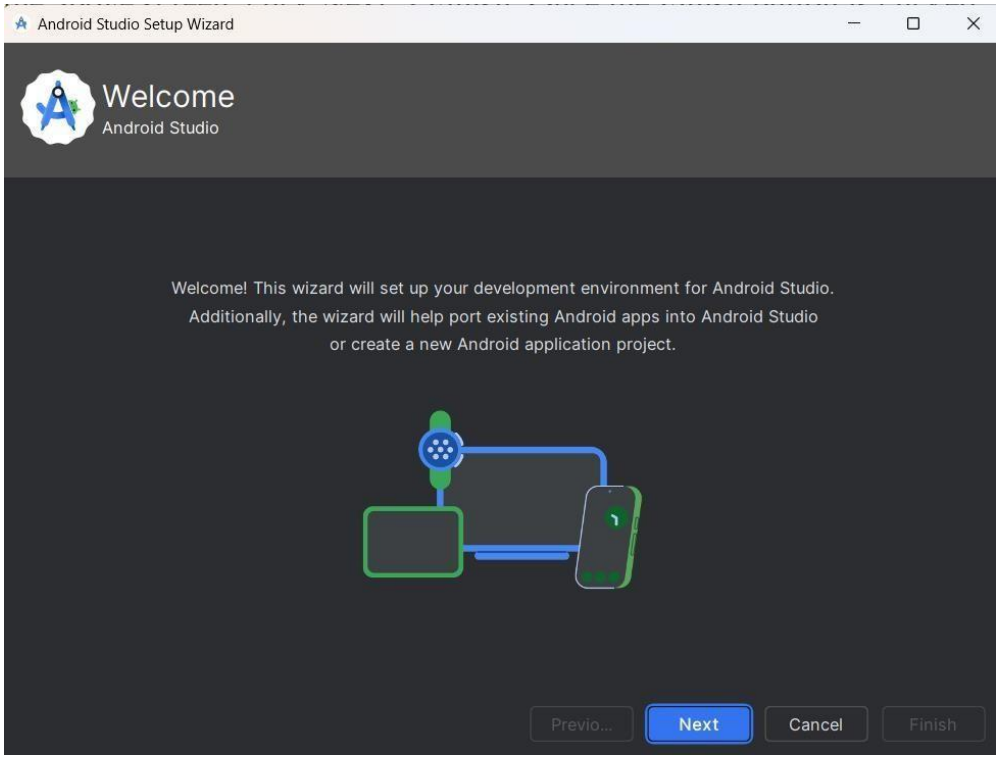


**Step 8.3: -** Change the destination as per your convenience and click on 'Next' Button.

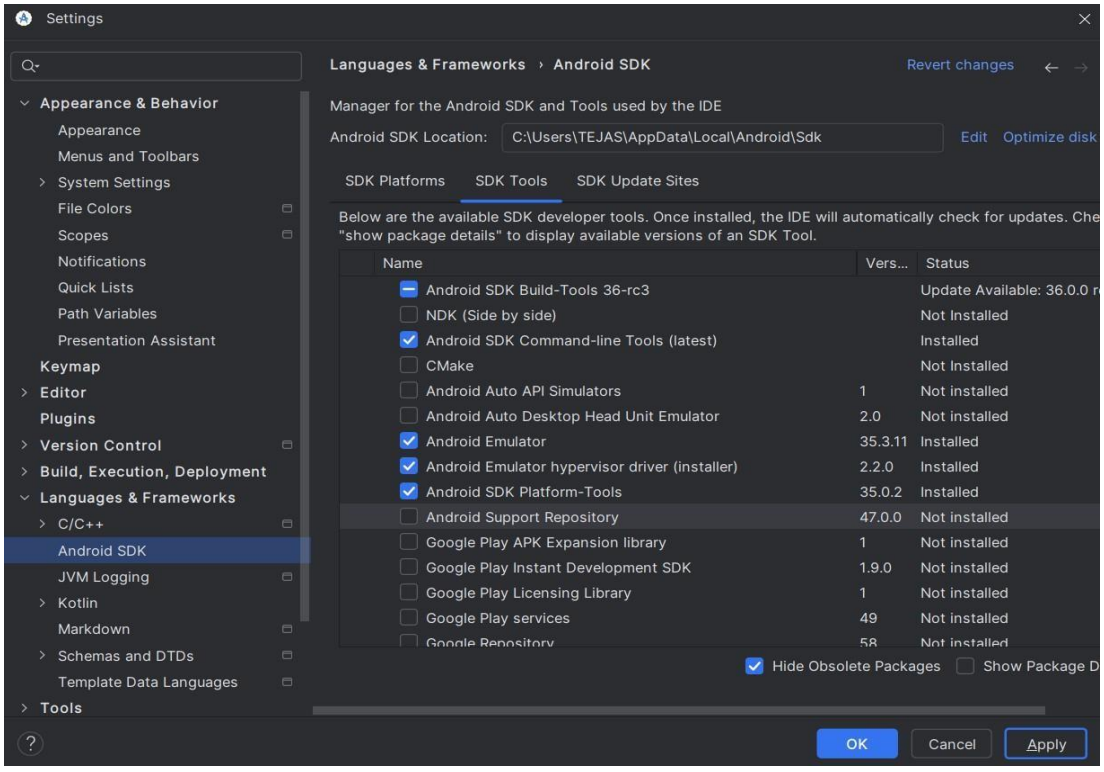



**Step 8.4: -** Follow the steps of the installation wizard. Once the installation wizard completes, you will get the following screen.






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

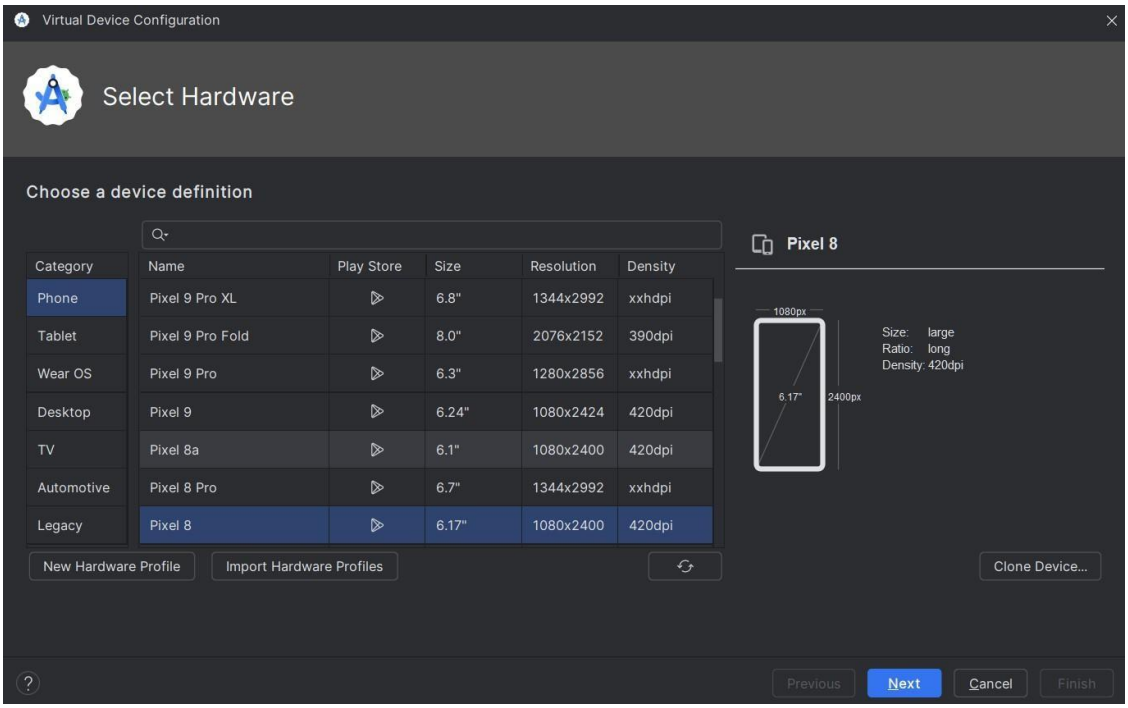


**Step 9: -** Open a terminal and run the following command

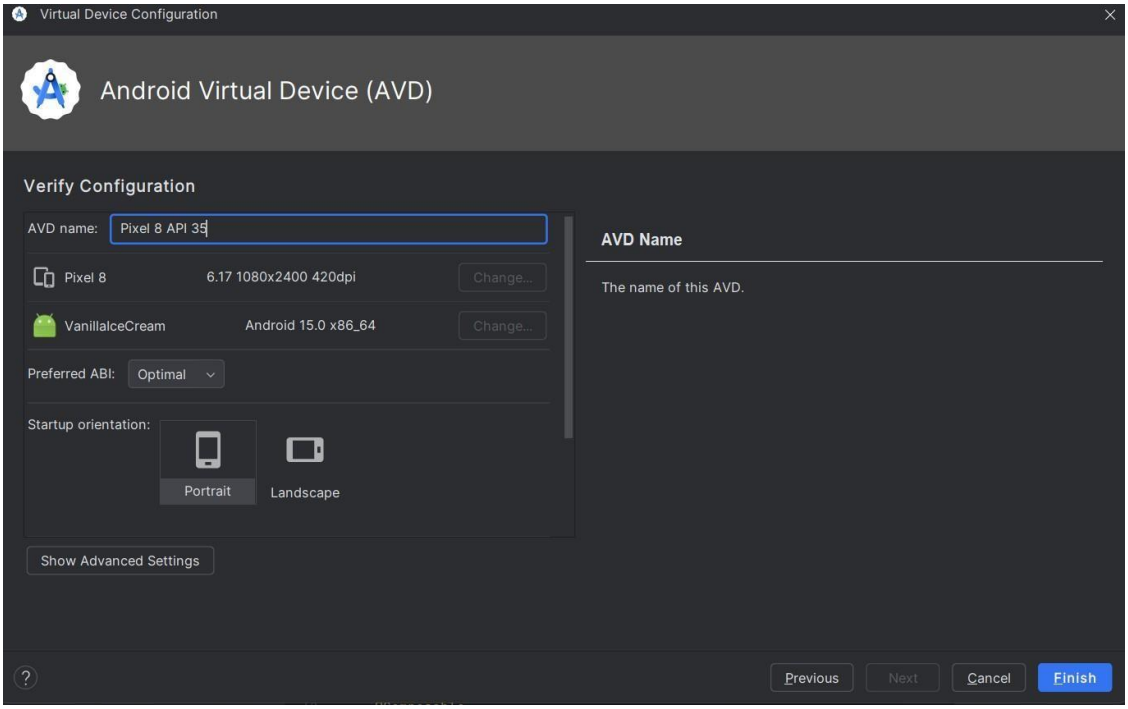
```
(base) PS C:\Users\prana>  flutter doctor --android-licenses
Warning: Observed package id 'platform-tools' in inconsistent location 'C:\Users\prana\AppData\Local\Android\Sdk\platfor
m-tools-2' (Expected 'C:\Users\prana\AppData\Local\Android\Sdk\platform-tools')
Warning: Observed package id 'platform-tools' in inconsistent location 'C:\Users\prana\AppData\Local\Android\Sdk\platfor
m-tools-2' (Expected 'C:\Users\prana\AppData\Local\Android\Sdk\platform-tools')
                                                                    Warning: Errors during XML parse:
Warning: Additionally, the fallback loader failed to parse the XML.ry...
Warning: Errors during XML parse:       ] 54% Fetch remote repository...
Warning: Additionally, the fallback loader failed to parse the XML.ry...
[=======================================] 100% Computing updates...
All SDK package licenses accepted.
```

```
(base) PS C:\Users\prana> flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.27.3, on Microsoft Windows [Version 10.0.26100.3037], locale en-IN)
[✓] Windows Version (Installed version of Windows is version 10 or higher)
[✓] Android toolchain - develop for Android devices (Android SDK version 35.0.1)
[✓] Chrome - develop for the web
[!] Visual Studio - develop Windows apps (Visual Studio Community 2022 17.12.4)
    ✗ Visual Studio is missing necessary components. Please re-run the Visual Studio installer for the "Desktop
      development with C++" workload, and include these components:
        MSVC v142 - VS 2019 C++ x64/x86 build tools
         - If there are multiple build tool versions available, install the latest
        C++ CMake tools for Windows
        Windows 10 SDK
[✓] Android Studio (version 2024.2)
[✓] IntelliJ IDEA Ultimate Edition (version 2024.2)
[✓] VS Code (version 1.96.4)
[✓] Connected device (4 available)
[✓] Network resources

! Doctor found issues in 1 category.
(base) PS C:\Users\prana>
```

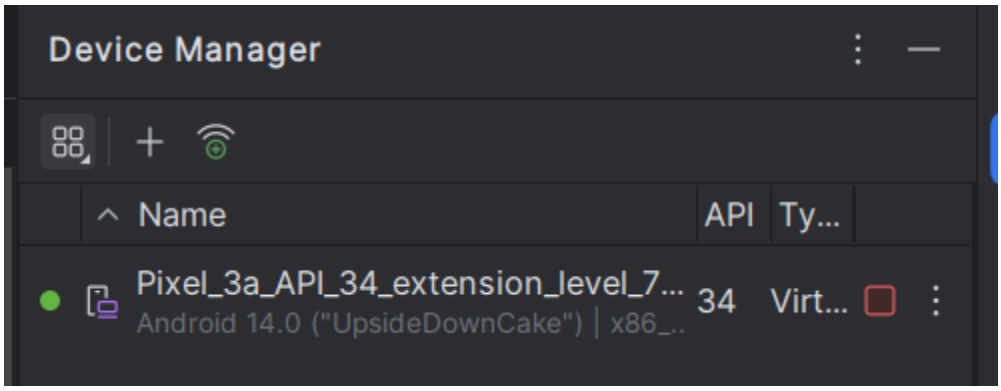
**Step 10: -** Next, you need to set up an Android emulator. It is responsible for running and testing the Flutter application




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

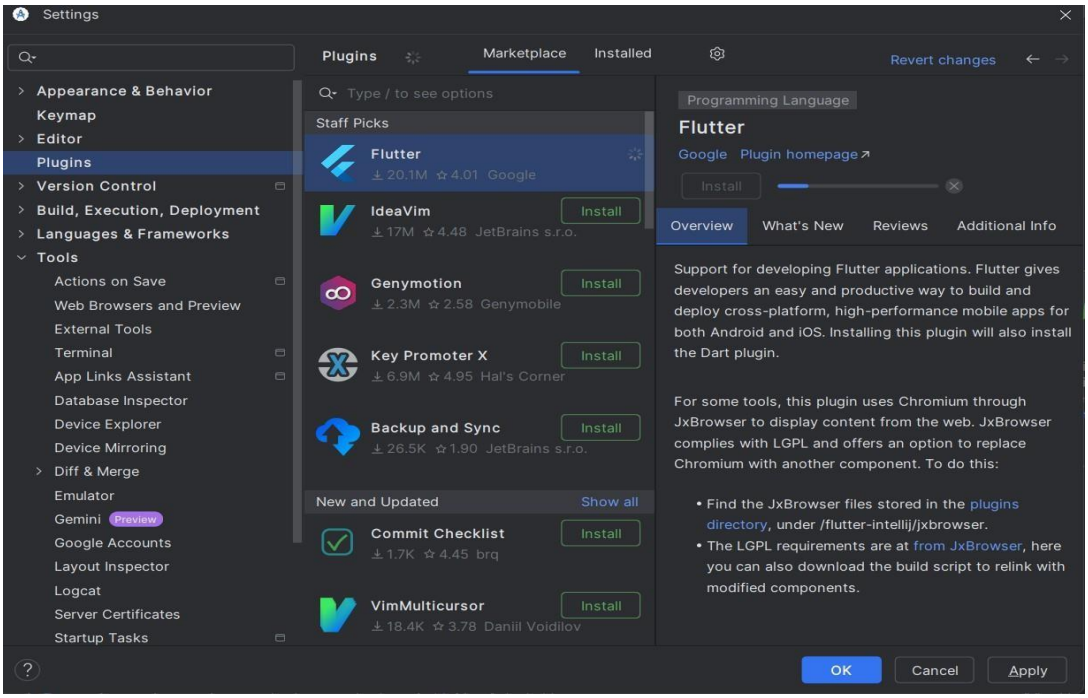


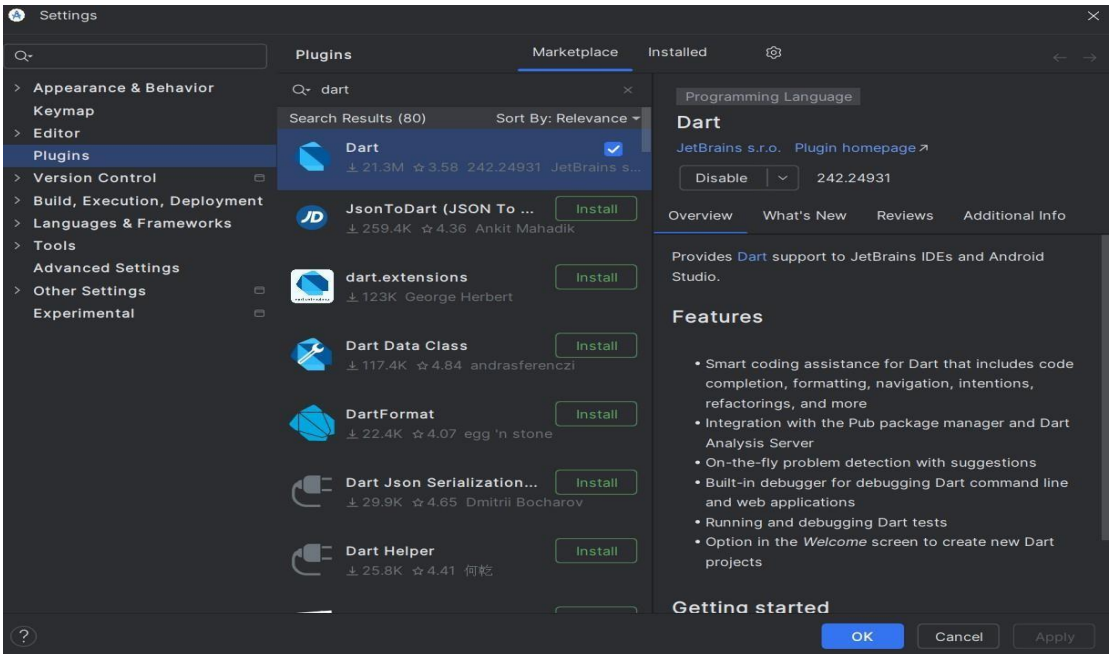
**Step 10.2: -** Click on the icon pointed into the red color rectangle. The Android emulator displayed as below screen




**Step 11: -** Now, install Flutter and Dart plugin for building Flutter application in Android Studio. These plugins provide a template to create a Flutter application, give an option to run and debug Flutter application in the Android Studio itself

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




**Step 11.2: -** Restart the Android Studio

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

# MAD & PWA Lab
# Journal

| | |
|---|---|
| Experiment No. | 02 |
| Experiment Title. | To design Flutter UI by including common widgets. |
| Roll No. | 60 |
| Name | Pranav Pramod Titambe |
| Class | D15A |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation |
| Grade: | |

**Project Title: DailyHunt** **Roll No. 60**

**Aim:** To design Flutter UI by including common widgets.

**Theory:**

Flutter follows a widget-based approach where everything in the UI is a widget. Widgets can be classified into two main types:

- Stateless Widgets: Do not change their state once built (e.g., Text, Container).
- Stateful Widgets: Can update dynamically based on user interaction (e.g., TextField, Checkbox).

Commonly Used Widgets in Flutter-

(a) Scaffold Widget

The Scaffold widget provides the basic structure for a Flutter app, including an AppBar, Drawer, FloatingActionButton, and BottomNavigationBar. It is a fundamental widget used to create a standard screen layout in Flutter.

(b) Container Widget

A Container is a box model widget that can hold other widgets. It is commonly used for adding padding, margins, borders, and background decorations.

(c) Row and Column Widgets

- Row: Arranges widgets horizontally.
- Column: Arranges widgets vertically.
  These two widgets are fundamental for designing layouts in Flutter.

(d) ListView Widget

The ListView widget is used for displaying a scrollable list of items. It is useful for showing large amounts of data dynamically.

(e) Stack Widget

The Stack widget is used to place widgets on top of each other. This is useful for creating overlapping UI elements such as banners, profile images, or layered designs.

(f) ElevatedButton Widget

The ElevatedButton widget is used for clickable buttons with a raised effect. It is a commonly used button in Flutter applications.

(g) TextField Widget

The TextField widget is used to take user input, such as entering a name, email, or password. It is commonly used in forms and authentication screens.

**Project Title: DailyHunt** **Roll No. 60**

**Code:**

## main.dart file

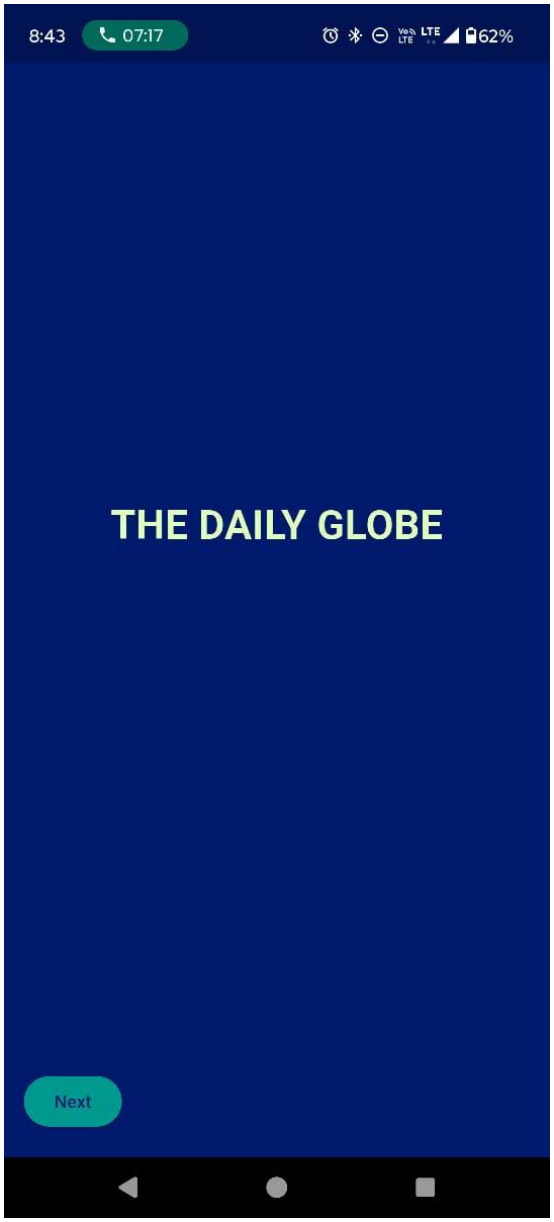
```
import 'package:flutter/material.dart';
import 'package:dailyhunt/login.dart';
import
'package:permission_handler/permission_handler.dart
';

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: "The Daily Globe",
      color: Color(0xFFE1FFBB),
      theme: ThemeData(
        primarySwatch: Colors.blue,
        scaffoldBackgroundColor: const
Color(0xFF001A6E),
        // textTheme: const TextTheme(
        //   bodyLarge: TextStyle(color:
Color(0xFFEFE9D5)),
        //   bodyMedium: TextStyle(color:
Color(0xFFEFE9D5)),
        //   titleLarge: TextStyle(color:
Color(0xFFEFE9D5)),
        // ),
        // textTheme: const TextTheme(
        //   titleLarge:TextStyle(
        //     color: Color(0xFFEFE9D5)
        //   )
        // )
      ),
      home: const OnboardingFlow(),
      debugShowCheckedModeBanner: false,
    );
  }
}

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

  @override
  State<OnboardingFlow> createState() =>
_OnboardingFlowState();
}

class _OnboardingFlowState extends
State<OnboardingFlow> {
  final PageController _pageController =
PageController();
  int _currentPage = 0;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Column(
        children: [
          Expanded(
            child: PageView(
              controller: _pageController,
              onPageChanged: (index) {
                setState(() {
                  _currentPage = index;
                });
              },
              children: const [
                SplashScreen(),
                TrackingScreen(),
                NotificationScreen(),
              ],
            ),
          ),
          // Page Navigation Buttons
          Padding(
            padding: const
EdgeInsets.symmetric(horizontal: 16, vertical: 20),
            child: Row(
              mainAxisAlignment:
MainAxisAlignment.spaceBetween,
              children: [
                if (_currentPage > 0)
                  TextButton(
                    onPressed: () {
                      _pageController.previousPage(
                        duration: const Duration(milliseconds:
300),
                        curve: Curves.easeInOut,
                      );
                    },
                    child: const Text(
                      "Back",
                      style: TextStyle(color:
Color(0xFFE1FFBB)),
                    ),
                  ),
                ElevatedButton(
                  onPressed: () {
                    if (_currentPage < 2) {
                      _pageController.nextPage(
                        duration: const Duration(milliseconds:
300),
                        curve: Curves.easeInOut,
                      );
                    } else {
                      // Navigate to login screen
                      Navigator.pushReplacement(
```

```
                    context,
                  MaterialPageRoute(builder: (context)
=> const Login()),
                );
              }
            },
            style: ElevatedButton.styleFrom(
              foregroundColor: Color(0xFF001A6E),
              backgroundColor: Color(0xFF009990),
            ),
            child: Text(_currentPage == 2 ? "Get
Started" : "Next"),
          ),
        ],
      ),
    ),
  ],
 ),
 );
 }
}

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

 @override
 Widget build(BuildContext context) {
  return const Center(
   child: Text(
    'THE DAILY GLOBE',
    style: TextStyle(
     color: Color(0xFFE1FFBB),
     fontSize: 32,
     fontWeight: FontWeight.bold,
    ),
   ),
  );
 }
}

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

 @override
 Widget build(BuildContext context) {
  return SafeArea(
   child: Padding(
    padding: const EdgeInsets.all(24.0),
    child: Column(
     mainAxisAlignment:
MainAxisAlignment.center,
     children: [
      const Text(
       'Welcome to\nThe Daily Globe!',
       textAlign: TextAlign.center,
       style: TextStyle(
        color: Color(0xFFE1FFBB),
        fontSize: 24,
        fontWeight: FontWeight.bold,
       ),
      ),
      const SizedBox(height: 8),
      const Text(
       'Please set your preferences to\nget the app
up and running.',
       textAlign: TextAlign.center,
       style: TextStyle(
        color: Colors.white70,
        fontSize: 16,
       ),
      ),
      const SizedBox(height: 24),
      Card(
       child: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
         crossAxisAlignment:
CrossAxisAlignment.stretch,
         mainAxisSize: MainAxisSize.min,
         children: [
          const Text(
           'This publication is\nad-supported.',
           textAlign: TextAlign.center,
           style: TextStyle(
            fontSize: 16,
            fontWeight: FontWeight.w500,
           ),
          ),
          const SizedBox(height: 8),
          const Text(
           'Enable ad tracking to see personalized
advertising that is relevant to your interests.',
           textAlign: TextAlign.center,
           style: TextStyle(
            fontSize: 14,
            color: Colors.black54,
           ),
          ),
          const SizedBox(height: 16),
          ElevatedButton(
           onPressed: () {},
           style: ElevatedButton.styleFrom(
            backgroundColor: const
Color(0xFF074799),
            padding: const
EdgeInsets.symmetric(vertical: 12),
           ),
           child: const Text(
            'Personalize My Ads',
            style: TextStyle(
             color: Color(0xFFE1FFBB),
            ),
           ),
          ),
          TextButton(
           onPressed: () {
            print('No Thanks');
           },
           child: const Text(
            'No Thanks',
            style: TextStyle(color:
Colors.black54),
           ),
```

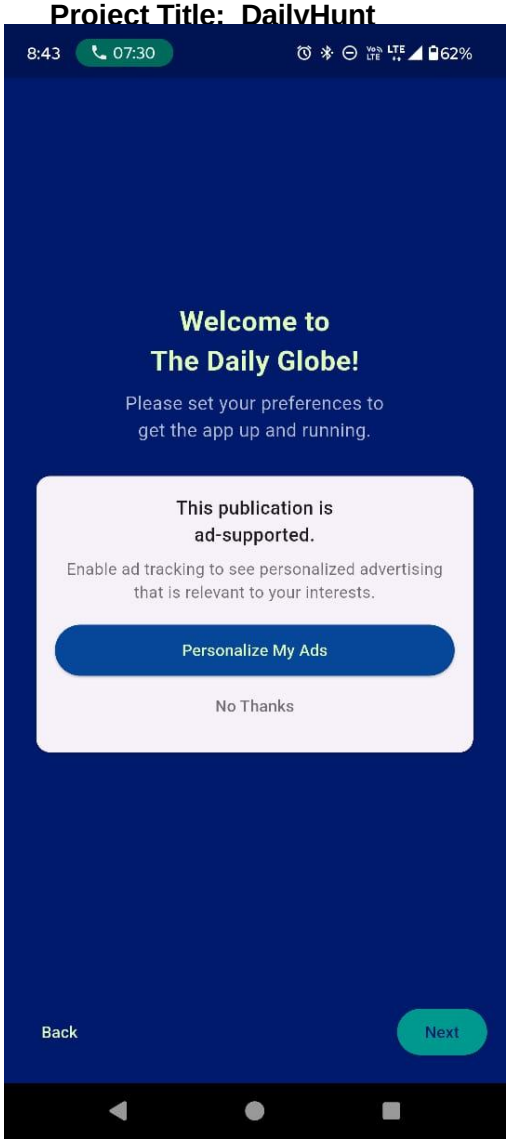
```
                ),
              ],
            ),
          ),
        ),
      ],
    ),
  ),
 );
 }
}

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

 Future<void>
requestNotificationPermission(BuildContext context)
async {
  var status = await
Permission.notification.request();

  if (status.isGranted) {
   // Show success message
   // ignore: use_build_context_synchronously
   ScaffoldMessenger.of(context).showSnackBar(
    const SnackBar(content: Text('Notification
permission granted!')),
   );
  } else if (status.isDenied) {
   // Show warning message
   // ignore: use_build_context_synchronously
   ScaffoldMessenger.of(context).showSnackBar(
    const SnackBar(content: Text('Notification
permission denied!')),
   );
  } else if (status.isPermanentlyDenied) {
   // Open app settings if permanently denied
   // ignore: use_build_context_synchronously
   ScaffoldMessenger.of(context).showSnackBar(
    const SnackBar(
     content: Text(
       'Notification permission permanently denied.
Enable from settings.'),
    ),
   );
   await openAppSettings();
  }
 }

 @override
 Widget build(BuildContext context) {
  return SafeArea(
   child: Padding(
    padding: const EdgeInsets.all(24.0),
    child: Column(
     mainAxisAlignment:
MainAxisAlignment.center,
     children: [
      const Text(
       'Welcome to\nThe Daily Globe!',
       textAlign: TextAlign.center,
       style: TextStyle(
        color: Color(0xFFE1FFBB),
        fontSize: 24,
        fontWeight: FontWeight.bold,
       ),
      ),
      const SizedBox(height: 8),
      const Text(
       'Please set your preferences to\nget the app
up and running.',
       textAlign: TextAlign.center,
       style: TextStyle(
        color: Colors.white70,
        fontSize: 16,
       ),
      ),
      const SizedBox(height: 24),
      Card(
       child: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
         crossAxisAlignment:
CrossAxisAlignment.stretch,
         mainAxisSize: MainAxisSize.min,
         children: [
          const Text(
           "Don't miss our\ntop stories!",
           textAlign: TextAlign.center,
           style: TextStyle(
            fontSize: 16,
            fontWeight: FontWeight.w500,
           ),
          ),
          const SizedBox(height: 8),
          const Text(
           'Subscribe to our notifications to stay
up to date on the latest breaking news.',
           textAlign: TextAlign.center,
           style: TextStyle(
            fontSize: 14,
            color: Colors.black54,
           ),
          ),
          const SizedBox(height: 16),
          ElevatedButton(
           onPressed: ()=>
requestNotificationPermission(context),
           style: ElevatedButton.styleFrom(
            backgroundColor: const
Color(0xFF074799),
            padding: const
EdgeInsets.symmetric(vertical: 12),
           ),
           child: const Text(
            'Subscribe to Notifications Now',
            style: TextStyle(
             color: Color(0xFFE1FFBB),
            ),
           ),
          ),
          TextButton(
```

```
                    onPressed: () {},
                    child: const Text(

                          'No Thanks',
                      style: TextStyle(color:
Colors.black54),
                    ),
                  ),
                ],
              ),
            ),
          ),
        ],
      ),
    ),
  );
 }
}
```

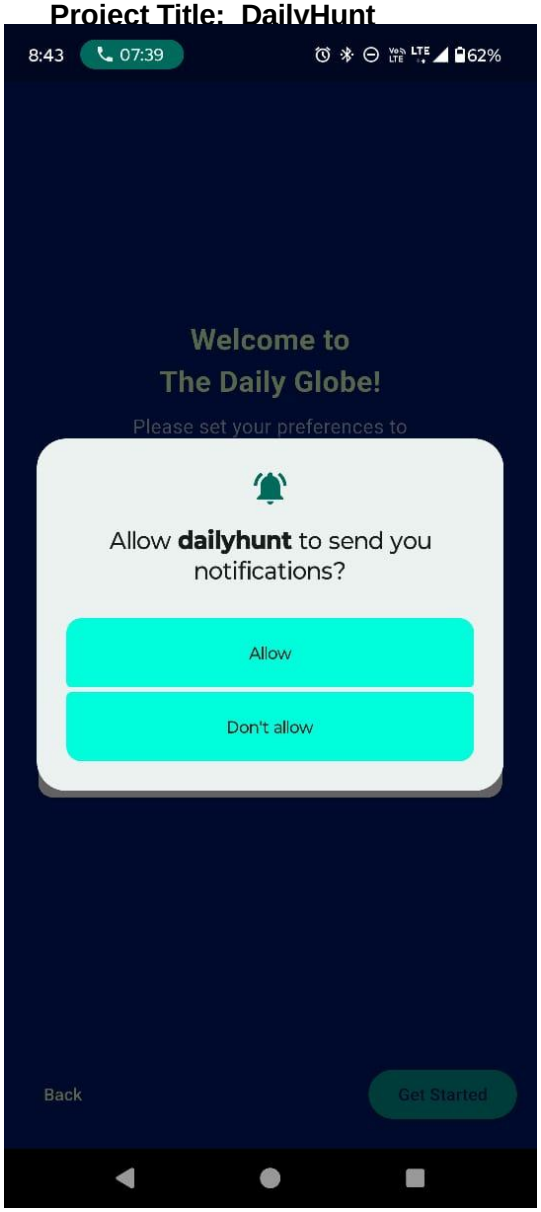
**Project Title: DailyHunt** **Roll No. 60**



**Project Title: DailyHunt** **Roll No. 60**



**Project Title: DailyHunt** **Roll No. 60**



# MAD & PWA Lab
# Journal

| | |
|---|---|
| Experiment No. | 03 |
| Experiment Title. | To include icons, images, fonts in Flutter app |
| Roll No. | Pranav Pramod Titambe |
| Name | 60 |
| Class | D15A |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation |
| Grade: | |

**Project Title: DailyHunt** **Roll No. 60**

**Aim:** To include icons, images, fonts in Flutter app

**Theory:**

## *Using Icons in Flutter*

Icons in Flutter can be added using the built-in Material Icons or custom icon packs.

*(a) Material Icons*

Flutter provides a collection of built-in Material Icons, which can be used with the Icon widget.

Eg: Icon(Icons.home, size: 30, color: Colors.blue)

*(b) Custom Icons*

If you need icons that are not available in the Material Icons set, you can use external icon packs like:

- Font Awesome (font_awesome_flutter package)
- Custom SVG Icons (flutter_svg package)

  Eg in pubspec.yaml file -

  dependencies:
    font_awesome_flutter: ^10.5.0

In code -

```
import 'package:font_awesome_flutter/font_awesome_flutter.dart';

IconButton(
  icon: FaIcon(FontAwesomeIcons.heart, color: Colors.red),
  onPressed: () {},
)
```

## *Adding Images in Flutter*

Images can be loaded in Flutter from different sources like assets, network, or memory.

*(a) Using Network Images*

Network images are loaded from an online URL. Example:

Eg: Image.network("https://example.com/sample.jpg", width: 200, height: 150)

*(b) Using Asset Images*

To use images from the local project folder (assets/), follow these steps:

1. Place the image inside the assets/images/ folder.
2. Declare the image in pubspec.yaml:

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

In code: Image.asset("assets/images/sample.png", width: 200, height: 150)

## *Adding Custom Fonts in Flutter*

Custom fonts improve the visual identity of an app.

*Steps to Add a Custom Font:*

1. Download the font and place it inside the assets/fonts/ folder.
2. Declare the font in pubspec.yaml:

```
flutter:
  fonts:
    - family: CustomFont
      fonts:
        - asset: assets/fonts/CustomFont-Regular.ttf
        - asset: assets/fonts/CustomFont-Bold.ttf
          weight: 700
```

In code -

```
Text(
  "Hello, Flutter!",
  style: TextStyle(fontFamily: "CustomFont", fontSize: 20, fontWeight: FontWeight.bold),
)
```

***main.dart file***

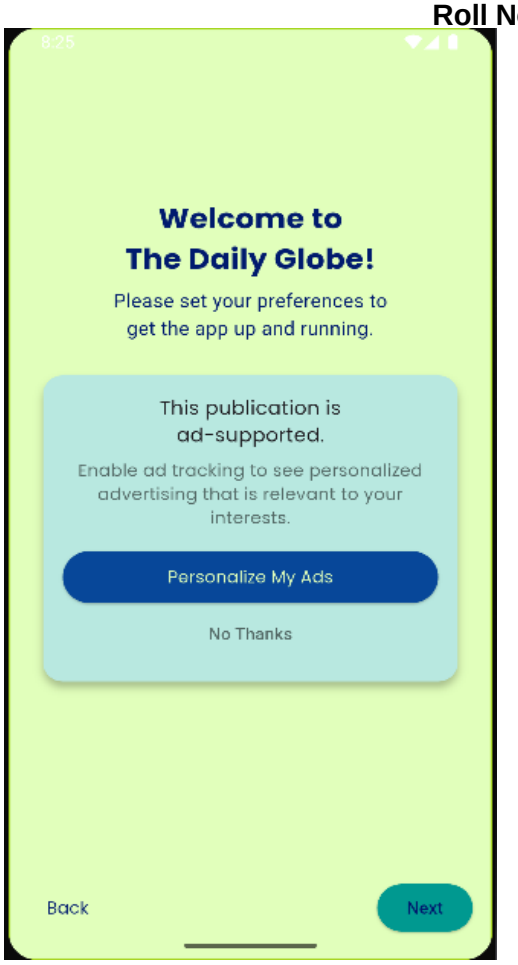
```
import 'package:flutter/material.dart';
import 'package:dailyhunt/login.dart';
import
'package:permission_handler/permission_handle
r.dart';

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: "The Daily Globe",
      color: Color(0xFF001A6E),
      theme: ThemeData(
        primarySwatch: Colors.blue,
        scaffoldBackgroundColor: const
Color(0xFFE1FFBB),
        // textTheme: const TextTheme(
        //   bodyLarge: TextStyle(color:
Color(0xFFEFE9D5)),
        //   bodyMedium: TextStyle(color:
Color(0xFFEFE9D5)),
        //   titleLarge: TextStyle(color:
Color(0xFFEFE9D5)),
        // ),
        // textTheme: const TextTheme(
        //   titleLarge:TextStyle(
        //     color: Color(0xFFEFE9D5)
        //   )
        // )
      ),
      home: const OnboardingFlow(),
      debugShowCheckedModeBanner: false,
    );
  }
}

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

  @override
  State<OnboardingFlow> createState() =>
_OnboardingFlowState();
}

class _OnboardingFlowState extends
State<OnboardingFlow> {
  final PageController _pageController =
PageController();
  int _currentPage = 0;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Column(
        children: [
          Expanded(
            child: PageView(
              controller: _pageController,
              onPageChanged: (index) {
                setState(() {
                  _currentPage = index;
                });
              },
              children: const [
                SplashScreen(),
                TrackingScreen(),
                NotificationScreen(),
              ],
            ),
          ),
          // Page Navigation Buttons
          Padding(
            padding: const
EdgeInsets.symmetric(horizontal: 16, vertical:
20),
            child: Row(
              mainAxisAlignment:
MainAxisAlignment.spaceBetween,
              children: [
                if (_currentPage > 0)
                  TextButton(
                    onPressed: () {
                      _pageController.previousPage(
                        duration: const
Duration(milliseconds: 300),
                        curve: Curves.easeInOut,
                      );
                    },
                    child: const Text(
                      "Back",
                      style: TextStyle(color:
Color(0xFF001A6E),fontFamily:
"CustomPoppins"),
                    ),
                  ),
                ElevatedButton(
                  onPressed: () {
                    if (_currentPage < 2) {
```

```
                    _pageController.nextPage(
                      duration: const
Duration(milliseconds: 300),
                      curve: Curves.easeInOut,
                    );
                   } else {
                    // Navigate to login screen
                    Navigator.pushReplacement(
                      context,
                      MaterialPageRoute(builder:
(context) => const Login()),
                    );
                   }
                  },
                  style: ElevatedButton.styleFrom(
                    foregroundColor:
Color(0xFF001A6E),
                    backgroundColor:
Color(0xFF009990),
                  ),
                  child: Text(_currentPage == 2 ? "Get
Started" : "Next"),
                ),
              ],
            ),
          ),
        ],
      ),
    );
  }
}

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

  @override
  Widget build(BuildContext context) {
    return const Center(
      child: Text(
        'THE DAILY GLOBE',
        style: TextStyle(
          color: Color(0xFF001A6E),
          fontSize: 32,
          fontWeight: FontWeight.bold,
          fontFamily: "CustomPoppins"
        ),
      ),
    );
  }
}

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

  @override
  Widget build(BuildContext context) {
    return SafeArea(
      child: Padding(
        padding: const EdgeInsets.all(24.0),
        child: Column(
          mainAxisAlignment:
MainAxisAlignment.center,
          children: [
            const Text(
              'Welcome to\nThe Daily Globe!',
              textAlign: TextAlign.center,
              style: TextStyle(
                color: Color(0xFF001A6E),
                fontSize: 24,
                fontWeight: FontWeight.bold,
                fontFamily: "CustomPoppins"
              ),
            ),
            const SizedBox(height: 8),
            const Text(
              'Please set your preferences to\nget the
app up and running.',
              textAlign: TextAlign.center,
              style: TextStyle(
                color: Color(0xFF001A6E),
                fontSize: 16,
              ),
            ),
            const SizedBox(height: 24),
            Card(
              color: const Color(0xFFB8E8E0),
              elevation: 5,
              shape: RoundedRectangleBorder(
                borderRadius:
BorderRadius.circular(16),
              ),
              child: Padding(
                padding: const EdgeInsets.all(16.0),
                child: Column(
                  crossAxisAlignment:
CrossAxisAlignment.stretch,
                  mainAxisSize: MainAxisSize.min,
                  children: [
                    const Text(
                      'This publication is\nad-
supported.',
                      textAlign: TextAlign.center,
                      style: TextStyle(
                        fontSize: 16,
                        fontWeight: FontWeight.w500,
                        fontFamily: "CustomPoppins"
                      ),
                    ),
                    const SizedBox(height: 8),
                    const Text(
```

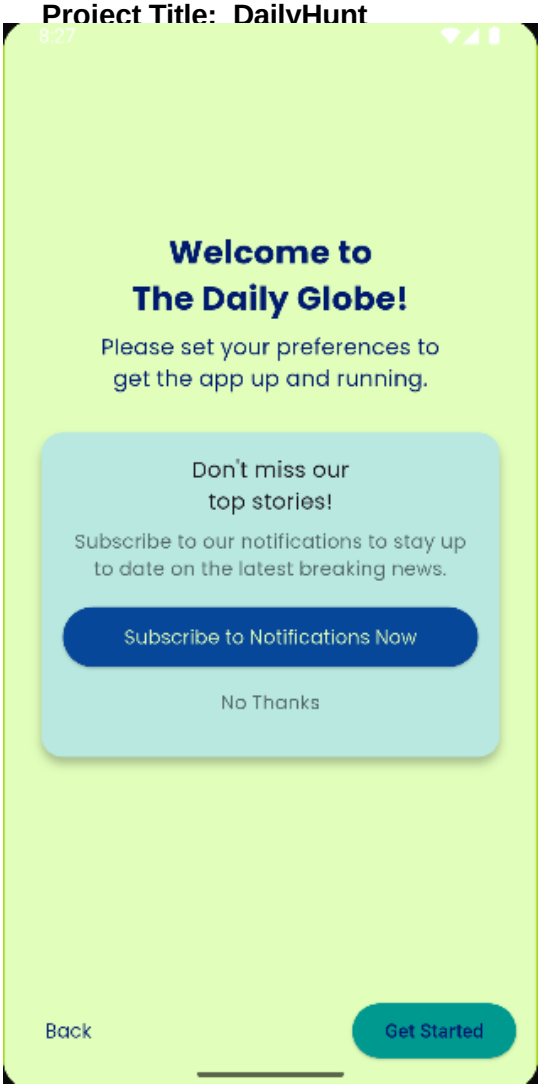
```
                'Enable ad tracking to see
personalized advertising that is relevant to your
interests.',
                textAlign: TextAlign.center,
                style: TextStyle(
                  fontSize: 14,
                  color: Colors.black54,
                  fontFamily: "CustomPoppins"
                ),
              ),
              const SizedBox(height: 16),
              ElevatedButton(
                onPressed: () {},
                style: ElevatedButton.styleFrom(
                  backgroundColor: const
Color(0xFF074799),
                  padding: const
EdgeInsets.symmetric(vertical: 12),
                ),
                child: const Text(
                  'Personalize My Ads',
                  style: TextStyle(
                    color: Color(0xFFE1FFBB),
                    fontFamily: "CustomPoppins",
                  ),
                ),
              ),
              TextButton(
                onPressed: () {
                  print('No Thanks');
                },
                child: const Text(
                  'No Thanks',
                  style: TextStyle(color:
Colors.black54),
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }
}

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

  Future<void>
requestNotificationPermission(BuildContext
context) async {
    var status = await
Permission.notification.request();

    if (status.isGranted) {
      // Show success message
      // ignore: use_build_context_synchronously

ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Notification
permission granted!')),
      );
    } else if (status.isDenied) {
      // Show warning message
      // ignore: use_build_context_synchronously

ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Notification
permission denied!')),
      );
    } else if (status.isPermanentlyDenied) {
      // Open app settings if permanently denied
      // ignore: use_build_context_synchronously

ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(
          content: Text(
            'Notification permission permanently
denied. Enable from settings.'),
        ),
      );
      await openAppSettings();
    }
  }

  @override
  Widget build(BuildContext context) {
    return SafeArea(
      child: Padding(
        padding: const EdgeInsets.all(24.0),
        child: Column(
          mainAxisAlignment:
MainAxisAlignment.center,
          children: [
            const Text(
              'Welcome to\nThe Daily Globe!',
              textAlign: TextAlign.center,
              style: TextStyle(
                color: Color(0xFF001A6E),
                fontSize: 24,
                fontWeight: FontWeight.bold,
                fontFamily: "CustomPoppins"
              ),
            ),
            const SizedBox(height: 8),
            const Text(
```

```
          'Please set your preferences to\nget the
app up and running.',
          textAlign: TextAlign.center,
          style: TextStyle(
            color: Color(0xFF001A6E),
            fontSize: 16,
            fontFamily: "CustomPoppins"
          ),
        ),
        const SizedBox(height: 24),
        Card(
          color: const Color(0xFFB8E8E0),
          elevation: 5,
          shape: RoundedRectangleBorder(
            borderRadius:
BorderRadius.circular(16),
          ),
          child: Padding(
            padding: const EdgeInsets.all(16.0),
            child: Column(
              crossAxisAlignment:
CrossAxisAlignment.stretch,
              mainAxisSize: MainAxisSize.min,
              children: [
                const Text(
                  "Don't miss our\ntop stories!",
                  textAlign: TextAlign.center,
                  style: TextStyle(
                    fontSize: 16,
                    fontWeight: FontWeight.w500,
                    fontFamily: "CustomPoppins"
                  ),
                ),
                const SizedBox(height: 8),
                const Text(
                  'Subscribe to our notifications to
stay up to date on the latest breaking news.',
                  textAlign: TextAlign.center,
                  style: TextStyle(
                    fontSize: 14,
                    color: Colors.black54,
                    fontFamily: "CustomPoppins"
                  ),
                ),
                const SizedBox(height: 16),
                ElevatedButton(
                  onPressed: () =>
requestNotificationPermission(context),
                  style: ElevatedButton.styleFrom(
                    backgroundColor: const
Color(0xFF074799),
                    padding: const
EdgeInsets.symmetric(vertical: 12),
                  ),
                  child: const Text(
                    'Subscribe to Notifications Now',
                    style: TextStyle(
                      color: Color(0xFFE1FFBB),
                      fontFamily: "CustomPoppins"
                    ),
                  ),
                ),
                TextButton(
                  onPressed: () {},
                  child: const Text(
                    'No Thanks',
                    style: TextStyle(color:
Colors.black54,fontFamily: "CustomPoppins"),
                  ),
                ),
              ],
            ),
          ),
        ),
      ],
    ),
  );
 }
}
```

**Project Title: DailyHunt** **Roll No. 60**

**Project Title: DailyHunt** **Roll No. 60**



***Login.dart file***

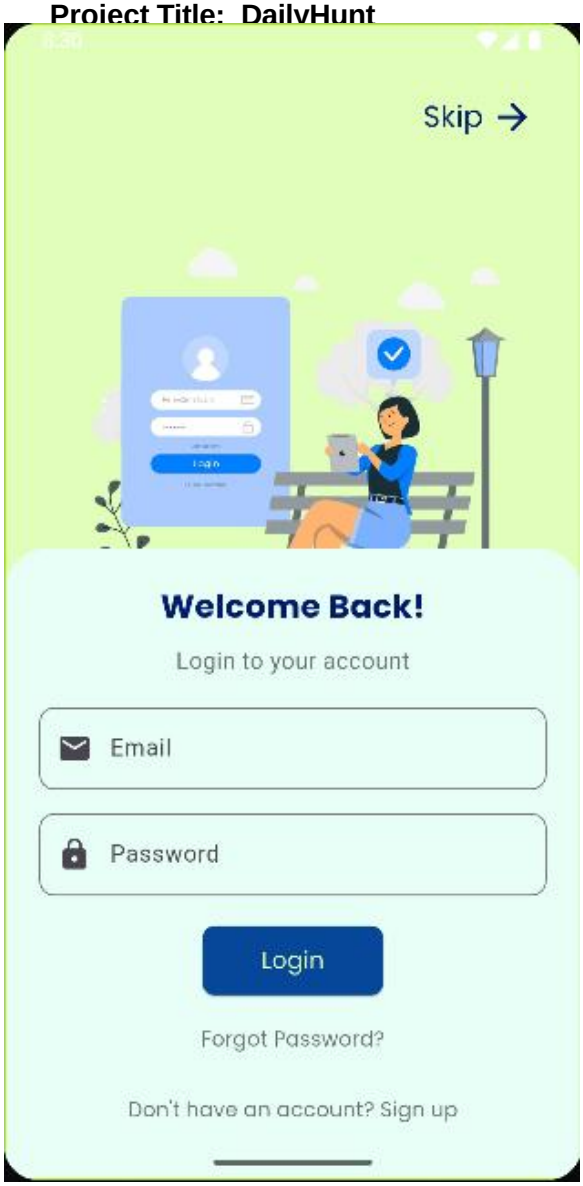
```
import 'package:dailyhunt/languages.dart';
import 'package:flutter/material.dart';
import 'home.dart';

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

  @override
  State<Login> createState() => _LoginState();
}

class _LoginState extends State<Login> {
  final TextEditingController emailController =
TextEditingController();
  final TextEditingController passwordController =
TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Stack(
        children: [
          // Background Image
          Align(
            alignment: Alignment.topCenter,
            child: Container(
              height: 580, // Adjust the height as needed
              decoration: BoxDecoration(
                image: DecorationImage(
                  image:
AssetImage("assets/background.png"),
                  fit: BoxFit.scaleDown,
                ),
              ),
            ),
          ),

          // Skip Button
          Positioned(
            top: 40,
            right: 20,
            child: TextButton(
              onPressed: () {
                debugPrint("Skipped");
                Navigator.push(context,
MaterialPageRoute(builder: (context) {
                  return Languages();
                }));
              },
              child: Row(
                mainAxisSize: MainAxisSize.min,
                children: const [
                  Text(
                    "Skip",
                    style: TextStyle(fontSize: 20, color:
Color(0xFF001A6E),fontFamily: "CustomPoppins"),
                  ),
                  SizedBox(width: 5),
                  Icon(Icons.arrow_forward, size: 30, color:
Color(0xFF001A6E)),
                ],
              ),
            ),
          ),

          // Login Card
          Align(
            alignment: Alignment.bottomCenter,
            child: Container(
              width: double.infinity,
              padding: const EdgeInsets.all(24),
              decoration: const BoxDecoration(
                color: Color.fromARGB(255, 231, 255, 246),
// Light Teal Card
                borderRadius: BorderRadius.only(
                  topLeft: Radius.circular(30),
                  topRight: Radius.circular(30),
                ),
              ),
              child: Column(
                mainAxisSize: MainAxisSize.min,
                children: [
                  const Text(
                    "Welcome Back!",
                    style: TextStyle(
                      fontSize: 22,
                      fontWeight: FontWeight.bold,
                      color: Color(0xFF001A6E),
                      fontFamily: "CustomPoppins"
                    ),
                  ),
                  const SizedBox(height: 10),
                  const Text(
                    "Login to your account",
                    style: TextStyle(fontSize: 16, color:
Colors.black54),
                  ),
                  const SizedBox(height: 20),

                  // Email Field
                  TextField(
                    controller: emailController,
                    decoration: InputDecoration(
                      labelText: "Email",
                      prefixIcon: const Icon(Icons.email),
                      border: OutlineInputBorder(
                        borderRadius: BorderRadius.circular(10),
                      ),
                    ),
                  ),
                  const SizedBox(height: 16),
```

```
            // Password Field
            TextField(
              controller: passwordController,
              obscureText: true,
              decoration: InputDecoration(
                labelText: "Password",
                prefixIcon: const Icon(Icons.lock),
                border: OutlineInputBorder(
                  borderRadius: BorderRadius.circular(10),
                ),
              ),
            ),
            const SizedBox(height: 20),

            // Login Button
            ElevatedButton(
              onPressed: () {
                // TODO: Implement login logic
              },
              style: ElevatedButton.styleFrom(
                backgroundColor: const
Color(0xFF074799), // Deep Blue
                padding: const EdgeInsets.symmetric(
                    horizontal: 40, vertical: 12),
                shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.circular(8),
                ),
              ),
              child: const Text(
                "Login",
                style: TextStyle(
                  fontSize: 16,
                  color: Color(0xFFE1FFBB), // Light
Greenish-Yellow
                  fontFamily: "CustomPoppins"
                ),
              ),
            ),
            const SizedBox(height: 5),

            // Forgot Password & Signup Text
            TextButton(
              onPressed: () {
                // TODO: Implement Forgot Password
Logic
              },
              child: const Text(
                "Forgot Password?",
                style: TextStyle(color:
Colors.black54,fontFamily: "CustomPoppins"),
              ),
            ),
            TextButton(
              onPressed: () {
                // TODO: Implement Signup Navigation
              },
              child: const Text(
                "Don't have an account? Sign up",
                style: TextStyle(color:
Colors.black54,fontFamily: "CustomPoppins"),
              ),
            ),
          ],
        ),
      ),
    ),
  ],
 ),
 );
 }
}
```

***pubspec.yaml***

```
flutter:
  fonts:
    - family: CustomPoppins
      fonts:
        - asset: assets/fonts/Poppins-Regular.ttf
        - asset: assets/fonts/Poppins-Bold.ttf
  uses-material-design: true
  assets:
    - assets/background.png
        - asset: assets/Poppins-Bold.ttf
          weight: 700
```

**Project Title: DailyHunt** **Roll No. 60**



# MAD & PWA Lab
# Journal

| | |
|---|---|
| Experiment No. | 04 |
| Experiment Title. | To create an interactive Form using form widget |
| Roll No. | 60 |
| Name | Pranav Pramod Titambe |
| Class | D15A |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation |
| Grade: | |

**Project Title: DailyHunt** **Roll No. 60**

**Aim:**

To design Flutter UI by including common widgets.

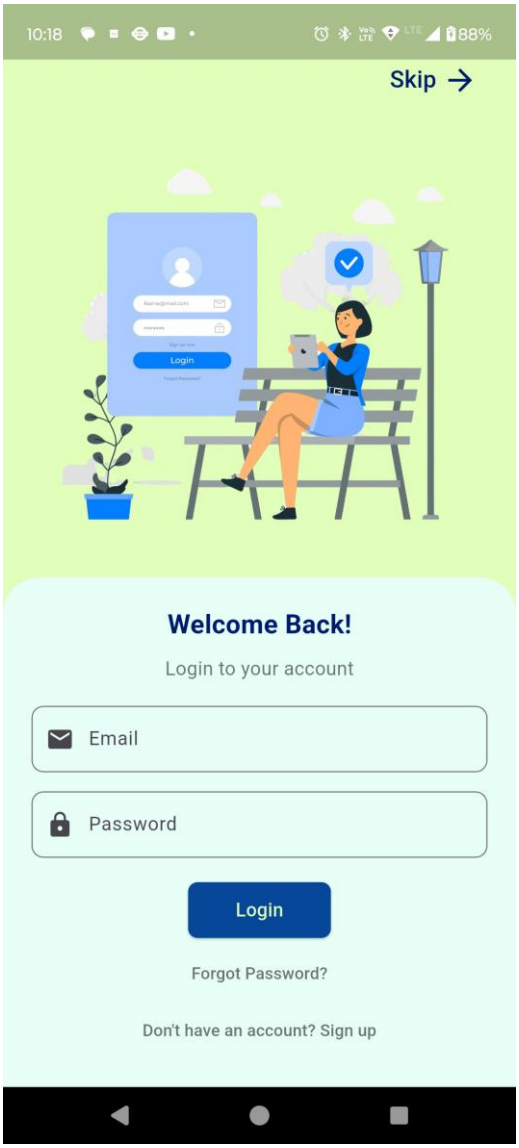
**Program:**

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

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

  @override
  State<Login> createState() =>
_LoginState();
}

class _LoginState extends State<Login> {
  final TextEditingController emailController
= TextEditingController();
  final TextEditingController
passwordController =
TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Stack(
        children: [
          // Background Image
          Align(
            alignment: Alignment.topCenter,
            child: Container(
              height: 580, // Adjust the height as
needed
              decoration: BoxDecoration(
                image: DecorationImage(
                  image:
AssetImage("assets/background.png"),
                  fit: BoxFit.scaleDown,
                ),
              ),
            ),
          ),

          // Skip Button
          Positioned(
            top: 40,
            right: 20,
            child: TextButton(
              onPressed: () {
                debugPrint("Skipped");
              },
              child: Row(
                mainAxisSize: MainAxisSize.min,
                children: const [
                  Text(
                    "Skip",
                    style: TextStyle(fontSize: 20,
color: Color(0xFF001A6E)),
                  ),
                  SizedBox(width: 5),
                  Icon(Icons.arrow_forward, size:
30, color: Color(0xFF001A6E)),
                ],
              ),
            ),
          ),

          // Login Card
          Align(
            alignment: Alignment.bottomCenter,
            child: Container(
              width: double.infinity,
              padding: const EdgeInsets.all(24),
              decoration: const BoxDecoration(
                color: Color.fromARGB(255, 231,
255, 246), // Light Teal Card
                borderRadius: BorderRadius.only(
                  topLeft: Radius.circular(30),
                  topRight: Radius.circular(30),
                ),
              ),
              child: Column(
                mainAxisSize: MainAxisSize.min,
                children: [
                  const Text(
                    "Welcome Back!",
                    style: TextStyle(
                      fontSize: 22,
                      fontWeight: FontWeight.bold,
                      color: Color(0xFF001A6E),
                    ),
                  ),
                  const SizedBox(height: 10),
                  const Text(
                    "Login to your account",
                    style: TextStyle(fontSize: 16,
color: Colors.black54),
                  ),
                  const SizedBox(height: 20),
```

```
            // Email Field
            TextField(
              controller: emailController,
              decoration: InputDecoration(
                labelText: "Email",
                prefixIcon: const
Icon(Icons.email),
                border: OutlineInputBorder(
                  borderRadius:
BorderRadius.circular(10),
                ),
              ),
            ),
            const SizedBox(height: 16),

            // Password Field
            TextField(
              controller: passwordController,
              obscureText: true,
              decoration: InputDecoration(
                labelText: "Password",
                prefixIcon: const
Icon(Icons.lock),
                border: OutlineInputBorder(
                  borderRadius:
BorderRadius.circular(10),
                ),
              ),
            ),
            const SizedBox(height: 20),

            // Login Button
            ElevatedButton(
              onPressed: () {
                // TODO: Implement login
logic
              },
              style: ElevatedButton.styleFrom(
                backgroundColor: const
Color(0xFF074799), // Deep Blue
                padding: const
EdgeInsets.symmetric(
                    horizontal: 40, vertical: 12),
                shape:
RoundedRectangleBorder(
                  borderRadius:
BorderRadius.circular(8),
                ),
              ),
              child: const Text(
                "Login",
                style: TextStyle(
                  fontSize: 16,
                  color: Color(0xFFE1FFBB), //
Light Greenish-Yellow
                ),
              ),
            ),
            const SizedBox(height: 5),

            // Forgot Password & Signup Text
            TextButton(
              onPressed: () {
                // TODO: Implement Forgot
Password Logic
              },
              child: const Text(
                "Forgot Password?",
                style: TextStyle(color:
Colors.black54),
              ),
            ),
            TextButton(
              onPressed: () {
                // TODO: Implement Signup
Navigation
              },
              child: const Text(
                "Don't have an account? Sign
up",
                style: TextStyle(color:
Colors.black54),
              ),
            ),
          ],
        ),
      ),
    ],
  ),
 );
 }
}
```

**Project Title: DailyHunt** **Roll No. 60**

**Output:**




Common widgets used-

1. Scaffold
    - The main structure of the screen.
    - Provides the base layout like body.

2. Stack
    - Places widgets on top of each other (background image, button, and login form).

3. Container
    - Used for styling and layout purposes (background image, white login box).

4. BoxDecoration
    - Adds styling inside Container (background image, rounded corners).

5. DecorationImage
    - Used inside BoxDecoration to set an image as a background.

6. Align
    - Positions widgets at specific places (Skip button at the top-right, login box at the bottom).

7. Padding
    - Adds spacing around widgets.

8. TextButton
    - Used for the "Skip" button.

9. Column
    - Arranges widgets vertically inside the login box.

10. Text
    - Displays static text (NYKAA FASHION, Login or Signup, Stay stylish...).

11. SizedBox
    - Adds empty space between widgets.

12. TextField
    - Takes user input (mobile number).

13. InputDecoration
    - Adds a border and placeholder text (Enter mobile no.) inside TextField.

14. OutlineInputBorder
    - Provides a border for the TextField.

15. ElevatedButton
    - The "Get OTP" button.

16. TextStyle
    - Used to style text (color, font size, weight).

17. BorderRadius
    - Rounds corners of the white login box.

# MAD & PWA Lab
# Journal

| | |
|---|---|
| Experiment No. | 05 |
| Experiment Title. | To apply navigation, routing and gestures in Flutter App |
| Roll No. | 60 |
| Name | Pranav Pramod Titambe |
| Class | D15A |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation |
| Grade: | |

**Project Title: DailyHunt** **Roll No. 60**

**Aim:** To apply navigation, routing and gestures in Flutter App

**Theory:**

Flutter provides tools to handle navigation, routing, and gestures, allowing users to move between screens and interact with the app smoothly. These features help create a user-friendly experience in mobile applications.

---

**1. Navigation in Flutter**

Navigation is the process of moving between different screens (or pages) in a Flutter app. Flutter uses a stack-based approach for navigation, where new screens are pushed onto the stack and removed when the user navigates back.

Types of Navigation:

- Push Navigation: Moves to a new screen and adds it to the stack.
- Pop Navigation: Removes the current screen and returns to the previous one.
- Named Routes: Uses pre-defined route names to navigate.
- Navigation with Data: Allows passing data between screens when navigating.

---

**2. Routing in Flutter**

Routing helps in managing different screens efficiently. Instead of manually handling each screen transition, Flutter allows defining routes in a structured way.

Types of Routing:

- Direct Routing: Navigates to a specific screen using explicit methods.
- Named Routing: Uses a predefined route name to navigate, making the app more organized.

Routing improves app maintainability, especially in apps with multiple screens.

---

**3. Gestures in Flutter**

Gestures enable user interaction in Flutter applications. Flutter provides built-in gesture detection capabilities for touch-based interactions.

Common Gestures:

- Tap: A single touch interaction.
- Double Tap: Two quick consecutive taps.
- Long Press: Holding a touch for a longer duration.
- Swipe: Moving a finger across the screen.
- Drag: Moving an object by pressing and holding it.

Gestures are essential for making apps interactive and responsive.

---

**4. Combining Navigation and Gestures**

Navigation and gestures can be combined to enhance user experience. For example:

- Tapping on a button can navigate to another screen.
- Swiping a card can delete an item or move to another page.
- Dragging an element can reposition items within the app.

---

Navigation, routing, and gestures are fundamental to creating an interactive Flutter application. Navigation allows movement between screens, routing helps manage screens efficiently, and gestures enable touch interactions. Mastering these concepts helps in developing dynamic and user-friendly Flutter applications.

**Code:**

***login.dart file***

```
import 'package:dailyhunt/languages.dart';
import 'package:flutter/material.dart';
import 'home.dart';

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

  @override
  State<Login> createState() =>
_LoginState();
}

class _LoginState extends State<Login> {
  final TextEditingController
emailController = TextEditingController();
  final TextEditingController
passwordController =
TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Stack(
        children: [
          // Background Image
          Align(
            alignment: Alignment.topCenter,
            child: Container(
              height: 580, // Adjust the height as
needed
              decoration: BoxDecoration(
                image: DecorationImage(
                  image:
AssetImage("assets/background.png"),
                  fit: BoxFit.scaleDown,
                ),
              ),
            ),
          ),

          // Skip Button
          Positioned(
            top: 40,
            right: 20,
            child: TextButton(
              onPressed: () {
                debugPrint("Skipped");
                Navigator.push(context,
MaterialPageRoute(builder: (context) {
                  return Languages();
                }));
              },
```

```
          child: Row(
            mainAxisSize: MainAxisSize.min,
            children: const [
              Text(
                "Skip",
                style: TextStyle(fontSize: 20,
color: Color(0xFF001A6E)),
              ),
              SizedBox(width: 5),
              Icon(Icons.arrow_forward, size:
30, color: Color(0xFF001A6E)),
            ],
          ),
        ),
      ),

      // Login Card
      Align(
        alignment: Alignment.bottomCenter,
        child: Container(
          width: double.infinity,
          padding: const EdgeInsets.all(24),
          decoration: const BoxDecoration(
            color: Color.fromARGB(255, 231,
255, 246), // Light Teal Card
            borderRadius: BorderRadius.only(
              topLeft: Radius.circular(30),
              topRight: Radius.circular(30),
            ),
          ),
          child: Column(
            mainAxisSize: MainAxisSize.min,
            children: [
              const Text(
                "Welcome Back!",
                style: TextStyle(
                  fontSize: 22,
                  fontWeight: FontWeight.bold,
                  color: Color(0xFF001A6E),
                ),
              ),
              const SizedBox(height: 10),
              const Text(
                "Login to your account",
                style: TextStyle(fontSize: 16,
color: Colors.black54),
              ),
              const SizedBox(height: 20),

              // Email Field
              TextField(
                controller: emailController,
                decoration:
InputDecoration(
                  labelText: "Email",
                  prefixIcon: const
Icon(Icons.email),
                  border:
OutlineInputBorder(
                    borderRadius:
BorderRadius.circular(10),
                  ),
                ),
              ),
              const SizedBox(height:
16),

              // Password Field
              TextField(
                controller:
passwordController,
                obscureText: true,
                decoration:
InputDecoration(
                  labelText: "Password",
                  prefixIcon: const
Icon(Icons.lock),
                  border:
OutlineInputBorder(
                    borderRadius:
BorderRadius.circular(10),
                  ),
                ),
              ),
              const SizedBox(height:
20),

              // Login Button
              ElevatedButton(
                onPressed: () {
                  // TODO: Implement
login logic
                },
                style:
ElevatedButton.styleFrom(
                  backgroundColor:
const Color(0xFF074799), // Deep
Blue
                  padding: const
EdgeInsets.symmetric(
                    horizontal: 40,
vertical: 12),
```

```
            hape:
        RoundedRectangleBorder(
                      borderRadius:
        BorderRadius.circular(8),
                    ),
                  ),
                  child: const Text(
                    "Login",
                    style: TextStyle(
                      fontSize: 16,
                      color:
        Color(0xFFE1FFBB), // Light
        Greenish-Yellow
                    ),
                  ),
                ),
                const SizedBox(height:
        5),

                // Forgot Password &
        Signup Text
                TextButton(
                  onPressed: () {
                    // TODO: Implement
        Forgot Password Logic
                  },
                  child: const Text(
                    "Forgot Password?",
                    style: TextStyle(color:
        Colors.black54),
                  ),
                ),
                TextButton(
                  onPressed: () {
                    // TODO: Implement
        Signup Navigation
                  },
                  child: const Text(
                    "Don't have an
        account? Sign up",
                    style: TextStyle(color:
        Colors.black54),
                  ),
                ),
              ],
            ),
          ),
        ],
      ),
    );
  }
}
```

***languages.dart file***

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

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

  @override
```

```
State<Languages> createState() =>
_LanguagesState();
}

class _LanguagesState extends
State<Languages> {
  // Dictionary with language names as keys
and language codes as values
  final Map<String, String> languages = {
    "English": "en",
    "Spanish": "es",
    "French": "fr",
    "German": "de",
    "Hindi": "hi",
    "Mandarin": "zh",
    "Japanese": "ja",
    "Arabic": "ar",
    "Marathi":"mr"
  };

  // Store selected language code
  String? _selectedLanguageCode;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("Select Language"),
        backgroundColor: const
Color(0xFF001A6E),
        foregroundColor: Colors.white,
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment:
CrossAxisAlignment.start,
          children: [
            const Text(
              "Languages",
              style: TextStyle(
                color: Color(0xFF001A6E),
                fontSize: 20,
                fontWeight: FontWeight.bold,
              ),
            ),
            const SizedBox(height: 10),
            Expanded(
              child: ListView(
                children:
languages.entries.map((entry) {
return Card(
                  elevation: 2,
                  child: RadioListTile<String>(
                    title: Text(entry.key), //
Display language name
                    value: entry.value, // Use
language code as value
                    groupValue:
_selectedLanguageCode,
                    onChanged: (value) {
                      setState(() {
                        _selectedLanguageCode =
value;
                      });
                    },
                  ),
                );
              }).toList(),
            ),
          ),
        ],
      ),
    ),
    floatingActionButton:
FloatingActionButton.extended(
      onPressed: () {
        if (_selectedLanguageCode != null) {
          debugPrint("Selected Language:
$_selectedLanguageCode");
          // Pass selected language code to
API or next screen

ScaffoldMessenger.of(context).showSnackB
ar(
            SnackBar(content: Text("Selected
Language: $_selectedLanguageCode")),
          );
          Navigator.push(context,
MaterialPageRoute(builder: (context) {
            return Home(lang:
_selectedLanguageCode!);
          }));
        }
      },
      label: const Text("Confirm",style:
TextStyle(color: Colors.white),),
      icon: const Icon(Icons.check,color:
Colors.white,),
      backgroundColor: const
Color(0xFF001A6E),
    ),
```

```
 );
  }
}
```

***home.dart file***

```
import
'package:dailyhunt/NewsPageDetail.dart';
import
'package:dailyhunt/api/news_api.dart';
import
'package:dailyhunt/model/news_model.dart';
import 'package:flutter/material.dart';
import 'package:intl/intl.dart';

class Home extends StatefulWidget {
  final String lang;
  Home({super.key, required this.lang});

  @override
  State<Home> createState() =>
_HomeState();
}

class _HomeState extends State<Home> {
  List<NewsModel> newsList = [];
  NewsApi newsApi = NewsApi();

  Future<void> getTopNews() async {
    List<NewsModel> fetchedNews = await
newsApi.getTopHeadlines(widget.lang);
    setState(() {
      newsList = fetchedNews;
    });
  }

  int _selectedIndex = 0;

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

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

  String formatDate(DateTime date) {
    String day =
DateFormat('d').format(date);
    String suffix =
getDaySuffix(int.parse(day));
```

```
 String formattedDate = "$day$suffix
${DateFormat('MMM
yyyy').format(date)}";
   return formattedDate;
  }

  String getDaySuffix(int day) {
   if (day >= 11 && day <= 13) {
    return "th";
   }
   switch (day % 10) {
    case 1:
     return "st";
    case 2:
     return "nd";
    case 3:
     return "rd";
    default:
     return "th";
   }
  }

  @override
  Widget build(BuildContext context) {
   return Scaffold(
    appBar: AppBar(
     backgroundColor: Colors.indigo[900],
     foregroundColor: Colors.green[200],
     title: const Text("The Daily Globe"),
     titleTextStyle: const TextStyle(
      fontWeight: FontWeight.bold,
      fontSize: 20,
     ),
     elevation: 0,
     automaticallyImplyLeading: false,
     actions: [
      IconButton(
       icon: const Icon(Icons.notifications,
color: Colors.white),
       onPressed: () {},
      ),
     ],
    ),
    body: Padding(
     padding: const EdgeInsets.all(5.0),
     child: SingleChildScrollView(
      child: Column(
       children: [
        Column(
         crossAxisAlignment:
CrossAxisAlignment.start,
children: [
          const Padding(padding: const
EdgeInsets.all(16),
          child: Column(
           crossAxisAlignment:
CrossAxisAlignment.start,
           children: [
            const Text(
             'Welcome back, Tyler!',
             style: TextStyle(fontSize: 24,
fontWeight: FontWeight.bold),
            ),
            const SizedBox(height: 5),
            const Text(
             'Discover a world of news
that matters to you',
             style: TextStyle(fontSize: 16,
color: Colors.grey),
            ),
            const SizedBox(height: 20),
            const SectionHeader(title:
'Trending news'),
           ],
          ),),

          SizedBox(
           height:
MediaQuery.of(context).size.height * 0.35,
           child: newsList.isEmpty
             ? const Center(child:
CircularProgressIndicator())
             : ListView.builder(
            scrollDirection:
Axis.horizontal,
            itemCount: newsList.length,
            itemBuilder: (context, index) {
             return NewsCard(
              category: 'Business',
              title: newsList[index].title,
              source:
newsList[index].source,
              image:
newsList[index].image,
              date: DateFormat('d MMM
yyyy')

.format(newsList[index].publishedAt),

content:newsList[index].content
             );
            },
```

```
              ),
              const SizedBox(height: 20),
              const SectionHeader(title:
'Recommendation'),
              const RecommendationCard(),
            ],
          ),
        ],
      ),
    ),
    bottomNavigationBar: Container(
      decoration: BoxDecoration(
        color: Colors.indigo[900],
        borderRadius: const
BorderRadius.only(
          topLeft: Radius.circular(20),
          topRight: Radius.circular(20),
        ),
        boxShadow: [
          BoxShadow(
            color:
Colors.black.withOpacity(0.3),
            blurRadius: 10,
            spreadRadius: 2,
          ),
        ],
      ),
      child: ClipRRect(
        borderRadius: const
BorderRadius.only(
          topLeft: Radius.circular(20),
          topRight: Radius.circular(20),
        ),
        child: BottomNavigationBar(
          backgroundColor:
Colors.indigo[900],
          currentIndex: _selectedIndex,
          onTap: _onItemTapped,
          selectedItemColor:
Colors.green[200],
          unselectedItemColor:
Colors.white70,
          showUnselectedLabels: false,
          type:
BottomNavigationBarType.fixed,
          items: const [
            BottomNavigationBarItem(
              icon: Icon(Icons.home, size: 28),
              label: 'Home',
            ),
            BottomNavigationBarItem(
              icon: Icon(Icons.search, size: 28),
              label: 'Search',
            ),
            BottomNavigationBarItem(
              icon: Icon(Icons.person, size: 28),
              label: 'Profile',
            ),
          ],
        ),
      ),
    ),
  );
 }
}

class SectionHeader extends
StatelessWidget {
 final String title;
 const SectionHeader({super.key, required
this.title});

 @override
 Widget build(BuildContext context) {
  return Row(
    mainAxisAlignment:
MainAxisAlignment.spaceBetween,
    children: [
      Text(
        title,
        style: const TextStyle(fontSize: 18,
fontWeight: FontWeight.bold),
      ),
      TextButton(
        onPressed: () {},
        child: Text('See all', style:
TextStyle(color: Colors.indigo[900])),
      ),
    ],
  );
 }
}

class NewsCard extends StatelessWidget {
 final String category, title, source, date,
image,content;
 const NewsCard(
    {super.key,
     required this.category,
     required this.title,
     required this.source,
```

```
required this.image,
      required this.date,
      required this.content});

  @override
  Widget build(BuildContext context) {
   return GestureDetector(
     onTap: (){
      Navigator.push(context,
MaterialPageRoute(builder:
(context)=>NewsDetailPage(title: title,
source: source, publishedAt: date, image:
image, content:content )));
     },
     child: Container(
      width: 250,
      margin: const EdgeInsets.only(right:
10),
      decoration: BoxDecoration(
       color: const Color(0xFF009990),
       borderRadius:
BorderRadius.circular(10),
       boxShadow: [BoxShadow(color:
Colors.grey.shade300, blurRadius: 5)],
      ),
      child: Column(
       crossAxisAlignment:
CrossAxisAlignment.start,
       children: [
        Container(
         height: 100,
         decoration: BoxDecoration(
          borderRadius:
          const BorderRadius.vertical(top:
Radius.circular(10)),
          image: DecorationImage(
           image: NetworkImage(image),
           fit: BoxFit.cover,
          ),
         ),
        ),
        Padding(
         padding: const EdgeInsets.all(8.0),
         child: Column(
          crossAxisAlignment:
CrossAxisAlignment.start,
          children: [
           Container(
            decoration: BoxDecoration(
             borderRadius:
BorderRadius.circular(10),
color: Colors.indigo[900],
             ),
             height: 20,
             width: 100,
             alignment: Alignment.center,
             child: Text(
              category,
              style: const TextStyle(color:
Colors.white),
             ),
            ),
            Text(title,
              style: const TextStyle(
                fontSize: 16, fontWeight:
FontWeight.bold)),
            const SizedBox(height: 5),
            Text('$source • $date', style:
const TextStyle(color: Colors.white)),
           ],
          ),
         ),
        ],
       ),
     ),
   );
  }
}

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

  @override
  State<RecommendationCard> createState()
=> _RecommendationCardState();
}

class _RecommendationCardState extends
State<RecommendationCard> {
  @override
  Widget build(BuildContext context) {
   return Container(
     padding: const EdgeInsets.all(16),
     decoration: BoxDecoration(
      color: Color(0xFF009990),
      borderRadius:
BorderRadius.circular(10),
      boxShadow: [BoxShadow(color:
Colors.grey.shade300, blurRadius: 5)],
     ),
     child: Column(
```

```
  crossAxisAlignment:
CrossAxisAlignment.start,
        children: [
          Row(
            mainAxisAlignment:
MainAxisAlignment.spaceBetween,
            children: [
              Row(
                children: const [
                  Icon(Icons.business, color:
Colors.black),
                  SizedBox(width: 5),
                  Text('Forbes', style:
TextStyle(fontWeight: FontWeight.bold)),
                ],
              ),
              TextButton(onPressed: () {}, child:
const Text('Follow')),
            ],
          ),
          const SizedBox(height: 5),
          const Text(
            'Tech Startup Secures \$50 Million
Funding for Expansion',
            style: TextStyle(fontSize: 16,

fontWeight: FontWeight.bold),
          ),
          const SizedBox(height: 5),
          Container(
            decoration: BoxDecoration(
              borderRadius:
BorderRadius.circular(10),
              color: Colors.indigo[900],
            ),
            height: 20,
            width: 100,
            alignment: Alignment.center,
            child: Text(
              "Business",
              style: const TextStyle(
                color: Colors.white,
              ),
            ),
          ),
        ],
      ),
    );
  }
}
```

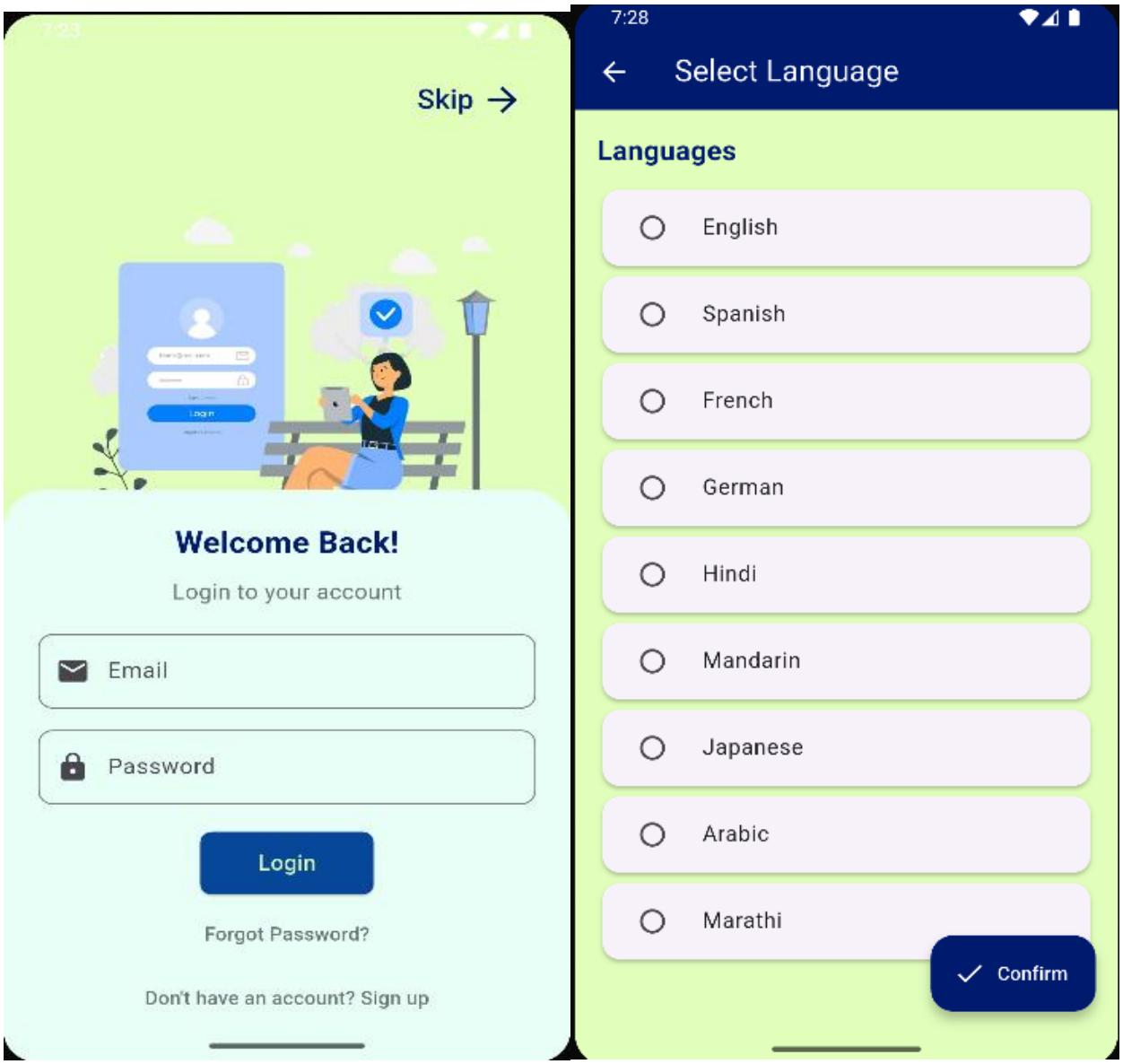
***NewsPageDetail.dart file***

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

class NewsDetailPage extends StatelessWidget {
  final String title, source, publishedAt, image,
content;

  const NewsDetailPage({
    super.key,
    required this.title,
    required this.source,
    required this.publishedAt,
    required this.image,
    required this.content,
  });

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: const Color(0xFFE1FFBB),
// Light green background
      body: CustomScrollView(
        slivers: [
          SliverAppBar(
            expandedHeight: 250,
            floating: false,
            pinned: true,
            backgroundColor: const
Color(0xFF001A6E), // Dark blue app bar
            flexibleSpace: FlexibleSpaceBar(
              background: ClipRRect(
                borderRadius:
                const BorderRadius.vertical(bottom:
Radius.circular(30)),
                child: Image.network(
                  image,
                  fit: BoxFit.cover,
                ),
              ),
            ),
          ),
          SliverToBoxAdapter(
            child: Container(
              decoration: BoxDecoration(
                color: Colors.white,
                borderRadius: const
BorderRadius.vertical(
                  top: Radius.circular(30),
                ),
                boxShadow: [
                  BoxShadow(
                    color: Colors.grey.withOpacity(0.3),
                    blurRadius: 10,
                    spreadRadius: 2,
                  ),
                ],
              ),
              padding: const
EdgeInsets.symmetric(horizontal: 20, vertical: 25),
              child: Column(
                crossAxisAlignment:
CrossAxisAlignment.start,
                children: [
                  Text(
                    title,
                    style: const TextStyle(
                      fontSize: 22,
                      fontWeight: FontWeight.bold,
                      color: Color(0xFF074799), // Medium
blue title
                    ),
                  ),
                  const SizedBox(height: 10),
                  Row(
                    children: [
                      const CircleAvatar(
                        radius: 14,
                        backgroundColor:
Color(0xFF009990), // Teal avatar
                        child: Icon(Icons.person, color:
Colors.white, size: 16),
                      ),
                      const SizedBox(width: 8),
                      Text(
                        source,
                        style: const TextStyle(
                          fontSize: 14,
                          fontWeight: FontWeight.w500,
                          color: Colors.black87,
                        ),
                      ),

    const SizedBox(width: 8),
                      Text(
                        "• $publishedAt",
                        style: const TextStyle(
                          fontSize: 14,
                          color: Colors.grey,
                        ),
                      ),
                    ],
                  ),
```

```
            const SizedBox(height: 20),
            Text(
              content,
              style: const TextStyle(
                fontSize: 16,
                color: Colors.black87,
                height: 1.5,
              ),
            ),
                                                    ],
                                                  ),
                                                ),
                                              ],
                                            ),
                                          );
                                        }
                                      }
```

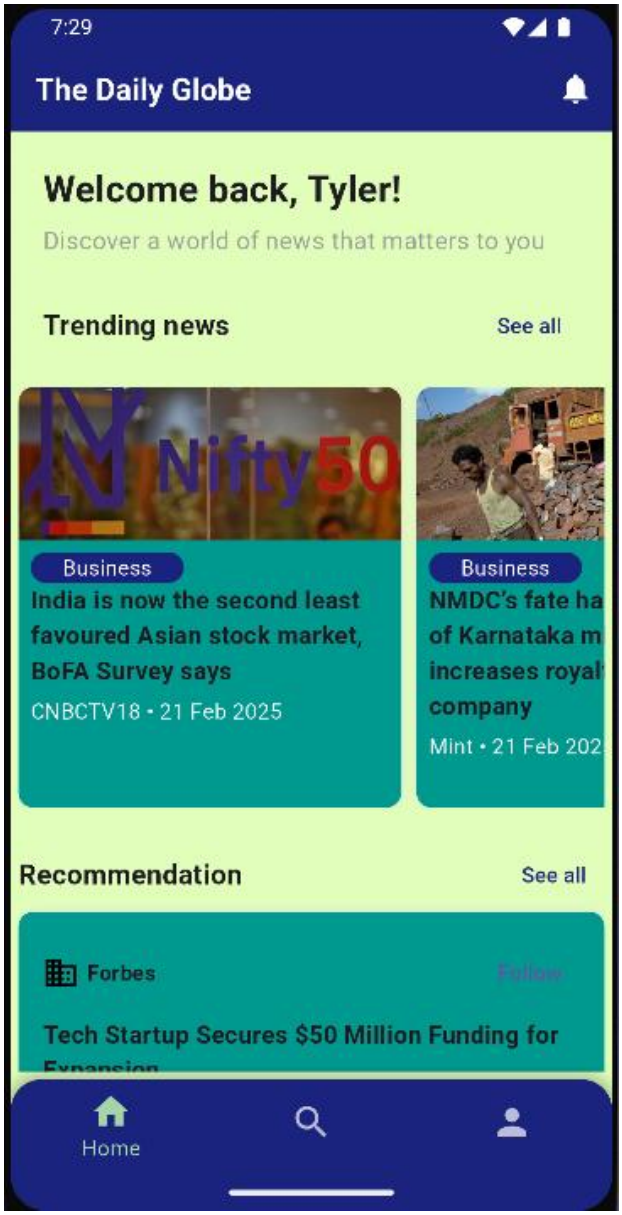
**Output:**

After clicking skip button it will navigate to languages.dart page
And
After clicking shopping cart icon in topbar it will redirect to bag.dart page



**Project Title: DailyHunt** **Roll No. 60**

After selecting language and clicking on confirm then it will navigate to home.dart
And
After clicking on any news card then it will navigate to newsdetailpage.dart



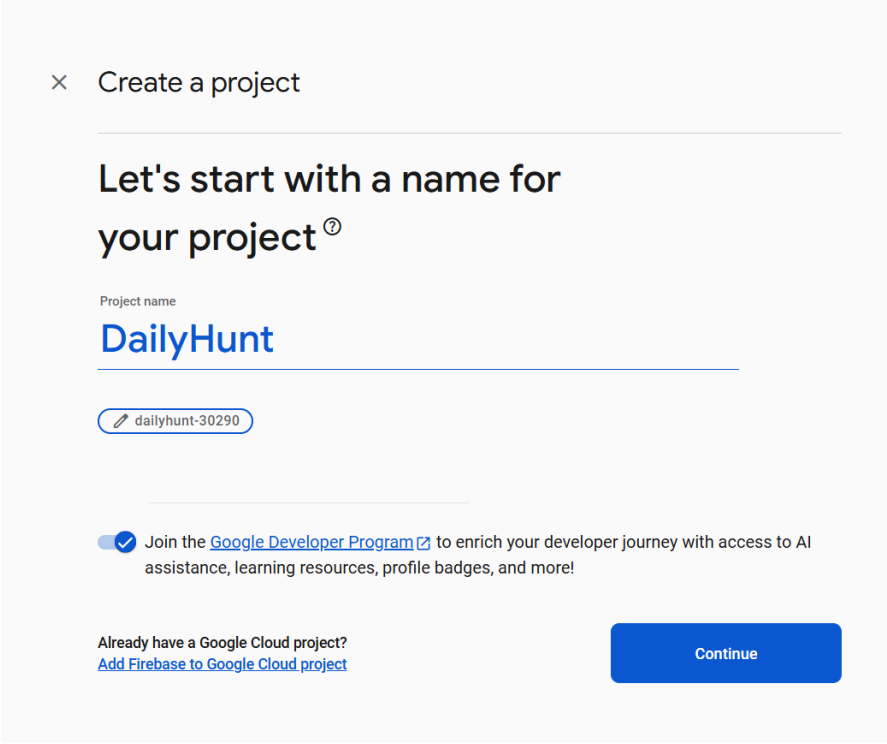

# MAD & PWA Lab
# Journal

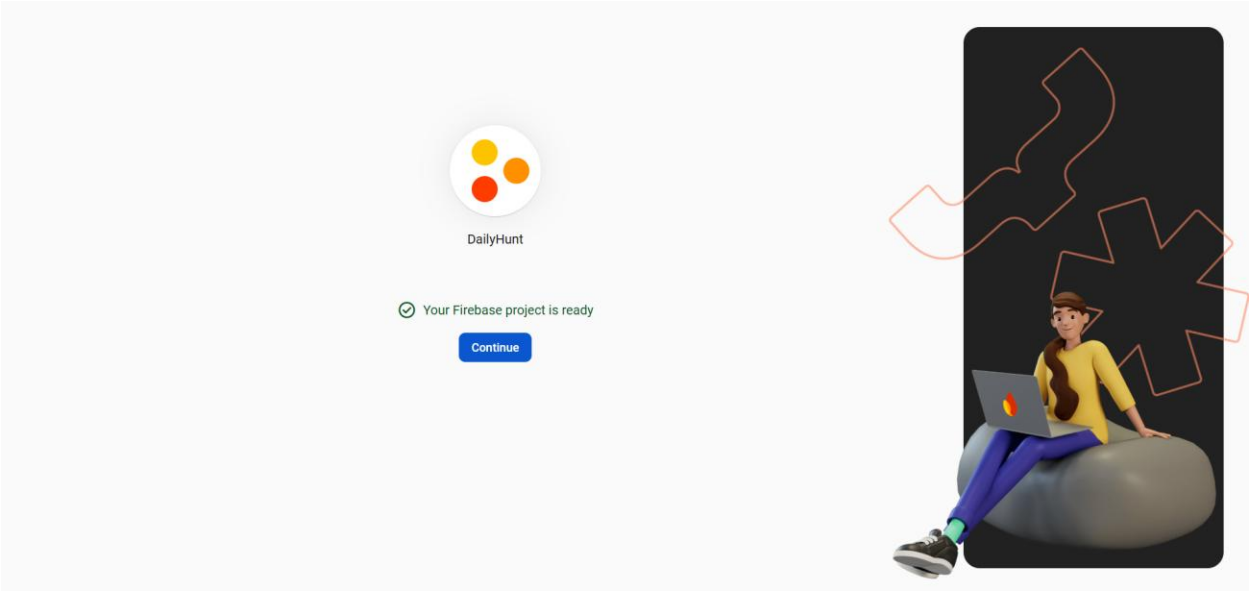
| | |
|---|---|
| Experiment No. | 06 |
| Experiment Title. | To Connect Flutter UI with fireBase database |
| Roll No. | 60 |
| Name | Pranav Pramod Titambe |
| Class | D15A |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS |
| Grade: | |

**Project Title: DailyHunt** **Roll No. 60**

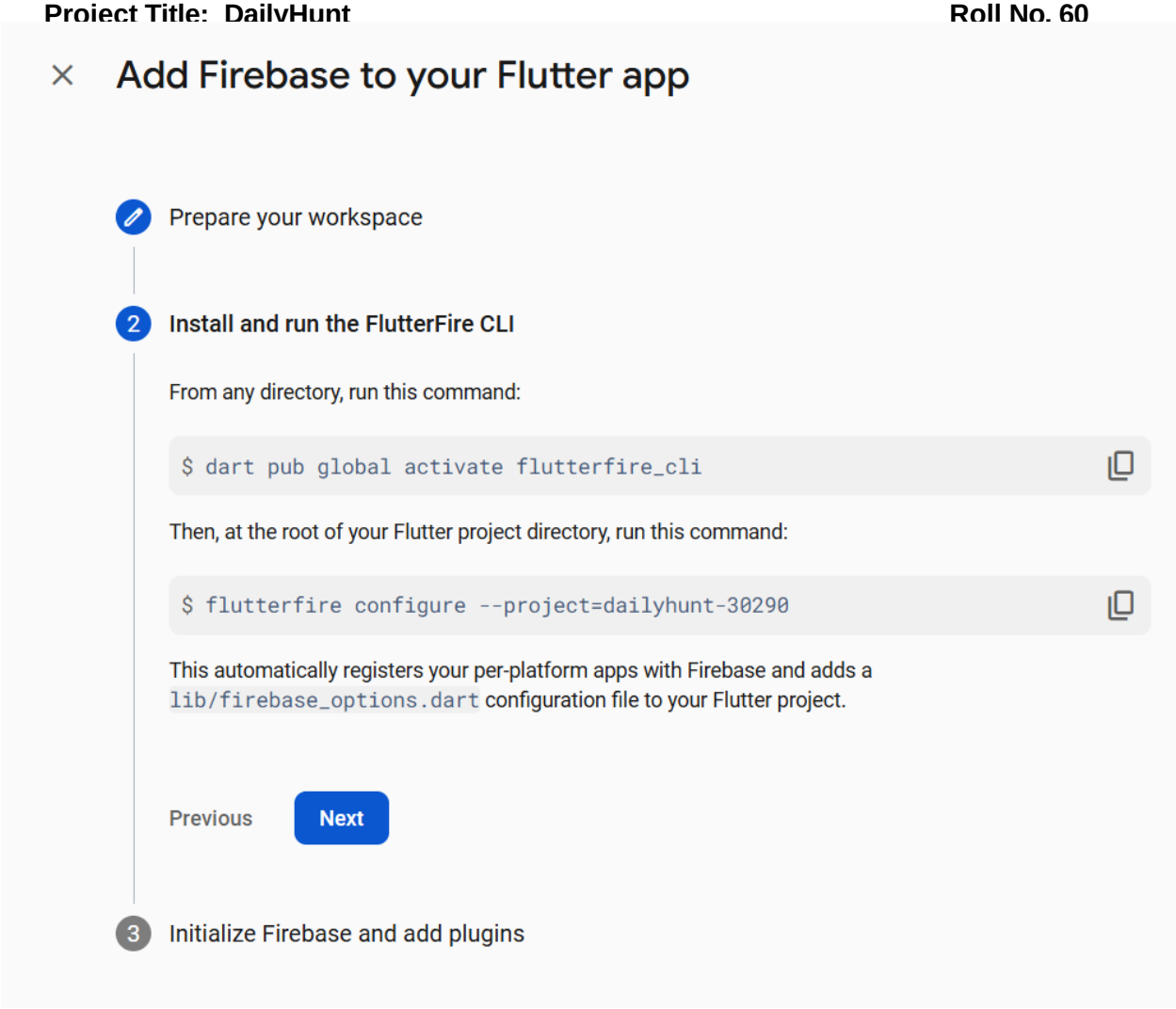
## Creating a New Firebase Project

First, log in with your Google account to manage your Firebase projects. From within the Firebase dashboard, select the Create new project button and give it a name




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




Run this above command as mentioned and firebase_options.dart will be generated automatically.

**In pubspec.yaml:**

```
dependencies:
  flutter:
    sdk: flutter
  permission_handler: ^11.3.1 # Use the latest version
  cupertino_icons: ^1.0.8
  http: ^1.3.0
  intl: ^0.20.2
  firebase_core: ^3.12.0
  firebase_auth: ^5.5.0
  cloud_firestore: ^5.6.4
  firebase_storage: ^12.4.3
  firebase_messaging: ^15.2.3
```

**Firebase connection in code:**

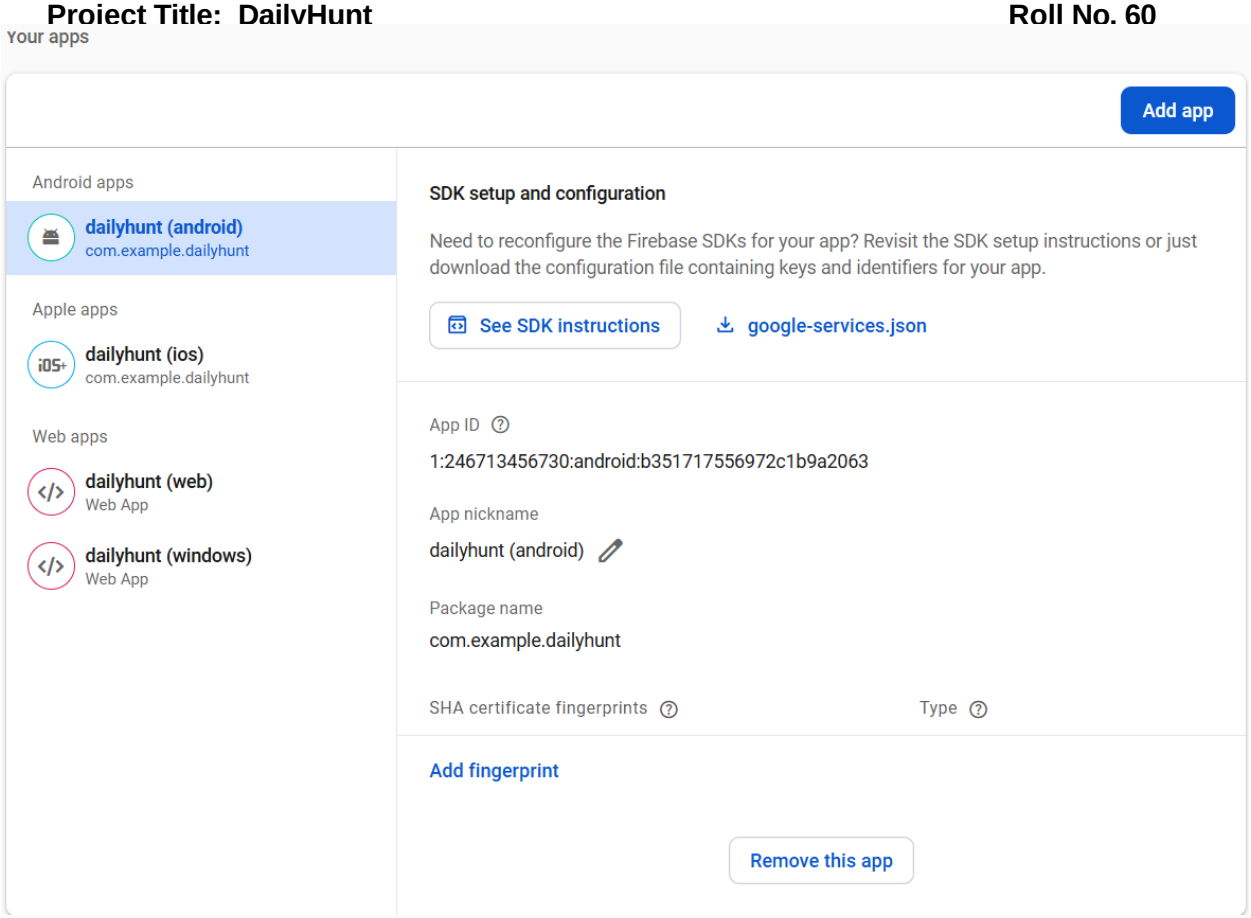
```
import 'package:flutter/material.dart';
import 'package:dailyhunt/login.dart';
import 'package:permission_handler/permission_handler.dart';
```

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

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  try {
    await Firebase.initializeApp(
      options: FirebaseOptions(
        apiKey: 'AIzaSyBpVLoaXlXO4fMcQFCIj26smZp49IPpVow',
        appId: '1:246713456730:android:b351717556972c1b9a2063',
        messagingSenderId: '246713456730',
        projectId: 'dailyhunt-30290',
        storageBucket: 'dailyhunt-30290.firebasestorage.app',
      )
    );
    print("✅ Firebase connected successfully!");
  } catch (e) {
    print("❌ Firebase initialization failed: $e");
  }
  runApp(const MyApp());
}
```

**Output:**


Check if andriod , ios, web apps are registered to firebase.

**Project Title: DailyHunt** **Roll No. 60**



# 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. | 60 |
| Name | Pranav Pramod Titambe |
| Class | D15A |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO4: Understand various PWA frameworks and their requirements |
| Grade: | |

**Project Title: DailyHunt** **Roll No. 60**

To write meta data of your Ecommerce PWA

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:-

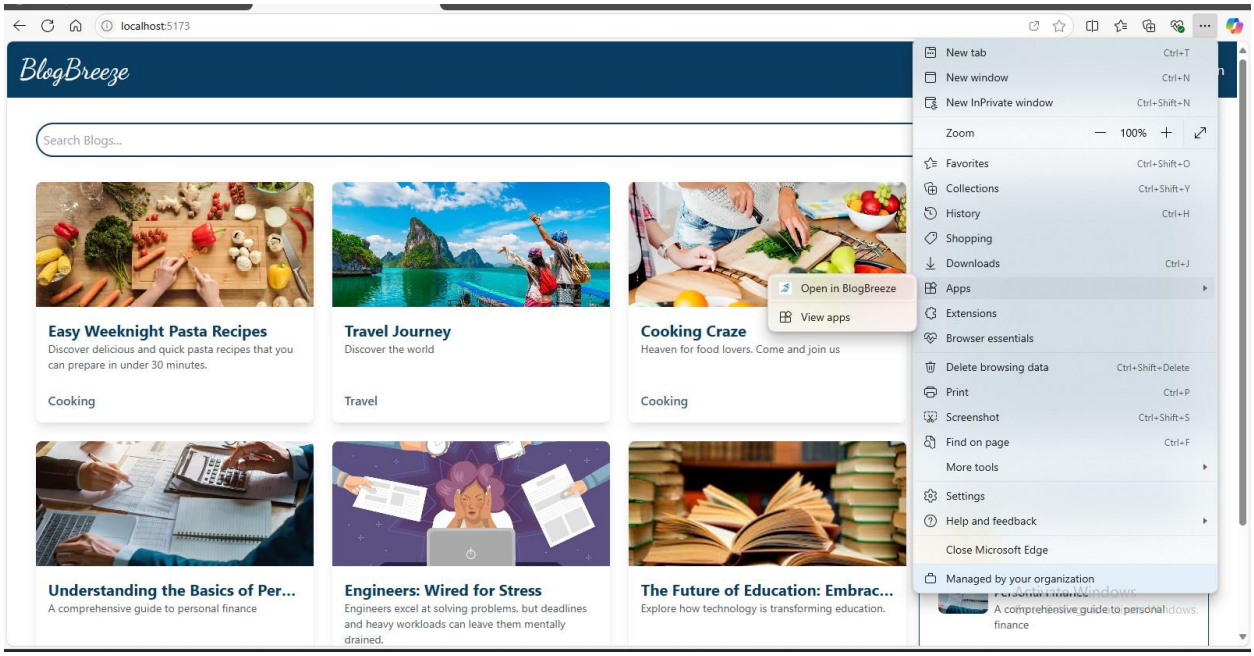
```
{
"name": "BlogBreeze", "short_name": "BlogBreeze", "start_url": "ani.html",
"scope": "./", "icons": [
{
"src": "/src/assets/react.svg", "sizes": "192x192",
"type": "image/svg"
}
],
"theme_color": "#ffd31d", "background_color": "#333", "display": "standalone"
}
```

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

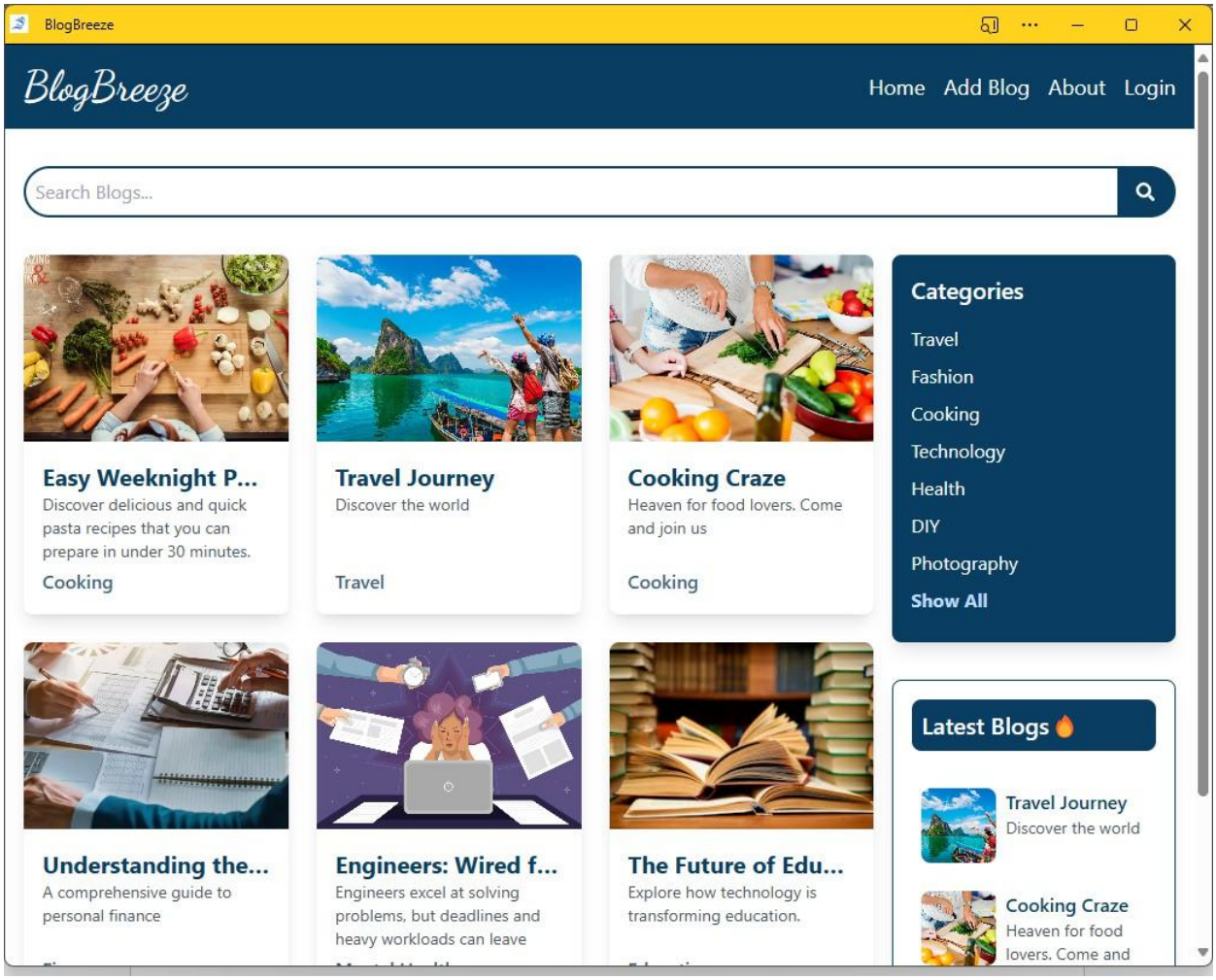
**Project Title: DailyHunt** **Roll No. 60**

```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:-



**Project Title: DailyHunt** **Roll No. 60**




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. | 60 |
| Name | Pranav Pramod Titambe |
| Class | D15A |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO5: Design and Develop a responsive User Interface by applying PWA Design techniques |
| Grade: | |

**Project Title: DailyHunt** **Roll No. 60**

**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);
})
);
});
```

**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. | 60 |
| Name | Pranav Pramod Titambe |
| Class | D15A |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO5: Design and Develop a responsive User Interface by applying PWA Design techniques |
| Grade: | |

**Project Title: DailyHunt** **Roll No. 60**

**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.');
}
}
}
});
```

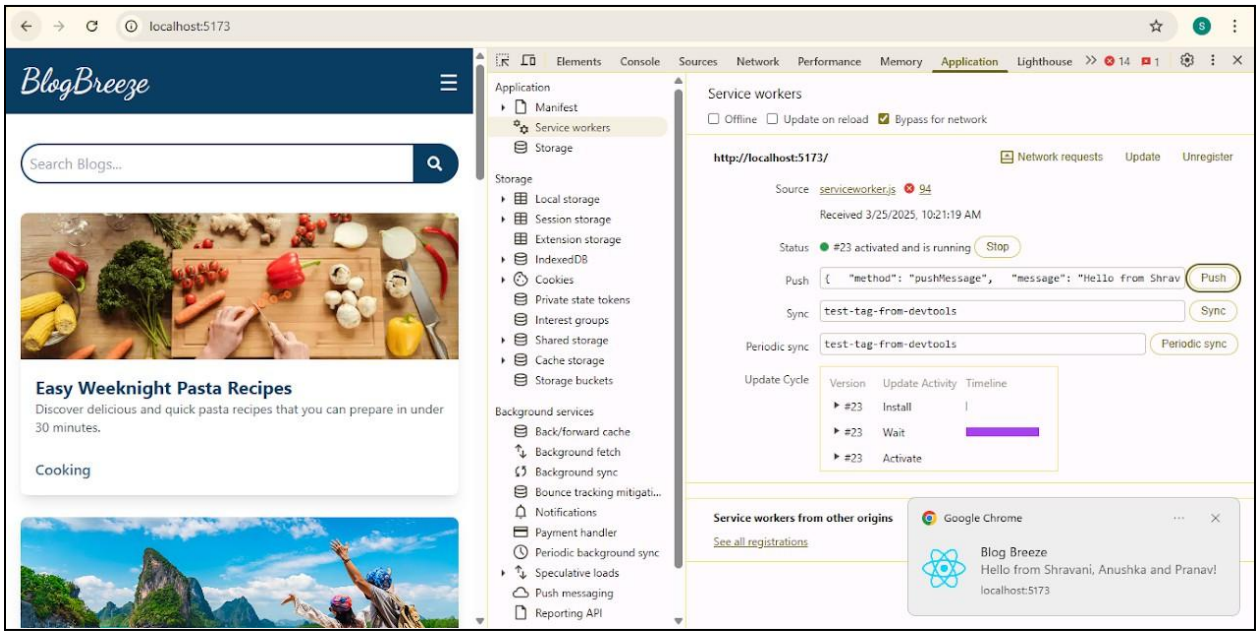
**SERVICE WORKER SCRIPT FOR SYNC AND FETCH :**

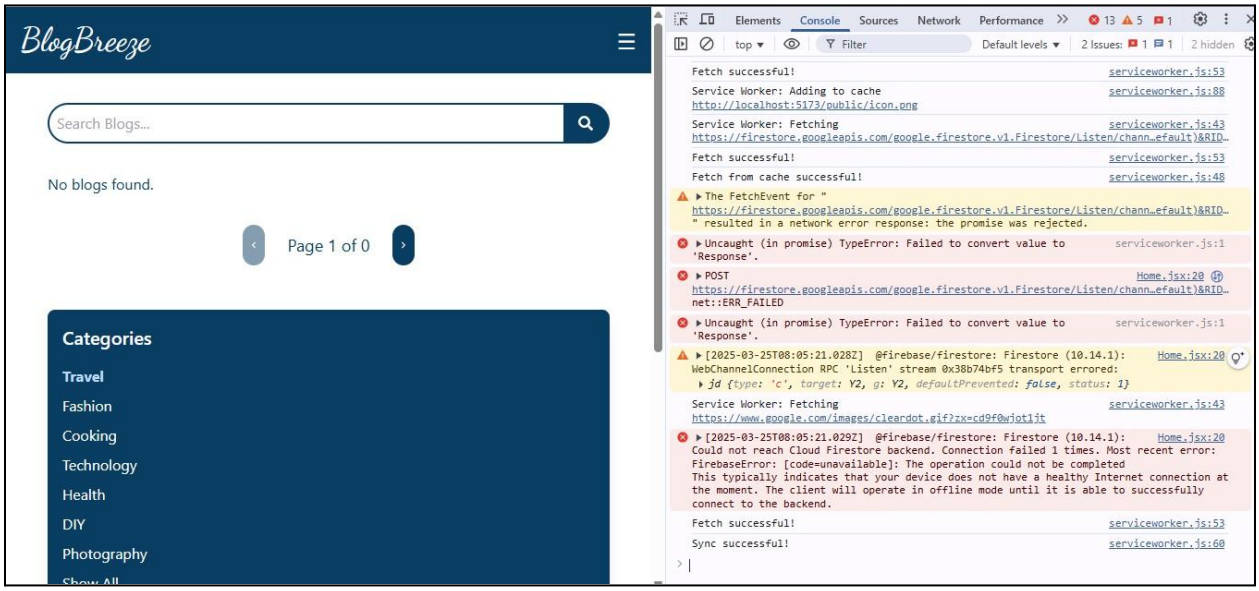
```
const CACHE_NAME = 'blogbreeze-v1'; const filesToCache = [
'/',
'index.html', 'manifest.json', '/src/assets/react.svg', '/public/icon.png',
'/src/index.css','/src/main.jsx',

];

// 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;
});
});
}
```

**Project Title: DailyHunt** **Roll No. 60**

**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. | 60 |
| Name | Pranav Pramod Titambe |
| Class | D15A |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO5: Design and Develop a responsive User Interface by applying PWA Design techniques |
| Grade: | |

**Project Title: DailyHunt** **Roll No. 60**

**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

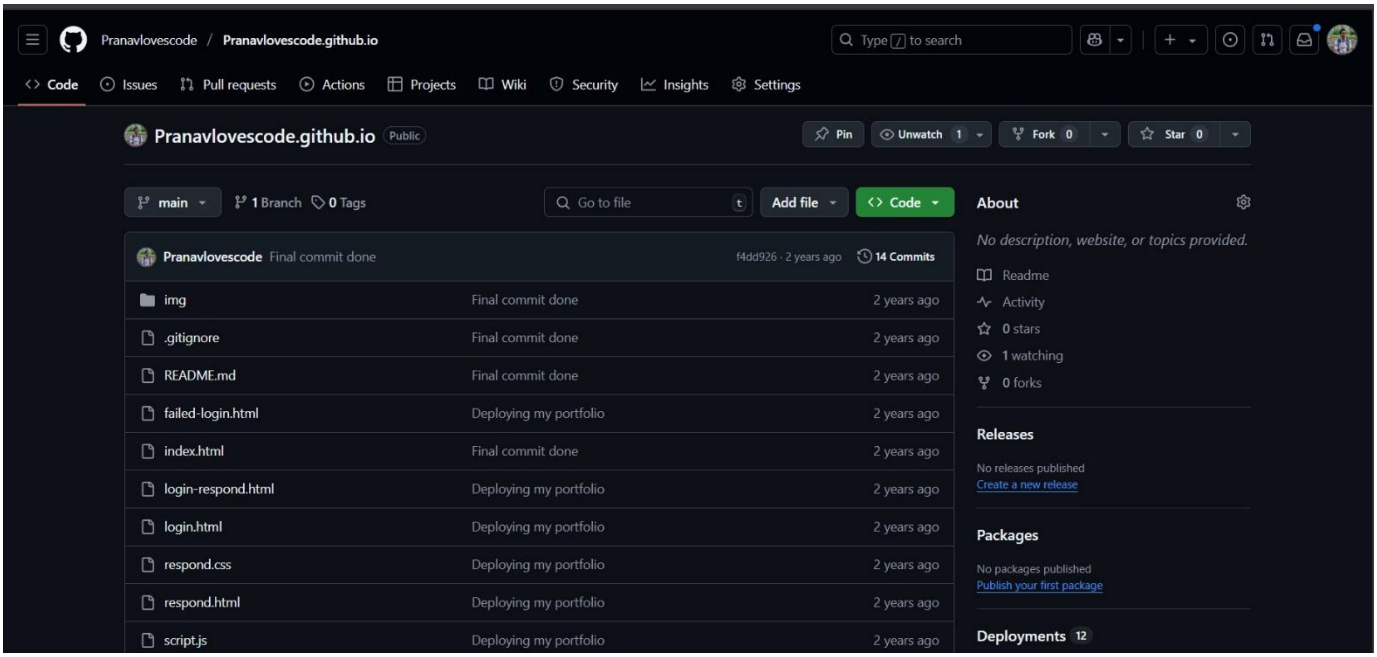
1. Hosted by Google. Enough said.
2. Authentication, Cloud Messaging, and a whole lot of other handy services will be available to you.
3. A real-time database will be available to you, which can store 1 GB of data.
4. You'll also have access to a blob store, which can store another 1 GB of data.
5.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

Cons

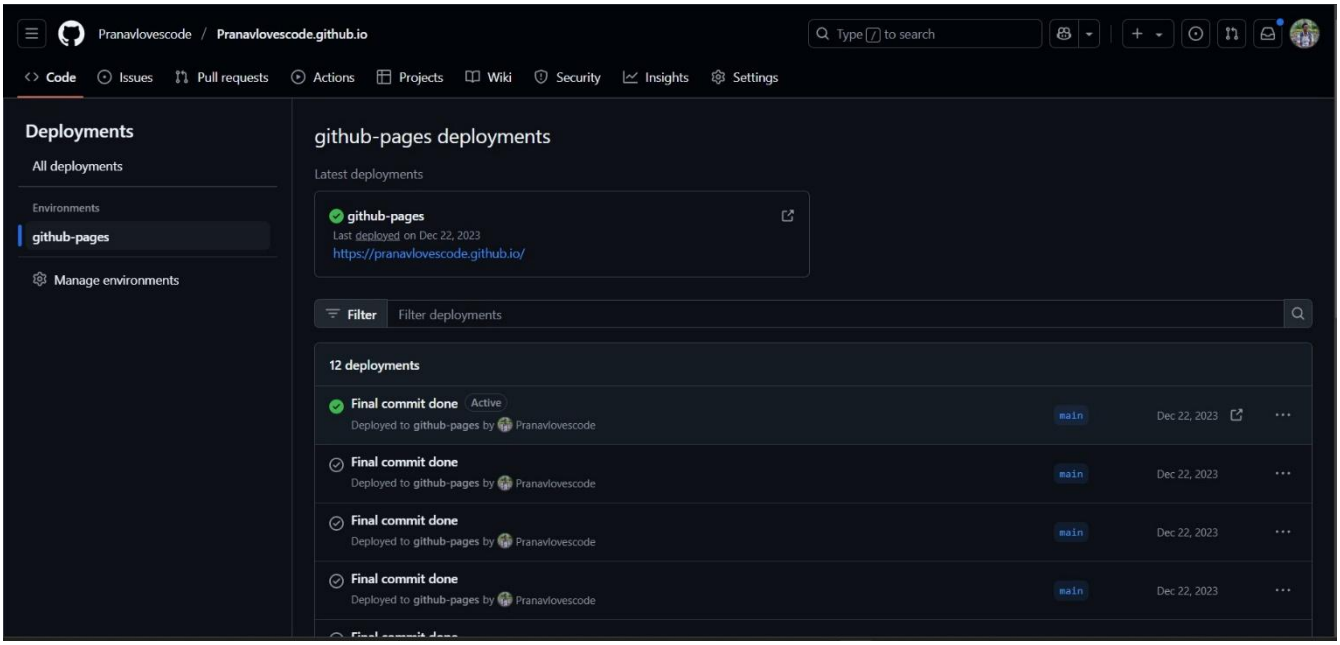
1. Only 10 GB of data transfer is allowed per month. But this is not really a big problem, if you use a CDN or AMP.
2. Command-line interface only.
3. No in-built support for any static site generator.

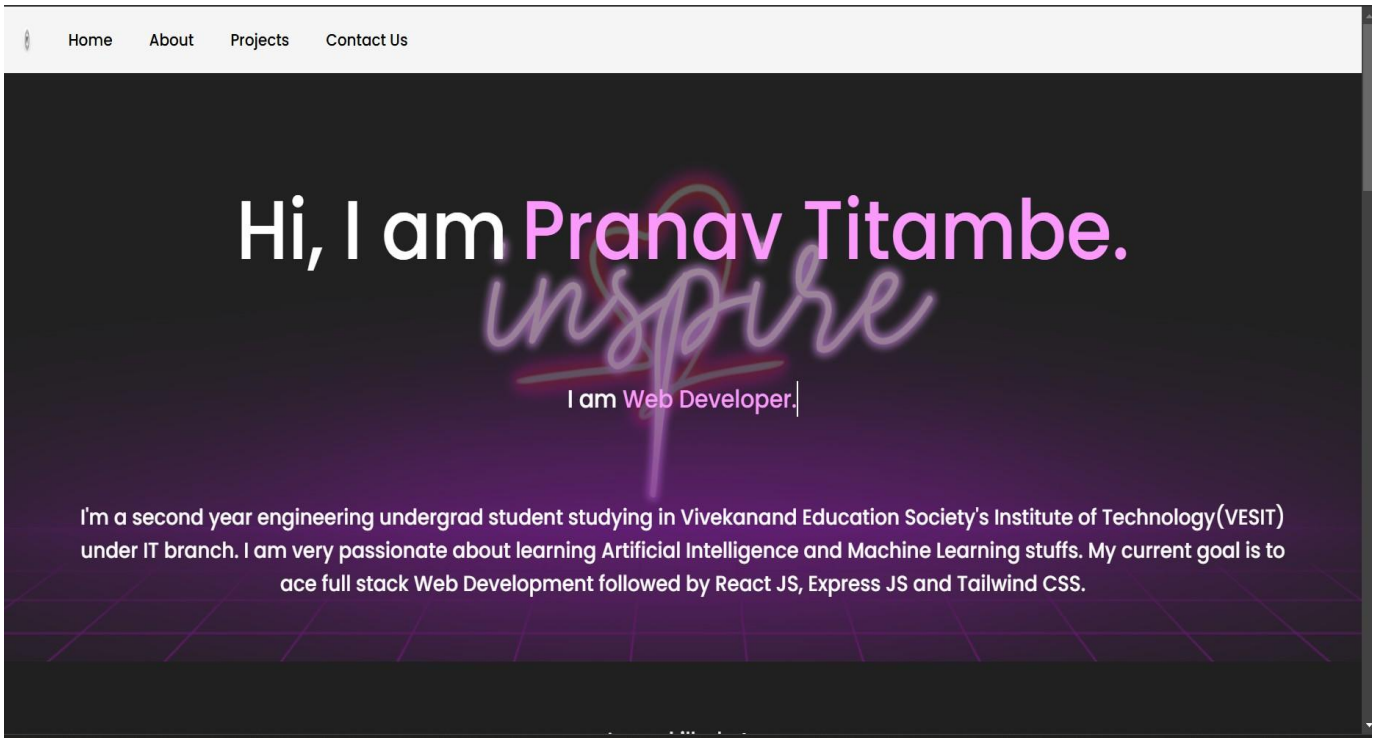
**Link to our GitHub repository: [https://github.com/ShravaniR2412/IP_ASSG2](https://github.com/ShravaniR2412/IP_ASSG2)**

**Github Screenshot:**



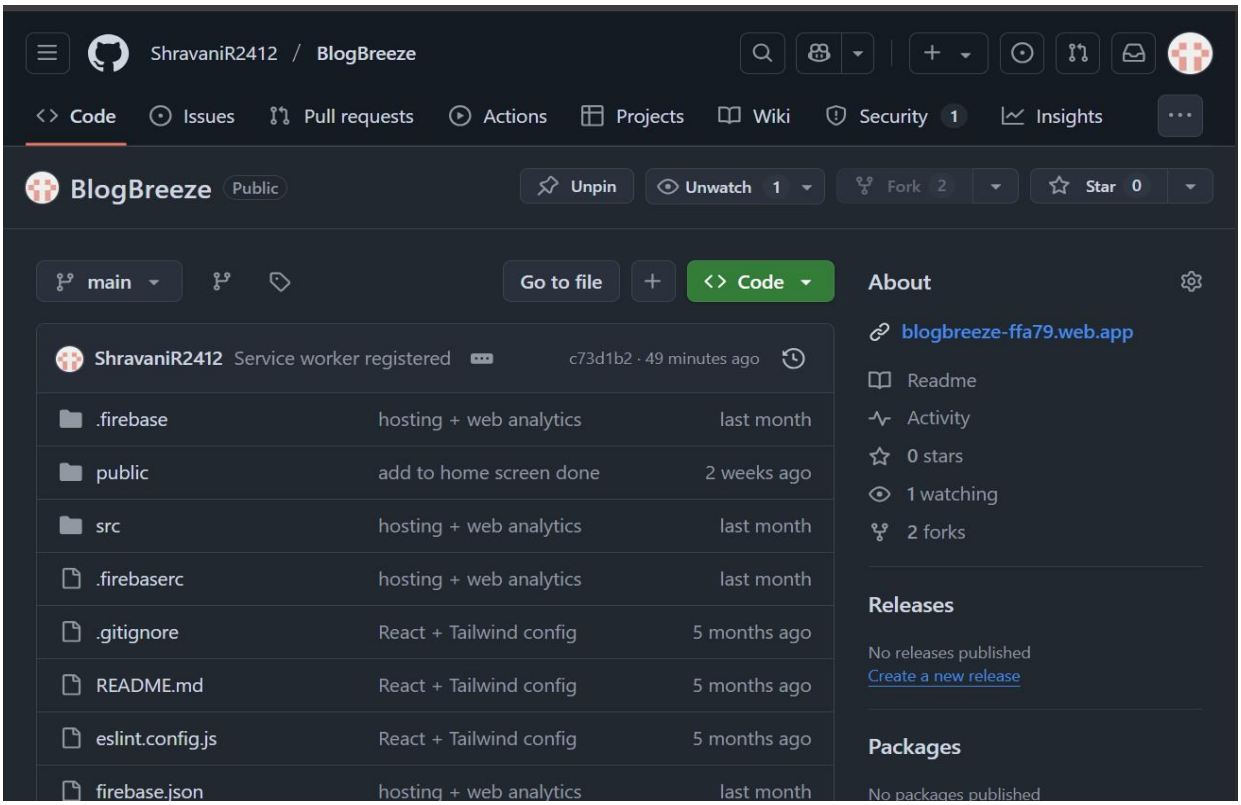
**Project Title: DailyHunt** **Roll No. 60**



**Project Title: DailyHunt** **Roll No. 60**

**Deployed Link: https://pranavlovescode.github.io/**
**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. | 60 |
| Name | Pranav Pramod Titambe |
| Class | D15A |
| Subject | MAD & PWA Lab |
| Lab Outcome | LO6: Develop and Analyze PWA Features and deploy it over app hosting solution |
| Grade: | |

**Project Title: DailyHunt** **Roll No. 60**

**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**

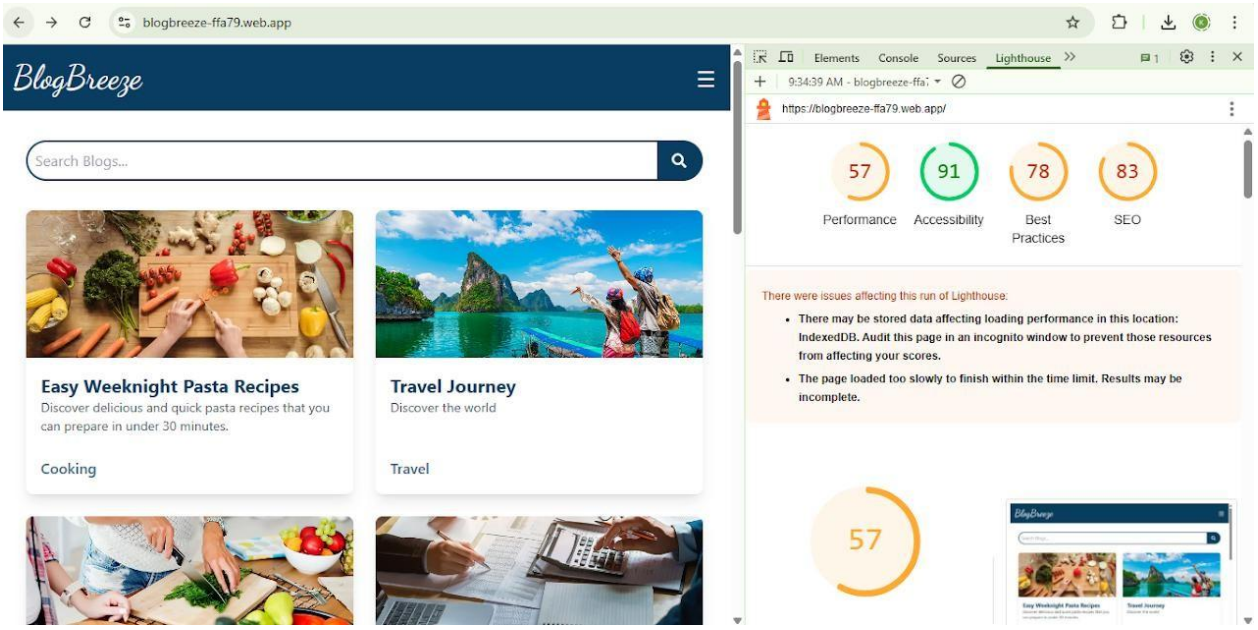
Google Lighthouse has the option of running the Audit for Desktop as well as mobile version of your page(s). The top metrics that will be measured in the Audit are:

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,

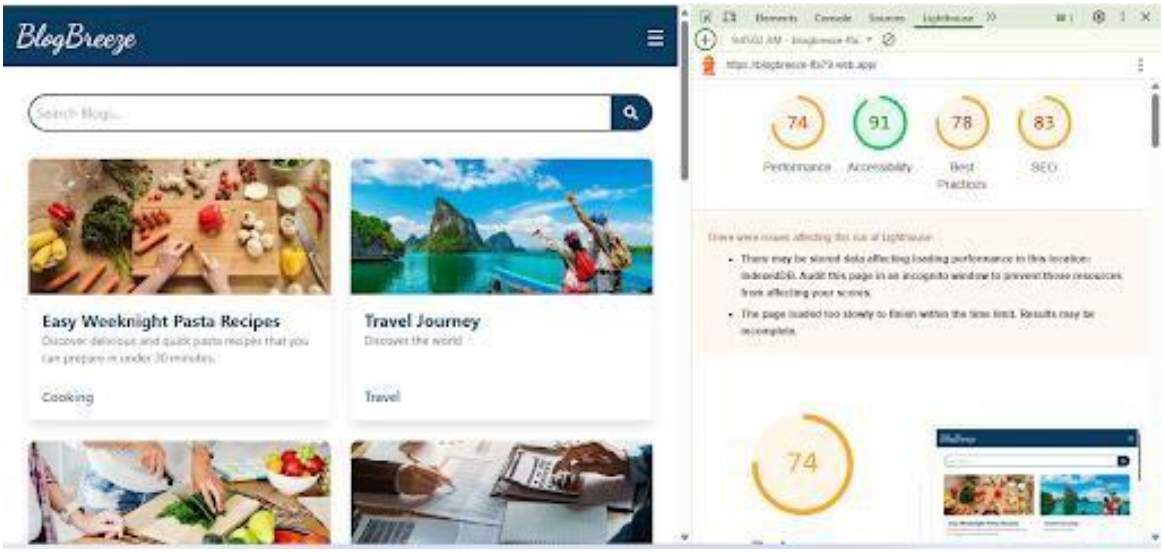
libraries, etc. Password input with paste-into disabled
Geo-Location and cookie usage alerts on load, etc.

**Project Title: DailyHunt** **Roll No. 60**

Performance before changes




Performance after changes



**Project Title: DailyHunt** **Roll No. 60**



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