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 KUNAL MAHESH PUNJABI 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: MyCart **Roll No : 43**

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: MyCart **Roll No:43**

Lab Objectives:

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

Lab Outcomes:

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

Project Title: **Roll No.**

Index

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

MAD & PWA Lab
Journal

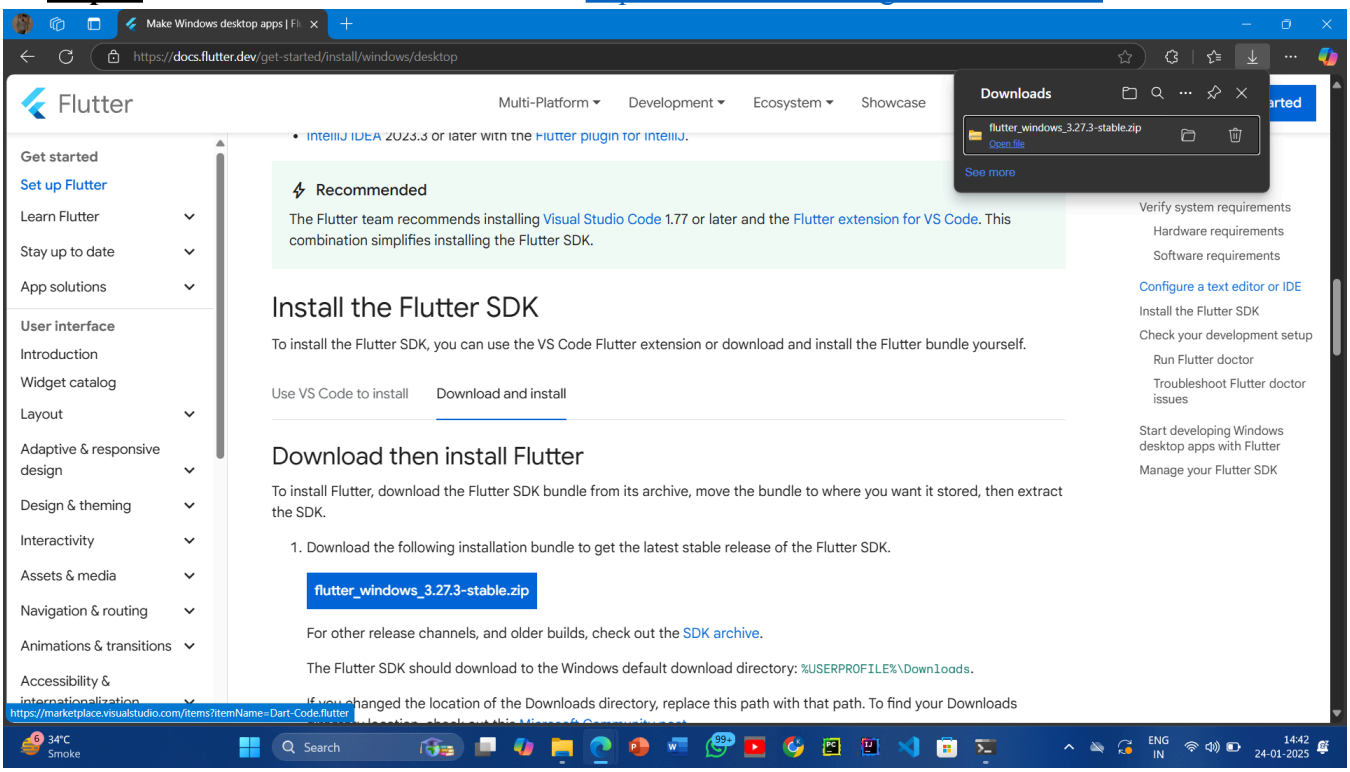
Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	43
Name	Kunal Punjabi
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	

EXPERIMENT NO: - 01

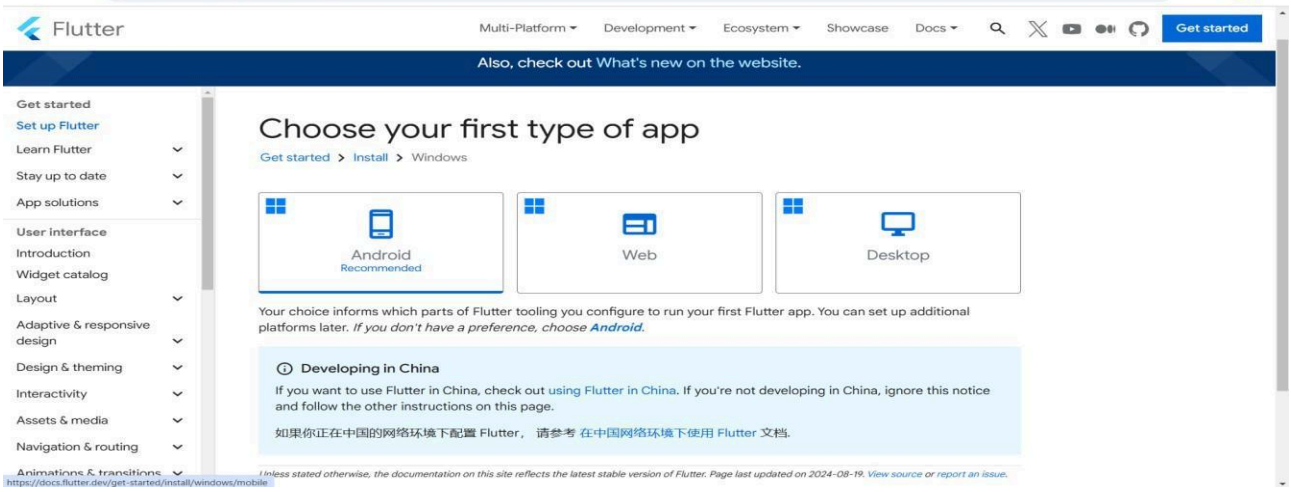
Name:- Kunal Punjabi **Class:-** D15A **Roll:No: -** 43

AIM: - Installation and Configuration of Flutter Environment.

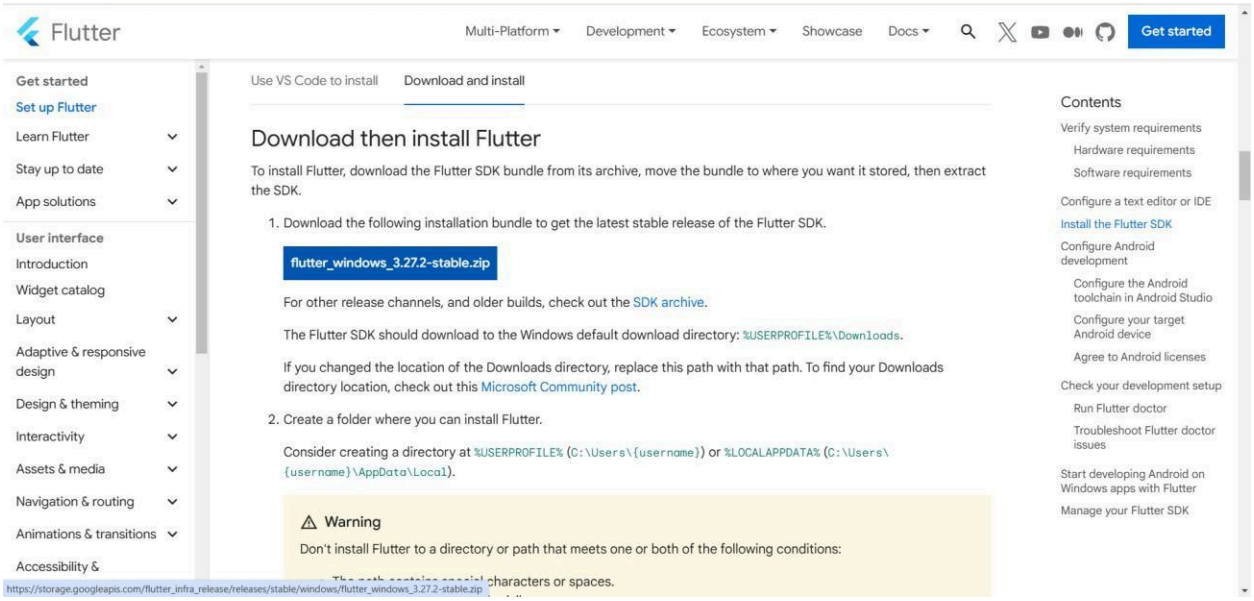
Step 1: Go to the official Flutter website: https://docs.flutter.dev/get-started/install



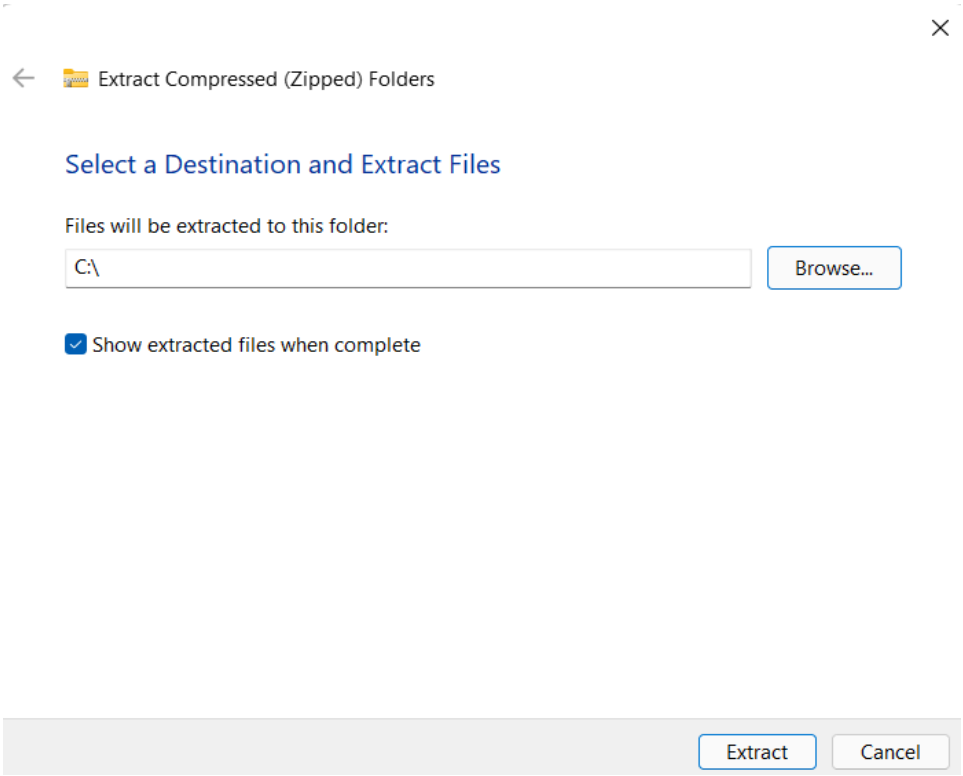
Step 2: To download the latest Flutter SDK, click on the Windows icon > Android



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



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

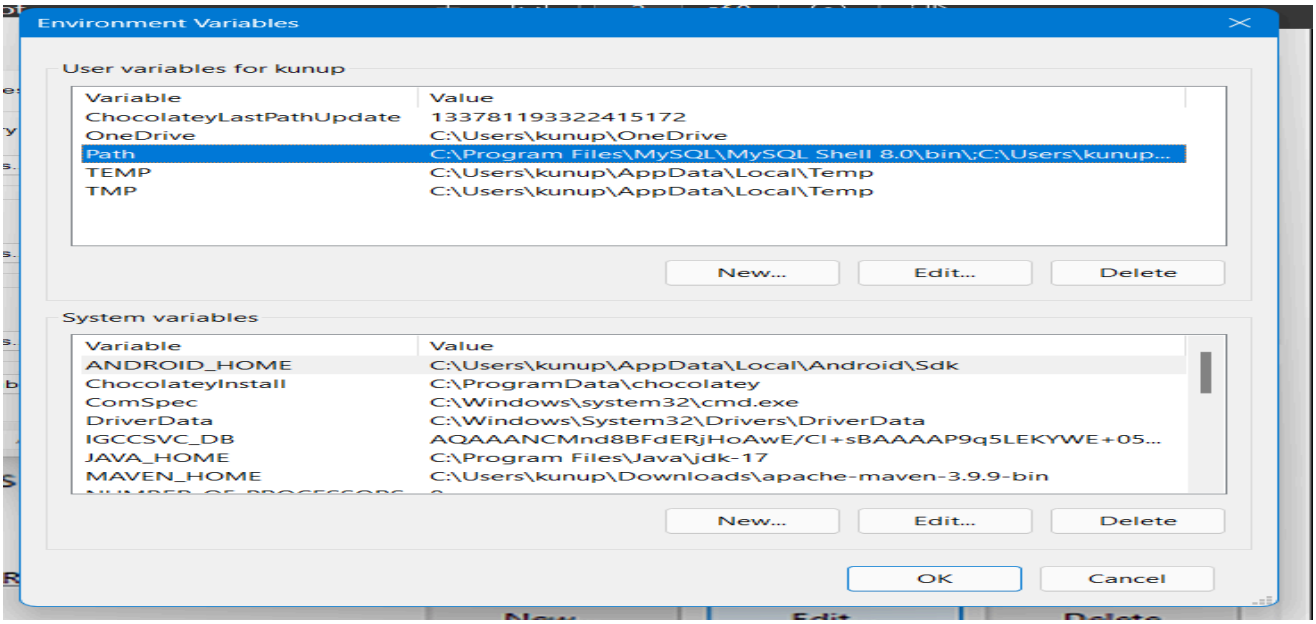


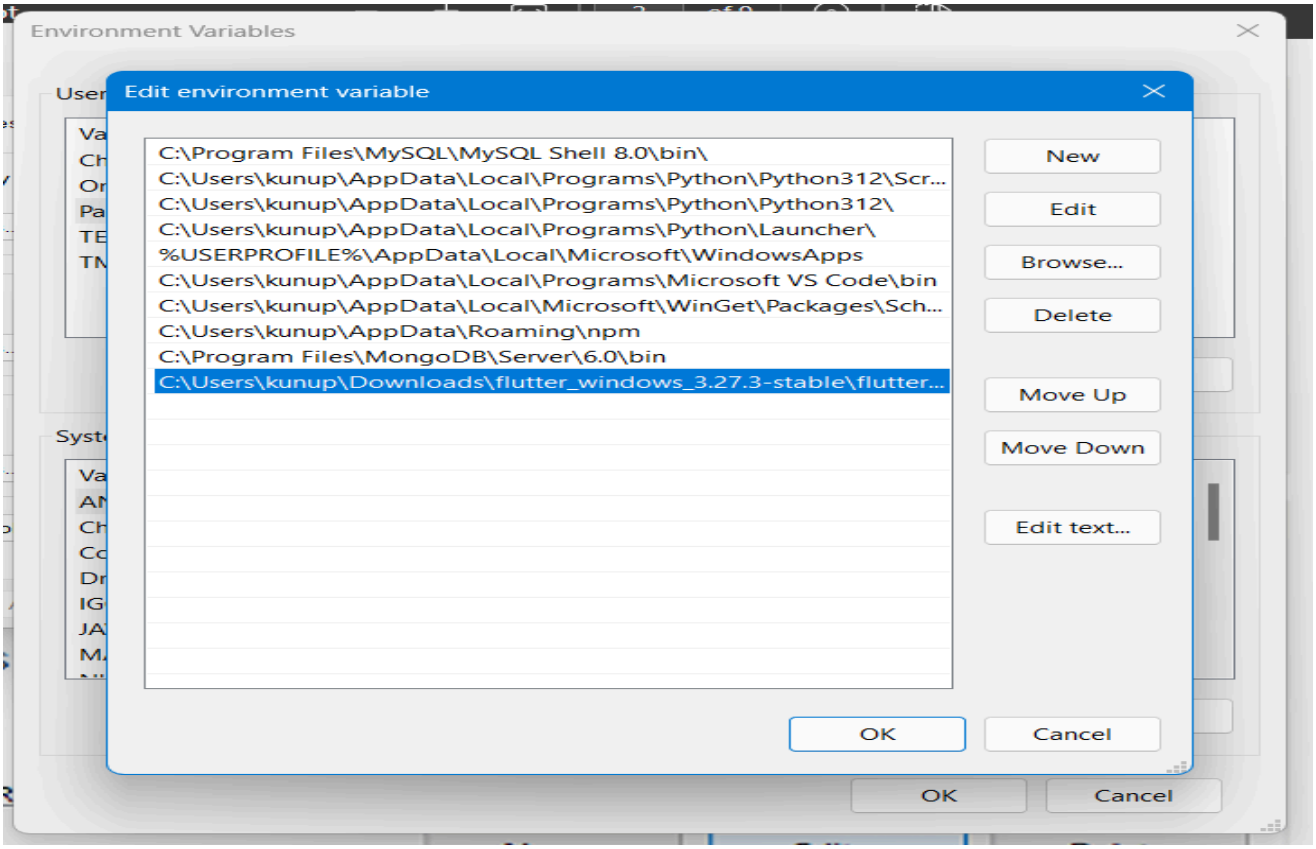
Step 5 :- Add Flutter to System PATH

Right-click on the Start Menu > System > Advanced system settings > Environment Variables.

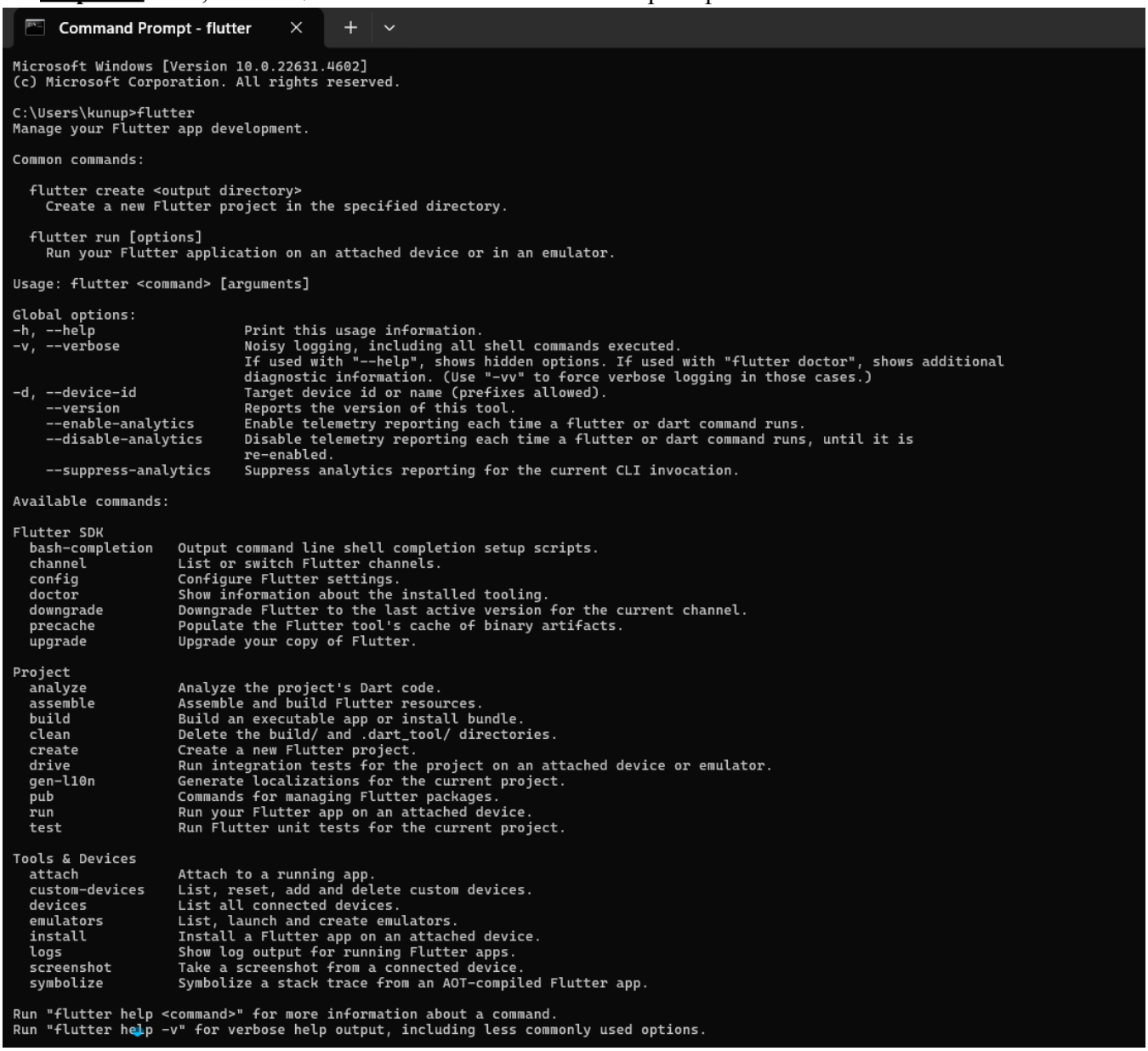
Under System Variables, find Path and click Edit.

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





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

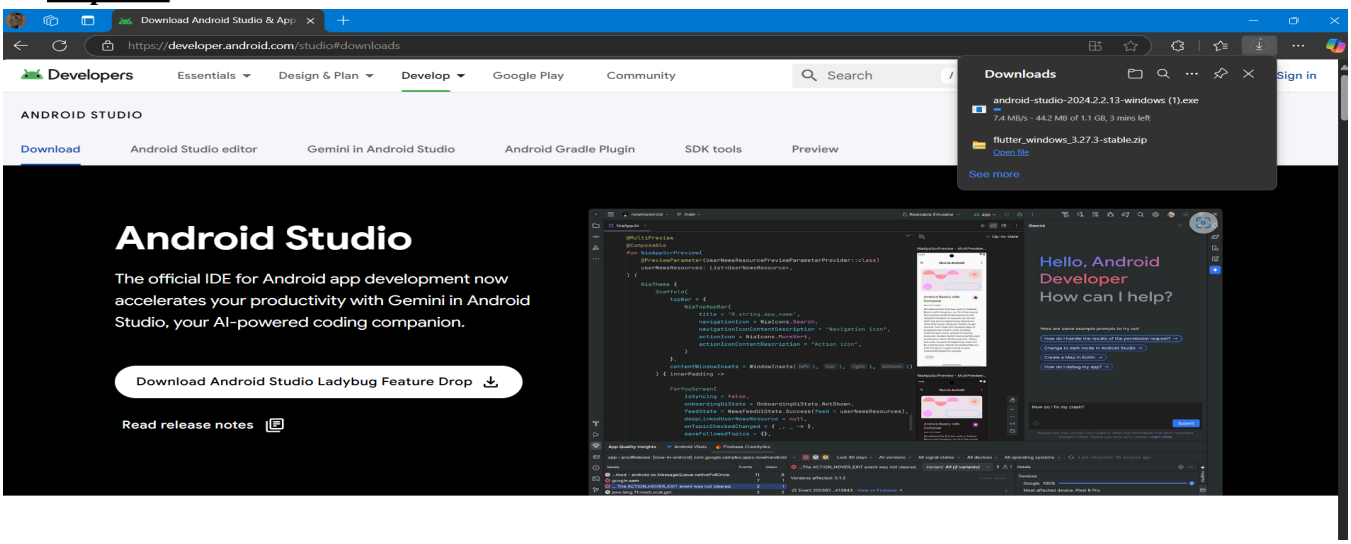


Step 7:- Run the $ flutter doctor command. This command checks for all the requirements of Flutter app development and displays a report of the status of your Flutter installation

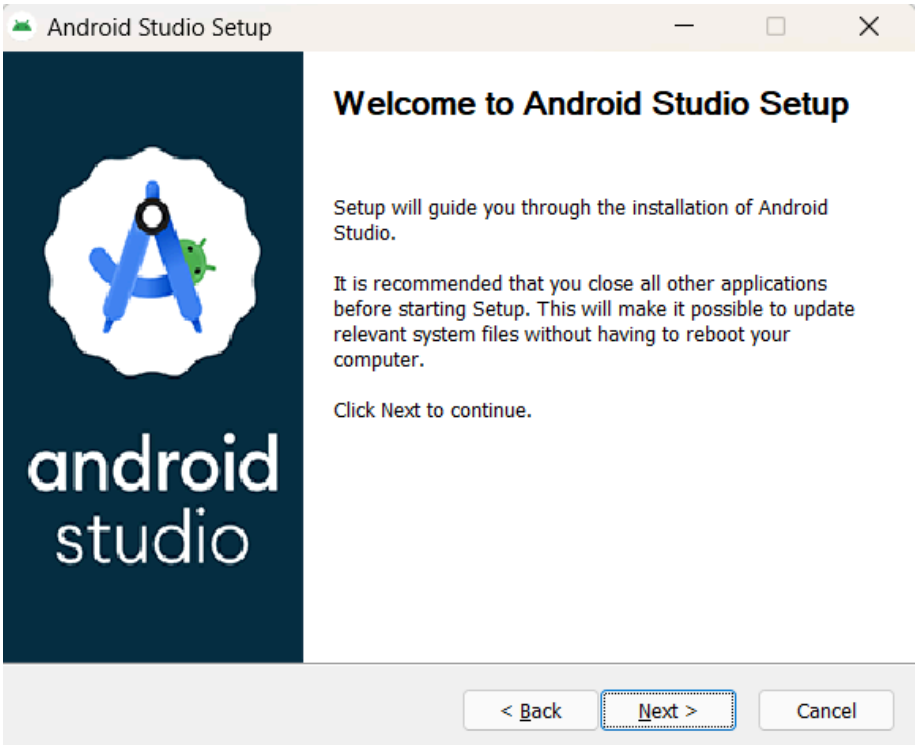
```
Microsoft Windows [Version 10.0.22631.4751]
(c) Microsoft Corporation. All rights reserved.

C:\Users\kunup>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.27.3, on Microsoft Windows [Version 10.0.22631.4751], locale en-IN)
[√] Windows Version (Installed version of Windows is version 10 or higher)
[√] Android toolchain - develop for Android devices (Android SDK version 35.0.1)
[√] Chrome - develop for the web
[X] Visual Studio - develop Windows apps
    X Visual Studio not installed; this is necessary to develop Windows apps.
      Download at https://visualstudio.microsoft.com/downloads/.
      Please install the "Desktop development with C++" workload, including all of its default components
[√] Android Studio (version 2024.2)
[√] VS Code (version 1.96.4)
[√] Connected device (3 available)
```

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

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

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

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

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

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

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

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

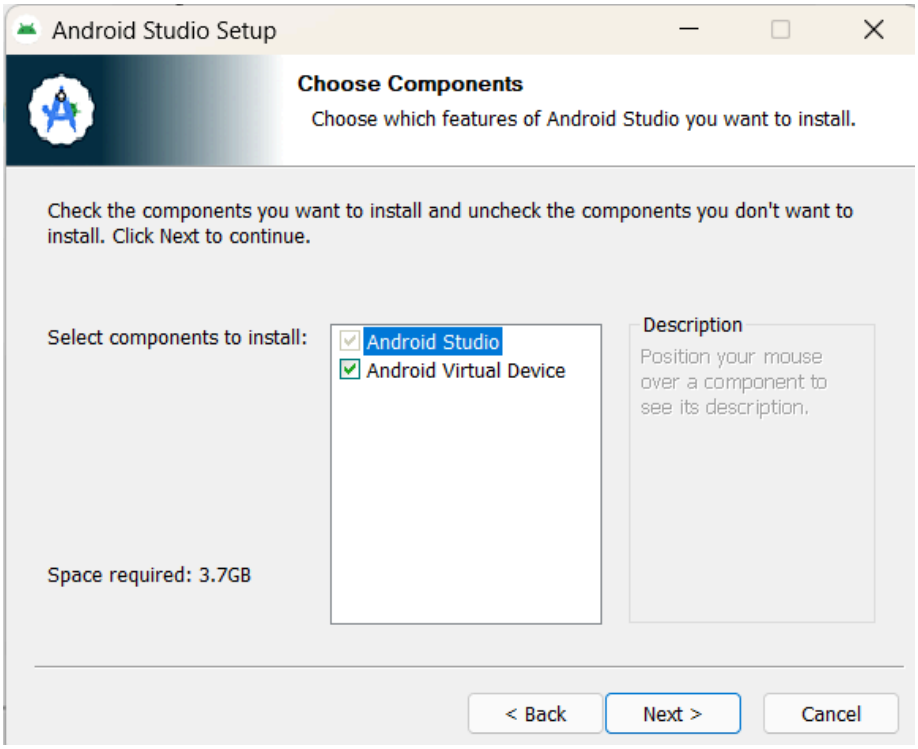
Step 8 : - Go to Android Studio and download the installer.



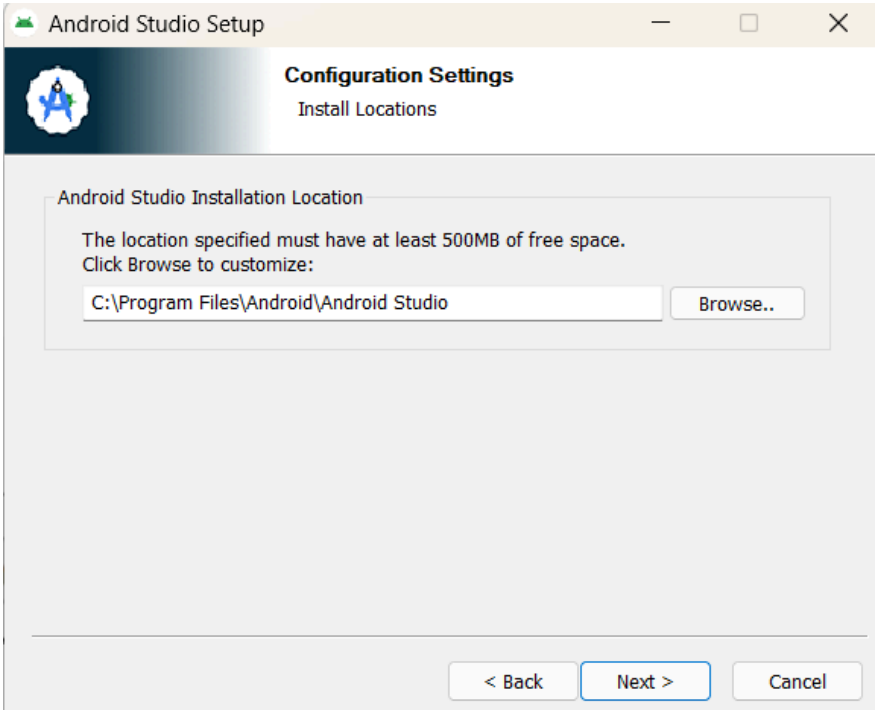
Step 8.1: - When the download is complete, open the .exe file and run it. You will get the following dialog box



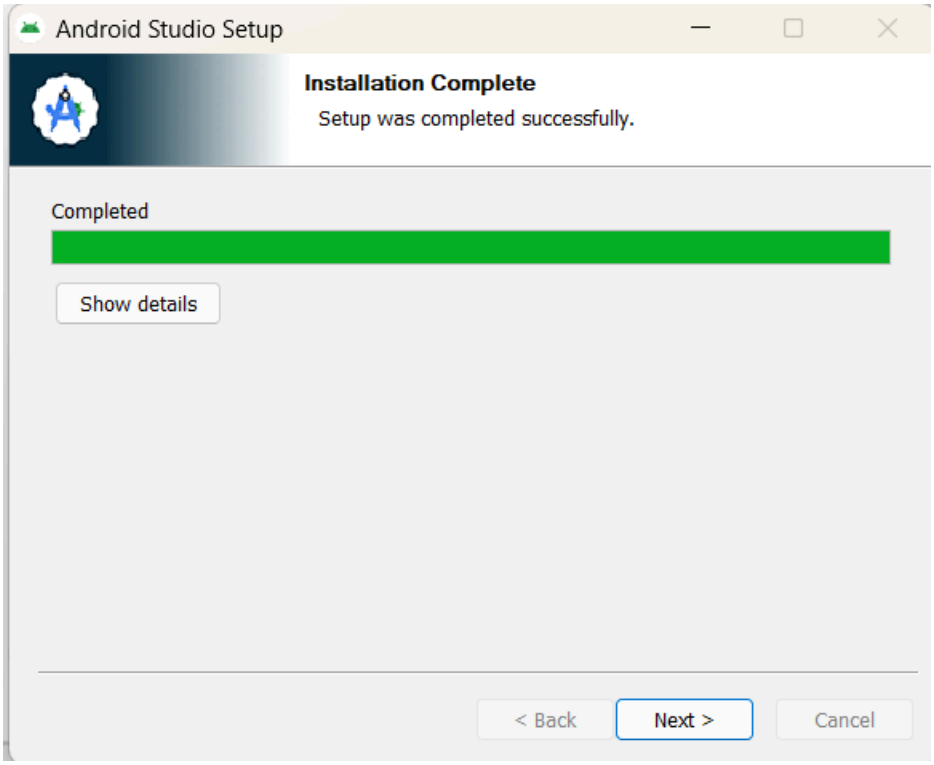
Step 8.2: - Select all the Checkboxes and Click on ‘Next’ Button.

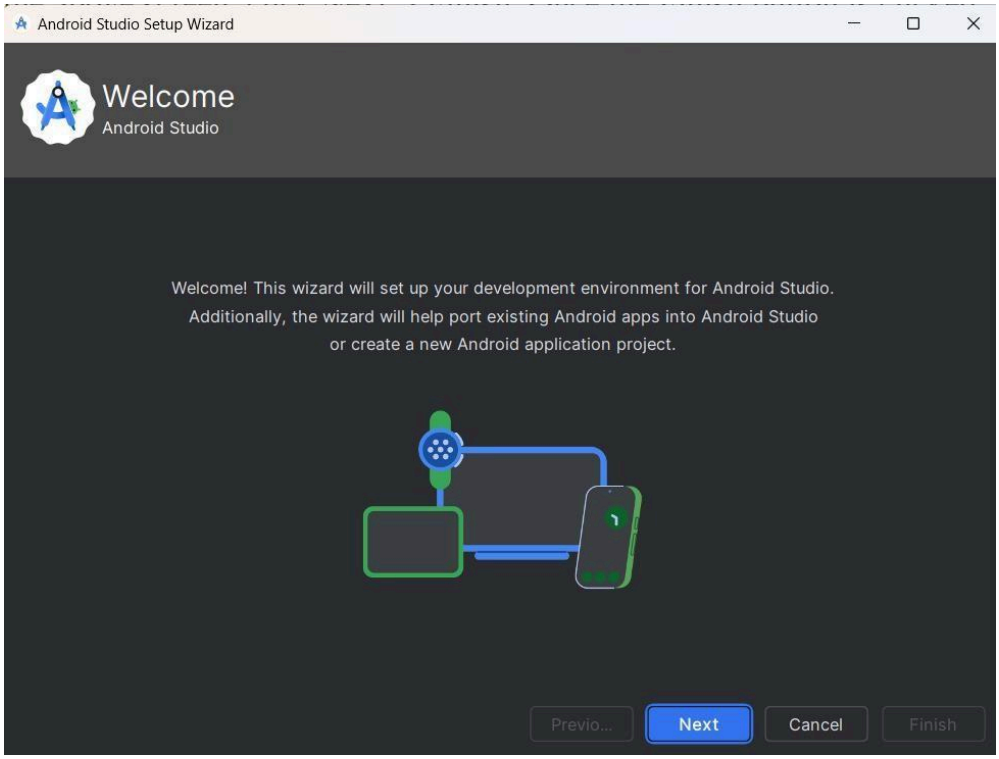


Step 8.3: - Change the destination as per your convenience and click on ‘Next’ Button.

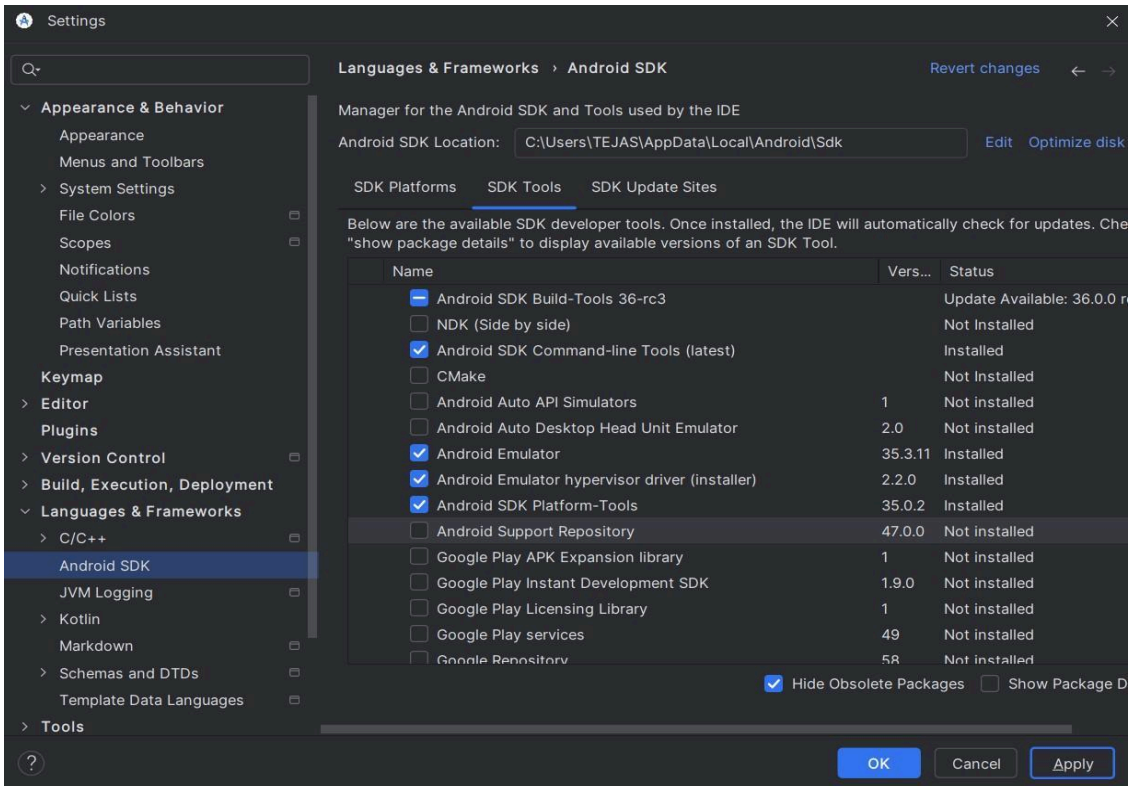


Step 8.4: - Follow the steps of the installation wizard. Once the installation wizard completes, you will get the following screen.





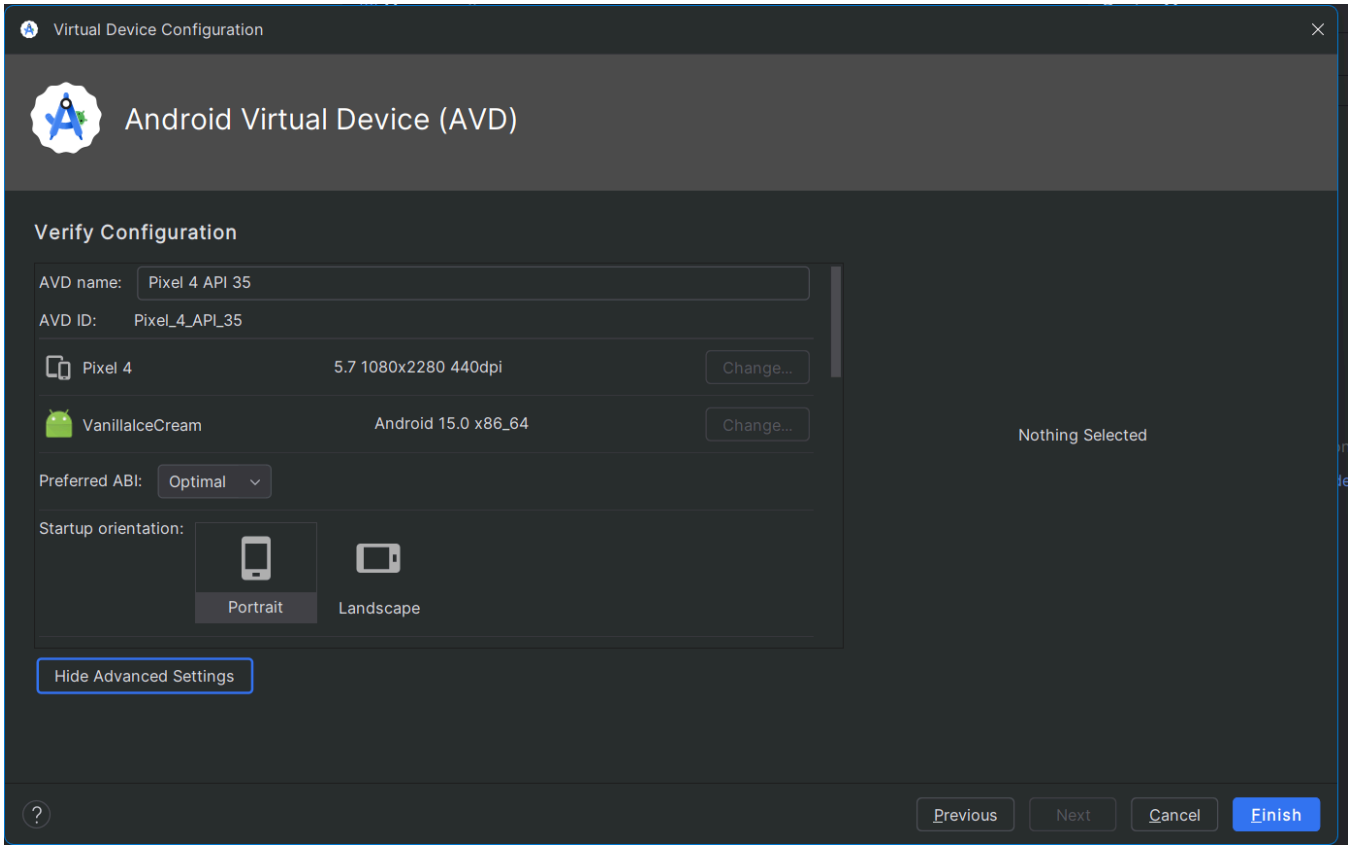
Step 8.5: - Go to Preferences > Appearance & Behavior > System Settings > Android SDK. Select the SDK Tools tab and check Android SDK Command-line Tools and Install it.



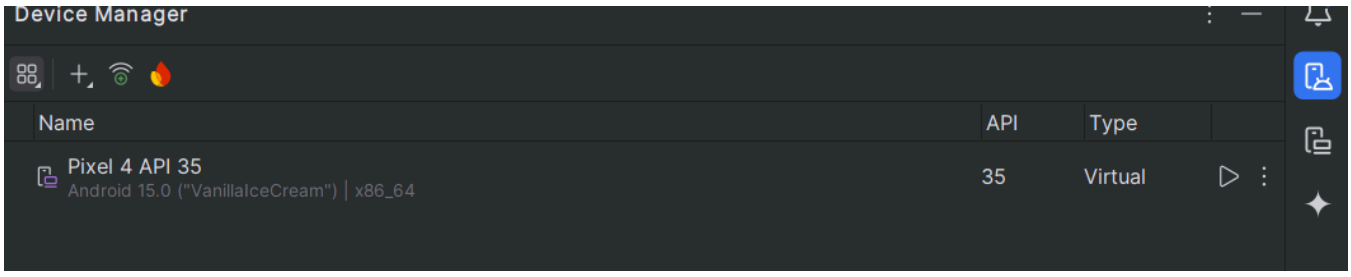
Step 9: - Open a terminal and run the following command

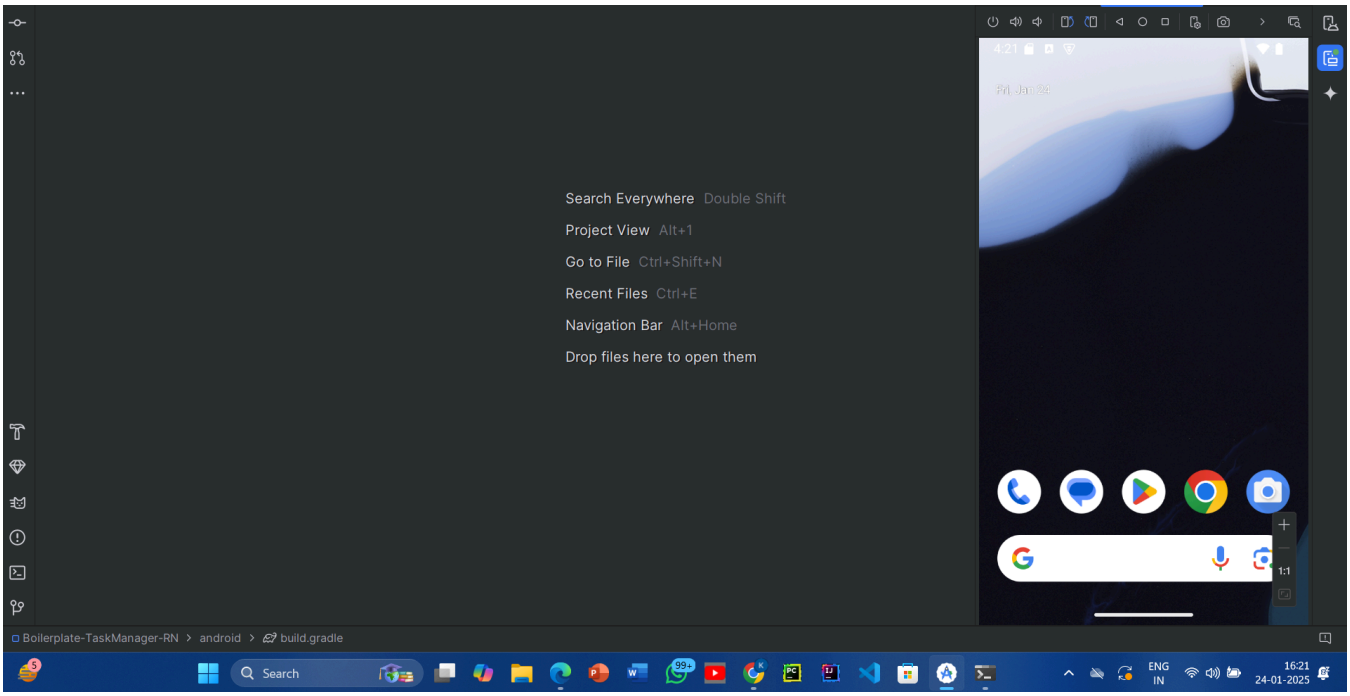
```
All SDK package licenses accepted


C:\Users\kunup>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.27.3, on Microsoft Windows [Version 10.0.22631.4602], 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
[X] Visual Studio - develop Windows apps
    X Visual Studio not installed; this is necessary to develop Windows apps.
      Download at https://visualstudio.microsoft.com/downloads/.
      Please install the "Desktop development with C++" workload, including all of its default components
[√] Android Studio (version 2024.2)
[√] VS Code (version 1.96.4)
[√] Connected device (3 available)
[√] Network resources

! Doctor found issues in 1 category.
```

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



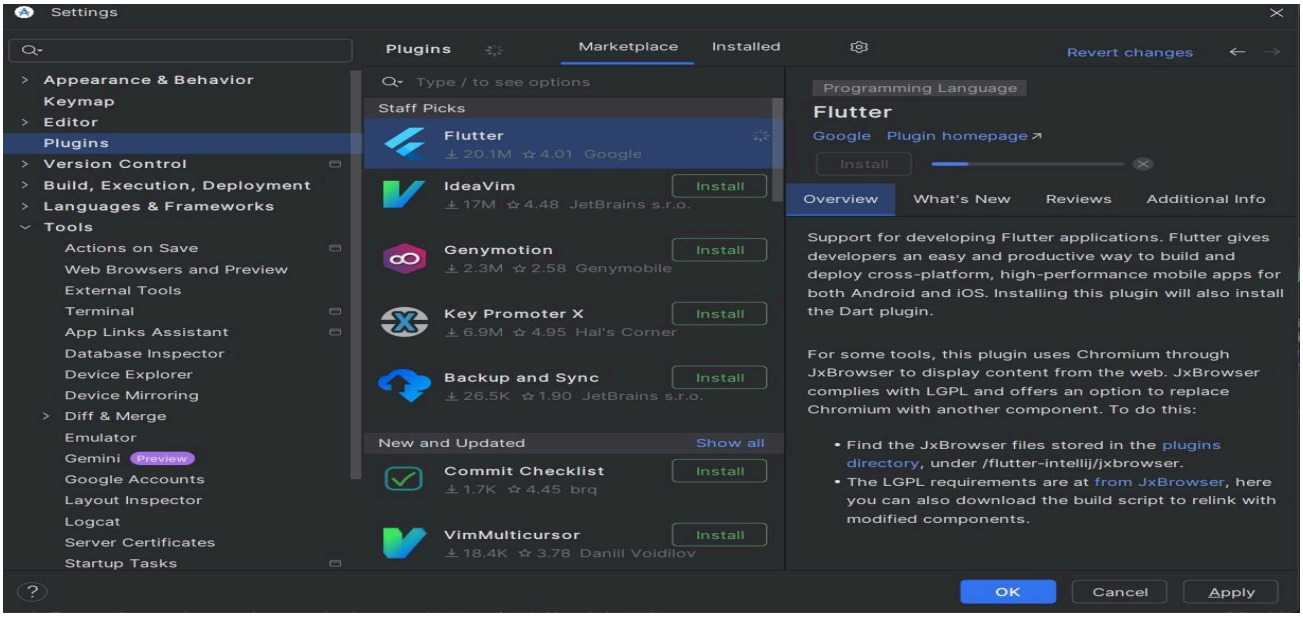
Step 10.2: - Click on the icon pointed into the red color rectangle. The Android emulator displayed as below screen

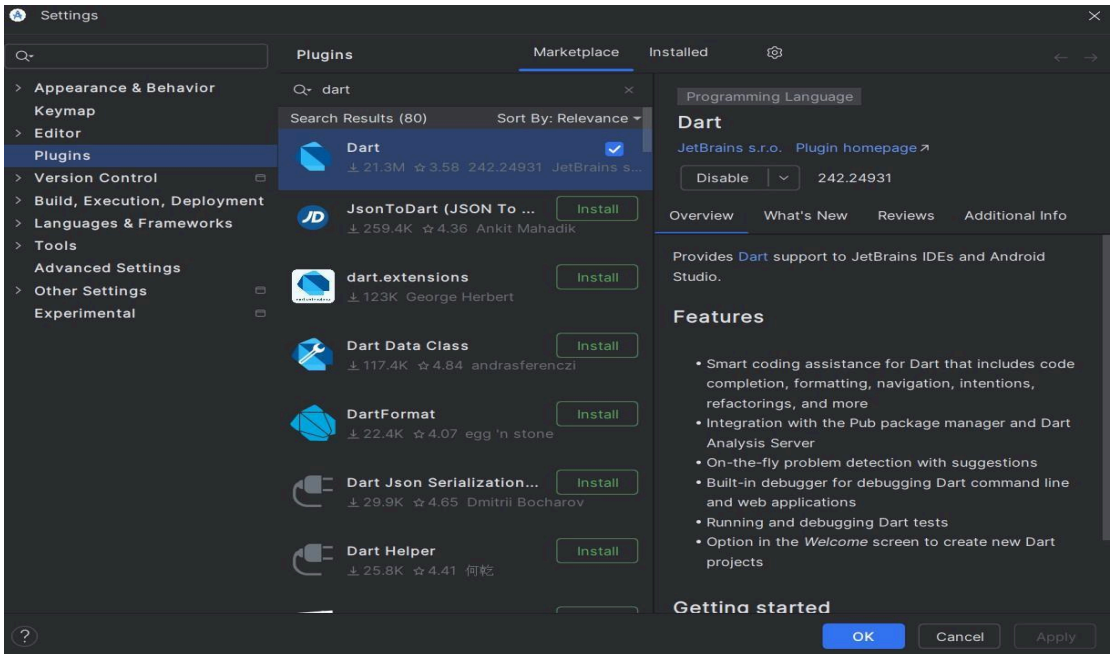




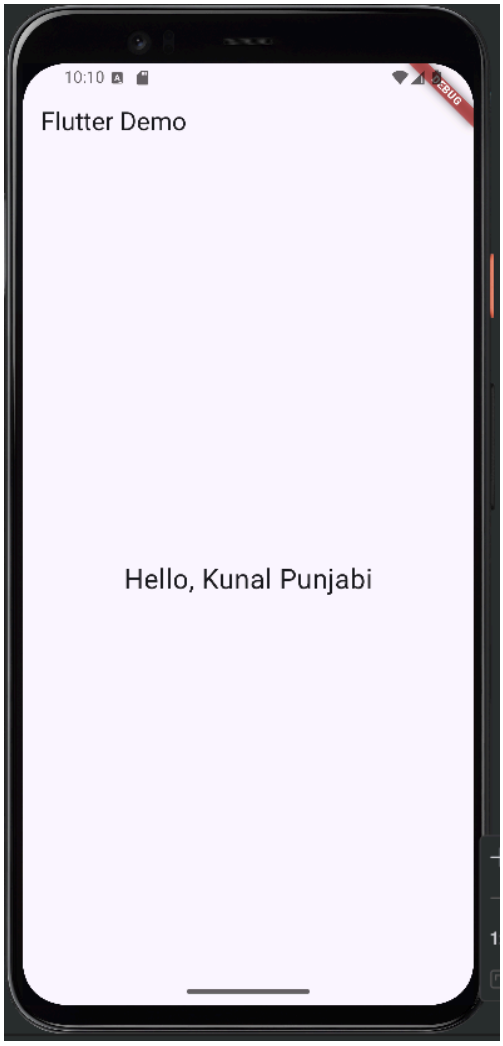
Step 11: - Now, install Flutter and Dart plugin for building Flutter application in Android Studio. These plugins provide a template to create a Flutter application, give an option to run and debug Flutter application in the Android Studio itself

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



Project Title: **Roll No.**

MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	43
Name	Kunal Punjabi
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

EXPERIMENT NO: - 02

Name:- Kunal Punjabi **Class:-** D15A **Roll:No: -** 43

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

Theory: Flutter UI in Jeans Shopping App

Each element on the screen of a Flutter app is a widget. The UI structure is built using a tree of widgets, where each widget defines either a visual component, a layout structure, or user interactions.
The appearance and behavior of the app are determined by the choice, sequence, and nesting of these widgets.
Flutter follows a reactive UI approach, meaning when any code modification occurs, only the necessary widgets are rebuilt based on the difference between the previous and the current state. This ensures efficient UI rendering.
In this Jeans Shopping App, various single-child and multi-child widgets are used to construct the UI efficiently.

Types of Widgets in Jeans Shopping App

1. Stateful and Stateless Widgets

- **StatelessWidget**: These widgets do not change once built. The HomePage class extends StatelessWidget, displaying a list of jeans products.
- **StatefulWidget**: These widgets maintain state and update dynamically. The CartProvider class (used with Provider) helps manage the cart's state, ensuring live updates when products are added.

Commonly Used Widgets in Jeans Shopping App

1. **Layout Widgets**
   - Scaffold: Provides the base structure, including the AppBar and Body.
   - GridView.builder: Used to display the list of jeans in a structured grid format, making product browsing easier.
   - Card: Each product is enclosed within a material Card widget for better visual appeal.
   - Column & Row: Used to align text, buttons, and product details properly inside each product card.
2. **Interactive Widgets**
   - GestureDetector: Detects taps on a product and navigates to ProductDetailsPage.
   - IconButton: The shopping cart icon on the AppBar allows users to navigate to the cart page.
   - Stack & Positioned: Used to display a dynamic cart count badge over the shopping cart icon.
3. **Image and Text Widgets**
   - Image.asset: Displays product images stored in the app's assets.
   - Text: Displays product names and prices with customized styles.
4. **Navigation and State Management**
   - Navigator.push: Handles page transitions when the user selects a product.
   - Provider & ValueListenableBuilder: Manage cart updates and dynamically display the number of items in the cart.

Code : homepage.dart

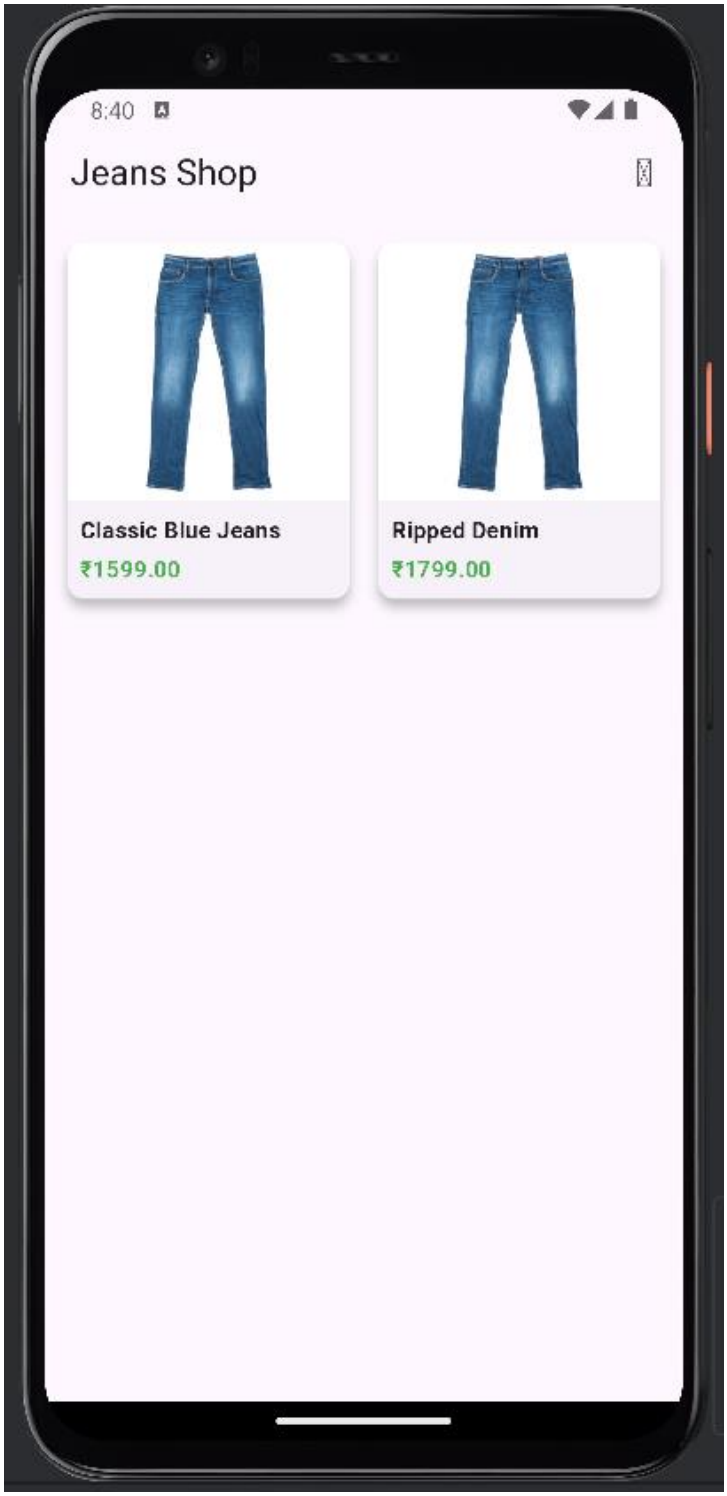
```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'cart_page.dart';
import 'cart_provider.dart';
import 'product_details.dart';

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

  final List<Map<String, dynamic>> jeansProducts = const [
    {
      'imagePath': 'assets/jeans0.jpg',
      'productName': 'Classic Blue Jeans',
      'price': 1599.0,
      'colors': ['Blue', 'Dark Blue', 'Light Blue'],
    },
    {
      'imagePath': 'assets/jeans1.jpg',
      'productName': 'Ripped Denim',
      'price': 1799.0,
      'colors': ['Black', 'Gray', 'Navy Blue'],
    },
  ];

  @override
  Widget build(BuildContext context) {
    final cartProvider = Provider.of<CartProvider>(context);

    return Scaffold(
      appBar: AppBar(
        title: const Text('Jeans Shop'),
        actions: [
          Stack(
            children: [
              IconButton(
                icon: const Icon(Icons.shopping_cart),
                onPressed: () {
                  Navigator.push(
                    context,
                    MaterialPageRoute(
                      builder: (context) => const CartPage(), // ☑
FIXED: `cartItems` hata diya
                    ),
                  );
                },
              ),
              ValueListenableBuilder<List<Map<String,
dynamic>>>(
```

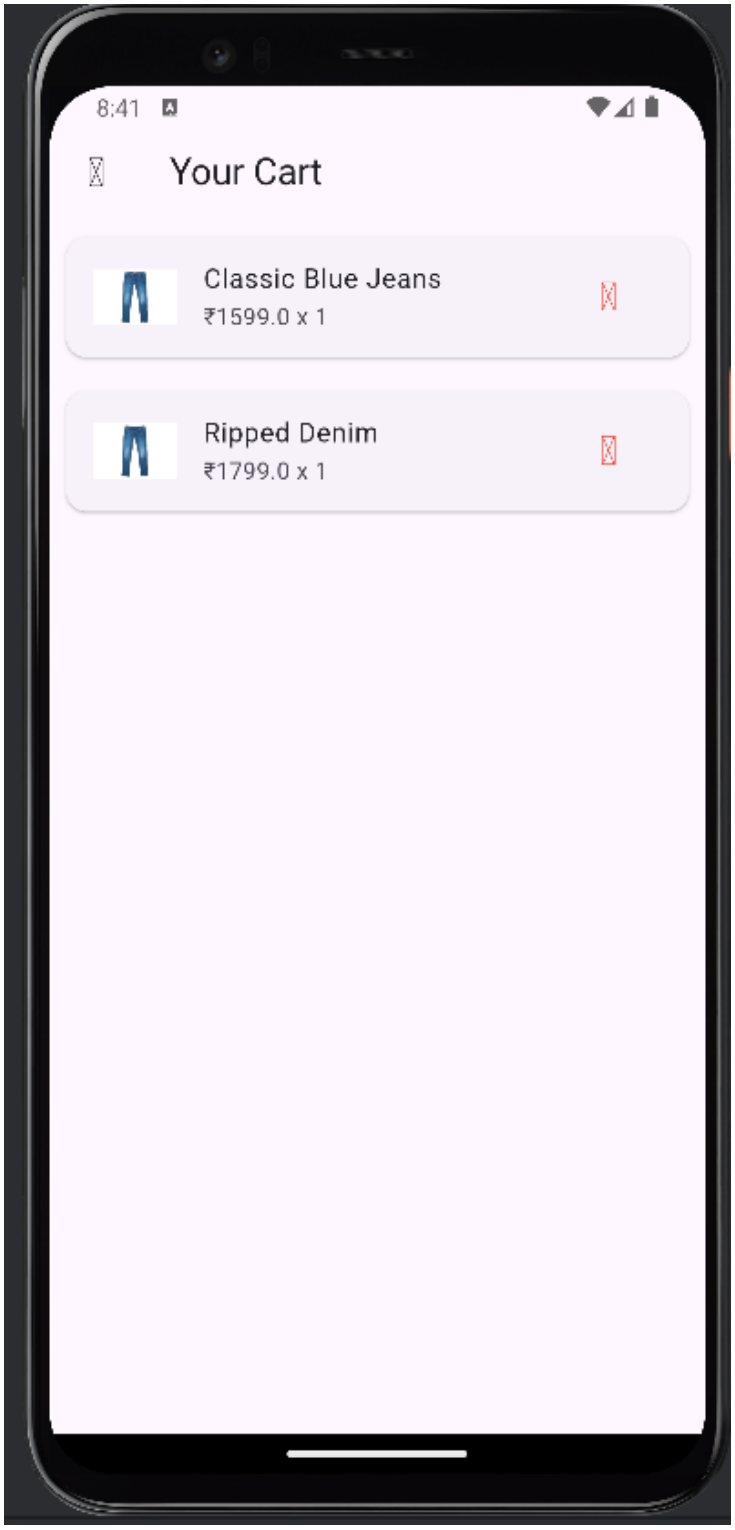
```
            valueListenable: cartProvider.cartItems,
            builder: (context, cartItems, child) {
              return cartItems.isNotEmpty
                  ? Positioned(
                right: 10,
                top: 5,
                child: Container(
                  padding: const EdgeInsets.all(6),
                  decoration: const BoxDecoration(
                    color: Colors.red,
                    shape: BoxShape.circle,
                  ),
                  child: Text(
                    cartItems.length.toString(),
                    style: const TextStyle(
                      color: Colors.white,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                ),
              )
                  : const SizedBox();
            },
          ),
        ],
      ),
    ],
  ),
  body: GridView.builder(
    padding: const EdgeInsets.all(10),
    gridDelegate: const
SliverGridDelegateWithFixedCrossAxisCount(
      crossAxisCount: 2,
      crossAxisSpacing: 10,
      mainAxisSpacing: 10,
      childAspectRatio: 0.8,
    ),
    itemCount: jeansProducts.length,
    itemBuilder: (context, index) {
      var product = jeansProducts[index];
      return GestureDetector(
        onTap: () {
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) => ProductDetailsPage(
                imagePath: product['imagePath'],
                productName: product['productName'],
                price: product['price'],
                colors: product['colors'],
              ),
```

```
            ),
          );
        },
        child: Card(
          elevation: 5,
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(10),
          ),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Expanded(
                child: ClipRRect(
                  borderRadius: const BorderRadius.vertical(
                    top: Radius.circular(10),
                  ),
                  child: Image.asset(
                    product['imagePath'],
                    fit: BoxFit.cover,
                    width: double.infinity,
                  ),
                ),
              ),
              Padding(
                padding: const EdgeInsets.all(8),
                child: Column(
                  crossAxisAlignment:
CrossAxisAlignment.start,
                  children: [
                    Text(
                      product['productName'],
                      style: const TextStyle(
                        fontWeight: FontWeight.bold,
                        fontSize: 14,
                      ),
                    ),
                    const SizedBox(height: 4),
                    Text(
                      "₹${product['price'].toStringAsFixed(2)}",
                      style: const TextStyle(
                        color: Colors.green,
                        fontWeight: FontWeight.bold,
                      ),
                    ),
                  ],
                ),
              ),
            ],
          ),
        ),
      );
```

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

Screenshots:


8:40
Jeans Shop
Classic Blue Jeans
₹1599.00
Ripped Denim
₹1799.00


8:41
Your Cart
Classic Blue Jeans
₹1599.0 x 1
Ripped Denim
₹1799.0 x 1

Project Title: **Roll No.**

MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	43
Name	Kunal Punjabi
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

EXPERIMENT NO: - 03

Name:- Kunal Punjabi **Class:-** D15A **Roll:No: -** 43

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

Theory: -

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

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

Visual elements play a significant role in app development.

- **Enhanced User Experience**: Images and icons make your app visually appealing and user-friendly.
- **Information Conveyance**: They convey information quickly and intuitively. A well-chosen icon can replace lengthy text.
- **Branding**: Custom icons and images reinforce your app's branding, making it memorable.

Adding Icons in Flutter

Flutter provides built-in material design icons through the Icons class. Custom icons can also be added using third-party packages such as flutter_launcher_icons and font_awesome_flutter.

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

Adding Images in Flutter

Flutter supports images from three sources:

1. **Assets** (Stored locally in the project)

- Place the image inside the assets/images folder in the project. ● Declare the image in pubspec.yaml

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

- Display the image in the app

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

2. **Network** (Fetched from the internet)

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

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

3. **Memory or File** (Stored on the device)

Adding Custom Fonts in Flutter

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

- Download the font and place it in the assets/fonts/ folder.
- Declare the font in pubspec.yaml
- Use the font in the app

```
Text(
  'Custom Font Example', style: TextStyle(fontFamily:
  'CustomFont', fontSize: 24),
);
```

Code: - product_details.dart

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import
'package:url_launcher/url_launcher.dart
';
import 'cart_provider.dart';

class ProductDetailsPage extends
StatefulWidget {
 final String imagePath;
 final String productName;
 final double price;
 final List<String> colors;

 const ProductDetailsPage({
  Key? key,
  required this.imagePath,
  required this.productName,
  required this.price,
  required this.colors,
 }) : super(key: key);

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

class _ProductDetailsPageState extends
State<ProductDetailsPage> {
 late String selectedColor;
 int selectedSize = 28;
 final List<int> availableSizes = [28,
30, 32, 34, 36, 38];

 @override
 void initState() {
  super.initState();
  selectedColor = widget.colors.first;
 }

 void _addToCart() {
  final cartProvider =
Provider.of<CartProvider>(context,
listen: false);

  cartProvider.addToCart({
   'imagePath': widget.imagePath,
   'productName':
widget.productName,
   'price': widget.price,
   'color': selectedColor,
   'size': selectedSize,
   'quantity': 1,
  });


ScaffoldMessenger.of(context).showSn
ackBar(
   SnackBar(
    content:
Text("${widget.productName} added to
cart!"),
    duration: const Duration(seconds:
2),
   ),
  );
 }

 Future<void> _orderNow() async {
  String message =
   "Hello, I want to order:\n-
*${widget.productName}*\n- Price:
₹${widget.price}\n- Color:
$selectedColor\n- Size: $selectedSize";
  Uri url =
```

```
Uri.parse("https://wa.me/+919876543210?text=${Uri.encodeComponent(message)}");

  if (await canLaunchUrl(url)) {
   await launchUrl(url, mode: LaunchMode.externalApplication);
  } else {
   if (mounted) {
ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text("Could not open WhatsApp")),
     );
   }
  }
 }

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(title: Text(widget.productName)),
   body: SingleChildScrollView(
    child: Padding(
     padding: const EdgeInsets.all(16.0),
     child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
       // Product Image
       ClipRRect(
        borderRadius: BorderRadius.circular(10),
        child: Image.asset(
         widget.imagePath,
         width: double.infinity,
         height: 250,
         fit: BoxFit.cover,
        ),
       ),
       const SizedBox(height: 16),

       // Product Name
       Text(
        widget.productName,
        style: const TextStyle(fontSize: 22, fontWeight: FontWeight.bold),
       ),
       const SizedBox(height: 8),

       // Product Price
       Text(
"₹${widget.price.toStringAsFixed(2)}",
        style: const TextStyle(fontSize: 18, color: Colors.green, fontWeight: FontWeight.bold),
       ),
       const SizedBox(height: 16),

       // Select Color
       const Text("Select Color:", style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold)),
       Wrap(
        spacing: 8,
        children: widget.colors.map((color) {
         return ChoiceChip(
          label: Text(color),
```

```
                selected: selectedColor == color,
                selectedColor: Colors.blueAccent,
                onSelected: (selected) => setState(() => selectedColor = color),
                labelStyle: TextStyle(color: selectedColor == color ? Colors.white : Colors.black),
              );
            }).toList(),
          ),
          const SizedBox(height: 16),

          // Select Size
          const Text("Select Size:", style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold)),
          Wrap(
            spacing: 8,
            children: availableSizes.map((size) {
              return ChoiceChip(
                label: Text(size.toString()),
                selected: selectedSize == size,
                selectedColor: Colors.blueAccent,
                onSelected: (selected) => setState(() => selectedSize = size),
                labelStyle: TextStyle(color: selectedSize == size ? Colors.white : Colors.black),
              );
            }).toList(),
          ),
          const SizedBox(height: 24),

          // Buttons
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
            children: [
              ElevatedButton.icon(
                onPressed: _addToCart,
                icon: const Icon(Icons.shopping_cart),
                label: const Text("Add to Cart"),
              ),
              ElevatedButton.icon(
                onPressed: _orderNow,
                icon: const Icon(Icons.shopping_bag),
                label: const Text("Order Now"),
                style: ElevatedButton.styleFrom(backgroundColor: Colors.green),
              ),
            ],
          ),
        ],
      ),
    ),
  );
 }
}
```

Screeshots :


8:40
Classic Blue Jeans
Classic Blue Jeans
₹1599.00
Select Color:
Blue
Dark Blue
Light Blue
Select Size:
28
30
32
34
36
38
Add to Cart
Order Now

Project Title: **Roll No.**

MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	43
Name	Kunal Punjabi
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

EXPERIMENT NO: - 04

Name:- Kunal Punjabi **Class:-** D15A **Roll:No: -** 43

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

Theory: -

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

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

Creation of a Form

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

A **GlobalKey<FormState>** is required to uniquely identify the form and enable validation or data retrieval from the form fields.

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

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

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

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

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

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

Some Properties of Form Widget

- **key:** A GlobalKey that uniquely identifies the Form. You can use this key to interact with the form, such as validating, resetting, or saving its state.
- **child:** The child widget that contains the form fields. Typically, this is a Column, ListView, or another widget that allows you to arrange the form fields vertically.
- **autovalidateMode:** An enum that specifies when the form should automatically validate its fields.

Some Methods of Form Widget

- **validate():** This method is used to trigger the validation of all the form fields within the Form. It returns true if all fields are valid, otherwise false. You can use it to check the overall validity of the form before submitting it.
- **save():** This method is used to save the current values of all form fields. It invokes the onSaved callback for each field. Typically, this method is called after validation succeeds.
- **reset():** Resets the form to its initial state, clearing any user-entered data.
- **currentState:** A getter that returns the current FormState associated with the Form.

Code: -

```
import
'package:flutter/material.d
art';
import 'homepage.dart';  //
Correct import


class LoginPage extends
StatefulWidget {
  @override
  _LoginPageState
createState() =>
_LoginPageState();
}


class _LoginPageState extends
State<LoginPage> {
  final TextEditingController
_emailController =
TextEditingController();
```

```
  final
TextEditingController
_passwordController =
TextEditingController();
  final
GlobalKey<FormState>
_formKey =
GlobalKey<FormState>();

  void _login() {
    if
(_formKey.currentState!.v
alidate()) {
      String email =
_emailController.text.trim(
);
      String password =
_passwordController.text.t
rim();

      if (email ==
"test@example.com" &&
password ==
"password123") {

Navigator.pushReplaceme
nt(
          context,

MaterialPageRoute(builde
r: (context) =>
HomePage()), // cartItems
removed
        );
      } else {

ScaffoldMessenger.of(context)
.showSnackBar(
          const SnackBar(content:
Text("Invalid email or
password")),
        );
      }
    }
  }

  @override
  Widget build(BuildContext
context) {
    return Scaffold(
      appBar: AppBar(title:
const Text('Login')),
      body: Padding(
        padding: const
EdgeInsets.all(16.0),
        child: Form(
          key: _formKey,
          child: Column(
            crossAxisAlignment:
CrossAxisAlignment.start,
            children: [
              TextFormField(
                controller:
_emailController,
                decoration: const
InputDecoration(labelText:
'Email'),
                validator: (value) {
```

```
              if (value ==
null || value.isEmpty) {
                return 'Please
enter your email';
              }
              return null;
            },
          ),
          TextFormField(
            controller:
_passwordController,
            decoration: const
InputDecoration(labelText
: 'Password'),
            obscureText:
true,
            validator:
(value) {
              if (value ==
null || value.isEmpty) {
                return 'Please
enter your password';
              }
              return null;
            },
          ),
          const
SizedBox(height: 20),
          ElevatedButton(
            onPressed:
_login,
            child: const
Text('Login'),
          ),
        ],
      ),
    ),
  );
 }
}
```

signup.dart

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

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

class _SignUpPageState
extends State<SignUpPage> {
 final _formKey =
GlobalKey<FormState>();
 final TextEditingController
_passwordController =
TextEditingController();
 final TextEditingController
_confirmPasswordController
= TextEditingController();

 bool _isPasswordVisible =
false;
 bool
```

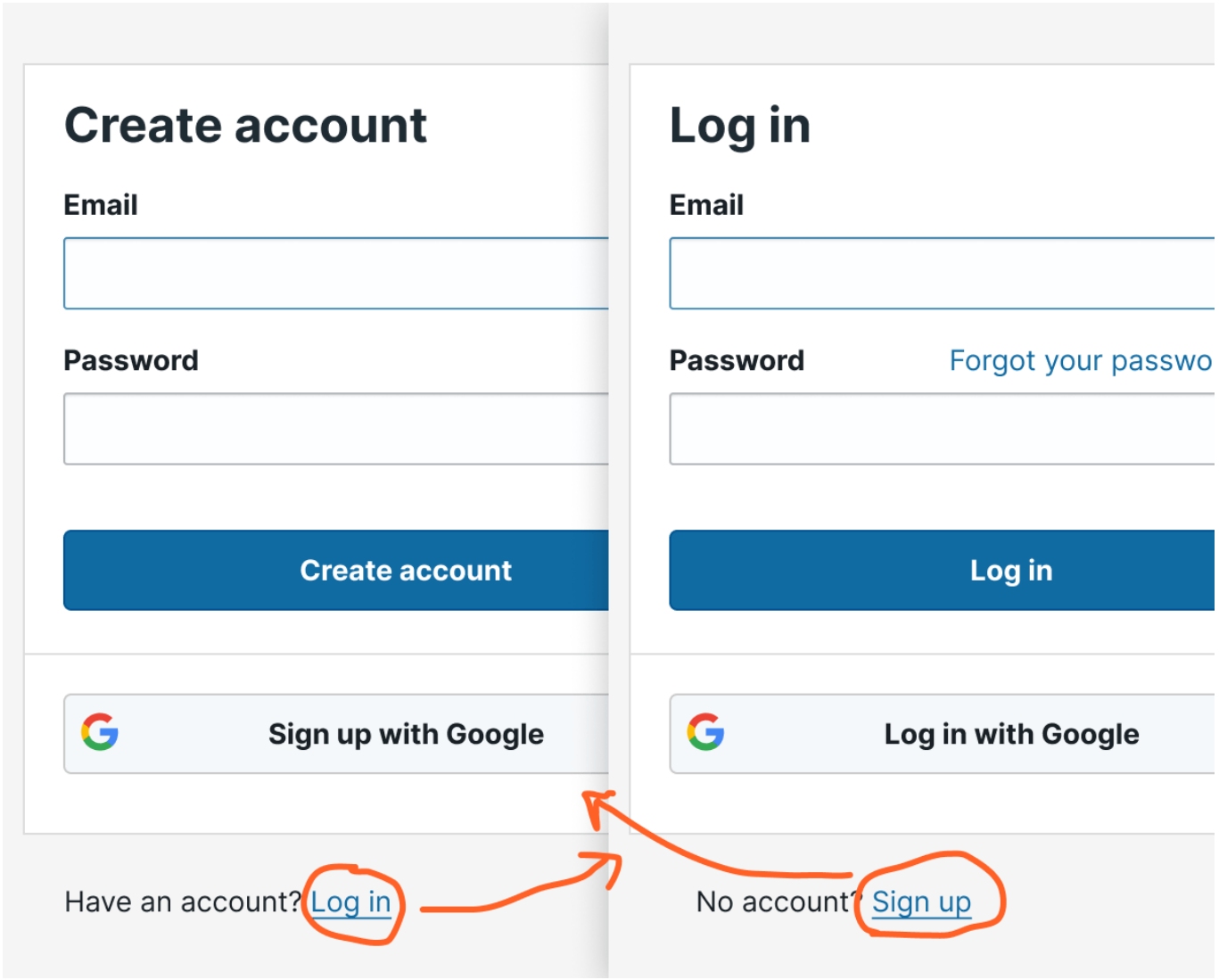
```
_isConfirmPasswordVisible = false;

  void _signUp() {
    if (_formKey.currentState!.validate()) {

ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text("Sign-up successful! Redirecting...")),
      );

      Future.delayed(const Duration(seconds: 2), () {

Navigator.pushReplacement(
          context,
          MaterialPageRoute(
            builder: (context) => const HomePage(), // Removed cartItems argument
          ),
        );
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Sign Up')),
      body: SingleChildScrollView(
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Form(
            key: _formKey,
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                const Text("Create Account", style: TextStyle(fontSize: 22, fontWeight: FontWeight.bold)),
                const SizedBox(height: 10),

                // Name Field
                TextFormField(
                  decoration: const InputDecoration(
                    labelText: 'Full Name',
                    border: OutlineInputBorder(),
                    prefixIcon: Icon(Icons.person),
```

```
              ),
              validator:
(value) {
                if (value ==
null || value.isEmpty) {
                  return
'Please enter your name';
                }
                return null;
              },
            ),
            const
SizedBox(height: 12),

            // Email Field
            TextFormField(
              decoration:
const InputDecoration(
                labelText:
'Email Address',
                border:
OutlineInputBorder(),
                prefixIcon:
Icon(Icons.email),
              ),
              keyboardType:
TextInputType.emailAddre
ss,
              validator:
(value) {
                if (value ==
null || value.isEmpty) {
                  return
'Please enter your email';
                }
                if (!RegExp(r'^[a-
zA-Z0-9_.+-]+@[a-zA-Z0-9-
]+\.[a-zA-Z0-9-
.]+$').hasMatch(value)) {
                  return 'Enter a
valid email';
                }
                return null;
              },
            ),
            const
SizedBox(height: 12),

            // Password Field
            TextFormField(
              controller:
_passwordController,
              decoration:
InputDecoration(
                labelText:
'Password',
                border: const
OutlineInputBorder(),
                prefixIcon: const
Icon(Icons.lock),
                suffixIcon:
IconButton(
                  icon:
Icon(_isPasswordVisible ?
Icons.visibility :
Icons.visibility_off),
                  onPressed: () =>
setState(() =>
```

```
_isPasswordVisible =
!_isPasswordVisible),
                ),
              ),
              obscureText:
!_isPasswordVisible,
              validator:
(value) {
                if (value ==
null || value.isEmpty) {
                  return
'Please enter a password';
                }
                if
(value.length < 6) {
                  return
'Password must be at least
6 characters';
                }
                return null;
              },
            ),
            const
SizedBox(height: 12),

            // Confirm
Password Field
            TextFormField(
              controller:
_confirmPasswordControll
er,
              decoration:
InputDecoration(
                labelText:
'Confirm Password',
                border: const
OutlineInputBorder(),
                prefixIcon: const
Icon(Icons.lock),
                suffixIcon:
IconButton(
                  icon:
Icon(_isConfirmPasswordVisi
ble ? Icons.visibility :
Icons.visibility_off),
                  onPressed: () =>
setState(() =>
_isConfirmPasswordVisible =
!_isConfirmPasswordVisible),
                ),
              ),
              obscureText:
!_isConfirmPasswordVisible,
              validator: (value) {
                if (value == null ||
value.isEmpty) {
                  return 'Please
confirm your password';
                }
                if (value !=
_passwordController.text) {
                  return
'Passwords do not match';
                }
                return null;
              },
            ),
            const
```

```
SizedBox(height: 20),

            // Sign-Up
Button
            SizedBox(
              width:
double.infinity,
              child:
ElevatedButton(
                onPressed:
_signUp,
                child: const
Text('Sign Up', style:
TextStyle(fontSize: 16)),
              ),
            ),
          ],
        ),
      ),
    ),
  );
 }
}
```

SCREENSHOTS :



Project Title: **Roll No.**

MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	43
Name	Kunal Punjabi
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

EXPERIMENT NO: - 05

Name:- Kunal Punjabi **Class:-** D15A **Roll:No: -** 43

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

Theory: -

In Flutter, navigation, routing, and gestures play a key role in making an app interactive and user-friendly. The Navigator class is used to manage app navigation by pushing and popping screens (routes). The CartPage in this application demonstrates how navigation and state management work together to create a smooth user experience.

Navigation and Routing in Flutter

Navigation allows switching between different screens in an app. Flutter provides a Navigator widget to handle this using a stack-based approach.

1. Using Navigator for Routing

- The CartPage is navigated to when the user taps the cart icon on the home screen.
- This is done using Navigator.push(), which adds a new screen to the navigation stack.
- To go back, Navigator.pop(context) can be used.

2. State Management in Navigation

The Provider package is used in this app for managing cart data. The CartProvider class:

- Stores cart items using ValueNotifier.
- Updates UI automatically when items are added or removed.
- Provides the total price calculation for the cart.

Handling Gestures in Flutter

Flutter provides the GestureDetector widget and built-in button widgets to handle gestures. In CartPage, gestures are used for user interactions:

1. Tap Gesture (Adding & Removing Items)

- The IconButton with a delete icon allows users to remove an item from the cart when tapped.

2. ListView Scrolling Gesture

- ListView.builder() is used to display cart items in a scrollable list.
- The user can scroll vertically to see all items.

Conclusion:

The CartPage effectively demonstrates navigation, routing, and gesture handling in Flutter.

- Navigation: Uses Navigator.push() to open the cart screen.
- State Management: Uses Provider to dynamically update the cart.
- Gestures: Uses tap gestures for adding/removing items and scrolling for viewing the cart list.

Code: -

Cart_page.dart:

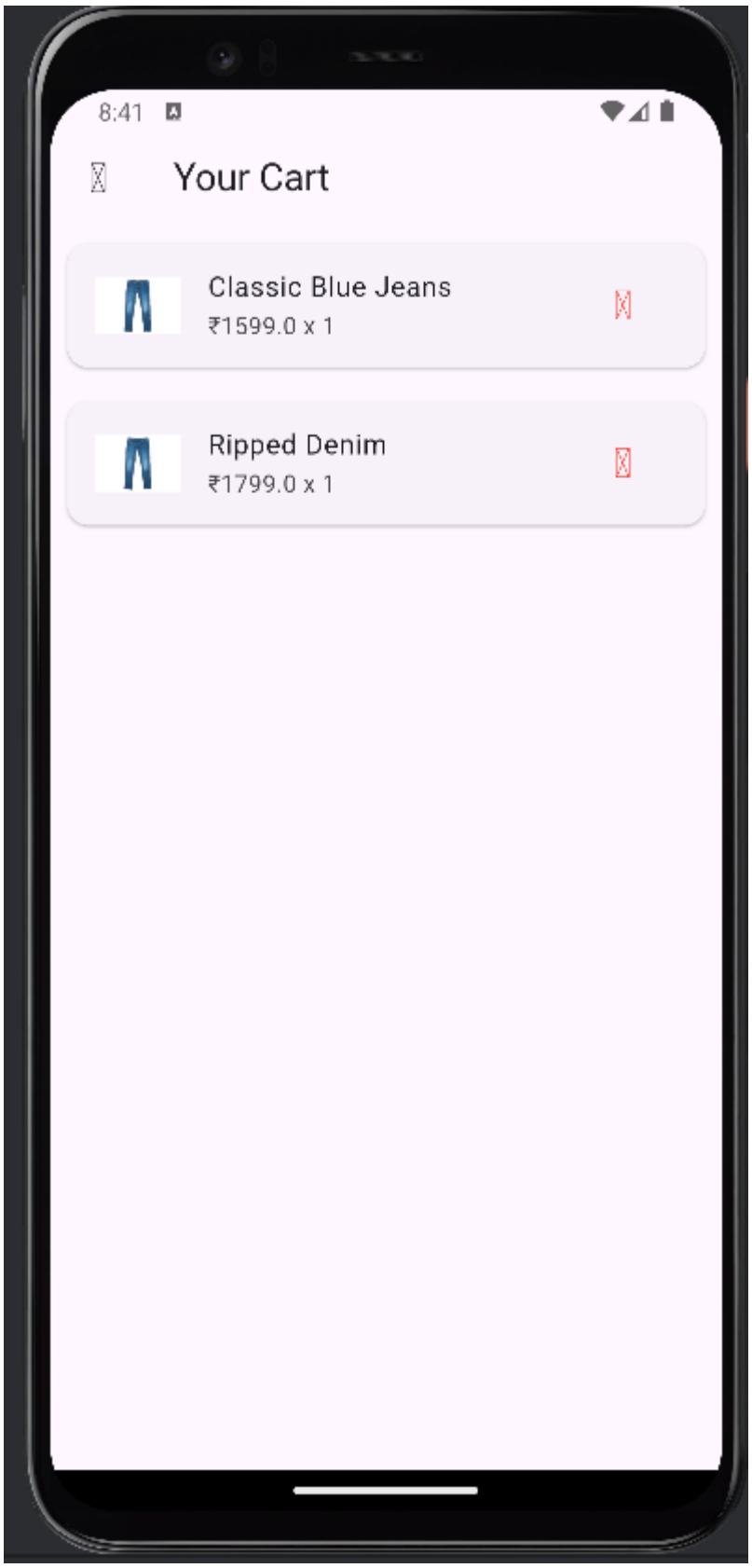
```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'cart_provider.dart';

class CartPage extends StatelessWidget {
  const CartPage({Key? key}) : super(key:
key);

  @override
  Widget build(BuildContext context) {
    final cartProvider =
Provider.of<CartProvider>(context);

    return Scaffold(
      appBar: AppBar(title: const Text('Your
Cart')),
      body:
ValueListenableBuilder<List<Map<String,
dynamic>>>(
        valueListenable:
cartProvider.cartItems,
        builder: (context, cartItems, child) {
          return cartItems.isEmpty
              ? const Center(child: Text("Your
cart is empty!"))
              : ListView.builder(
            itemCount: cartItems.length,
            itemBuilder: (context, index) {
              var item = cartItems[index];
              return Card(
                margin: const
```

```
EdgeInsets.symmetric(vertical: 10,
horizontal: 10),
              child: ListTile(
                leading:
Image.asset(item['imagePath'] ??
'assets/placeholder.jpg', width: 50),
                title: Text(item['productName']
?? 'No Name'),
                subtitle: Text("₹${item['price']}
x ${item['quantity'] ?? 1}"),
                trailing: IconButton(
                  icon: const Icon(Icons.delete,
color: Colors.red),
                  onPressed: () {

cartProvider.removeFromCart(index);
                  },
                ),
              ),
            );
          },
        );
      },
    ),
  );
 }
}
```

cart_provider.dart

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

class CartProvider with
ChangeNotifier {
  final
ValueNotifier<List<Map<String
, dynamic>>> _cartItems =
ValueNotifier([]);


ValueNotifier<List<Map<String
, dynamic>>> get cartItems =>
_cartItems;
```

```
  void addToCart(Map<String,
dynamic> item) {
    _cartItems.value =
[..._cartItems.value, item];
    _cartItems.notifyListeners();
  }

  void removeFromCart(int
index) {
    _cartItems.value =
List.from(_cartItems.value)..re
moveAt(index);
    _cartItems.notifyListeners();
  }

  void updateQuantity(int index,
int quantity) {
    if (quantity > 0) {

_cartItems.value[index]['quantit
y'] = quantity;
    } else {
      removeFromCart(index);
    }
    _cartItems.notifyListeners();
  }

  double getTotalPrice() {
    return
_cartItems.value.fold(0, (total,
item) => total + (item['price'] *
(item['quantity'] ?? 1)));
  }
}
```


8:41
Your Cart
Classic Blue Jeans
₹1599.0 x 1
Ripped Denim
₹1799.0 x 1

Project Title: **Roll No.**

MAD & PWA Lab
Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	43
Name	Kunal Punjabi
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	

EXPERIMENT NO: - 06

Name : Kunal Punjabi

Class : D15A

Roll:No : 43

AIM: To connect flu er UI with firebase database.

Introduc on to Firebase and Flu er Integra on

Firebase is a comprehensive pla orm developed by Google, designed to help developers build highquality applica ons for both mobile and web. It provides essen al services such as real- me databases, authen ca on, cloud storage, hos ng, and much more. One of the most widely used Firebase services is the Firebase Real me Database, which is a NoSQL cloud database that allows data to be stored and synced in real- me across all connected devices. Flu er, on the other hand, is an open-source UI so ware development kit created by Google, which allows developers to build na vely compiled applica ons for mobile, web, and desktop from a single codebase. Its rich set of pre-designed widgets and powerful tools makes Flu er an a rac ve op on for developing visually appealing and performant applica ons. Integra ng Firebase with Flu er allows developers to leverage the full poten al of Firebase services in their applica ons. By using Firebase's Real me Database, Flu er apps can achieve features such as real- me data synchroniza on, secure authen ca on, and cloud-based storage. This combina on enables developers to create powerful, scalable, and feature-rich mobile and web applica ons.

Firebase Real me Database Overview

Firebase Real me Database is a cloud-hosted NoSQL database that stores data in a JSON-like format. The key characteris c of this database is its real- me synchroniza on feature, meaning that any changes made to the database are instantly reflected on all clients (i.e., devices) connected to it. This makes it an ideal solu on for applica ons that require frequent updates and need to maintain synchronized data across mul ple users or devices, such as messaging apps, social media pla orms, or collabora ve tools.

The Firebase Real me Database is structured as a tree of data, where each node in the tree can contain key value pairs. This structure allows for easy data retrieval and modifica on. Firebase's real- me capabili es enable apps to immediately receive updates to the data whenever it changes, without the need to refresh or reload the page. Addi onally, the database supports offline data persistence, meaning that even if the user's device loses its internet connec on, the app can s ll func on by using the locally cached data.

Se ng Up Firebase in Flu er :
To connect a Flu er app with Firebase, the following steps are typically followed:

1. Crea ng a Firebase Project: To start using Firebase with Flu er, the first step is to create a Firebase project in the Firebase Console. Once the project is created, developers can associate their Flu er app with the Firebase project by following the pla orm-specific instruc ons for Android or iOS. This usually involves configuring API keys, downloading configura on files, and adding them to the Flu er project.
2. Integra ng Firebase SDK in Flu er: A er the Firebase project is set up, developers need to integrate Firebase's SDK into the Flu er app. This involves adding the necessary dependencies to the Flu er project's pubspec.yaml file. For Firebase's Real me Database, the package firebase_database

is used. Addi onally, Firebase's core SDK (firebase_core) must also be included to ini alize Firebase services.

3. Ini alizing Firebase: Before any Firebase func onality can be used, it is essen al to ini alize Firebase in the Flu er app. This is done by calling Firebase.ini alizeApp() in the main entry point of the app (usually in the main.dart file). Firebase needs to be ini alized before interac ng with any Firebase services, such as the Real me Database, Cloud Firestore, or Authen ca on.

Code:

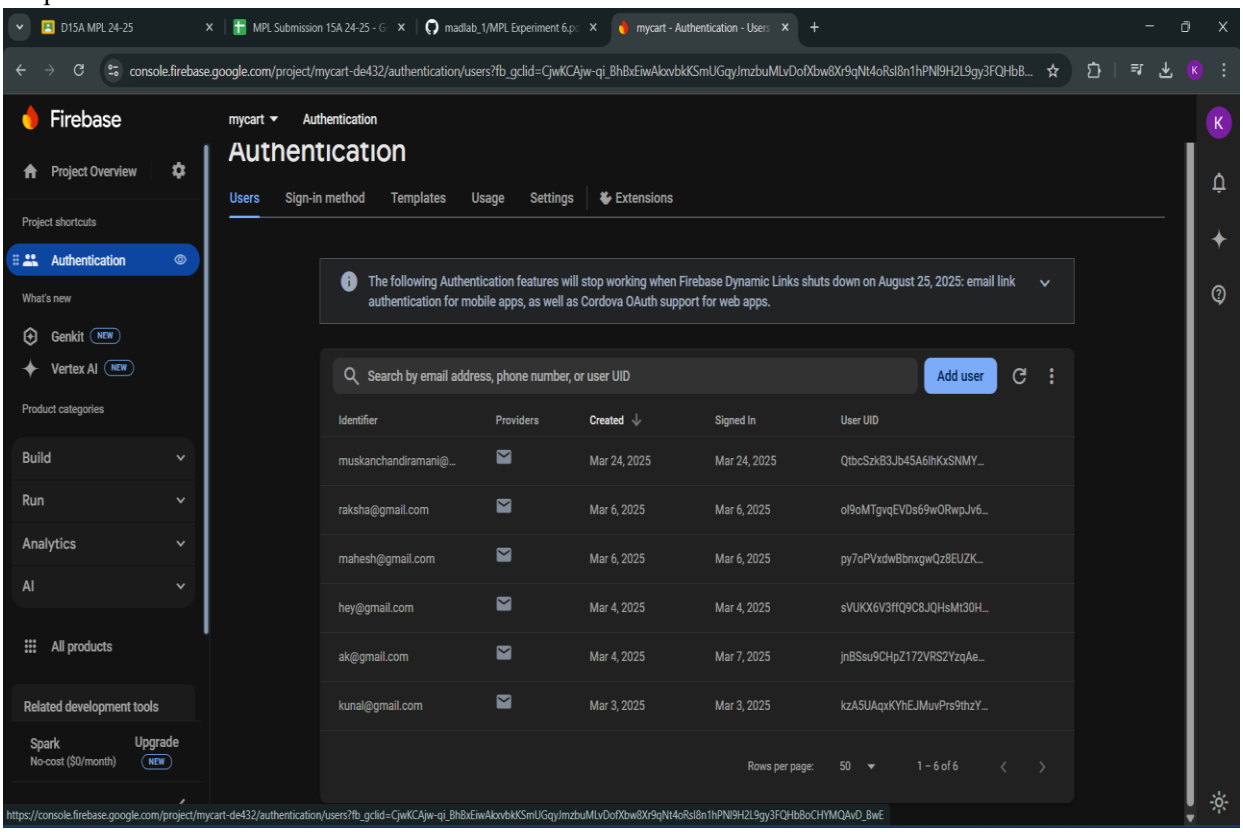
```
Signup.dart import 'package:flu
er/material.dart'; import
'../widgets/custom_text_field.dart'; import
'../widgets/custom_bu on.dart'; import
'../widgets/gender_selec on.dart'; import
'../widgets/profile_avatar.dart';

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

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

class _SignupPageState extends State<SignupPage> {  final TextEdi
ngController _emailController = TextEdi ngController();  final TextEdi
ngController _passwordController = TextEdi ngController();  final
TextEdi ngController _firstNameController = TextEdi ngController();
final TextEdi ngController _lastNameController = TextEdi
ngController();  final TextEdi ngController _birthdayController =
TextEdi ngController();
 String gender = "Male";

 void _signup() {
  // Handle signup logic here
  Navigator.pushNamed(context, '/home');
 }

 @override
 Widget build(BuildContext context) {
  return Scaffold(  appBar: AppBar(
leading: IconBu on(  icon: const
Icon(Icons.arrow_back),
    onPressed: () {
     Navigator.pop(context);
    },
   ),
  ),
  body: Padding(
```

```
      padding: const
EdgeInsets.all(20.0),        child:
Column(          children: [
          ProfileAvatar(onTap: () {
print("Profile Avatar Clicked!");            }),
          const     SizedBox(height:
20),          Row(            children:
[                              Expanded(
child: CustomTextField(
                    controller: _firstNameController, label: "First name")),
              const SizedBox(width:
10),               Expanded(
child: CustomTextField(
                    controller: _lastNameController, label: "Last name")),
            ],
          ),
          const SizedBox(height: 10),
          CustomTextField(controller: _birthdayController, label: "Birthday"),
const SizedBox(height: 10),
          GenderSelec on(onGenderSelected: (selectedGender) {
setState(() {
              gender = selectedGender;
            });
          }),
          CustomTextField(controller: _emailController, label: "Email"),
          const     SizedBox(height:     10),
CustomTextField(                  controller:
_passwordController,
              label:         "Password",
isPassword: true),               const
SizedBox(height:                  20),
GestureDetector(               onTap:
_signup,
            child: CustomBu on(
text: "Next",
              onPressed: () {},
            ),
          ),
        ],
      ),
    ),
  );
 }
}

Login.dart  import  'package:flu  er/material.dart';
import            'signup_page.dart';              import
'../widgets/custom_text_field.dart';              import
```

```
'../widgets/custom_bu           on.dart';           import
'home_page.dart'; // Import the HomePage

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

  @override
  Widget build(BuildContext context) {     final TextEdi ngController
_emailController = TextEdi ngController();     final TextEdi ngController
_passwordController = TextEdi ngController();

    void _login() {
      // Handle login logic
here       Navigator.push(
context,
       MaterialPageRoute(builder: (context) => const HomePage()),
      );
    }

    return Scaffold(
body: Padding(
       padding: const EdgeInsets.all(20.0),
child: Column(
         mainAxisAlignment: MainAxisAlignment.center,
children: [
           const Icon(Icons.facebook, size: 80, color: Colors.blue),
           const  SizedBox(height:  20),
CustomTextField(            controller:
_emailController,
             label: "Email",
           ),
           const     SizedBox(height:     10),
CustomTextField(                  controller:
_passwordController,
             label: "Password",
isPassword: true,
           ),
           const      SizedBox(height:
20),                  GestureDetector(
onTap: _login,                     child:
CustomBu on(                       text:
"Login",                onPressed: () {
                 Navigator.pushNamed(context, '/home');
               },
             ),
           ),
           TextBu on(
onPressed: () {},
             child: const Text("Forgot Password?"),
```

```
          ),
          const     SizedBox(height:
20),                GestureDetector(
onTap:                ()              {
Navigator.push(             context,
              MaterialPageRoute(builder: (context) => const SignupPage()),
            );
           },
           child: CustomBu on(
text: "Create new account",
             onPressed: () {
               Navigator.pushNamed(context, '/signup');
             },
           ),
          ),
          const SizedBox(height: 10),
          const Text("Meta", style: TextStyle(color: Colors.grey)),
        ],
      ),
    ),
  );
 }
}
```

Output



Project Title: **Roll No.**

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.	43
Name	Kunal Punjabi
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	

PWA Experiment -7

Devansh Wadhwani - 64
Kunal Punjabi - 43
Manav Punjabi - 44
Hitesh Rohra - 46

Class : D15A

❖ **AIM**

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

❖ **Theory**:-

Regular Web Application

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

Progressive Web Application (PWA)

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

❖ Key Differences Between PWAs and Regular Web Apps

1. Native-Like Experience

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

2. Ease of Access

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

3. Enhanced Performance

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

4. Improved User Engagement

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

5. Real-Time Updates

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

6. Search Engine Optimization (SEO) Benefits

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

7. Cost-Effective Development

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

❖ Advantages and Limitations of PWAs

➢ **Advantages:**

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

➢ **Limitations:**

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

CODE:

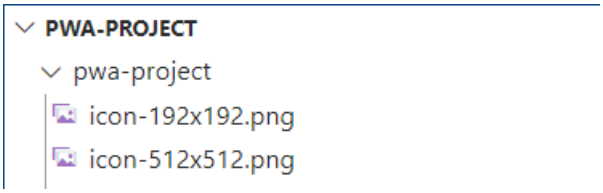
Index.html :

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

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

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

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

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

    <title>Delicia Food App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

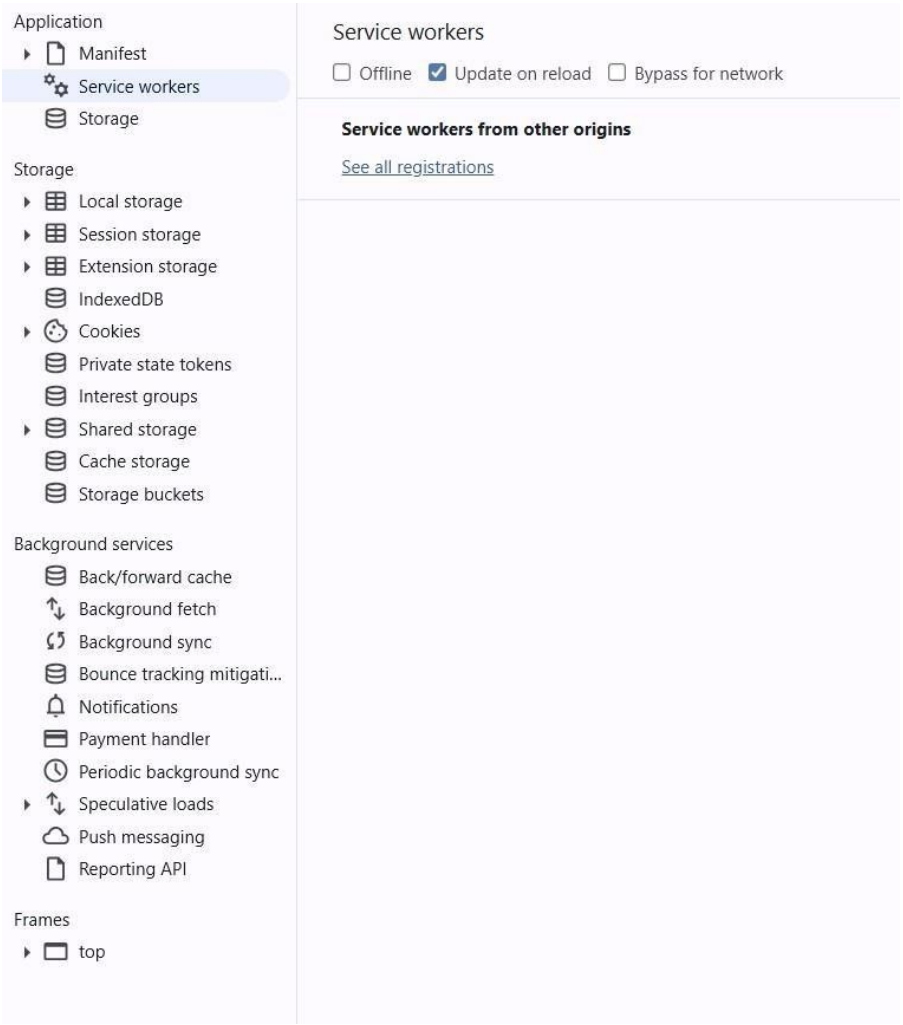
Manifest.json

```
  "name": "Delice Food App",
  "short_name": "Delice",
  "start_url": "./index.html",
  "scope": "./",
  "icons": [
    {
      "src": "icon-192x192.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "icon-512x512.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ],
  "theme_color": "#ffd31d",
  "background_color": "#333",
  "display": "standalone"
}
```

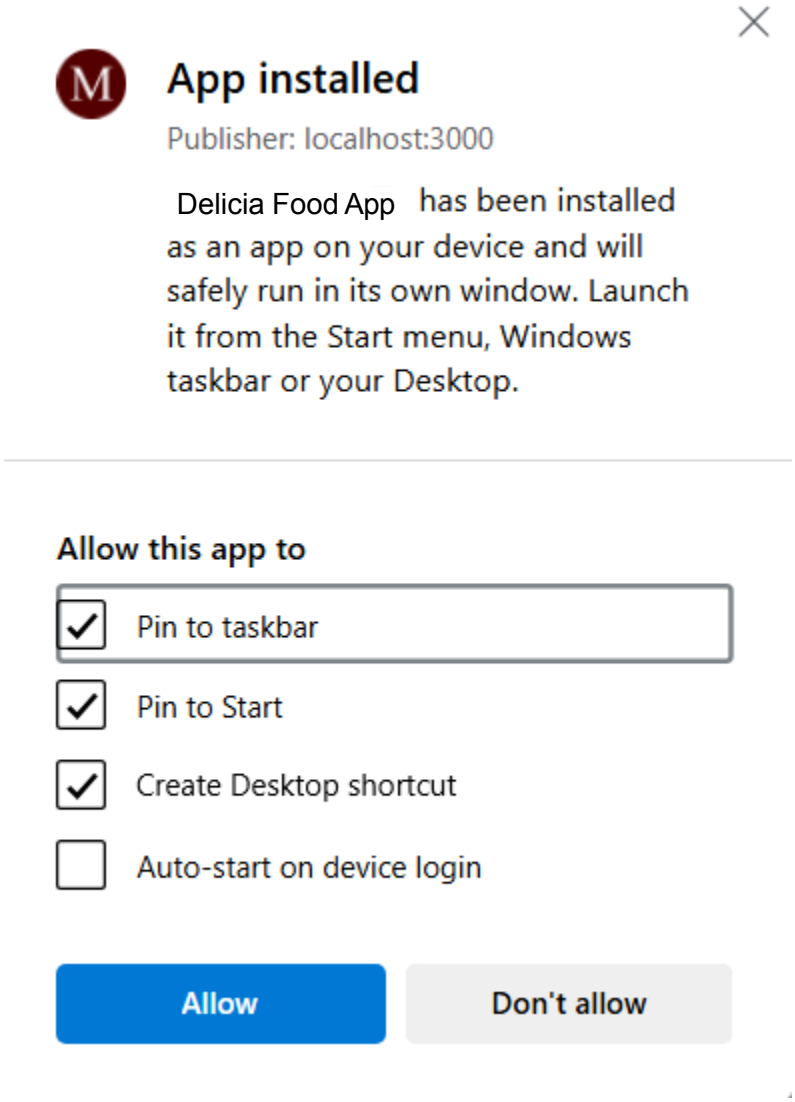
Icons

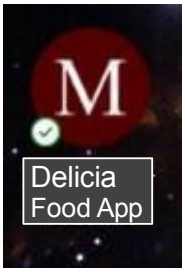


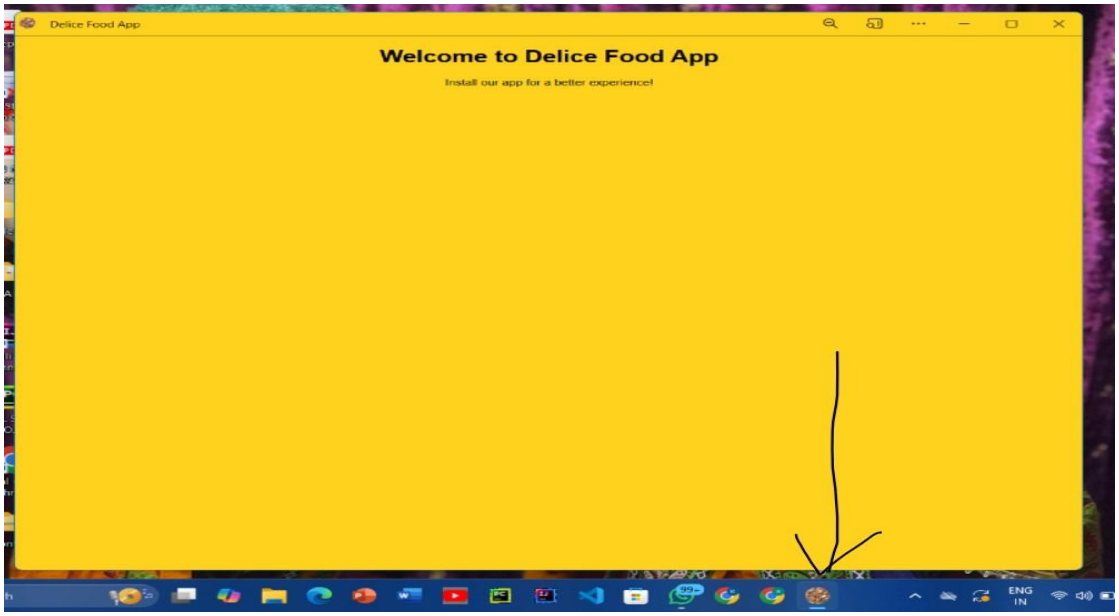
Google Dev

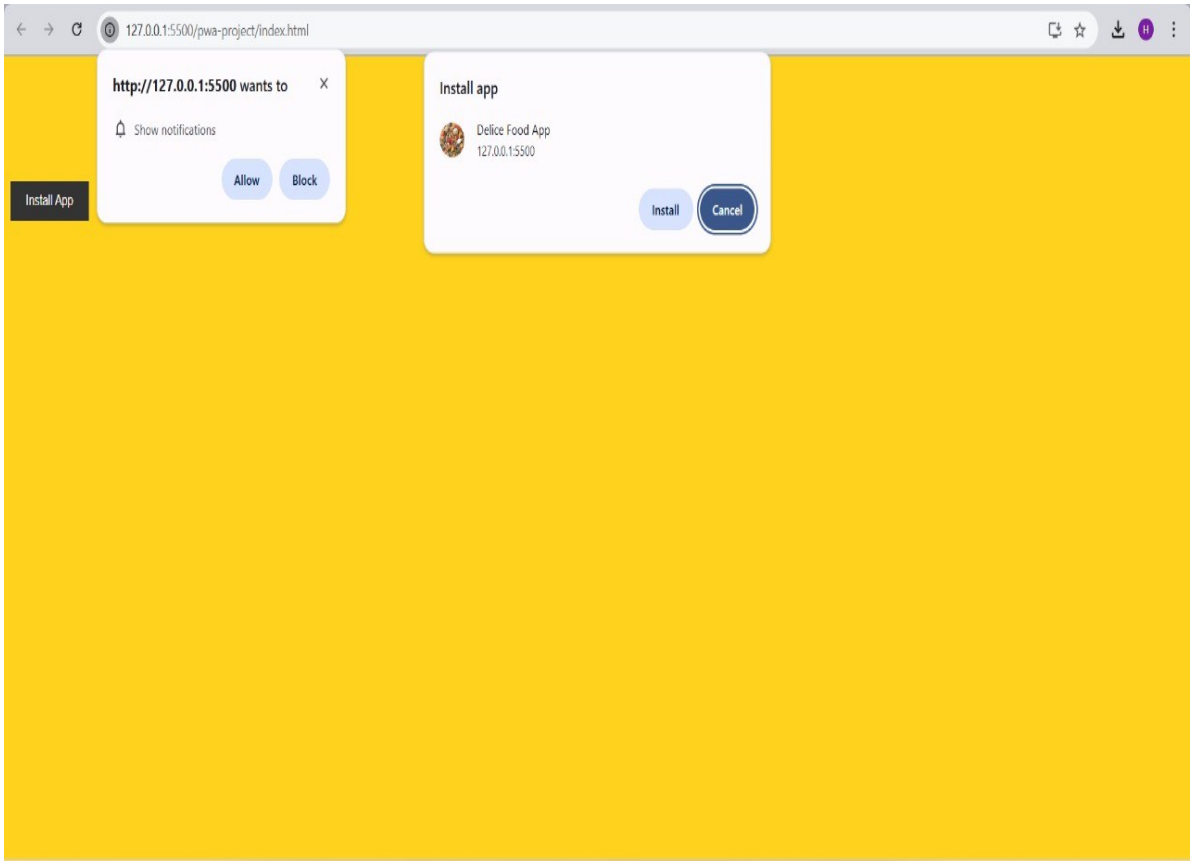


❖ **Output:-**


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


Delicia
Food App


Delice Food App
Welcome to Delice Food App
Install our app for a better experience!


127.0.0.1:5500/pwa-project/index.html
http://127.0.0.1:5500 wants to
Show notifications
Allow
Block
Install App
Install app
Delice Food App
127.0.0.1:5500
Install
Cancel

Project Title: **Roll No.**

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.	43
Name	Kunal Punjabi
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

MPL Experiment 8 (PWA)

Name: Devansh Wadhwani - 64
Kunal Punjabi - 43
Manav Punjabi - 44
Hitesh Rohra - 46 **Class:** D15A

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

Theory:

Service Worker

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

Things to note about Service Worker:

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

What can we do with Service Workers?

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

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

- You can manage Push Notifications
  You can manage push notifications with Service Worker and show any information message to the user.

- You can Continue
  Although Internet connection is broken, you can start any process with Background Sync of Service Worker.

What can’t we do with Service Workers?

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

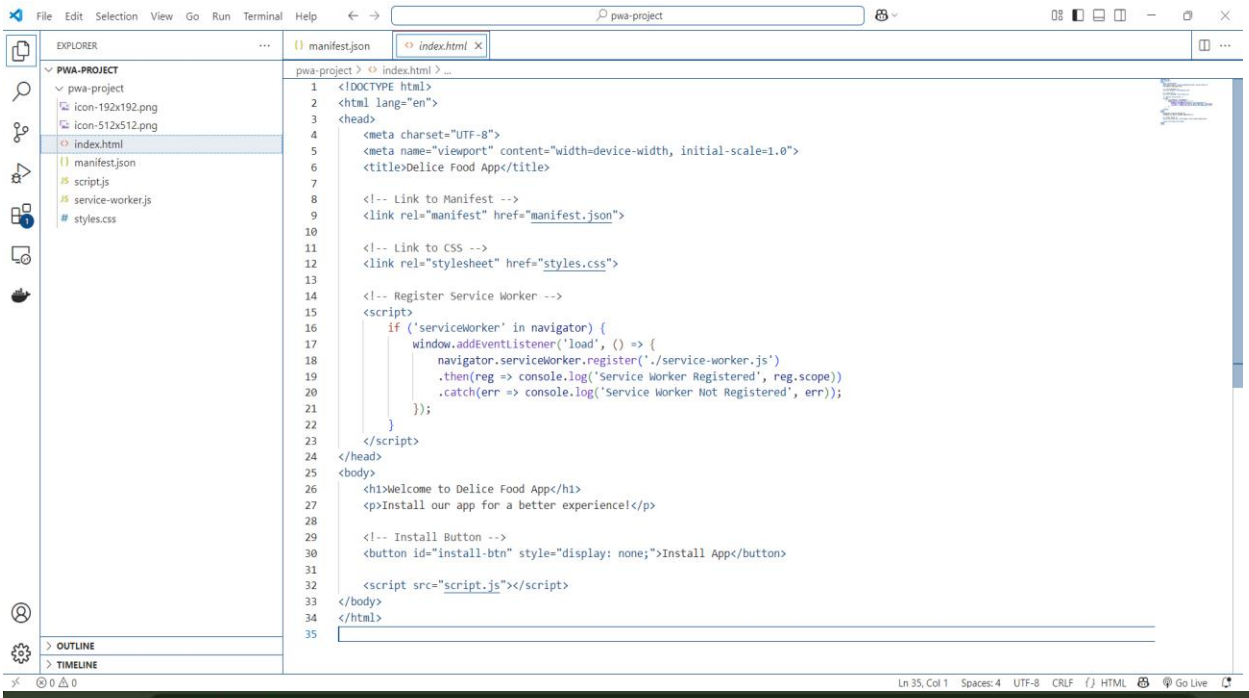
- You can’t work it on 80 Port
  Service Worker just can work on HTTPS protocol. But you can work on localhost during development.

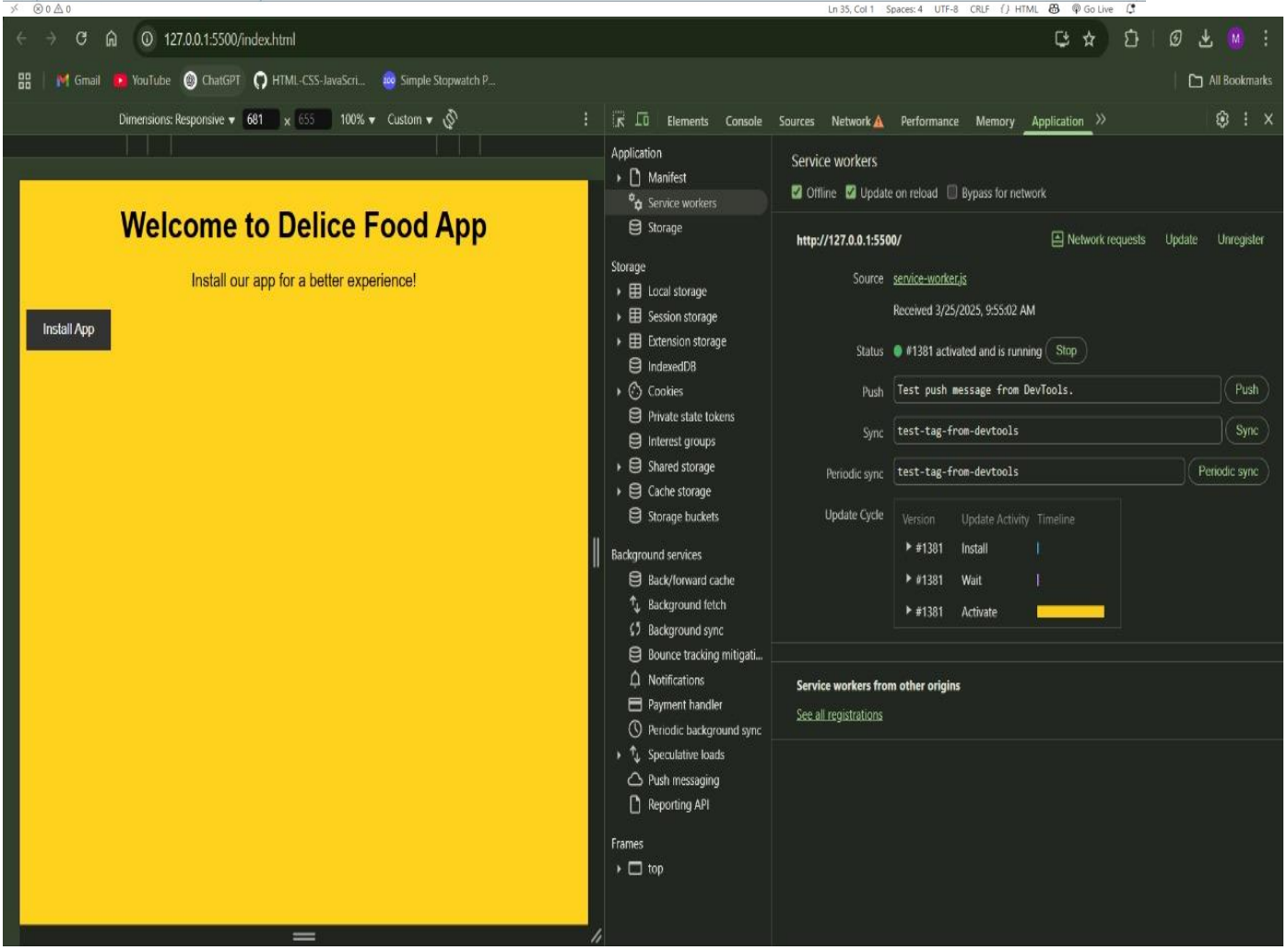
Codes:

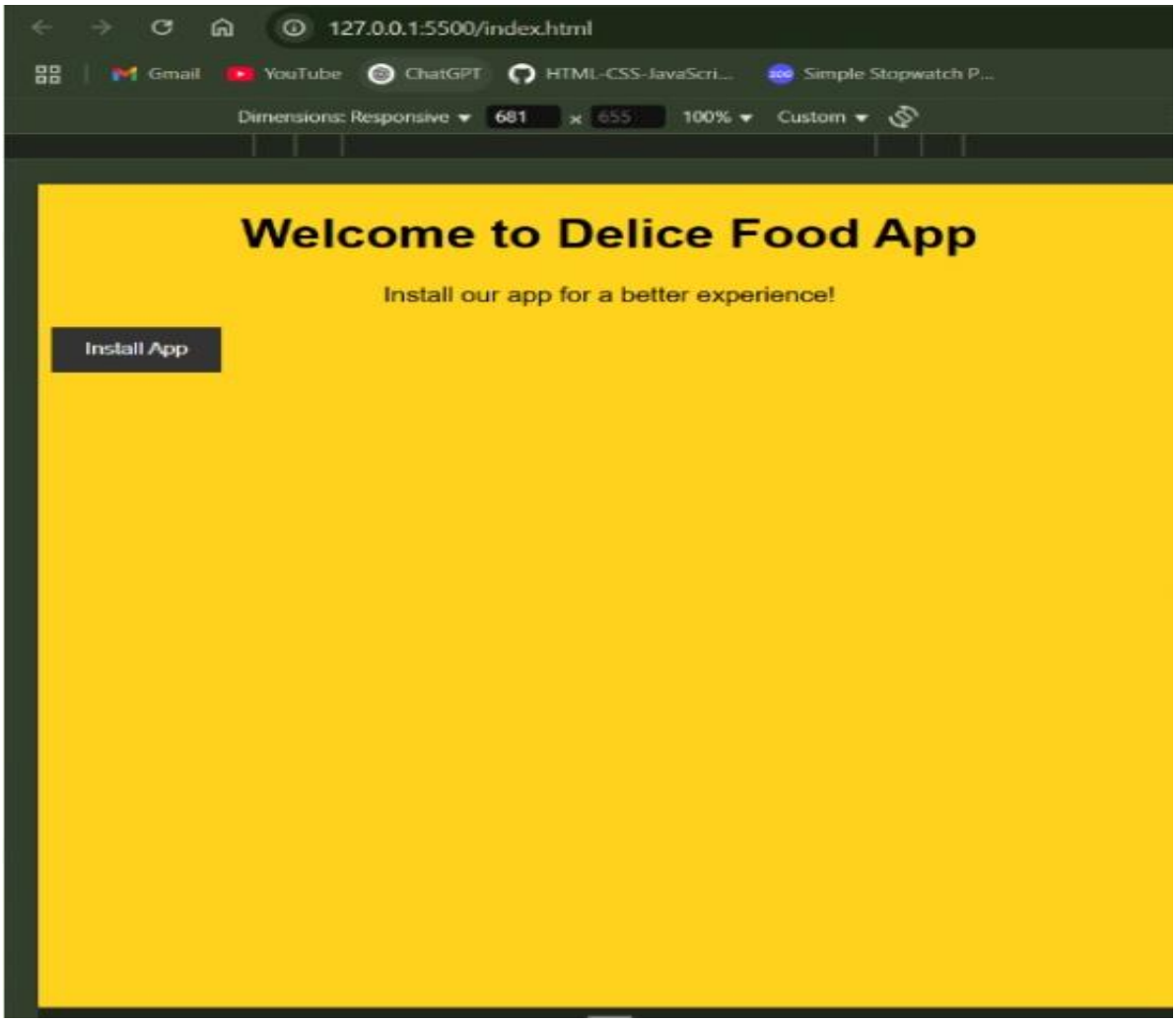
```
//Serviceworker.js


const CACHE_NAME = 'delice-cache-v1';
const urlsToCache = [
    './',
    './index.html',
    './styles.css',
    './script.js',
    './icon-192x192.png',
    './icon-512x512.png'
];

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

// Activate Service Worker (Old cache cleanup)
self.addEventListener('activate', event => {
    event.waitUntil(
        caches.keys().then(cacheNames => {
            return Promise.all(
                cacheNames.filter(cache => cache !==
CACHE_NAME).map(cache => caches.delete(cache))
            );
        })
    );
});

// Fetch Requests with Network Fallback
self.addEventListener('fetch', event => {
    event.respondWith(
        caches.match(event.request).then(response => {
            return response || fetch(event.request)
                .then(networkResponse => {
                    return caches.open(CACHE_NAME).then(cache => {
                        cache.put(event.request,
networkResponse.clone());
```

```
                        return networkResponse;
                    });
                });
        }).catch(() => {
            return caches.match('./index.html'); // Offline fallback
        })
    );
});
```

Output:


File Edit Selection View Go Run Terminal Help
pwa-project
EXPLORER
PWA-PROJECT
pwa-project
icon-192x192.png
icon-512x512.png
index.html
manifest.json
script.js
service-worker.js
styles.css
manifest.json
index.html
pwa-project > index.html > ...
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>Delice Food App</title>
7
8 <!-- Link to Manifest -->
9 <link rel="manifest" href="manifest.json">
10
11 <!-- Link to CSS -->
12 <link rel="stylesheet" href="styles.css">
13
14 <!-- Register Service Worker -->
15 <script>
16 if ('serviceWorker' in navigator) {
17 window.addEventListener('load', () => {
18 navigator.serviceWorker.register('./service-worker.js')
19 .then(reg => console.log('Service Worker Registered', reg.scope))
20 .catch(err => console.log('Service Worker Not Registered', err));
21 });
22 }
23 </script>
24 </head>
25 <body>
26 <h1>Welcome to Delice Food App</h1>
27 Install our app for a better experience!
28
29 <!-- Install Button -->
30 <button id="install-btn" style="display: none;">Install App</button>
31
32 <script src="script.js"></script>
33 </body>
34 </html>
35
OUTLINE
TIMELINE
Ln 35, Col 1 Spaces: 4 UTF-8 CRLF HTML Go Live


127.0.0.1:5500/index.html
Gmail YouTube ChatGPT HTML-CSS-JavaScri... Simple Stopwatch P...
All Bookmarks
Dimensions: Responsive 681 x 655 100% Custom
Welcome to Delice Food App
Install our app for a better experience!
Install App
Elements Console Sources Network Performance Memory Application
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Storage buckets
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking mitigati...
Notifications
Payment handler
Periodic background sync
Speculative loads
Push messaging
Reporting API
Frames
top
Service workers
Offline Update on reload Bypass for network
http://127.0.0.1:5500/
Network requests Update Unregister
Source service-worker.js
Received 3/25/2025, 9:55:02 AM
Status #1381 activated and is running Stop
Push Test push message from DevTools. Push
Sync test-tag-from-devtools Sync
Periodic sync test-tag-from-devtools Periodic sync
Update Cycle
Version Update Activity Timeline
#1381 Install
#1381 Wait
#1381 Activate
Service workers from other origins
See all registrations


127.0.0.1:5500/index.html
Gmail
YouTube
ChatGPT
HTML-CSS-JavaScri...
Simple Stopwatch P...
Dimensions: Responsive
681
655
100%
Custom
Welcome to Delice Food App
Install our app for a better experience!
Install App

Project Title: **Roll No.**

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.	43
Name	Kunal Punjabi
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

MPL Experiment 9 (PWA)

Name: Devansh Wadhwani - 64

Kunal Punjabi - 43

Manav Punjabi – 44

Hitesh Rohra - 46 **Class:** D15A

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

Theory:

Service Worker

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

Things to note about Service Worker:

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

Fetch Event

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

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

- **CacheFirst** - In this function, if the received request has cached before, the cached response is returned to the page. But if not, a new response requested from the network.

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

Sync Event

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

Push Event

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

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

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

Code:

Serviceworker.js

```
const CACHE_NAME = 'delice-cache-v1';
const urlsToCache = [
    './',
    './index.html',
    './styles.css',
    './script.js',
    './icon-192x192.png',
    './icon-512x512.png'
];

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

// Activate Service Worker (Old cache cleanup)
self.addEventListener('activate', event => {
    event.waitUntil(
        caches.keys().then(cacheNames => {
            return Promise.all(
                cacheNames.filter(cache => cache !== CACHE_NAME).map(cache =>
caches.delete(cache))
            );
        })
    );
});

// Fetch Requests with Network Fallback
self.addEventListener('fetch', event => {
    event.respondWith(
        caches.match(event.request).then(response => {
            return response || fetch(event.request)
                .then(networkResponse => {
                    return caches.open(CACHE_NAME).then(cache => {
                        cache.put(event.request, networkResponse.clone());
                        return networkResponse;
                    });
                });
        }).catch(() => {
            return caches.match('./index.html'); // Offline fallback
        })
    );
```

});

Push Notification Code:

1. Push Notification code:

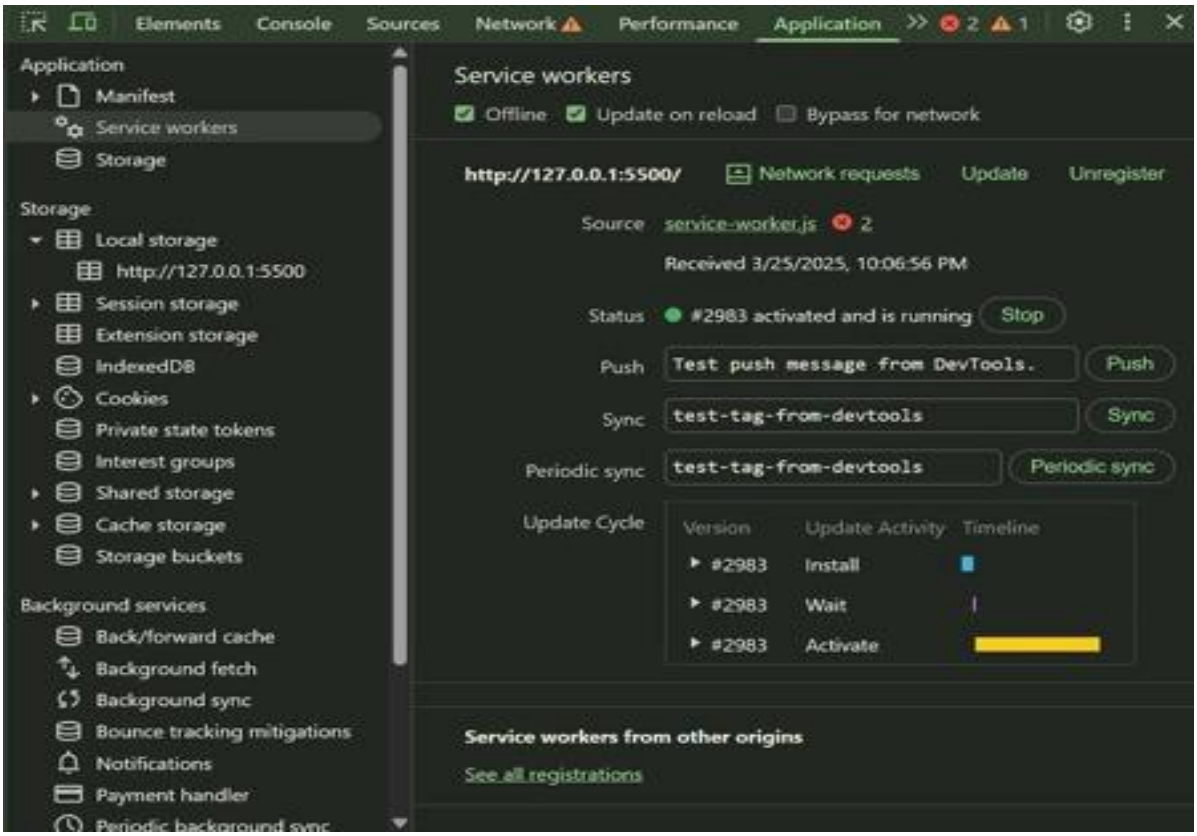
```
// Push Notification Handling
self.addEventListener("push", (event) => {
    console.log("Push notification received:", event);

    let data = { title: "New Notification", body: "Hello, this is a default message!" };

    if (event.data) {
        try {
            data = event.data.json();
        } catch (error) {
            console.warn("Push message is not JSON, using text instead.");
            data.body = event.data.text();
        }
    }

    const options = {
        body: data.body || "You have a new update!",
        icon: "./assets/images/logo_tourly_192.png",
        badge: "./assets/images/logo_tourly_192.png",
        vibrate: [200, 100, 200],
    };

    event.waitUntil(
        self.registration.showNotification(data.title || "Hello Sannidhi", options)
    );
});

// Click Event for Push Notifications
self.addEventListener("notificationclick", (event) => {
    event.notification.close();
    event.waitUntil(
        clients.openWindow("http://localhost:5500/")
    );
});
```

Fetch Code :

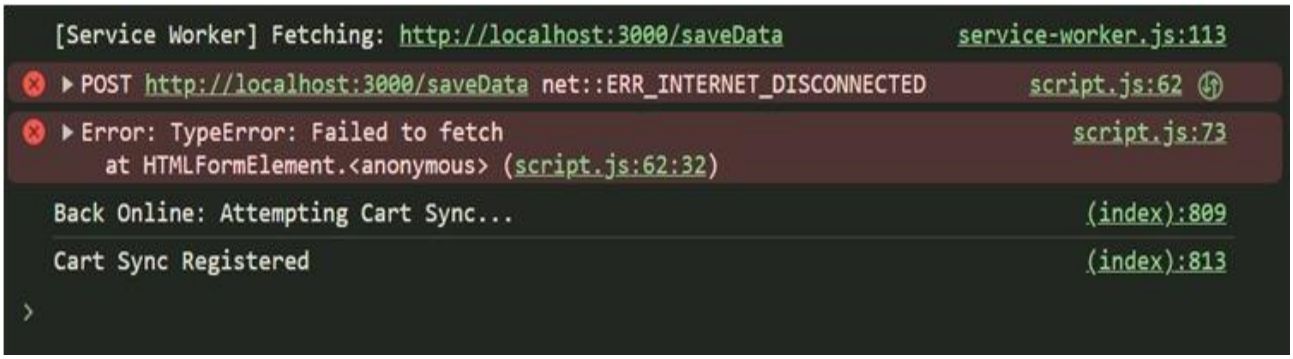
```
self.addEventListener("fetch", (event) => {
   console.log("[Service Worker] Fetching:", event.request.url);

   if (event.request.method !== "GET") return; // Skip non-GET requests

   event.respondWith(
      caches.match(event.request).then((response) => {
         return response || fetch(event.request).then((networkResponse) => {
            return networkResponse;
         });
      }).catch(() => console.warn("[Service Worker] Network request failed:",
event.request.url))
   );
});

// Sync Event - Send Stored Data When Online
self.addEventListener("sync", (event) => {
   if (event.tag === SYNC_TAG) {
      console.log("[Service Worker] Sync event triggered:", SYNC_TAG);
      event.waitUntil(syncOfflineData());
   }
});

// Function to Sync Offline Data & Save to JSON
async function syncOfflineData() {
   const storedData = await getFromLocalStorage();
   if (!storedData.length) {
      console.log("[Service Worker] No offline data to sync.");
      return;
   }

   console.log("[Service Worker] Attempting to sync offline data...", storedData);

   try {
      const response = await fetch("http://localhost:3000/saveData", {
         method: "POST",
         headers: { "Content-Type": "application/json" },
         body: JSON.stringify(storedData)
      });

      if (response.ok) {
         console.log("[Service Worker] Successfully synced data:", storedData);
         await saveDataToJson(storedData); // Save data to JSON file
         clearLocalStorage(); // Clear after successful sync
      } else {
         console.warn("[Service Worker] Sync failed. Server response not OK.");
      }
   } catch (error) {
      console.error("[Service Worker] Sync failed:", error);
   }
}
// Function to Retrieve Data from Local Storage
function getFromLocalStorage() {
```

```
    return new Promise((resolve) => {
        const data = localStorage.getItem("offlineData");
        resolve(data ? JSON.parse(data) : []);
    });
}

// Function to Clear Local Storage after Sync
function clearLocalStorage() {
    localStorage.removeItem("offlineData");
    console.log("[Service Worker] Local storage cleared after sync.");
}

// Save Synced Data to a JSON File
async function saveDataToJson(data) {
    try {
        const response = await fetch(JSON_FILE_URL, {
            method: "PUT", // Use PUT or POST based on server support
            headers: { "Content-Type": "application/json" },
            body: JSON.stringify(data)
        });

        if (response.ok) {
            console.log("[Service Worker] Data saved to JSON file:", JSON_FILE_URL);
        } else {
            console.error("[Service Worker] Failed to save data to JSON.");
        }
    } catch (error) {
        console.error("[Service Worker] Error saving to JSON:", error);
    }
}
```

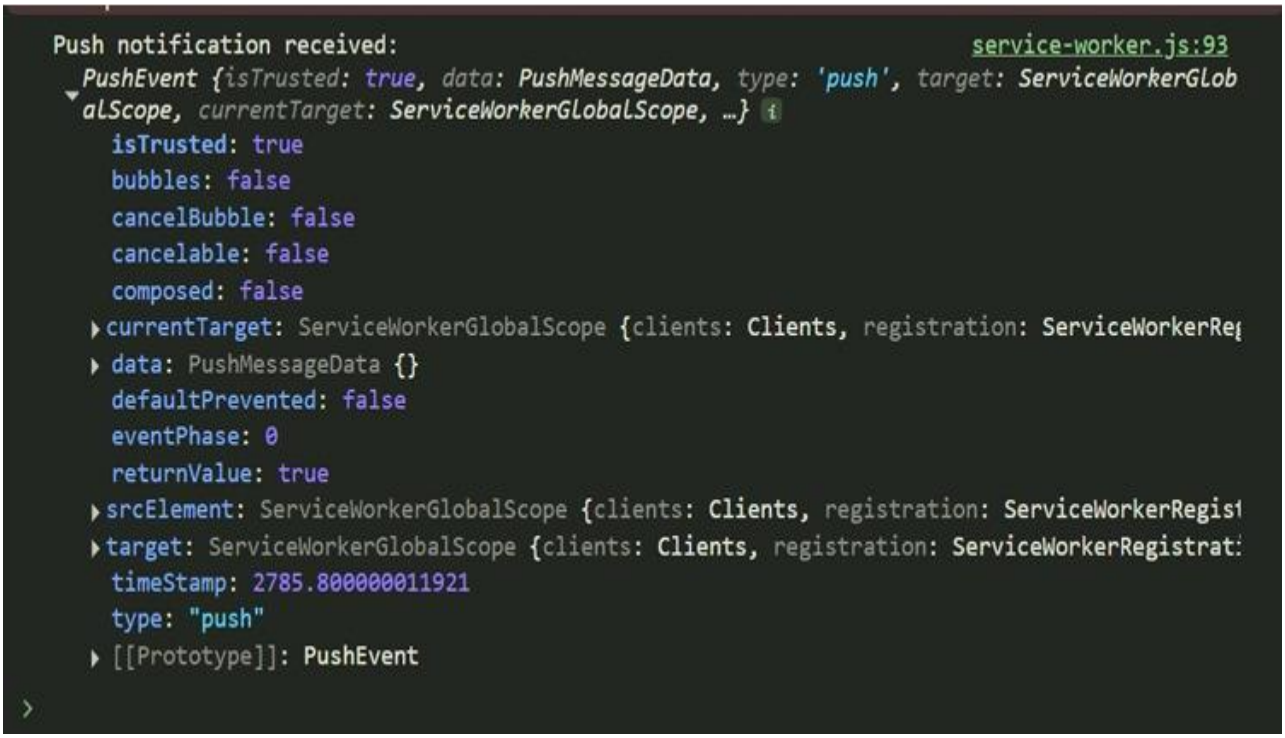
Output:

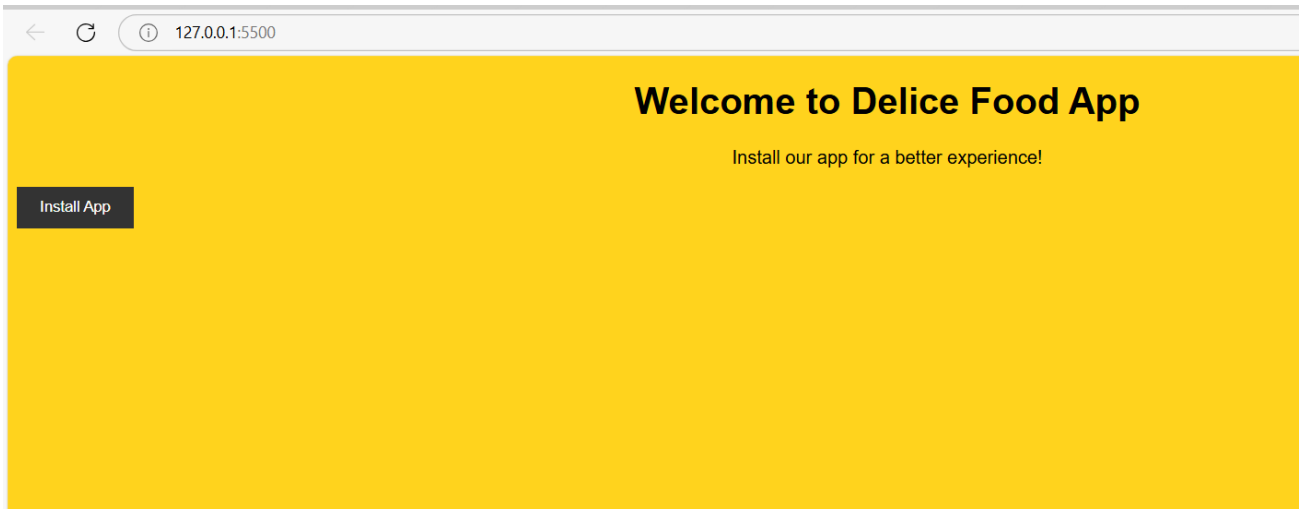

Elements
Console
Sources
Network
Performance
Application
Application
Manifest
Service workers
Storage
Storage
Local storage
http://127.0.0.1:5500
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Storage buckets
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking mitigations
Notifications
Payment handler
Service workers
Offline
Update on reload
Bypass for network
http://127.0.0.1:5500/
Network requests
Update
Unregister
Source
service-worker.js
2
Received 3/25/2025, 10:06:56 PM
Status
#2983 activated and is running
Stop
Push
Test push message from DevTools.
Push
Sync
test-tag-from-devtools
Sync
Periodic sync
test-tag-from-devtools
Periodic sync
Update Cycle
Version
Update Activity
Timeline
#2983
Install
#2983
Wait
#2983
Activate
Service workers from other origins
See all registrations

Sync Event



Push event




127.0.0.1:5500
Welcome to Delice Food App
Install our app for a better experience!
Install App

Project Title: **Roll No.**

MAD & PWA Lab
Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	43
Name	Kunal Punjabi
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

MPL Experiment 10 (PWA)

Class: D15A

Name:

Devansh Wadhwani – 64

Kunal Punjabi – 43

Manav Punjabi – 44

Hitesh Rohra - 46

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

Theory:

GitHub Pages

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

GitHub Pages provides the following key features:

- Blogging with Jekyll
- Custom URL
- Automatic Page Generator

Reasons for favoring this over Firebase:

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

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

Pros

- Very familiar interface if you are already using GitHub for your projects.
- Easy to set up. Just push your static website to the gh-pages branch and your website is ready.
- Supports Jekyll out of the box.

- Supports custom domains. Just add a file called CNAME to the root of your site, add an A record in the site's DNS configuration, and you are done.

Cons

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

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

Some of the features offered by Firebase are:

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

Reasons for favoring over GitHub Pages:

- Realtime backend made easy
- Fast and responsive

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

Pros

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

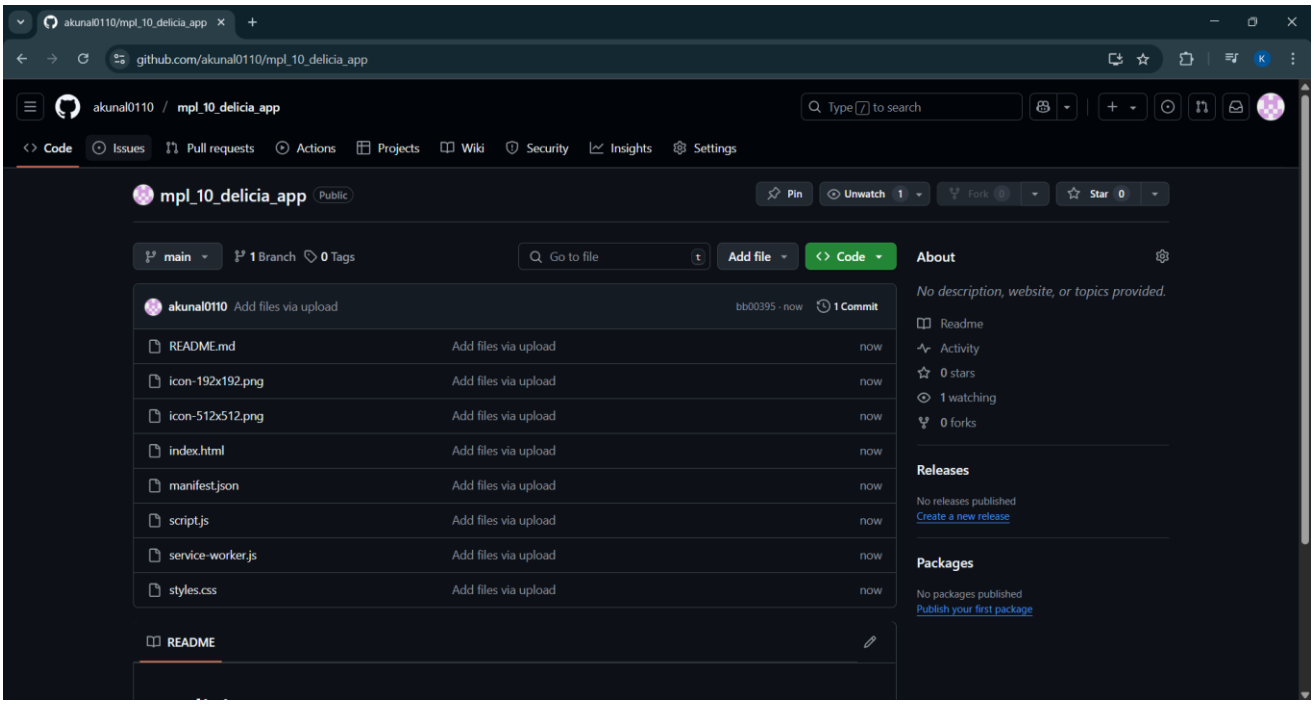
Cons

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

Link to our GitHub repository:
https://github.com/akunal0110/mpl_10_delicia_app.git

Link to our Hosted website:
https://devanshwadhwani2004/Delicia-App-PWA/

Github Screenshot:



Project Title: **Roll No.**

MAD & PWA Lab
Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	43
Name	Kunal Punjabi
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	

MPL Experiment 11 (PWA)

Name: Devansh Wadhwani – 64

Kunal Punjabi – 43

Manav Punjabi – 44

Hitesh Rohra - 46 **Class:** D15A

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

Theory:

Google Lighthouse :

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

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

Key Features and Audit Metrics

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

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

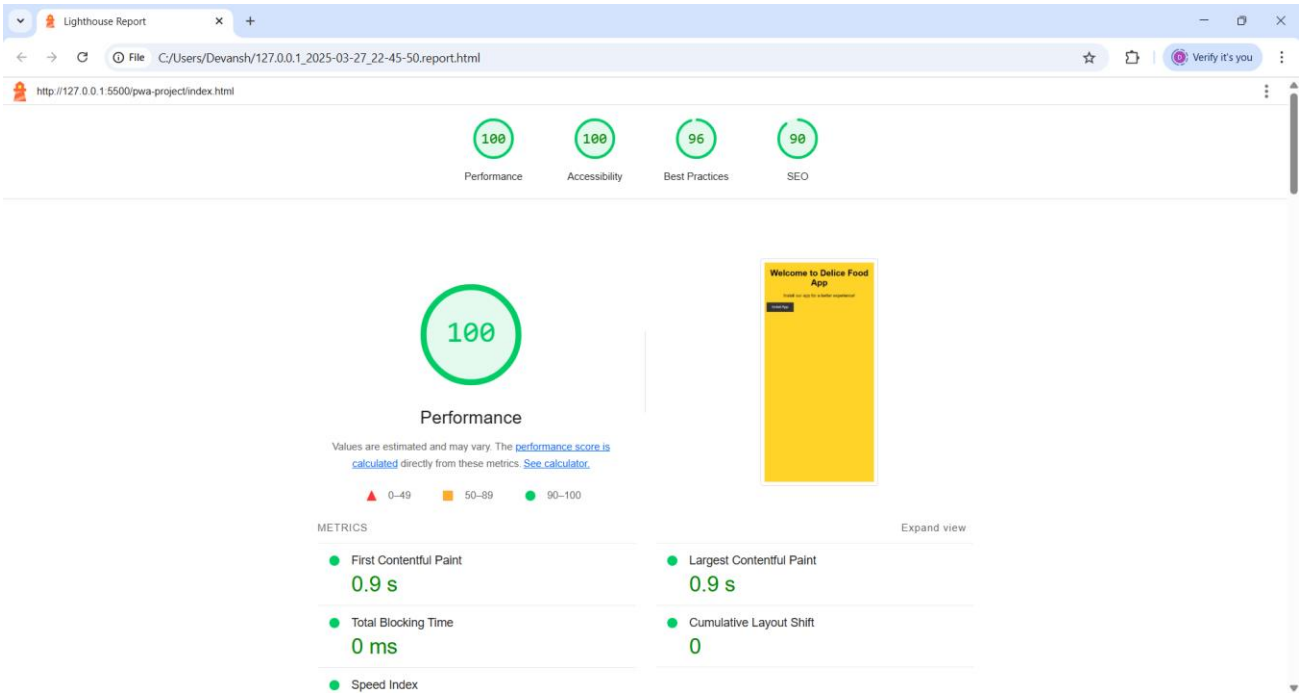
- **PWA Score (Mobile):**

Thanks to the rise of Service Workers, app manifests, etc., a lot of modern web applications are moving towards the PWA paradigm, where the objective is to make the application behave as close as possible to native mobile

applications. Scoring points are based on the Baseline PWA checklist laid down by Google which includes Service Worker implementation(s), viewport handling, offline functionality, performance in script-disabled environments, etc.

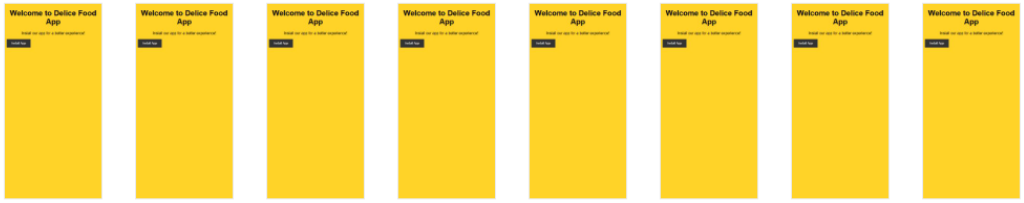
- **Accessibility:**
  As you might have guessed, this metric is a measure of how accessible your website is, across a plethora of accessibility features that can be implemented in your page (such as the 'aria-' attributes like aria-required, audio captions, button names, etc.). Unlike the other metrics though, Accessibility metrics score on a pass/fail basis i.e. if all possible elements of the page are not screen-reader friendly (HTML5 introduced features that would make pages easy to interpret for screen readers used by visually challenged people like tag names, tags such as <section>, <article>, etc.), you get a 0 on that score. The aggregate of these scores is your Accessibility metric score.
  Best Practices: As any developer would know, there are a number of practices that have been deemed 'best' based on empirical data. This metric is an aggregation of many such points, including but not limited to:Use of HTTPS

0utput:


Lighthouse Report
File C:/Users/Devansh/127.0.0.1_2025-03-27_22-45-50.report.html
Verify it's you
http://127.0.0.1:5500/pwa-project/index.html
100
Performance
100
Accessibility
96
Best Practices
90
SEO
100
Performance
Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.
0–49
50–89
90–100
Welcome to Delice Food App
METRICS
Expand view
First Contentful Paint
0.9 s
Largest Contentful Paint
0.9 s
Total Blocking Time
0 ms
Cumulative Layout Shift
0
Speed Index

0.9 s

View Treemap



Show audits relevant to: All FCP LCP

DIAGNOSTICS

- Eliminate render-blocking resources — Potential savings of 290 ms
- Page prevented back/forward cache restoration — 1 failure reason
- Avoid chaining critical requests — 2 chains found
- Largest Contentful Paint element — 910 ms

More information about the performance of your application. These numbers don't directly affect the Performance score.

Accessibility

These checks highlight opportunities to improve the accessibility of your web app. Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so manual testing is also encouraged.

ADDITIONAL ITEMS TO MANUALLY CHECK (10) Hide

- Interactive controls are keyboard focusable
- Interactive elements indicate their purpose and state
- The page has a logical tab order
- Visual order on the page follows DOM order
- User focus is not accidentally trapped in a region
- The user's focus is directed to new content added to the page

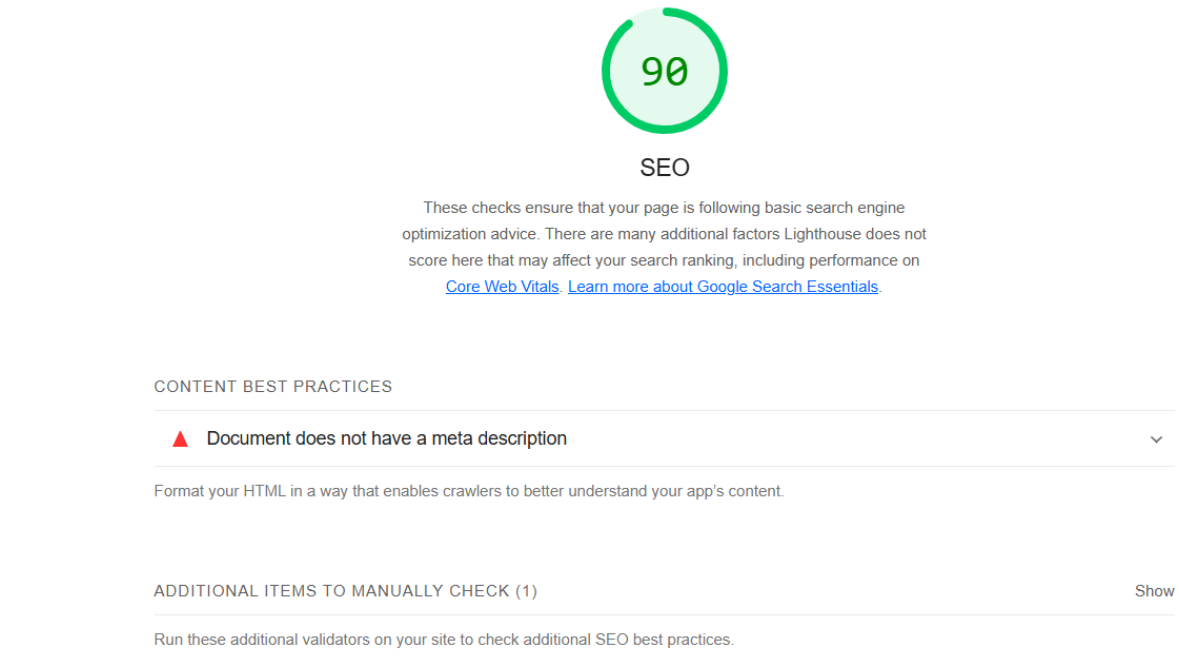
Best Practices

GENERAL

- Browser errors were logged to the console

TRUST AND SAFETY

- Ensure CSP is effective against XSS attacks
- Use a strong HSTS policy
- Ensure proper origin isolation with COOP
- Mitigate clickjacking with XFO or CSP



Conclusion:

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