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 **Ansh Ashish Sarfare** 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

Name of the Course : MAD & PWA Lab

Course Code : ITL604

Project Title: Tasty app **Roll No. 49**

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: Tasty app **Roll No. 49**

Lab Objectives:

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

Lab Outcomes:

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

Index

Sr. No	Experiment Title	LO	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			

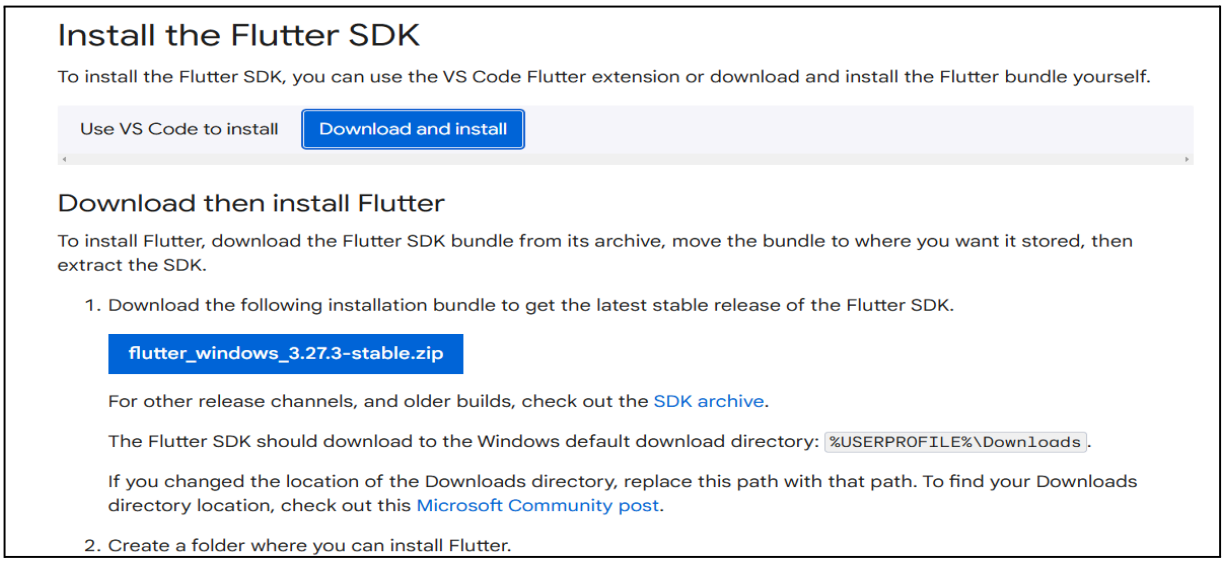
MAD & PWA Lab
Journal

Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	49
Name	Ansh Sarfare
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	

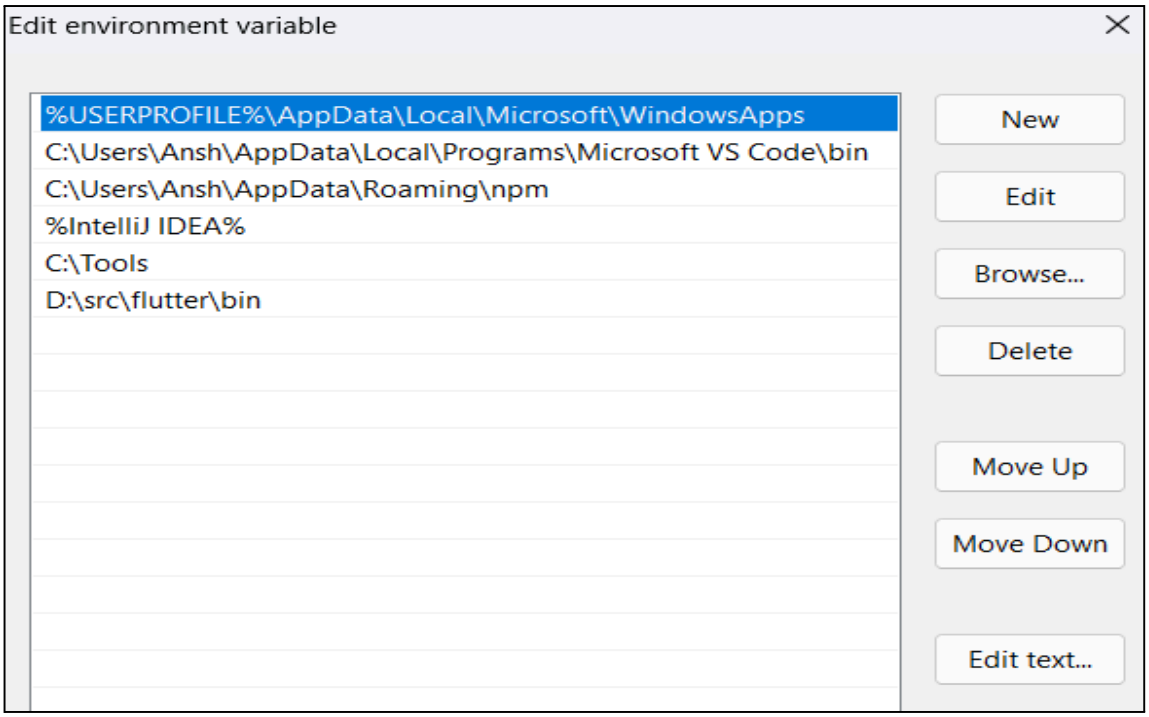
Project Title: Tasty app **Roll No. 49**

Aim: Installation and Configuration of Flutter Environment.

Step-1: Go to Flutter's official website and download the zip file to install flutter SDK



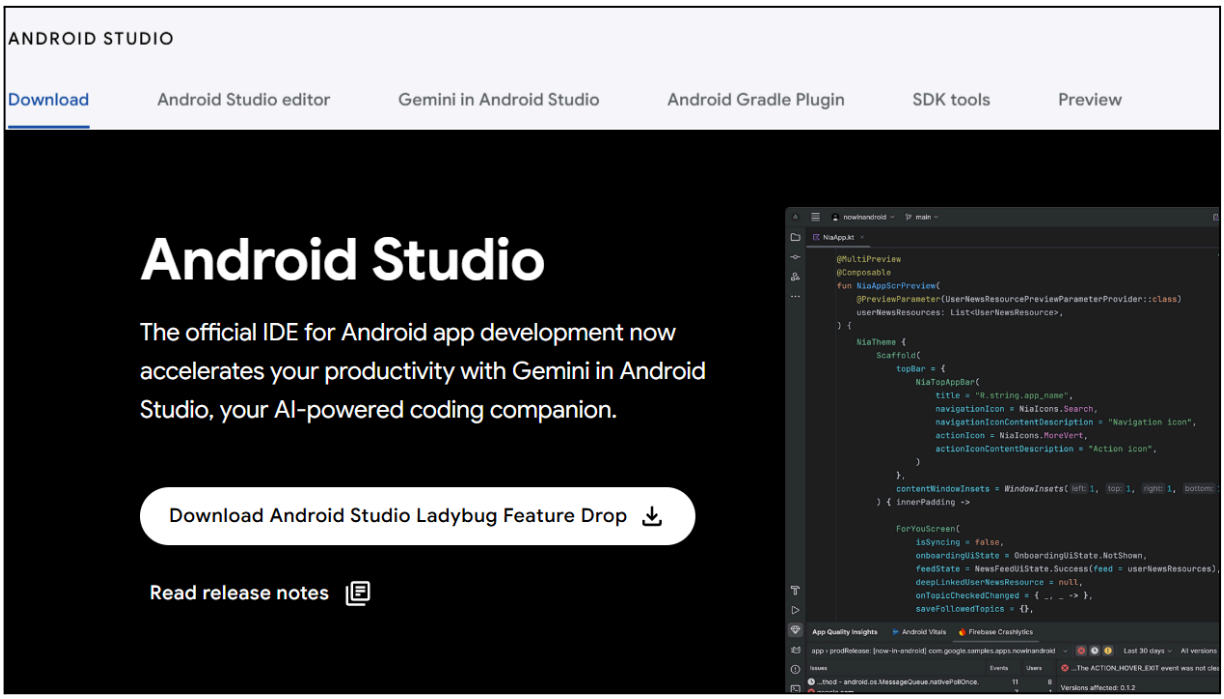
Step-2: Place the file in D drive and add its path in system environment variables

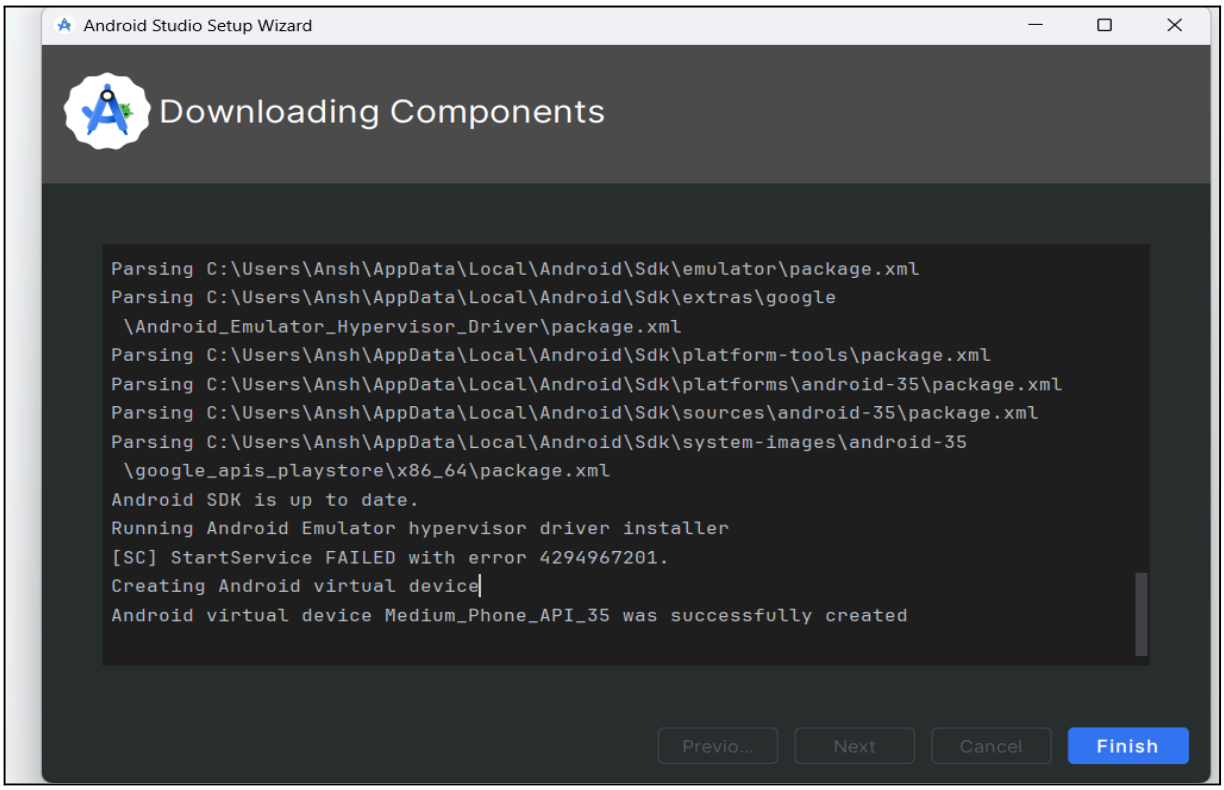


Step-3: Run ‘flutter doctor’ command to verify that flutter is installed successfully.

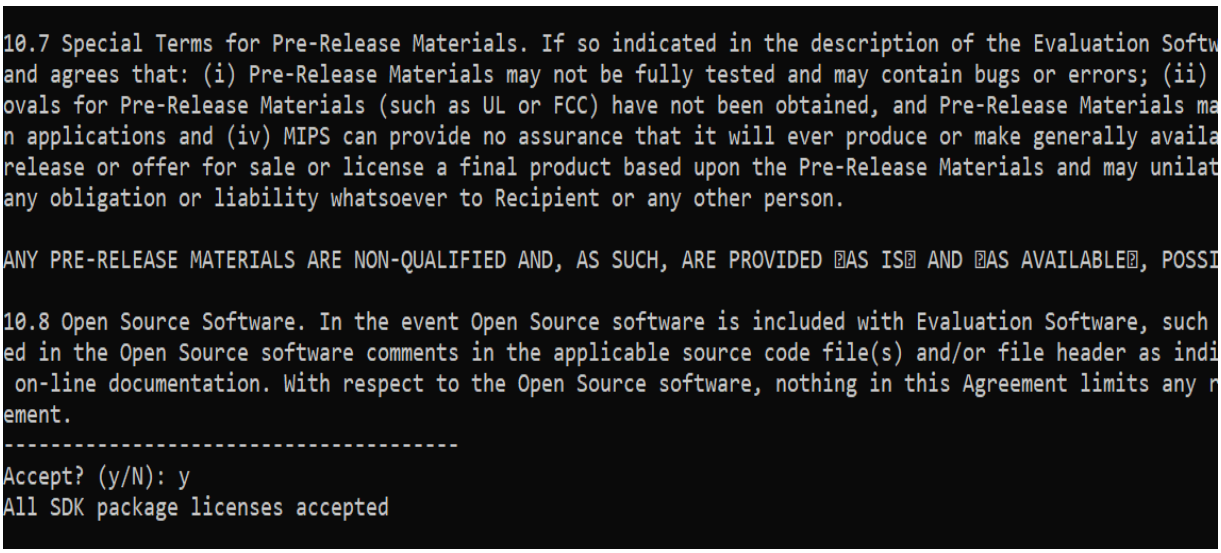
```
C:\Users\Ansh>flutter doctor

                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-4: Go to official website of Android studio and download the Ladybug Feature Drop for Windows





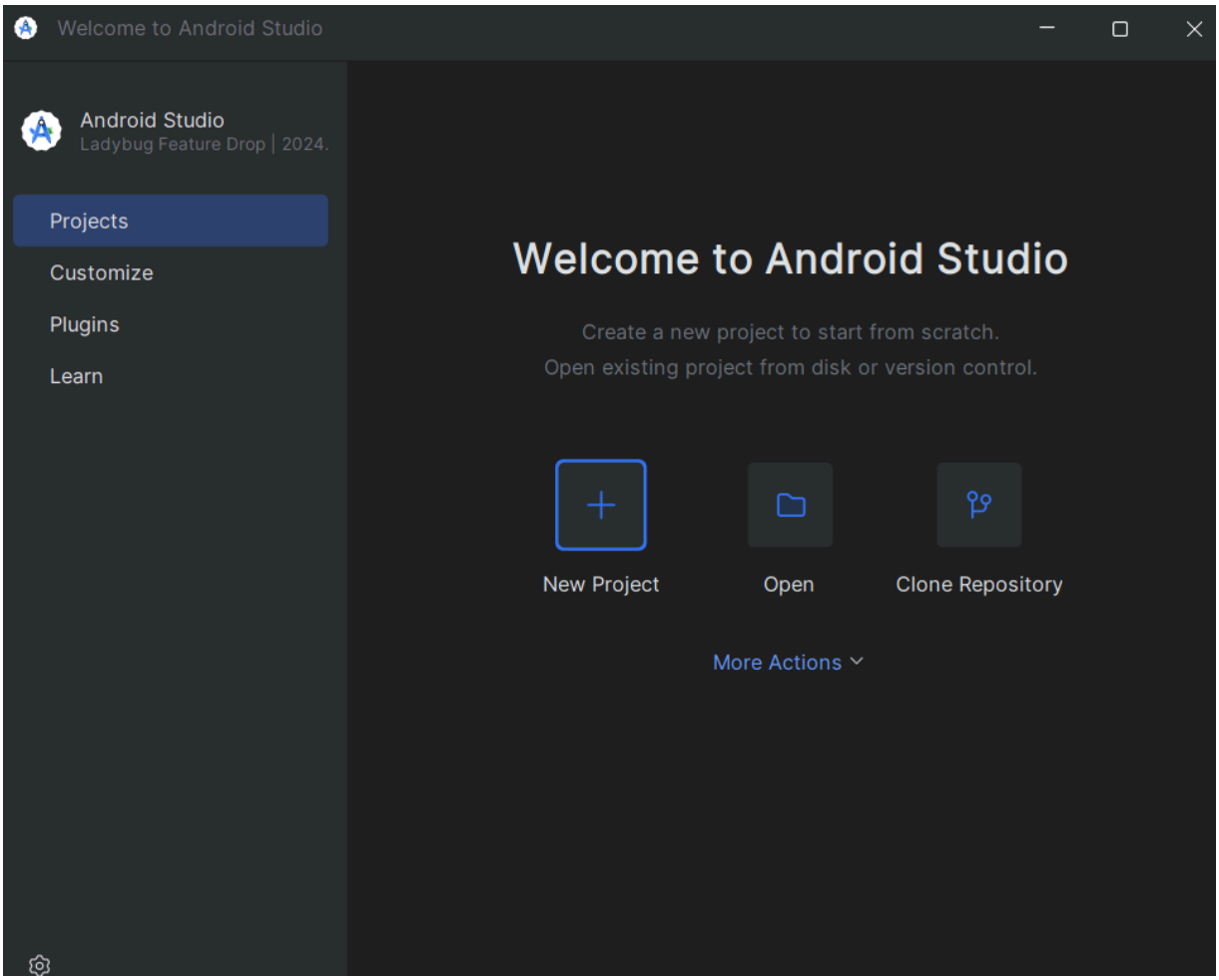
Step-5: After installation of Android studio run the command flutter doctor --android-licenses to accept required SDK licenses for Flutter to work with Android.



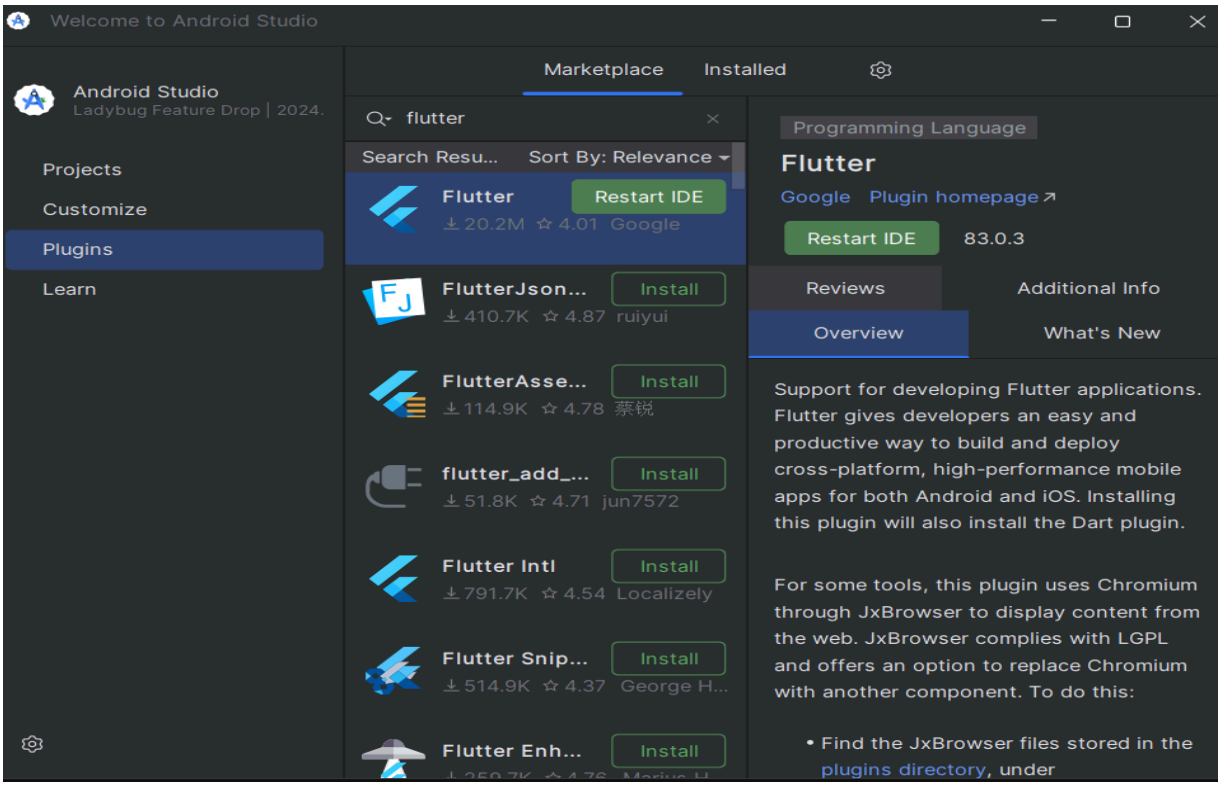
Step-7: Run flutter doctor again to ensure everything is installed properly.

```
C:\Users\Ansh>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.27.3, on Microsoft Windows [Version 10.0.22621.2715], 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)
[√] IntelliJ IDEA Ultimate Edition (version 2024.2)
[√] VS Code (version 1.96.4)
[√] Connected device (3 available)
[√] Network resources

! Doctor found issues in 1 category.
```

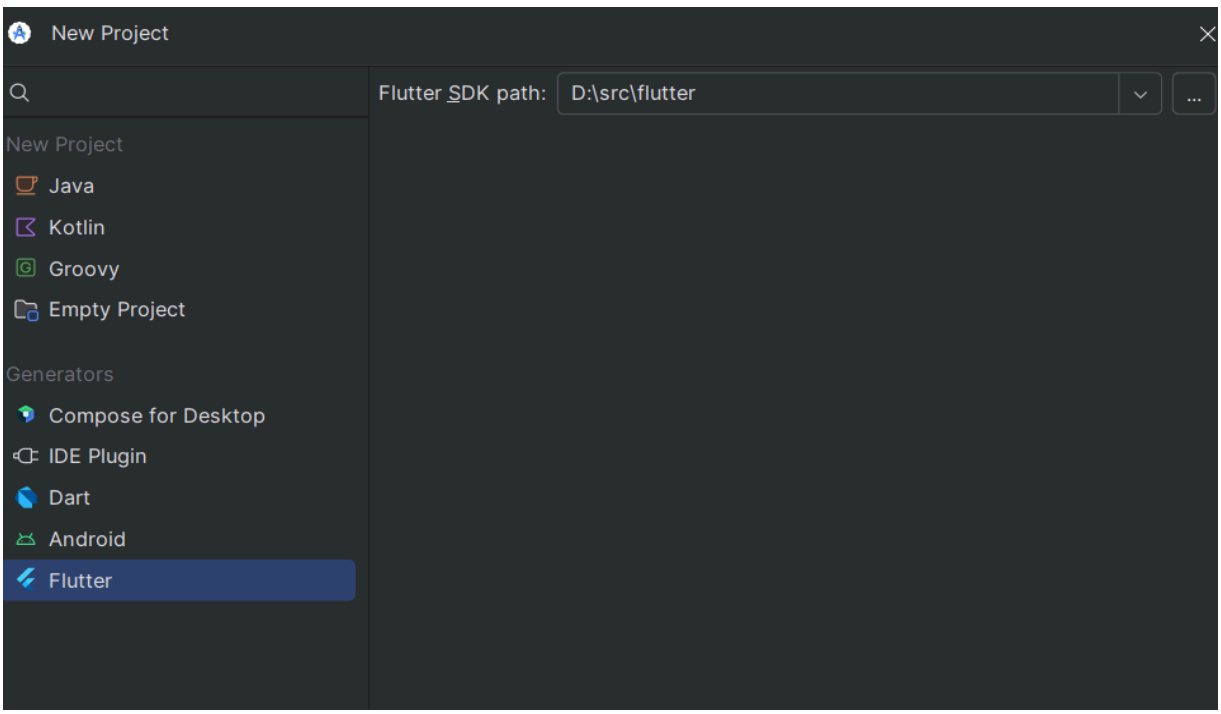
Android Studio:



Step-8: Add the Flutter Plugin

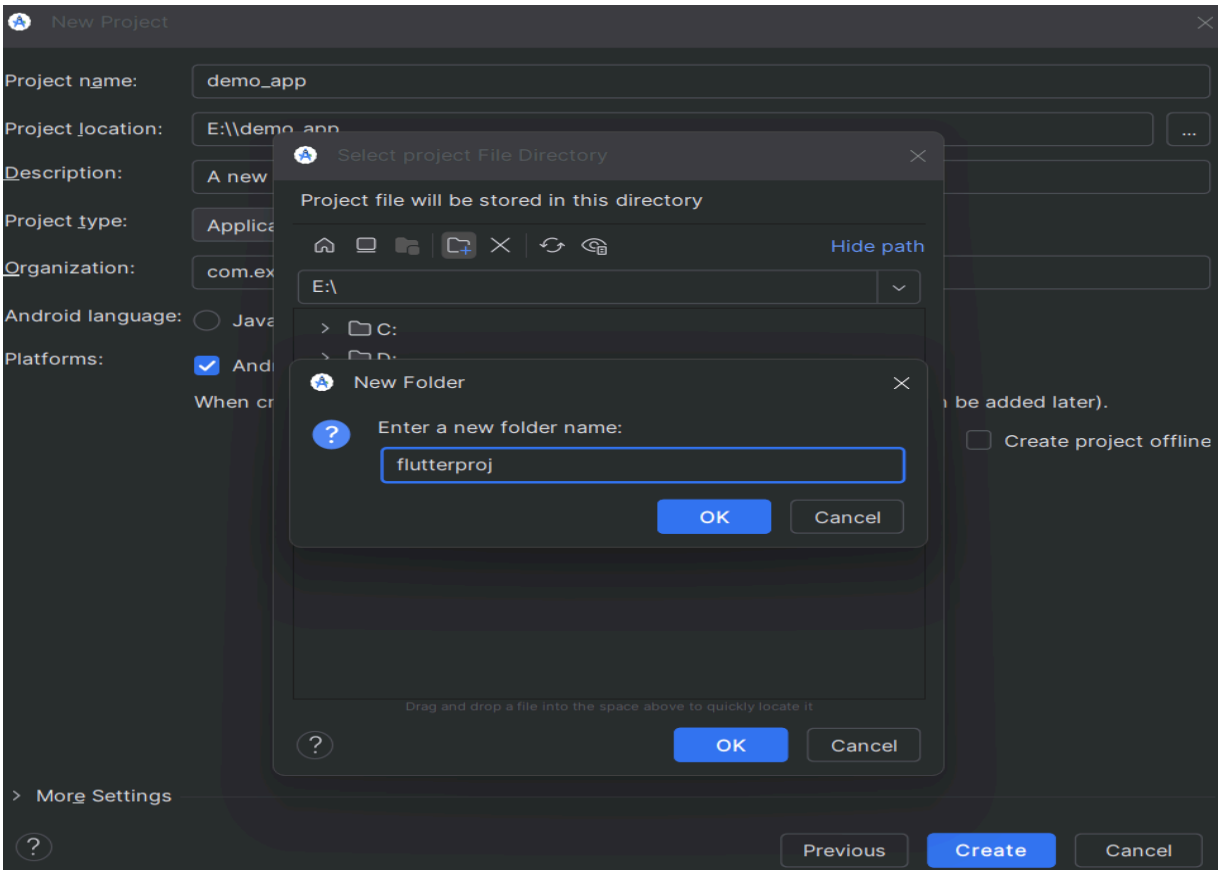


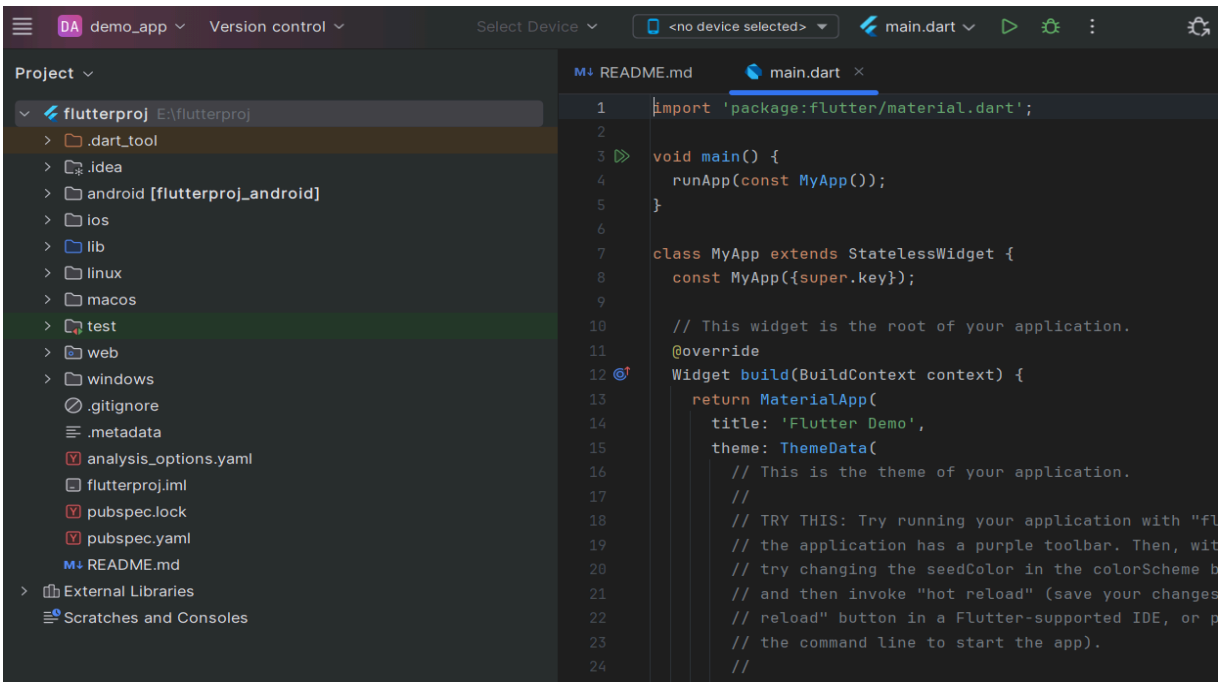
Step-9: Create a new Flutter Project



Project Title: Tasty app **Roll No. 49**

Step-10: Name the Project and set the project directory





Step-11: Edit the main.dart file with demo code for testing

main.dart × README.md

```dart
    ); // MaterialApp
  }
}

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

  final String title;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Theme.of(context).colorScheme.inversePrimary,
        title: Text(title),
      ), // AppBar
      body: const Center(
        child: Text(
          'Ansh Sarfare, I am 20yrs old',
          style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold),
        ), // Text
      ), // Center
    ); // Scaffold
  }
}

```

Step-11: Connect USB to your phone and enable USB Debugging in developer options , after that run ‘flutter devices’ command to check if device is listed

```
PS E:\flutterproj> flutter devices
Found 4 connected devices:
  AC2001 (mobile)    • 876c5c27 • android-arm64  • Android 12 (API 31)
  Windows (desktop) • windows  • windows-x64    • Microsoft Windows [Version 10.0.22621.2715]
  Chrome (web)      • chrome   • web-javascript • Google Chrome 132.0.6834.160
  Edge (web)        • edge     • web-javascript • Microsoft Edge 132.0.2957.127
```

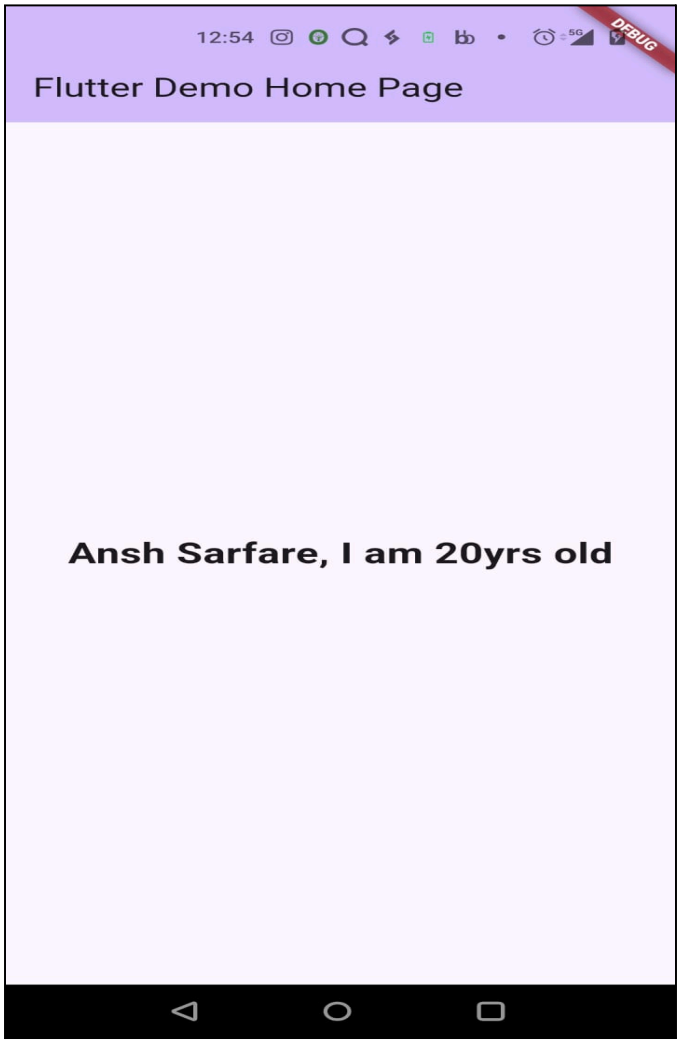
Step-12: Run ‘flutter run’ command to run the app on phone

```
PS E:\flutterproj> flutter run
Launching lib\main.dart on AC2001 in debug mode...
Warning: SDK processing. This version only understands SDK XML versions up to 3 but an SDK XML file of version 4
dio and the command-line tools that were released at different times.
Checking the license for package Android SDK Build-Tools 33.0.1 in D:\Android\Sdk\licenses
```

Project Title: Tasty app **Roll No. 49**

```
Terminal   Local
"Install Android SDK Build-Tools 33.0.1 v.33.0.1" finished.
warning: [options] source value 8 is obsolete and will be removed in a future release
warning: [options] target value 8 is obsolete and will be removed in a future release
warning: [options] To suppress warnings about obsolete options, use -Xlint:-options.
3 warnings
Running Gradle task 'assembleDebug'...                          139.5s
√ Built build\app\outputs\flutter-apk\app-debug.apk
Installing build\app\outputs\flutter-apk\app-debug.apk...         3.9s
I/flutter (25485): [INFO:flutter/shell/platform/android/android_context_vk_impeller.cc(64)] Kno
Syncing files to device AC2001...                                227ms

Flutter run key commands.
r Hot reload.
R Hot restart.
h List all available interactive commands.
d Detach (terminate "flutter run" but leave application running).
```

Output:



MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	49
Name	Ansh Sarfare
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Project Title: Tasty app **Roll No. 49**

Aim: To Design Flutter UI by including common widgets

Theory:

Common Widgets used in Flutter app:

1. **Scaffold:** Provides the basic visual structure for a screen, including app bars, bodies, and bottom navigation.
2. **AppBar:** A widget that is displayed at the top of the screen.
3. **Text:** Displays an immutable piece of text.
4. **TextField:** Allows the user to enter text.
5. **Image.network:** Displays an image from a URL.
6. **SizedBox:** A box with a specified size.
7. **ListView.builder:** Creates a scrollable list of widgets that are built on demand.
8. **Card:** A panel with slightly rounded corners and a shadow.
9. **InkWell:** A rectangular area of a UI that responds to touch.
10. **Column:** Arranges its children in a vertical array.
11. **Row:** Arranges its children in a horizontal array.
12. **Padding:** Inserts space around another widget.
13. **Align:** Aligns its child within itself.
14. **Container:** A widget that combines common painting, positioning, and sizing widgets.
15. **CircularProgressIndicator:** A widget that indicates that a task is in progress.
16. **BottomNavigationBar:** A bar at the bottom of the screen for selecting different destinations.
17. **BottomNavigationBarItem:** An item in a bottom navigation bar.
18. **SingleChildScrollView:** A box in which a single widget can be scrolled.
19. **AnimatedOpacity:** Animates the opacity of a widget.
20. **Center:** Centers its child within itself.
21. **MaterialPageRoute:** A route that replaces the entire screen with a platform-adaptive transition.
22. **Navigator.push:** Method to push a route onto the navigator's stack.
23. **Icon:** A graphical icon widget.
24. **InputDecoration:** Styles the visual appearance of a TextField.
25. **OutlineInputBorder:** A border for TextField with a rectangular outlinE.

Code:

main.dart:

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

```
import 'dart:convert';
import 'package:http/http.dart' as http;
import 'widgets/recipe_detail_screen.dart';
import 'models/recipe.dart';
import 'dart:async';
import 'login_screen.dart';

void main() {
 runApp(MyApp());
}
class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   title: 'Tasty',
   theme: ThemeData(
    scaffoldBackgroundColor: Colors.grey[900],
    appBarTheme: AppBarTheme(
     backgroundColor: Colors.grey[850],
     titleTextStyle: TextStyle(
      color: Colors.white,
      fontSize: 20,
      fontWeight: FontWeight.bold,
     ),
         textTheme: TextTheme(
     bodyMedium: TextStyle(color: Colors.white),
     titleLarge: TextStyle(
      color: Colors.white,
      fontWeight: FontWeight.bold,
     ),




    elevatedButtonTheme:
ElevatedButtonThemeData(
     style: ElevatedButton.styleFrom(
      backgroundColor: Colors.amber,
      foregroundColor: Colors.black,
     ),
    ),
   ),
   home: MyHomePage(),
  );
 }
}
class MyHomePage extends StatefulWidget {
 @override
 _MyHomePageState createState() =>
_MyHomePageState();
}
class _MyHomePageState extends
State<MyHomePage> {
 List<Recipe> _recipes = [];
 String _searchTerm = '';
 Recipe? _randomRecipe;
 Timer? _timer;
 double _opacity = 1.0;
 Map<String, List<Recipe>> _areaRecipes = {};
 List<String> areas = [
  'Indian',
  'Canadian',
  'Italian',
  'Chinese',
  'Mexican',
  'Thai'
 ];
 bool _isLoadingAreaRecipes = true;
 int _selectedIndex = 0;
 @override
 void initState() {
  super.initState();
  fetchRandomRecipe();
  fetchAreaRecipes();
```

```
    _timer = Timer.periodic(Duration(seconds: 5),
(Timer timer) {
      setState(() {
        _opacity = 0.0;
      });
      Future.delayed(Duration(milliseconds: 500), () {
        fetchRandomRecipe();
        setState(() {
          _opacity = 1.0;
        });
      });
  @override
  void dispose() {
    _timer?.cancel();
    super.dispose();
  }
  Future<void> fetchRandomRecipe() async {
    final response = await
http.get(Uri.parse('https://www.themealdb.com/api/js
on/v1/1/random.php'));
    if (response.statusCode == 200) {
      final data = json.decode(response.body);
      if (data['meals'] != null) {
        setState(() {
          _randomRecipe =
Recipe.fromJson(data['meals'][0]);
        });
      }
    } else {
      print('Failed to load random recipe');
    }
  }
  Future<void> fetchAreaRecipes() async {
    setState(() {
      _isLoadingAreaRecipes = true;
    });
    Map<String, List<Recipe>> tempAreaRecipes =
{};
    for (String area in areas) {
      final response = await
http.get(Uri.parse('https://www.themealdb.com/api/js
on/v1/1/filter.php?a=$area'));
      if (response.statusCode == 200) {
        final data = json.decode(response.body);
        if (data['meals'] != null) {
          List<Recipe> recipes = (data['meals'] as
List).map((json) => Recipe.fromJson(json)).toList();
          tempAreaRecipes[area] = recipes;
        }
      } else {
        print('Failed to load $area recipes');
      }
    }
    setState(() {
      _areaRecipes = tempAreaRecipes;
      _isLoadingAreaRecipes = false;
    });
  }
  Future<void> fetchRecipes() async {
    if (_searchTerm.isEmpty) {
      setState(() {
        _recipes = [];
      });
      return;
    }

    final response = await http.get(Uri.parse(

'https://www.themealdb.com/api/json/v1/1/search.php
?s=$_searchTerm'));
    if (response.statusCode == 200) {
      final data = json.decode(response.body);
      if (data['meals'] != null) {
```

```
      setState(() {
        _recipes = (data['meals'] as List)
            .map((json) => Recipe.fromJson(json))
            .toList();
      });
    } else {
      setState(() {
        _recipes = [];
      });
    }
  } else {
    throw Exception('Failed to load recipes');
  }
 }
 List<Recipe> get filteredRecipes {
  return _recipes.where((recipe) =>

recipe.strMeal.toLowerCase().contains(_searchTerm.t
oLowerCase()) ||
      recipe.ingredients.any((ingredient) =>

ingredient.name.toLowerCase().contains(_searchTer
m.toLowerCase()))).toList();
 }
 Future<Recipe?> fetchRecipeDetails(String
recipeName) async {
  final response = await
http.get(Uri.parse('https://www.themealdb.com/api/js
on/v1/1/search.php?s=$recipeName'));
  if (response.statusCode == 200) {
    final data = json.decode(response.body);
    if (data['meals'] != null &&
data['meals'].isNotEmpty) {
      return Recipe.fromJson(data['meals'][0]);
    }
  } else {
    print('Failed to load recipe details for
$recipeName');
  }
  return null;
 }
 @override
 Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Row(
        children: [
          Padding(
            padding: const EdgeInsets.only(right: 8.0),
            child: Image.asset(
              'assets/tasty_logo.png', // Path to your asset
              width: 30, // Adjust size as needed
              height: 30, // Adjust size as needed
            ),
          ),
          Text(
            'Tasty',
          ),
        ],
      ),
      centerTitle: false, // Align title to the left
    ),
    body: Column(
      children: [
        Padding(
          padding: const EdgeInsets.all(16.0),
          child: TextField(
            style: TextStyle(color: Colors.white),
            decoration: InputDecoration(
              hintText: 'Search recipes...',
              hintStyle: TextStyle(color:
Colors.grey[500]),
              fillColor: Colors.grey[800],
```

```
            filled: true,
            border: OutlineInputBorder(
              borderRadius:
BorderRadius.circular(30.0),
              borderSide: BorderSide.none,
            ),
            prefixIcon: Icon(Icons.search, color:
Colors.white),
          ),
          onChanged: (value) {
            setState(() {
              _searchTerm = value;
              fetchRecipes();
            });
        ),
        Expanded(
          child: _searchTerm.isEmpty
              ? SingleChildScrollView(
            child: Column(
              crossAxisAlignment:
CrossAxisAlignment.start,
              children: [
                AnimatedOpacity(
                  opacity: _opacity,
                  duration: Duration(milliseconds: 500),
                  child: SizedBox(
                    height: 300,
                    child: _randomRecipe != null
                        ? InkWell(
                      onTap: () {
                        Navigator.push(
                          context,
                          MaterialPageRoute(
                            builder: (context) =>
RecipeDetailScreen(recipe: _randomRecipe!),
                          ),
                        );
                      },
                      child: Card(
                        color: Colors.grey[800],
                        child: Column(
                          crossAxisAlignment:
CrossAxisAlignment.stretch,
                          children: [
                            Expanded(
                              child: Image.network(
                                _randomRecipe!.strMealThumb,
                                fit: BoxFit.cover,
                                errorBuilder: (context, error,
stackTrace) {
                                  return Container(
                                    color: Colors.grey[300],
                                    child: Center(
                                      child:
Icon(Icons.error_outline),
                                    ),
                                  );
                                },
                              ),
                            ),
                            Padding(
                              padding: const
EdgeInsets.all(8.0),
                              child: Align(
                                alignment: Alignment.center,
                                child: Text(
                                  _randomRecipe!.strMeal,
                                  style: TextStyle(
                                    color: Colors.white,
                                    fontSize: 18,
                                    fontWeight: FontWeight.bold,
                                  ),
                                )
                            : Center(child:
CircularProgressIndicator()),
                    ),
                  ),
```

```
            SizedBox(height: 20),
            if (!_isLoadingAreaRecipes)
              ...areas.map((area) {
                return Column(
                  crossAxisAlignment:
CrossAxisAlignment.start,
                  children: [
                    Padding(
                      padding: const
EdgeInsets.symmetric(vertical: 8.0),
                      child: Text(
                        area,
                        style: TextStyle(
                          color: Colors.white,
                          fontSize: 20,
                          fontWeight: FontWeight.bold,
                        ),
                      ),
                    ),
                    SizedBox(
                      height: 250,
                      child: ListView.builder(
                        scrollDirection: Axis.horizontal,
                        itemCount:
_areaRecipes[area]?.length ?? 0,
                        itemBuilder: (context, index) {
                          final recipe =
_areaRecipes[area]![index];
                          return InkWell(
                            onTap: () async {
                              Recipe? detailedRecipe =
await fetchRecipeDetails(recipe.strMeal);
                              if (detailedRecipe != null) {
                                Navigator.push(
                                  context,
                                  MaterialPageRoute(
                                    builder: (context) =>
RecipeDetailScreen(recipe: detailedRecipe),
                                  ),
                                );
                              } else {
                                // Handle the case where
detailed recipe is not found
                                print("Detailed recipe not
found");
                              }
                            },
                            child: Container(
                              width: 200,
                              margin: EdgeInsets.only(right:
10),
                              child: Column(
                                crossAxisAlignment:
CrossAxisAlignment.stretch,
                                children: [
                                  Expanded(
                                    child: ClipRRect(
                                      borderRadius:
BorderRadius.circular(10),
                                      child: Image.network(
                                        recipe.strMealThumb,
                                        fit: BoxFit.cover,
                                        errorBuilder: (context,
error, stackTrace) {
                                          return Container(
                                            color:
Colors.grey[300],
                                            child: Center(
                                              child:
Icon(Icons.error_outline),
                                            ),
                                          Padding(
```

```
                        padding: const
EdgeInsets.all(8.0),
                        child: Text(
                         recipe.strMeal,
                         style: TextStyle(
                          color: Colors.white,
                          fontWeight:
FontWeight.bold,
                        ),
      textAlign: TextAlign.center,
                      ),
                    );
                  }).toList()
                 else
                  Center(child:
CircularProgressIndicator()),
                ],
               ),
             )
              : ListView.builder(
             itemCount: filteredRecipes.length,
             itemBuilder: (context, index) {
              final recipe = filteredRecipes[index];
              return Card(
               color: Colors.grey[800],
               child: InkWell(
                onTap: () {
                 Navigator.push(
                  context,
                  MaterialPageRoute(
                   builder: (context) =>
RecipeDetailScreen(recipe: recipe),
                  ),
                 );
                },
                child: Padding(
                 padding: const EdgeInsets.all(8.0),
                 child: Column(
                  children: [
                   Image.network(
                    recipe.strMealThumb,
                    width: double.infinity,
                    height: 250,
                    fit: BoxFit.cover,
                    errorBuilder: (context, error,
stackTrace) {
                     return Container(
                      width: double.infinity,
                      height: 250,
                      color: Colors.grey[300],
                      child: Center(
                       child: Icon(Icons.error_outline),
                      ),
                     );
                    },
                   ),
                   SizedBox(height: 8),
                   Align(
                    alignment: Alignment.center,
                    child: Text(
                     recipe.strMeal,
                     style: TextStyle(
                      color: Colors.white,
                      fontSize: 18,
                      fontWeight: FontWeight.bold
                     ),
                    ),
   ),
   bottomNavigationBar: BottomNavigationBar(
    items: const <BottomNavigationBarItem>[
     BottomNavigationBarItem(
      icon: Icon(Icons.search),
      label: 'Search',
     ),
     BottomNavigationBarItem(
      icon: Icon(Icons.explore),
      label: 'Explore',
```

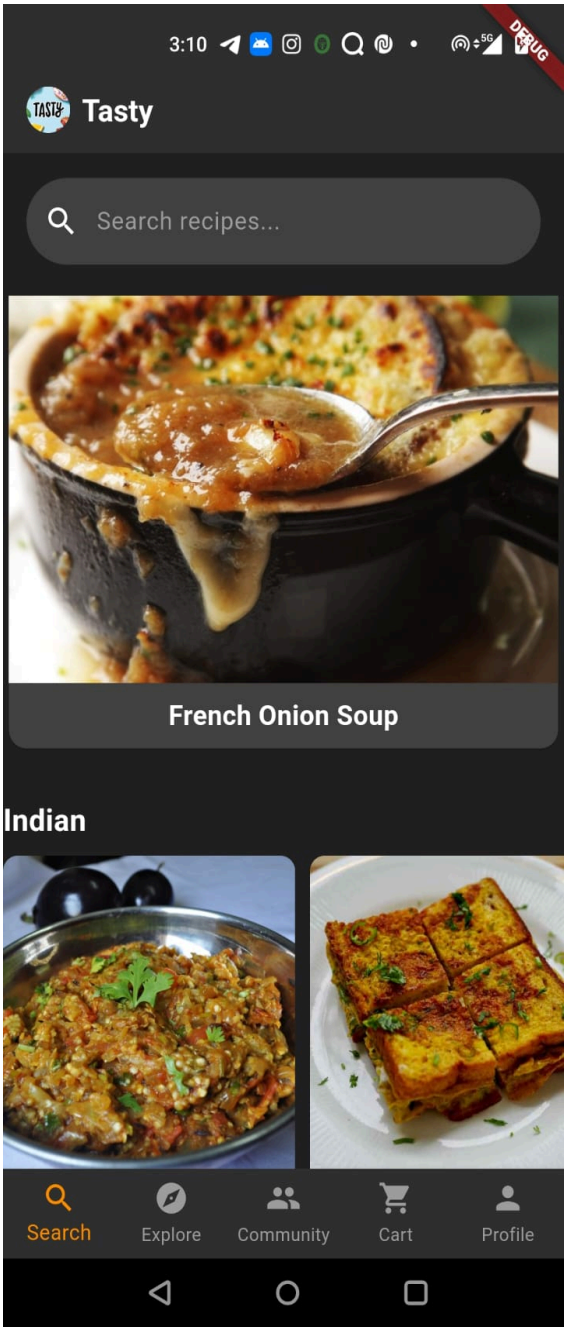
```
        ),
        BottomNavigationBarItem(
          icon: Icon(Icons.people),
          label: 'Community',
        ),
        BottomNavigationBarItem(
          icon: Icon(Icons.shopping_cart),
          label: 'Cart',
        ),
        BottomNavigationBarItem(
          icon: Icon(Icons.person),
          label: 'Profile',
        ),
      ],
      currentIndex: _selectedIndex,
      selectedItemColor: Colors.amber[800],
      unselectedItemColor: Colors.grey[500],
      backgroundColor: Colors.grey[850],
      type: BottomNavigationBarType.fixed,
      onTap: (index) {
        setState(() {
          _selectedIndex = index;
          if (index == 4) {
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) =>
LoginScreen()),
            );
          } else {
            // Handle navigation for other tabs
          }
}
```

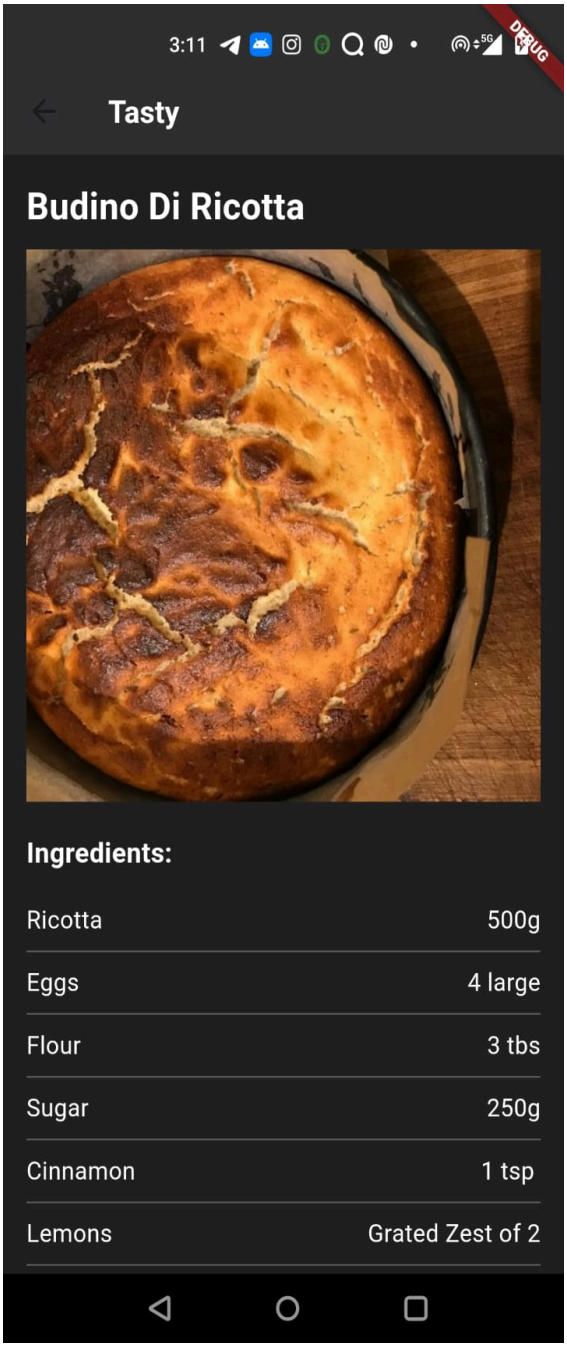
recipe_detail_screen:

```
import 'package:flutter/material.dart';
import '../models/recipe.dart';
import 'package:url_launcher/url_launcher.dart';

class RecipeDetailScreen extends StatelessWidget {
  final Recipe recipe;

  RecipeDetailScreen({required this.recipe});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Tasty'),
      ),
      body: SingleChildScrollView(
        padding: EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment:
CrossAxisAlignment.start,
          children: [
            Text(
              recipe.strMeal,
              style: TextStyle(fontSize: 24, fontWeight:
FontWeight.bold),
            ),
            SizedBox(height: 10),
            Image.network(
              recipe.strMealThumb,
              width: double.infinity,
              fit: BoxFit.cover,
              errorBuilder: (context, error, stackTrace) {
                return Container(
                  height: 200,
                  width: double.infinity,
                  color: Colors.grey[300],
                  child: Center(
                    child: Icon(Icons.error_outline),
                  ),
                );
              },
            ),
            SizedBox(height: 20),
```

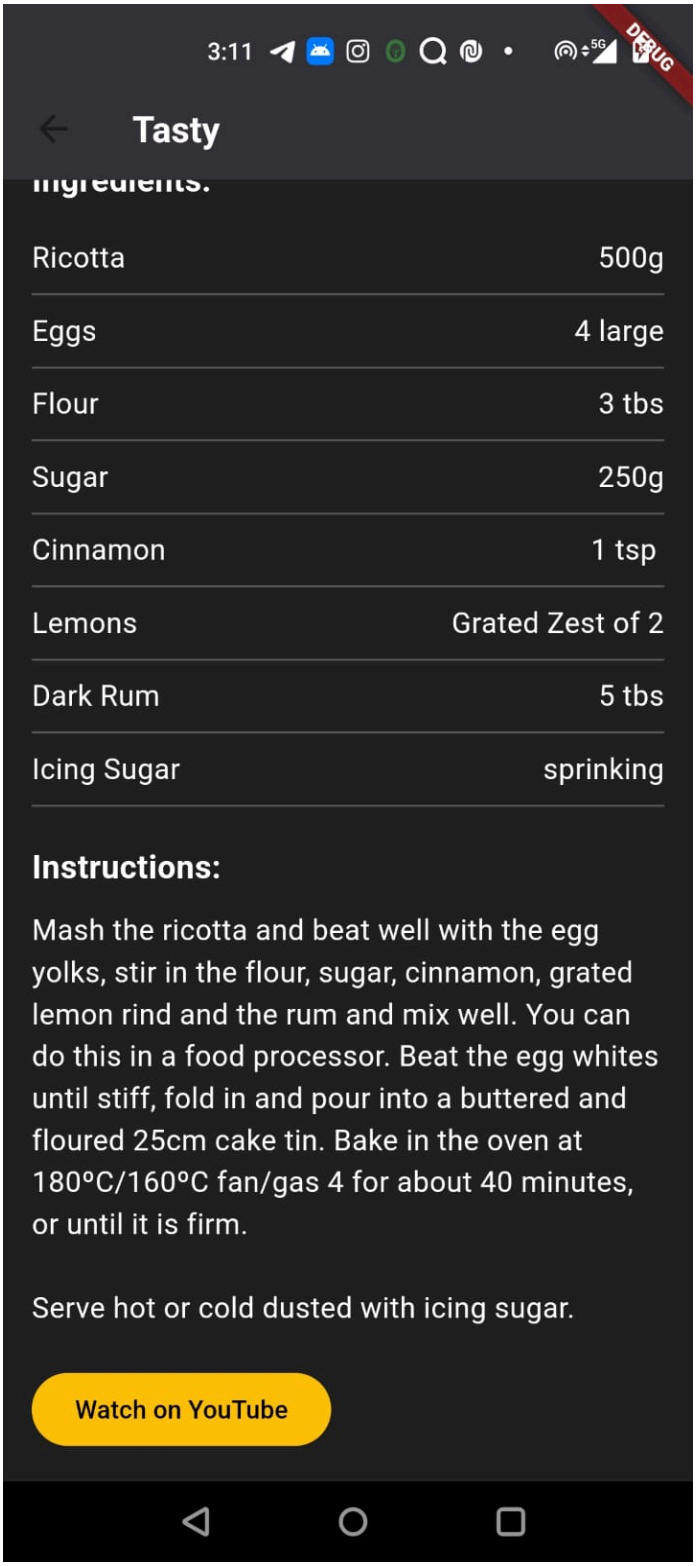
```
        Text(
          'Ingredients:',
          style: TextStyle(fontSize: 18, fontWeight:
FontWeight.bold),
        ),
        SizedBox(height: 10),
        ListView.builder(
          shrinkWrap: true,
          physics: NeverScrollableScrollPhysics(),
          itemCount: recipe.ingredients.length,
          itemBuilder: (context, index) {
            final ingredient = recipe.ingredients[index];
            return Column(
              children: [
                Padding(
                  padding: const
EdgeInsets.symmetric(vertical: 8.0),
                  child: Row(
                    mainAxisAlignment:
MainAxisAlignment.spaceBetween,
                    children: [
                      Text(
                        ingredient.name,
                        style: TextStyle(fontSize: 16),
                      ),
                      Text(
                        ingredient.measure,
                        style: TextStyle(fontSize: 16)
                    ),
                  ),
                  Divider(
                    color: Colors.grey[700], // Dull grey
line
                    height: 1,

        ),
        SizedBox(height: 20),
        Text(
          'Instructions:',
          style: TextStyle(fontSize: 18, fontWeight:
FontWeight.bold),
        ),
        SizedBox(height: 10),
        Text(
          recipe.strInstructions,
          style: TextStyle(fontSize: 16),
        ),
        SizedBox(height: 20),
        if (recipe.strYoutube != null &&
recipe.strYoutube!.isNotEmpty)
          ElevatedButton(
            onPressed: () async {
              final Uri url =
Uri.parse(recipe.strYoutube!);
              if (await canLaunchUrl(url)) {
                await launchUrl(url);
              } else {

ScaffoldMessenger.of(context).showSnackBar(
                  SnackBar(content: Text('Could not
launch ${recipe.strYoutube}')),
                );
              }
            },
            child: Text('Watch on YouTube'),
          ),
        ],
      )
```

Output:


3:10
DEBUG
Tasty
Search recipes...
French Onion Soup
Indian
Search
Explore
Community
Cart
Profile


3:11
DEBUG
Tasty
Budino Di Ricotta
Ingredients:
Ricotta 500g
Eggs 4 large
Flour 3 tbs
Sugar 250g
Cinnamon 1 tsp
Lemons Grated Zest of 2

Project Title: Tasty app **Roll No. 49**



MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	49
Name	Ansh Sarfare
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Project Title: Tasty app **Roll No. 49**

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

Theory: -
Incorporating Visual Elements in Flutter: Icons, Images, and Custom Fonts
Flutter is a powerful open-source UI framework that enables developers to build natively compiled applications for mobile, web, and desktop platforms—all from a single codebase. One of Flutter's greatest strengths is its flexibility in crafting highly customizable UIs.
This practical guide focuses on integrating essential visual elements—icons, images, and custom fonts into a Flutter application. These elements enhance visual appeal, improve usability, and create a more engaging user experience.
Importance of Visual Elements in App Development

- **Enhanced User Experience** – Icons and images make applications more visually appealing and user friendly.
- **Efficient Communication** – Well-designed icons convey information quickly, reducing the need for lengthy text.
- **Brand Identity** – Custom icons and images reinforce branding, making an app more memorable.

Managing Assets in a Flutter App
When a Flutter app is built, it consists of both code and assets. Assets include static files such as images, icons, fonts, and configuration files, which are deployed and available at runtime. Flutter supports multiple image formats, including JPEG, WebP, PNG, GIF, BMP, and WBMP.

Adding Icons in Flutter
Flutter provides built-in Material Design icons through the Icons class. Custom icons can also be integrated using third-party packages like flutter_launcher_icons and font_awesome_flutter.
Example (Built-in Material Icons):

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

Adding Images in Flutter
Flutter supports images from three primary sources: Assets, Network, and Local Storage (Memory or File
System).

1. Using Asset Images (Local Project Files)
To use an image stored in the project folder:
Place the image inside the assets/images/ folder.
Declare it in pubspec.yaml:

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

Display it in the app:
Image.asset('assets/images/sample.png');

2. Using Network Images (Fetched from the Internet)
Flutter simplifies loading images from the web using Image.network. Additional properties like height,
width, fit, and color can be specified.
Example:
Image.network('https://example.com/sample.jpg');

3. Using Local Storage (Memory or File System)
Images stored on the user's device can also be displayed using packages like image_picker or file_picker.
Adding Custom Fonts in Flutter
By default, Flutter uses the Roboto font. However, custom fonts can be added to create a unique visual
Identity.

Steps to Add a Custom Font:
1. Download the font and place it in the assets/fonts/ folder.

2. Declare the font in pubspec.yaml:
yaml
```
flutter:
 fonts:
 - family: CustomFont
 fonts:
 - asset: assets/fonts/CustomFont.ttf
```

3. Use the font in your app
```
Text(
 'Custom Font Example',
 style: TextStyle(fontFamily: 'CustomFont', fontSize: 24),
);
```

Project Title: Tasty app **Roll No. 49**

Code:

Profile_page:

```
import 'package:flutter/material.dart';
import
'package:cloud_firestore/cloud_firestore.dart
';
import
'package:firebase_auth/firebase_auth.dart';
import
'../services/firebase_auth_service.dart';
import
'../screens/liked_recipes_screen.dart';
import
'../screens/saved_recipes_screen.dart';
import '../screens/orders_page.dart';
import '../screens/address_page.dart';
import
'../screens/community_post_page.dart'; //
Import CommunityPostsPage

class ProfilePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Container(
          alignment: Alignment.topLeft,
          child: const Text("Profile"),
        ),
        actions: [
          IconButton(
            icon: const Icon(Icons.location_on,
color: Colors.white),
            onPressed: () {
              Navigator.push(
                context,
                MaterialPageRoute(builder:
(context) => AddressPage()),
              );
            },
          ),
          IconButton(
            icon: const Icon(Icons.image_sharp,
color: Colors.white), // Community Posts
icon
            onPressed: () {
              Navigator.push(
                context,
                MaterialPageRoute(builder:
(context) => CommunityPostsPage()),
              );
            },
          ),
          IconButton(
            icon: const Icon(Icons.logout, color:
Colors.white),
            onPressed: () async {
              final authService = AuthService();
              await authService.signOut();

Navigator.pushReplacementNamed(context,
'/login');
            },
          ),
        ],
      ),
      body:
FutureBuilder<DocumentSnapshot>(
        future:
FirebaseFirestore.instance.collection('users'
).doc(FirebaseAuth.instance.currentUser!.ui
d).get(),
        builder: (BuildContext context,
AsyncSnapshot<DocumentSnapshot>
snapshot) {
          if (snapshot.connectionState ==
ConnectionState.waiting) {
            return const Center(child:
CircularProgressIndicator());
          }
          if (!snapshot.hasData ||
!snapshot.data!.exists) {
```

```
          return const Center(child: Text("No
data found"));
        }

        var userData = snapshot.data!.data()
as Map<String, dynamic>;
        String username =
userData['username'] ?? 'User';

        return SingleChildScrollView(
          padding: const EdgeInsets.all(16.0),
          child: Column(
            children: [
              Text(
                'Hello, $username',
                style: const TextStyle(fontSize:
20, fontWeight: FontWeight.bold),
                textAlign: TextAlign.center,
              ),
              const SizedBox(height: 20),
              GestureDetector(
                onTap: () {
                  Navigator.push(context,
MaterialPageRoute(builder: (context) =>
LikedRecipesScreen()));
                },
                child: Container(
                  height: 150,
                  width: double.infinity,
                  decoration: BoxDecoration(
                    image: const
DecorationImage(
                      image:
AssetImage('assets/liked_recipes.png'),
                      fit: BoxFit.cover,
                    ),
                    borderRadius:
BorderRadius.circular(10),
                  ),
                ),
              ),
              const SizedBox(height: 20),
              GestureDetector(
                onTap: () {
                  Navigator.push(context,
MaterialPageRoute(builder: (context) =>
SavedRecipesScreen()));
                },
                child: Container(
                  height: 150,
                  width: double.infinity,
                  decoration: BoxDecoration(
                    image: const
DecorationImage(
                      image:
AssetImage('assets/saved_recipes.png'),
                      fit: BoxFit.cover,
                    ),
                    borderRadius:
BorderRadius.circular(10),
                  ),
                ),
              ),
              const SizedBox(height: 20),
              GestureDetector(
                onTap: () {
                  Navigator.push(
                    context,
                    MaterialPageRoute(
                      builder: (context) =>
OrdersPage(),
                    ),
                  );
                },
                child: Container(
                  height: 150,
                  width: double.infinity,
                  decoration: BoxDecoration(
                    image: const
DecorationImage(
                      image:
AssetImage('assets/orders.png'),
                      fit: BoxFit.cover,
                    ),
                    borderRadius:
BorderRadius.circular(10),
                  ),
                ),
```

Explore_screen:

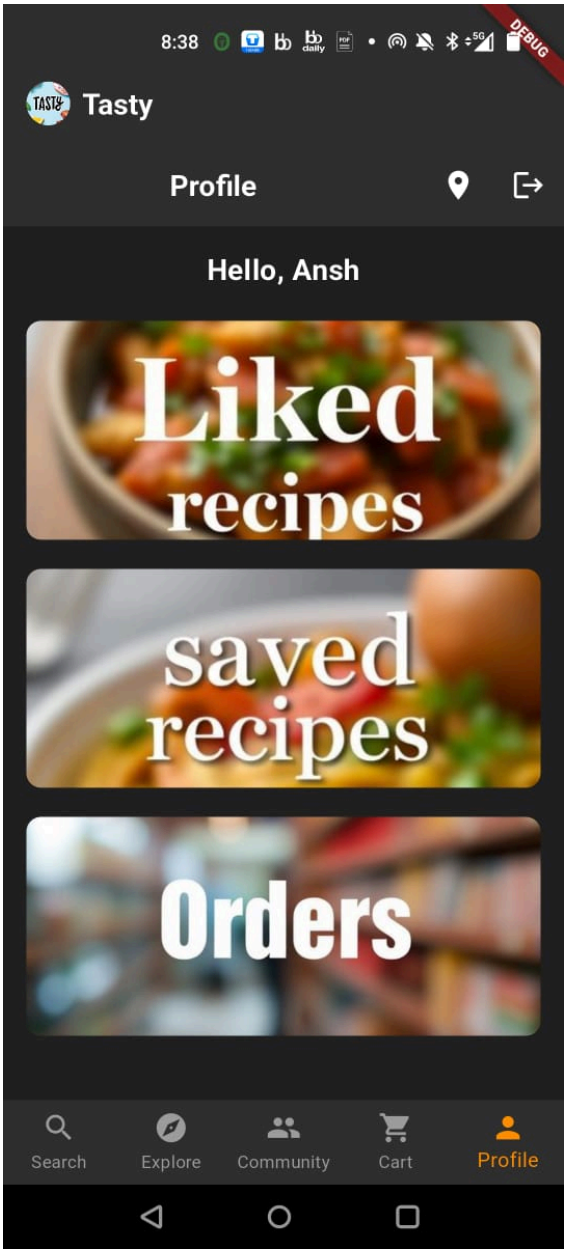
```
import 'package:flutter/material.dart';
import 'dart:convert';
import 'package:http/http.dart' as http;
import 'widgets/recipe_detail_screen.dart';
import 'models/recipe.dart';

class ExploreScreen extends
StatefulWidget {
  @override
  _ExploreScreenState createState() =>
_ExploreScreenState();
}
class _ExploreScreenState extends
State<ExploreScreen> {
  List<Category> categories = [];
  bool _isLoading = true;

  @override
  void initState() {
    super.initState();
    fetchCategories();
  }
  Future<void> fetchCategories() async {
    final response = await
http.get(Uri.parse('https://www.themealdb.co
m/api/json/v1/1/categories.php'));
    if (response.statusCode == 200) {
      final data =
json.decode(response.body);
      if (data['categories'] != null) {
        setState(() {
          categories = (data['categories'] as
List).map((json) =>
Category.fromJson(json)).toList();
          _isLoading = false;
        });
      }
    } else {
      print('Failed to load categories');
      setState(() {
        _isLoading = false;
  }

  Future<List<Recipe>>
fetchCategoryRecipes(String
categoryName) async {
    final response = await
http.get(Uri.parse('https://www.themealdb.co
m/api/json/v1/1/filter.php?c=$categoryName
'));
    if (response.statusCode == 200) {
      final data =
json.decode(response.body);
      if (data['meals'] != null) {
        return (data['meals'] as List).map((json)
=> Recipe.fromJson(json)).toList();
      } else {
        return [];
      }
    } else {
      print('Failed to load recipes for category:
$categoryName');
      return [];
    }

  Future<Recipe?>
fetchRecipeDetails(String recipeName)
async {
    final response = await
http.get(Uri.parse('https://www.themealdb.co
m/api/json/v1/1/search.php?s=$recipeName
'));
    if (response.statusCode == 200) {
      final data =
json.decode(response.body);
      if (data['meals'] != null &&
data['meals'].isNotEmpty) {
        return
Recipe.fromJson(data['meals'][0]);
      }
    } else {
      print('Failed to load recipe details for
$recipeName');
    }
    return null;
  }
```

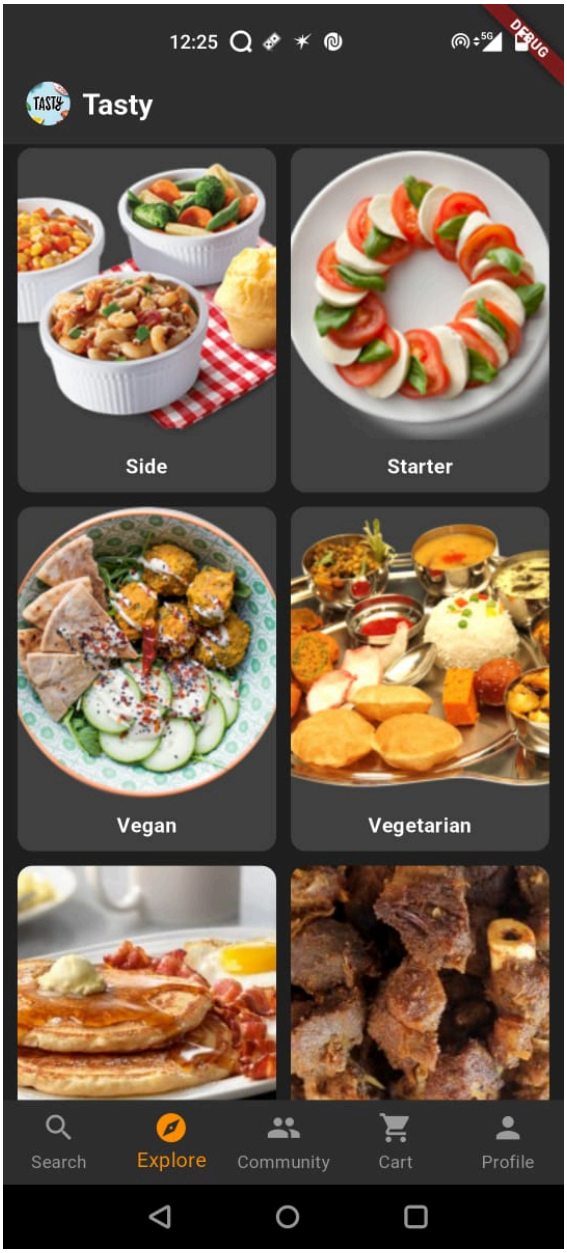
```
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   backgroundColor: Colors.grey[900],
   body: _isLoading
      ? Center(child:
CircularProgressIndicator())
      : GridView.builder(
     padding: EdgeInsets.all(10.0),
     gridDelegate:
SliverGridDelegateWithFixedCrossAxisCou
nt(
       crossAxisCount: 2,
       crossAxisSpacing: 10,
       mainAxisSpacing: 10,
       childAspectRatio: 0.75,
     ),
     itemCount: categories.length,
     itemBuilder: (context, index) {
       final category = categories[index];
       return InkWell(
         onTap: () async {
           List<Recipe> recipes = await
fetchCategoryRecipes(category.strCategory
);
           Navigator.push(
             context,
             MaterialPageRoute(
               builder: (context) =>
CategoryRecipeScreen(
                 categoryName:
category.strCategory,
                 recipes: recipes,
               ),
         child: Container(
           decoration: BoxDecoration(
             color: Colors.grey[800],
             borderRadius:
BorderRadius.circular(10),
           ),
           child: Column(
             crossAxisAlignment:
CrossAxisAlignment.stretch,
             children: [
               Expanded(
                 child: ClipRRect(
                   borderRadius:
BorderRadius.vertical(top:
Radius.circular(10)),
                   child: Image.network(
                     category.strCategoryThumb,
                     fit: BoxFit.cover,
                     errorBuilder: (context, error,
stackTrace) {
                       return Container(
                         color: Colors.grey[300],
                         child: Center(child:
Icon(Icons.error_outline)),
                       );
               Padding(
                 padding: const
EdgeInsets.all(8.0),
                 child: Text(
                   category.strCategory,
                   style: TextStyle(
                     color: Colors.white,
                     fontWeight: FontWeight.bold,
                   ),
                   textAlign: TextAlign.center,
                 ),
class CategoryRecipeScreen extends
StatelessWidget {
  final String categoryName;
  final List<Recipe> recipes;

  CategoryRecipeScreen({required
this.categoryName, required this.recipes});

  Future<Recipe?>
fetchRecipeDetails(String recipeName)
async {
    final response = await
http.get(Uri.parse('https://www.themealdb.co
m/api/json/v1/1/search.php?s=$recipeName
'));
    if (response.statusCode == 200) {
```

```
    final data =
json.decode(response.body);
    if (data['meals'] != null &&
data['meals'].isNotEmpty) {
      return
Recipe.fromJson(data['meals'][0]);
    }
   } else {
    print('Failed to load recipe details for
$recipeName');
   }
   return null;
 }
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
    title: Text(categoryName),
   ),
   backgroundColor: Colors.grey[900],
   body: recipes.isEmpty
      ? Center(
    child: Text(
     'No recipes found for this category.',
     style: TextStyle(color: Colors.white),
    ),
   )
      : ListView.builder(
    itemCount: recipes.length,
    itemBuilder: (context, index) {
     final recipe = recipes[index];
     return Card(
      color: Colors.grey[800],
      child: InkWell(
       onTap: () async {
        Recipe? detailedRecipe = await
fetchRecipeDetails(recipe.strMeal);
        if (detailedRecipe != null) {
         Navigator.push(
          context,
          MaterialPageRoute(
           builder: (context) =>
RecipeDetailScreen(recipe:
detailedRecipe),
        } else {
         print('Failed to load detailed
recipe');
       },
 {
            return Container(
             width: double.infinity,
             height: 250,
             color: Colors.grey[300],
             child: Center(
              child:
Icon(Icons.error_outline),
             ),
            SizedBox(height: 8),
            Align(
             alignment: Alignment.center,
             child: Text(
              recipe.strMeal,
              style: TextStyle(
               color: Colors.white,
               fontSize: 18,
               fontWeight:
FontWeight.bold,
             ),
}
class Category {
 final String strCategory;
 final String strCategoryThumb;
 final String idCategory;

 Category({
  required this.strCategory,
  required this.strCategoryThumb,
  required this.idCategory,
 });

 factory Category.fromJson(Map<String,
dynamic> json) {
  return Category(
   strCategory: json['strCategory'],
   strCategoryThumb:
json['strCategoryThumb'],
   idCategory: json['idCategory'],
  );
```

Project Title: Tasty app **Roll No. 49**

Output:





MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	49
Name	Ansh Sarfare
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Project Title: Tasty app **Roll No. 49**

Aim: To create an interactive form using form widget

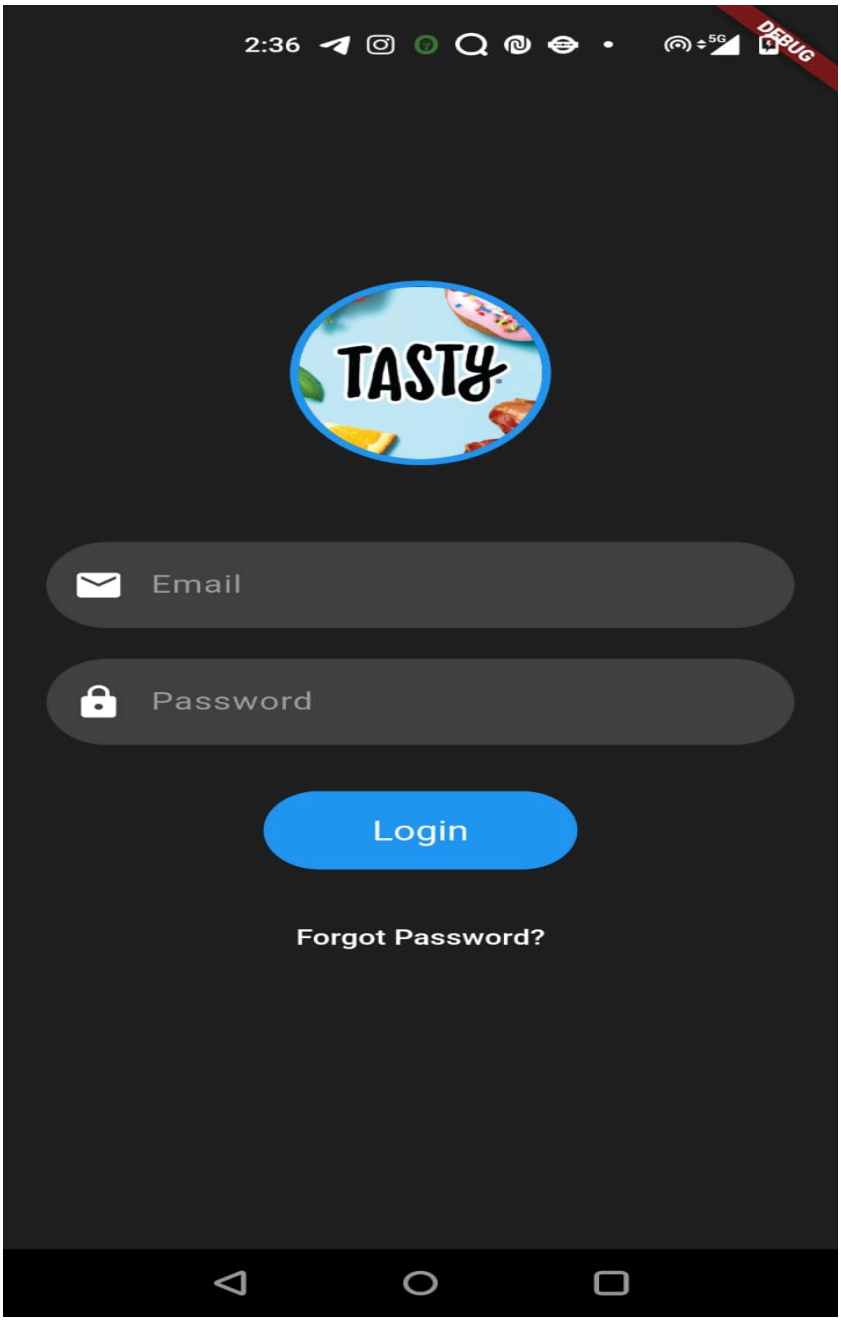
Code:

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

class LoginScreen extends StatefulWidget {
  @override
  _LoginScreenState createState() =>
_LoginScreenState();
}

class _LoginScreenState extends
State<LoginScreen> {
  final _formKey = GlobalKey<FormState>();
  String? _email, _password;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.grey[900],
      body: Center(
        child: SingleChildScrollView(
          padding: EdgeInsets.all(20.0),
          child: Form(
            key: _formKey,
            child: Column(
              mainAxisAlignment:
MainAxisAlignment.center,
              children: <Widget>[
                // Logo
                Container(
                  width: 120,
                  height: 120,
                  decoration: BoxDecoration(
                    shape: BoxShape.circle,
                    border: Border.all(
                      color: Colors.blue,
                      width: 4.0,
                    ),
                  ),
                  child: Center(
                    child: Image.asset(
'assets/tasty_logo.png', // Path to your asset
                      width: 120, // Adjust as
needed
                      height: 120, // Adjust as
needed
                    ),
                  ),
                ),
                SizedBox(height: 50),


                TextFormField(
                  decoration: InputDecoration(
                    hintText: 'Email',
                    hintStyle: TextStyle(color:
Colors.grey[500]),
                    fillColor: Colors.grey[800],
                    filled: true,
                    border: OutlineInputBorder(
                      borderRadius:
BorderRadius.circular(30.0),
                      borderSide: BorderSide.none,
                    ),
                    prefixIcon: Icon(Icons.email,
color: Colors.white),
                  ),
                  style: TextStyle(color:
Colors.white),
                  validator: (value) {
                    if (value == null ||
value.isEmpty) {
                      return 'Please enter your
email';
                    }
                    if (!value.contains('@')) {
                      return 'Please enter a valid
email';
                    }
                    return null;
                  },
```

```
            onSaved: (value) => _email =
value,
          ),
          SizedBox(height: 20),
          TextFormField(
            decoration: InputDecoration(
              hintText: 'Password',
              hintStyle: TextStyle(color:
Colors.grey[500]),
              fillColor: Colors.grey[800],
              filled: true,
              border: OutlineInputBorder(
                borderRadius:
BorderRadius.circular(30.0),
                borderSide: BorderSide.none,
              ),
              prefixIcon: Icon(Icons.lock,
color: Colors.white),
            ),
            style: TextStyle(color:
Colors.white),
            obscureText: true,
            validator: (value) {
              if (value == null ||
value.isEmpty) {
                return 'Please enter your
password';
              }
              if (value.length < 6) {
                return 'Password must be at
least 6 characters';
              }
              return null;
            },
            onSaved: (value) => _password
= value,
          ),
          SizedBox(height: 30),

          // Login Button
          ElevatedButton(
            style: ElevatedButton.styleFrom(
              backgroundColor: Colors.blue,
              foregroundColor: Colors.white,
              padding:
EdgeInsets.symmetric(horizontal: 50,
vertical: 15),
              textStyle: TextStyle(fontSize:
18),
              shape:
RoundedRectangleBorder(
                borderRadius:
BorderRadius.circular(30),
              ),
            ),
            onPressed: () {
              if
(_formKey.currentState!.validate()) {

_formKey.currentState!.save();
                // TODO: Implement login
functionality
                print('Email: $_email,
Password: $_password');
              }
            },
            child: Text('Login'),
          ),
          SizedBox(height: 20),

          TextButton(
            onPressed: () {
              // TODO: Implement forgot
password functionality
            },
            child: Text(
              'Forgot Password?',
              style: TextStyle(color:
Colors.white),
            )
      ),
```

Project Title: Tasty app **Roll No. 49**

Output:



MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	49
Name	Ansh Sarfare
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Project Title: Tasty app **Roll No. 49**

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

Theory: -

Navigation, Routing, and Gesture Handling in Flutter
In Flutter, screens or pages are referred to as routes, and each route is essentially a widget. This concept is similar to Activities in Android. Navigating between pages defines an app's workflow, and the mechanism for handling this is known as routing.
Flutter provides a built-in routing system using MaterialPageRoute, along with the Navigator.push() and Navigator.pop() methods to move between routes.
Additionally, gestures allow apps to respond to user interactions like taps, swipes, and drags, making applications more dynamic and user-friendly.

Navigation and Routing in Flutter,

1. Using the Navigator Widget

Flutter's Navigator widget manages a stack of routes, enabling seamless navigation between screens.
Pushing a Route: Moves to a new screen using Navigator.push().
Popping a Route: Returns to the previous screen using Navigator.pop().

Example:

```
ElevatedButton(
 onPressed: () {
 Navigator.push(
 context,
 MaterialPageRoute(builder: (context) => SecondScreen()),
 );
 },
 child: Text('Go to Second Screen'),
);
```

2. Using Named Routes

For larger applications, named routes provide a cleaner and more structured way to manage navigation.
Step 1: Define Routes in MaterialApp

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

);
Step 2: Navigate Using Navigator.pushNamed()
Navigator.pushNamed(context, '/second');
Handling Gestures in Flutter
Gestures enable user interaction through taps, swipes, pinches, and drags. Flutter provides various widgets and gesture detectors to manage these interactions effectively.

1. Tap Gestures

Taps are one of the most common interactions and can be handled using:
GestureDetector
InkWell
ElevatedButton
Example (Tap Gesture using GestureDetector):

```
GestureDetector(
 onTap: () {
 print("Tapped!");
 },
 child: Container(
 padding: EdgeInsets.all(20),
 color: Colors.blue,
 child: Text('Tap Me'),
 ),
);
```

2. Long Press Gestures

Long-press interactions can be captured using the onLongPress callback in GestureDetector or InkWell.

```
InkWell(
 onLongPress: () {
 print("Long Pressed!");
 },
 child: Container(
 padding: EdgeInsets.all(20),
 color: Colors.red,
 child: Text('Long Press Me'),
 ),
);
```

3. Swipe and Drag Gestures

Flutter provides built-in methods like onHorizontalDragUpdate and onVerticalDragUpdate to detect
swipe and drag actions.
Example (Swipe Detection):

```
GestureDetector(
 onHorizontalDragUpdate: (details) {
 if (details.primaryDelta! > 0) {
 print("Swiped Right!");
 } else {
 print("Swiped Left!");
 }
 },
 child: Container(
 padding: EdgeInsets.all(20),
 color: Colors.green,
 child: Text('Swipe Me'),
 ),
);
```

Code:

Cart page:

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'cart_provider.dart';
import 'personal_cart.dart';
import
'package:cloud_firestore/cloud_firestore.dart
';
import '../models/cart_model.dart';

class CartPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Cart'),
        actions: [
          IconButton(
            icon: Icon(Icons.shopping_cart,
color: Colors.white),
            onPressed: () {
              Navigator.push(
                context,
                MaterialPageRoute(builder:
(context) => PersonalCartPage()),
              );
            },
          ),
        ],
      ),
      body: StreamBuilder<QuerySnapshot>(
        stream:
FirebaseFirestore.instance.collection('cart').
snapshots(),
        builder: (context, snapshot) {
          if (!snapshot.hasData) {
```

```
          return Center(child:
CircularProgressIndicator());
        }

        return GridView.builder(
          padding: EdgeInsets.all(10),
          gridDelegate:
SliverGridDelegateWithFixedCrossAxisCou
nt(
            crossAxisCount: 2,
            crossAxisSpacing: 10,
            mainAxisSpacing: 10,
            childAspectRatio: 1,
          ),
          itemCount:
snapshot.data!.docs.length,
          itemBuilder: (context, index) {
            var category =
snapshot.data!.docs[index];
            return CategoryCard(category:
category);
          },
        );
      },
    ),
  );
 }
}

class CategoryCard extends
StatelessWidget {
 final QueryDocumentSnapshot category;

 CategoryCard({required this.category});

 @override
 Widget build(BuildContext context) {
   String categoryName = category.id;
   String imagePath =
'assets/${categoryName}.png'; // Category
image should be named categoryName.png
and present in assets.

   return GestureDetector(
     onTap: () {
      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) =>
ItemList(categoryName: categoryName),
        ),
      );
     },
     child: Card(
      elevation: 5,
      child: Column(
        mainAxisAlignment:
MainAxisAlignment.center,
        children: [
          Image.asset(
            imagePath,
            width: 130, // Increased image
width
            height: 130, // Increased image
height
            fit: BoxFit.cover,
          ),
          SizedBox(height: 10),
          Text(
            categoryName.toUpperCase(),
            style: TextStyle(fontWeight:
FontWeight.bold, color: Colors.black), //
Black text
          ),
        ],
      ),
     ),
   );
 }
}

class ItemList extends StatelessWidget {
 final String categoryName;

 ItemList({required this.categoryName});

 @override
 Widget build(BuildContext context) {
```

```
    return Scaffold(
      appBar: AppBar(
        title:
Text('${categoryName.toUpperCase()}'),
      ),
      body:
StreamBuilder<DocumentSnapshot>(
        stream:
FirebaseFirestore.instance.collection('cart').
doc(categoryName).snapshots(),
        builder: (context, snapshot) {
          if (!snapshot.hasData) {
            return Center(child:
CircularProgressIndicator());
          }

          var categoryData =
snapshot.data!.data() as Map<String,
dynamic>;
          List<dynamic> items =
categoryData['items'];

          return ListView.builder(
            itemCount: items.length,
            itemBuilder: (context, index) {
              Map<String, dynamic> item =
items[index];
              return ItemCard(item: item,
categoryName: categoryName);
            },
          );
        },
      ),
    );
  }
}

class ItemCard extends StatelessWidget {
  final Map<String, dynamic> item;
  final String categoryName;

  ItemCard({required this.item, required
this.categoryName});

  @override
  Widget build(BuildContext context) {
    final cartProvider =
Provider.of<CartProvider>(context);
    return Card(
      child: Padding(
        padding: const EdgeInsets.all(8.0),
        child: Row(
          children: [
            Image.network(
              item['imageURL'],
              width: 100,
              height: 100,
              fit: BoxFit.cover,
            ),
            SizedBox(width: 10),
            Expanded(
              child: Column(
                crossAxisAlignment:
CrossAxisAlignment.start,
                children: [
                  Text(
                    item['name'],
                    style: TextStyle(fontSize: 16,
fontWeight: FontWeight.bold, color:
Colors.black), // Black text
                  ),
                  Text(

'\₹${item['price'].toStringAsFixed(2)}',
                    style: TextStyle(color:
Colors.black), // Black text
                  ),
                ],
              ),
            ),
            QuantityControl(item: item,
cartProvider: cartProvider),
          ],
        ),
      ),
    );
  }
}
```

```
class QuantityControl extends
StatefulWidget {
  final Map<String, dynamic> item;
  final CartProvider cartProvider;

  QuantityControl({required this.item,
required this.cartProvider});

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

class _QuantityControlState extends
State<QuantityControl> {
  late int quantity;

  @override
  void initState() {
    super.initState();
    // Initialize quantity based on what's in the
cart
    quantity =
widget.cartProvider.getQuantity(widget.item[
'name']);
  }

  @override
  Widget build(BuildContext context) {
    return Row(
      mainAxisSize: MainAxisSize.min,
      children: [
        IconButton(
          icon: Icon(Icons.remove, color:
Colors.black), // Black icon
          onPressed: quantity > 0
              ? () {
            setState(() {
              quantity--;
            });
            widget.cartProvider.removeItem(
              CartItem(
                name: widget.item['name'],
                imageURL:
widget.item['imageURL'],
                price:
widget.item['price'].toDouble(),
                quantity: 1,
              ),
            );
          }
              : null,
        ),
        Text('$quantity', style: TextStyle(color:
Colors.black)), // Black text
        IconButton(
          icon: Icon(Icons.add, color:
Colors.black), // Black icon
          onPressed: () {
            setState(() {
              quantity++;
            });
            widget.cartProvider.addItem(
              CartItem(
                name: widget.item['name'],
                imageURL:
widget.item['imageURL'],
                price:
widget.item['price'].toDouble(),
                quantity: 1,
              ),
            );
          },
        ),
      ],
    );
  }
}
```

Community page:

```
import 'package:flutter/material.dart';
import
'package:cloud_firestore/cloud_firestore.dart
';
import
'package:firebase_auth/firebase_auth.dart';
import 'dart:io';
import '../services/cloudinary_service.dart';

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

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

class _CommunityPageState extends
State<CommunityPage> {
  final _formKey = GlobalKey<FormState>();
  final _captionController =
TextEditingController();
  File? _image;
  bool _uploading = false;

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

  @override
  void dispose() {
    _captionController.dispose();
    super.dispose();
  }

  Future<void> _pickImage() async {
    if (!mounted) return;

    try {
      final pickedImage = await
CloudinaryService.pickImage();
      if (pickedImage != null && mounted) {
        setState(() {
          _image = pickedImage;
        });
      }
    } catch (e) {
      if (!mounted) return;

ScaffoldMessenger.of(context).showSnackB
ar(
        const SnackBar(content: Text('Failed to
pick image')),
      );
    }
  }

  Future<void> _uploadPost() async {
    if (!_formKey.currentState!.validate() ||
_image == null) {
      return;
    }

    if (!mounted) return;
    setState(() {
      _uploading = true;
    });

    try {
      // Upload image to Cloudinary
      final imageUrl = await
CloudinaryService.uploadImage(_image!);
      if (imageUrl == null) {
        throw Exception('Failed to upload
image');
      }

      // Get current user
      final user =
FirebaseAuth.instance.currentUser;
      if (user == null) {
```

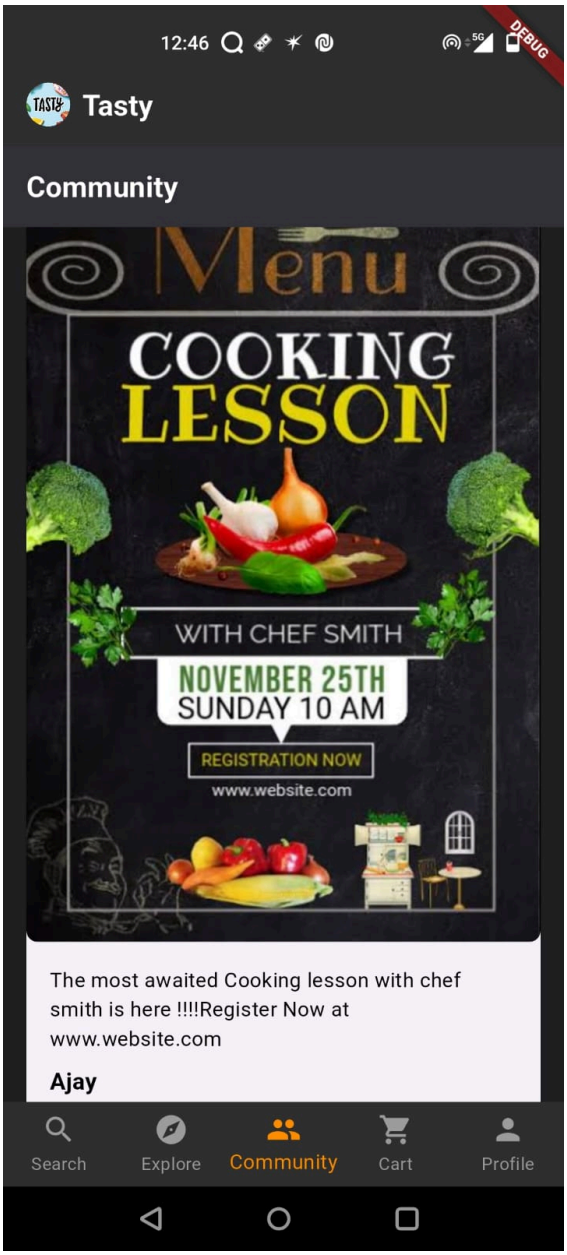
```
      throw Exception('User not logged in');
    }

    // Get user data from Firestore
    final userData = await
FirebaseFirestore.instance
        .collection('users')
        .doc(user.uid)
        .get();

    // Create post in Firestore
    await
FirebaseFirestore.instance.collection('comm
unityPosts').add({
      'userId': user.uid,
      'username': userData['username'],
      'imageUrl': imageUrl,
      'caption': _captionController.text.trim(),
      'timestamp':
FieldValue.serverTimestamp(),
    });

    if (!mounted) return;

    // Clear form
    _captionController.clear();
    setState(() {
      _image = null;
    });


ScaffoldMessenger.of(context).showSnackB
ar(
      const SnackBar(content: Text('Post
uploaded successfully!')),
    );
  } catch (e) {
    print('Error uploading post: $e');
    if (!mounted) return;


ScaffoldMessenger.of(context).showSnackB
ar(
      const SnackBar(content: Text('Failed to
upload post')),
    );
  } finally {
    if (!mounted) return;
    setState(() {
      _uploading = false;
    });
  }
 }

 @override
 Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Community'),
    ),
    body: SingleChildScrollView(
      key: const
PageStorageKey('community_scroll'),
      padding: const EdgeInsets.all(16.0),
      child: Column(
        children: [
          Form(
            key: _formKey,
            child: Column(
              children: [
                if (_image != null)
                  ClipRRect(
                    borderRadius:
BorderRadius.circular(8.0),
                    child: Image.file(
                      _image!,
                      height: 200,
                      width: double.infinity,
                      fit: BoxFit.cover,
                    ),
                  )
                else
                  OutlinedButton(
                    onPressed: _uploading ? null :
_pickImage,
                    child: const Text('Select
Image'),
```

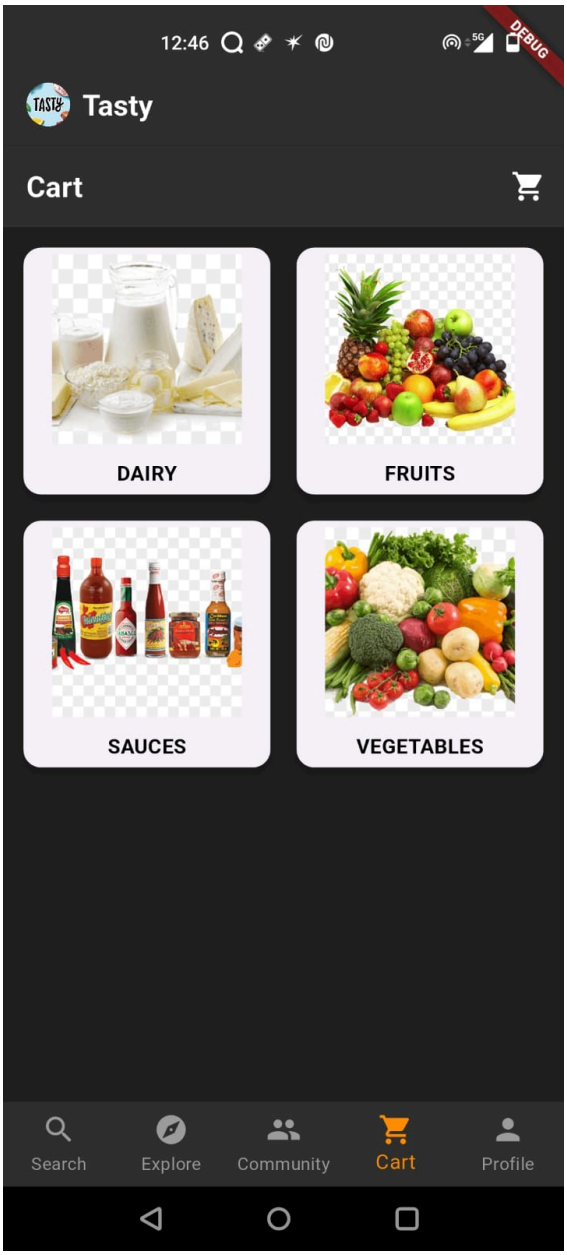
```
                ),
              const SizedBox(height: 16),

              TextFormField(
                controller: _captionController,
                enabled: !_uploading,
                decoration: const
InputDecoration(
                  labelText: 'Caption',
                  border: OutlineInputBorder(),
                ),
                maxLines: 3,
                style: const TextStyle(color:
Colors.white),
                validator: (value) {
                  if (value == null ||
value.trim().isEmpty) {
                    return 'Please enter a
caption';
                  }
                  return null;
                },
              ),
              const SizedBox(height: 16),

              SizedBox(
                width: double.infinity,
                child: ElevatedButton(
                  onPressed: _uploading ? null :
_uploadPost,
                  child: _uploading
                      ? const SizedBox(
                    height: 20,
                    width: 20,
                    child:
CircularProgressIndicator(strokeWidth: 2),
                  )
                      : const Text('Upload Post'),
                ),
              ),
            ],
          ),
        ),
        const SizedBox(height: 32),
            StreamBuilder<QuerySnapshot>(
              stream: FirebaseFirestore.instance
                  .collection('communityPosts')
                  .orderBy('timestamp',
descending: true)
                  .snapshots(),
              builder: (context, snapshot) {
                if (snapshot.hasError) {
                  return Center(
                    child: Text('Error:
${snapshot.error}'),
                  );
                }

                if (snapshot.connectionState ==
ConnectionState.waiting) {
                  return const Center(
                    child:
CircularProgressIndicator(),
                  );
                }

                // Get the documents or an empty
list if they are null
                final posts = snapshot.data?.docs
?? [];

                if (posts.isEmpty) {
                  return const Center(
                    child: Text('No posts yet!'),
                  );
                }

                return ListView.builder(
                  key: const
PageStorageKey('community_posts'),
                  shrinkWrap: true,
                  physics: const
NeverScrollableScrollPhysics(),
                  itemCount: posts.length,
                  itemBuilder: (context, index) {
                    final post = posts[index];
                    final data = post.data() as
Map<String, dynamic>;
```

```
                return Card(
                  key: ValueKey(post.id),
                  margin: const
EdgeInsets.only(bottom: 16.0),
                  child: Column(
                    crossAxisAlignment:
CrossAxisAlignment.start,
                    children: [
                      ClipRRect(
                        borderRadius:
BorderRadius.circular(8.0),
                        child: Image.network(
                          data['imageUrl'],
                          width: double.infinity,
                          fit: BoxFit.contain, //
Changed to BoxFit.contain
                          loadingBuilder: (context,
child, loadingProgress) {
                            if (loadingProgress ==
null) return child;
                            return SizedBox(
                              height: 300,
                              child: Center(
                                child:
CircularProgressIndicator(
                                  value:
loadingProgress.expectedTotalBytes != null
                                      ?
loadingProgress.cumulativeBytesLoaded /

loadingProgress.expectedTotalBytes!
                                      : null,
                                ),
                              ),
                            );
                          },
                          errorBuilder: (context,
error, stackTrace) {
                            return const SizedBox(
                              height: 300,
                              child: Center(
                                child:
Icon(Icons.error_outline, size: 40),
                                ),
                              );
                            },
                          ),
                        ),
                        Padding(
                          padding: const
EdgeInsets.all(16.0),
                          child: Column(
                            crossAxisAlignment:
CrossAxisAlignment.start,
                            children: [
                              Text(
                                data['caption'] ?? '',
                                style: const
TextStyle(color: Colors.black),
                              ),
                              const SizedBox(height:
8),
                              Text(
                                data['username'] ??
'Unknown User',
                                style: const TextStyle(
                                  fontWeight:
FontWeight.bold,
                                  fontSize: 16,
                                  color: Colors.black,
                                ),
                              ),
                            ],
                          ),
                        ),
                      ],
                    ),
                  ),
                ],
              ),
            );
          },
        );
      },
    ),
  ],
```

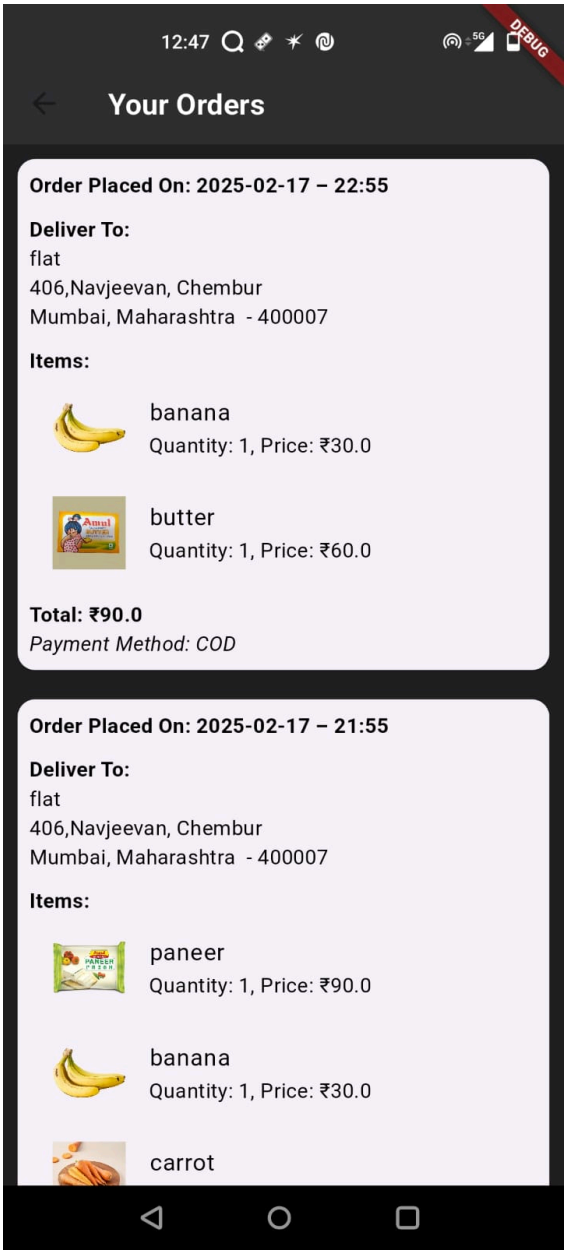
Project Title: Tasty app **Roll No. 49**

Output:





Project Title: Tasty app **Roll No. 49**



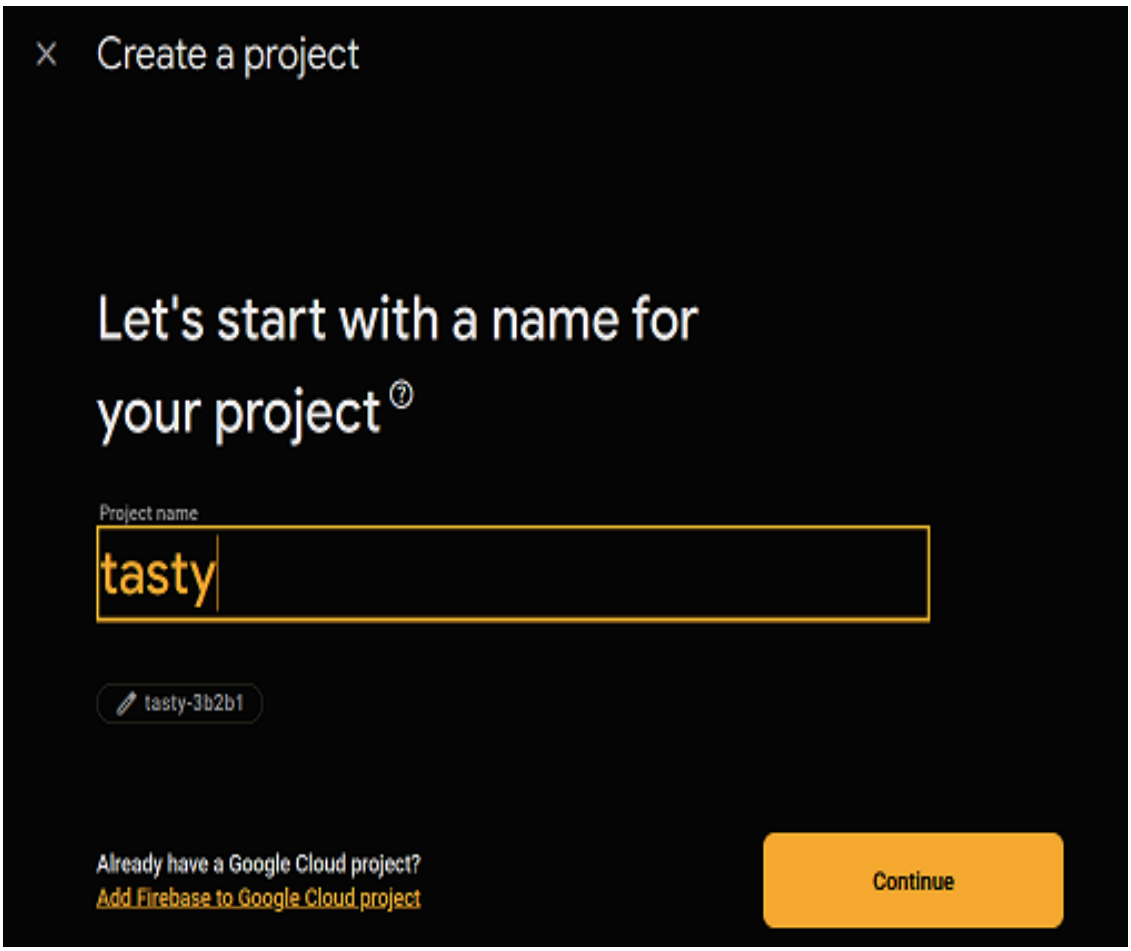
MAD & PWA Lab Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	49
Name	Ansh Sarfare
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	

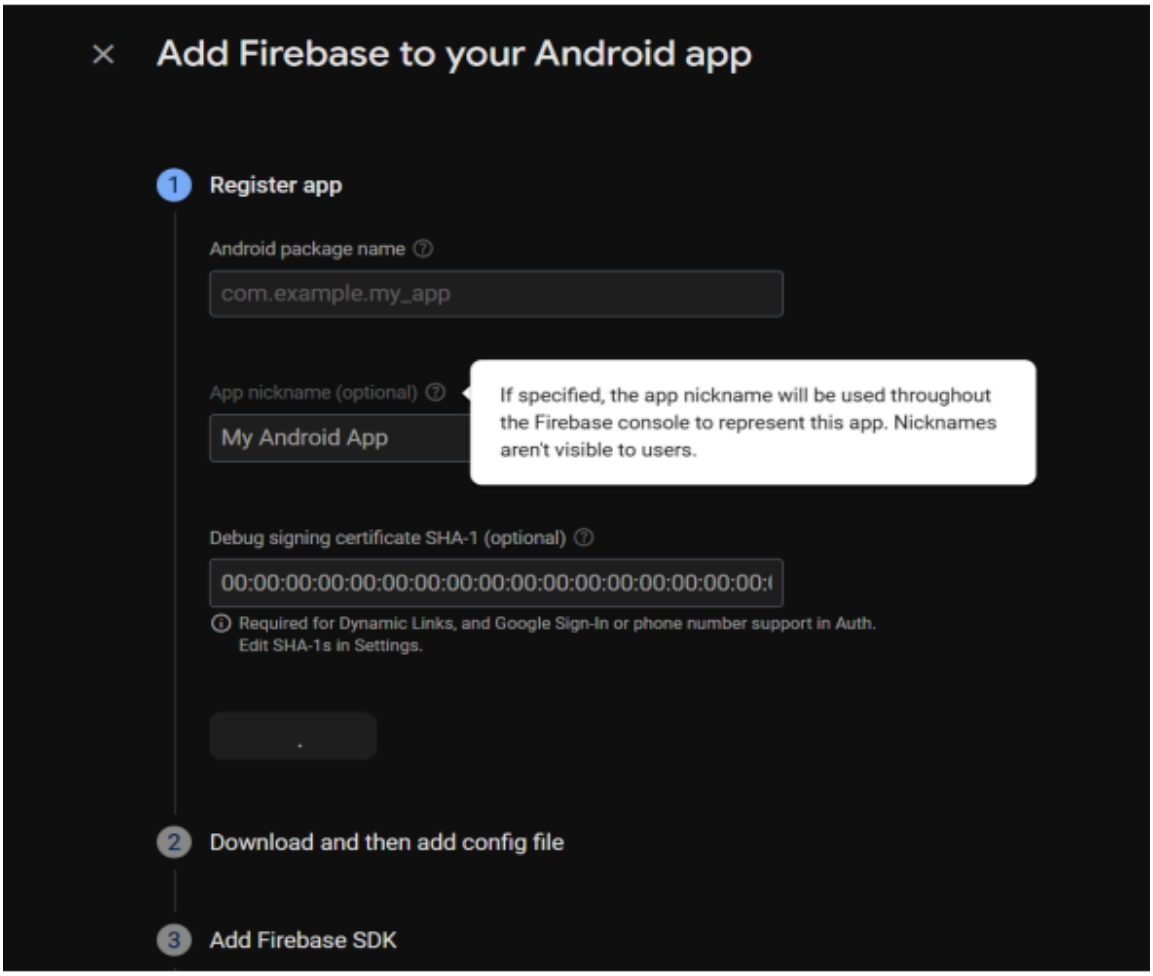
Project Title: Tasty app **Roll No. 49**

Aim: To Connect Flutter UI with Firebase Database.

Step-1:
First, log in with your Google account to manage your Firebase projects. From within the Firebase dashboard, select the Create new project button and give it a name



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



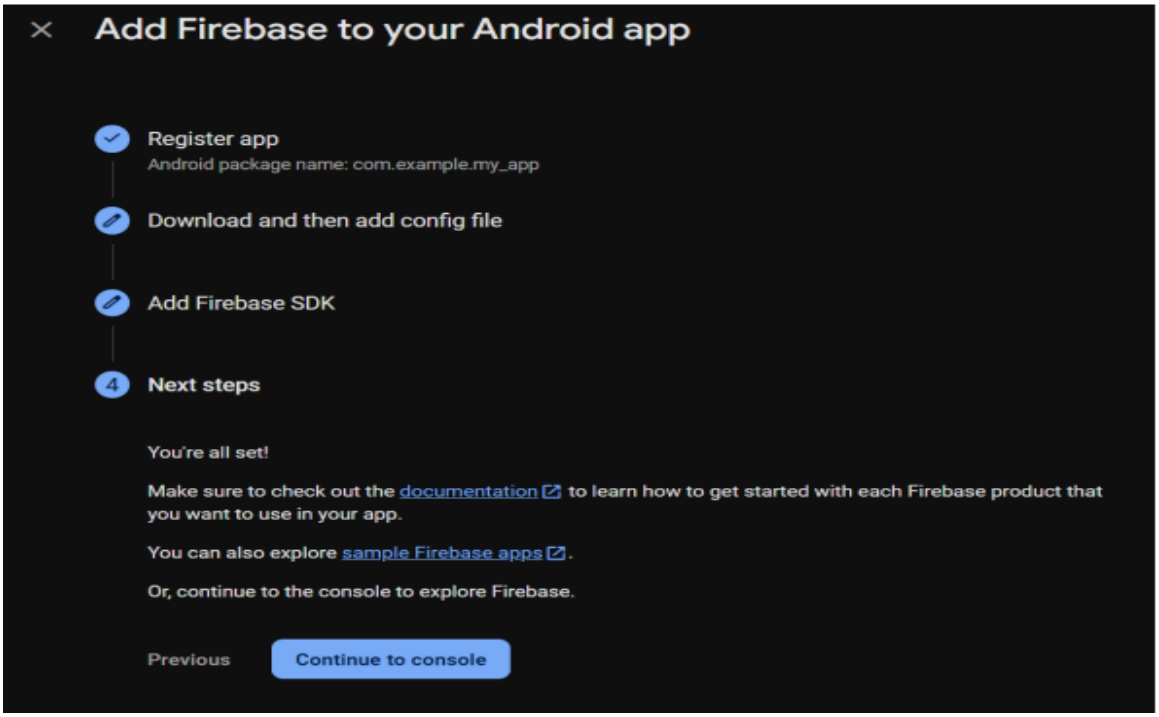
The most important thing here is to match up the Android package name that you choose here with the one inside of our application.

Downloading the Config File

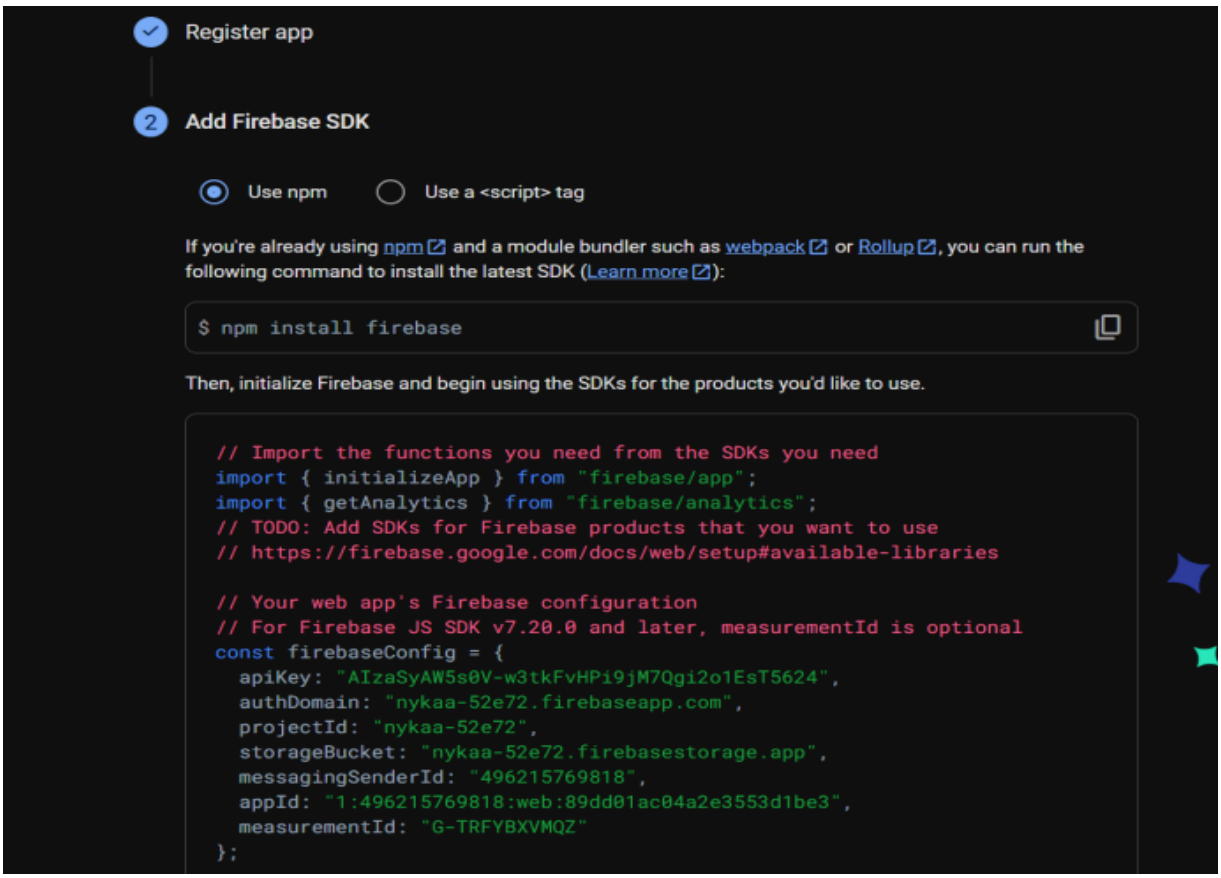
The next step is to add the Firebase configuration file into our Flutter project. This is important as it contains the API keys and other critical information for Firebase to use.

Select Download **google-services.json** from this page
In android/build.gradle adddependencies {
classpath 'com.google.gms.google-services:4.4.2'
}

In app/build.gradle adddependencies{
implementation platform('com.google.firebase:firebase-bom:33.9.0')
}



Now let us setup the app by adding a firebase_option.dart file in the lib directory similar to :



In pubspec.yaml:

```
dependencies:
flutter:
sdk: flutter
# The following adds the Cupertino Icons font to your application.
# Use with the CupertinoIcons class for iOS style icons.
cupertino_icons: ^1.0.8
firebase_core: ^3.11.0
firebase_auth: ^5.4.2
cloud_firestore: ^5.6.3
firebase_storage: ^12.4.2
```

Firebase connection Code:
(options is imported from the file: firebase_options.dart)

```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );
  CloudinaryService.init();
  runApp(
    MultiProvider(
      providers: [
        ChangeNotifierProvider(create: (context) => CartProvider()),
      ],
      child: MyApp(),
    ),
  );
}
```

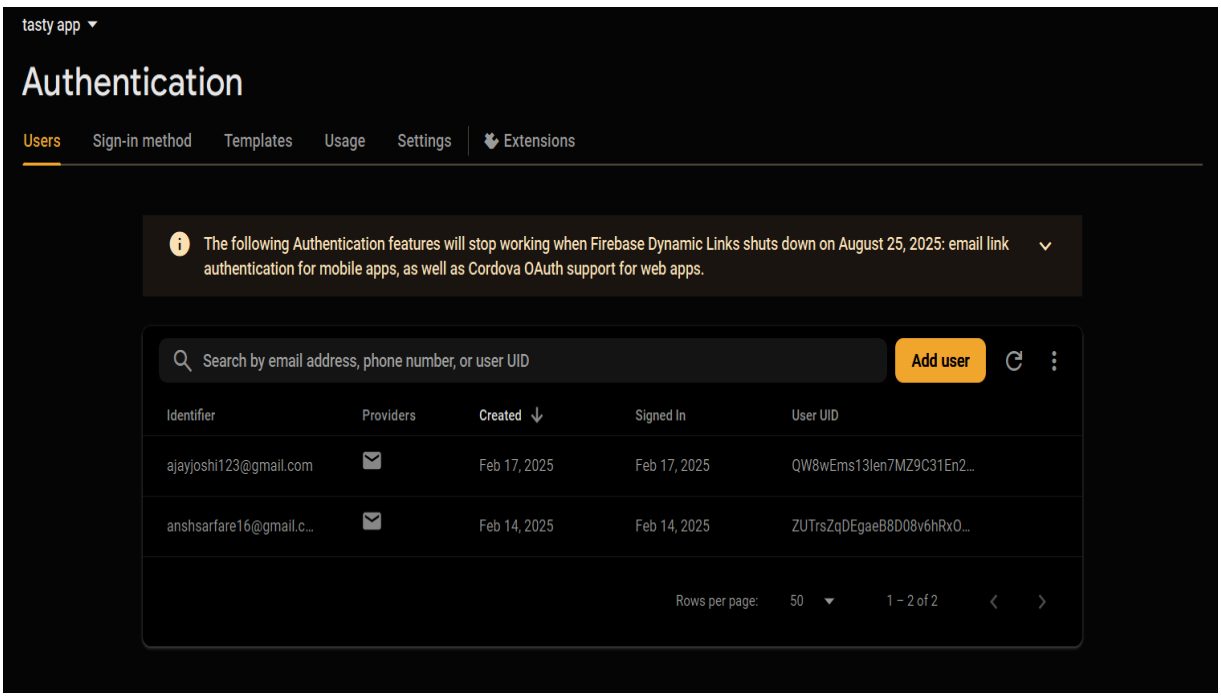
Authentication code:

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:cloud_firestore/cloud_firestore.dart';

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

  Future<User?> registerWithEmailAndPassword(String email, String password, String
username) async {
```

```
  try {
    UserCredential credential = await _auth.createUserWithEmailAndPassword(
      email: email,
      password: password,
    );
    User? user = credential.user;

    if (user != null) {
      await _firestore.collection('users').doc(user.uid).set({
        'email': email,
        'username': username, // Add username field
        'likedRecipes': [],
        'savedRecipes': [],
      });
    }

    return user;
  } catch (e) {
    print(e.toString());
    return null;
  }
}

Future<User?> signInWithEmailAndPassword(String email, String password) async {
  try {
    UserCredential credential = await _auth.signInWithEmailAndPassword(
      email: email,
      password: password,
    );
    User? user = credential.user;

    if (user != null) {
      final userDoc = await _firestore.collection('users').doc(user.uid).get();
      if (!userDoc.exists) {
        await _firestore.collection('users').doc(user.uid).set({
          'email': email,
          'likedRecipes': [],
          'savedRecipes': [],
        });
      }
    }
    return user;
  } catch (e) {
    print(e.toString());
```

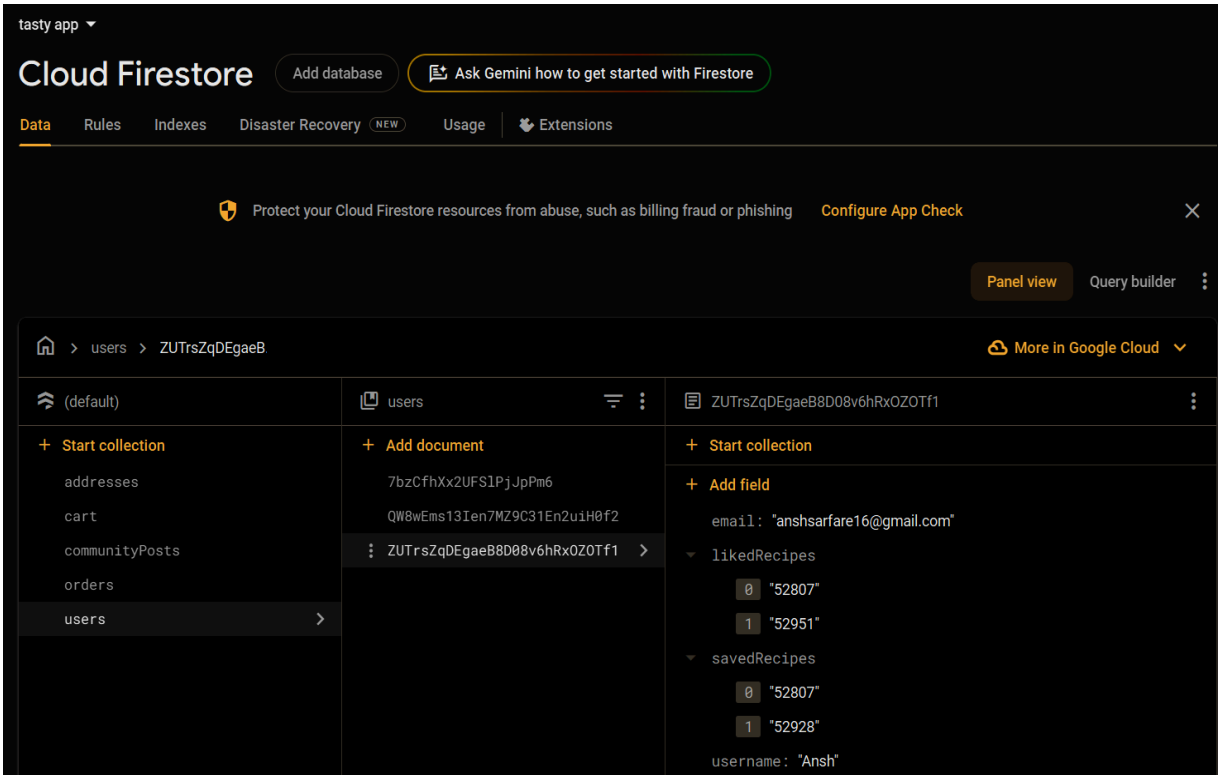
```
      return null;
    }
  }

  Stream<User?> authStateChanges() {
    return FirebaseAuth.instance.authStateChanges();
  }

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

  // Get the currently logged-in user
  User? getCurrentUser() {
    return _auth.currentUser;
  }
}
```

Authentication:



Project Title: Tasty app **Roll No. 49**

Firestore Database:



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.	49
Name	Ansh Sarfare
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	

Project Title: Tasty app **Roll No. 49**

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

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

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

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

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

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

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

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

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

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

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

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

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

App-like — They behave with the user as if they were native apps, in terms of interaction and navigation.
Updated — Information is always up-to-date thanks to the data update process offered by service workers.

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

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

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

Linkable — Easily shared via URL without complex installations.

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

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

Code:-

manifest.json:-

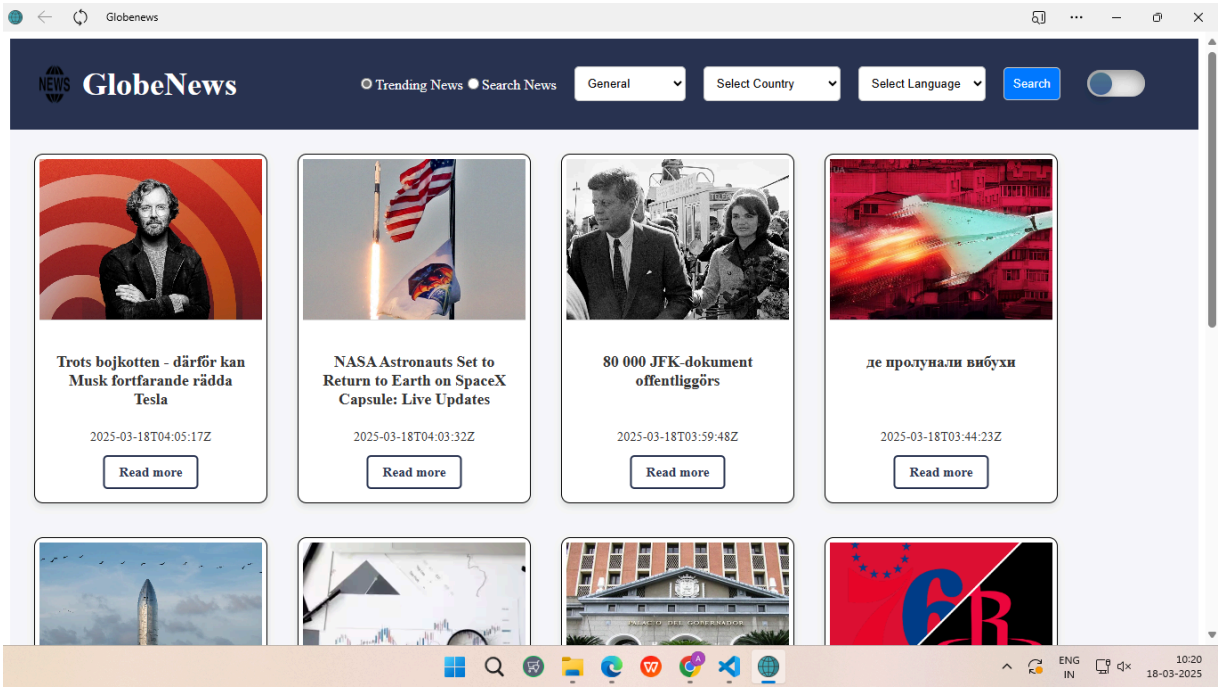
```
{
   "name": "GlobeNews",
   "short_name": "GNews",
   "start_url": "ani.html",
   "scope": "./",
   "icons": [
      {
        "src": "contract.png",
        "sizes": "192x192",
        "type": "image/png"
      },
      {
        "src": "contract.png",
        "sizes": "512x512",
        "type": "image/png"
      }
    ],
     "theme_color": "#ffd31d",
     "background_color": "#333",
   "display": "standalone"
 }
```

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

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

Project Title: Tasty app **Roll No. 49**

Output:



Conclusion:-

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

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

Install nodejs
https://nodejs.org/en/download/
In cmd
npm -v
npm install -g browser-sync
`npm install -g live-server`
Open vs code
Select proper folder where all files are available
Install extensions node js
In vs code terminal type following commands
node -v
Npx serve .
Reference https://developer.mozilla.org/en-US/docs/Web/Manifest
https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps

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.	49
Name	Ansh Sarfare
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

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

What can we do with Service Workers?

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

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

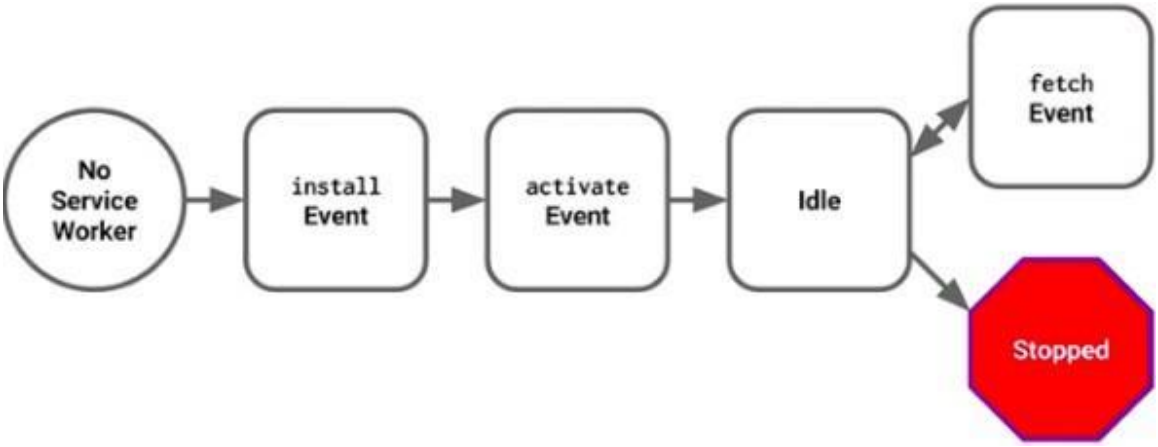
- You can manage **Push Notifications**
  You can manage push notifications with Service Worker and show any information message to the user.
- You can **Continue**
  Although Internet connection is broken, you can start any process with Background Sync of Service Worker.

What can't we do with Service Workers?

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

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

Service Worker Cycle



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

- Registration
- Installation
- Activation

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

main.js

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

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

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

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

main.js

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

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

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

main.js

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

Installation

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

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

service-worker.js

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

Activation

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

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

service-worker.js

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

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

Code:

Index.html

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

main.jsx

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

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

        // Check if the service worker is active
        if (registration.active) {
          console.log('[Service Worker] Active');
```

```
      }

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

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

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

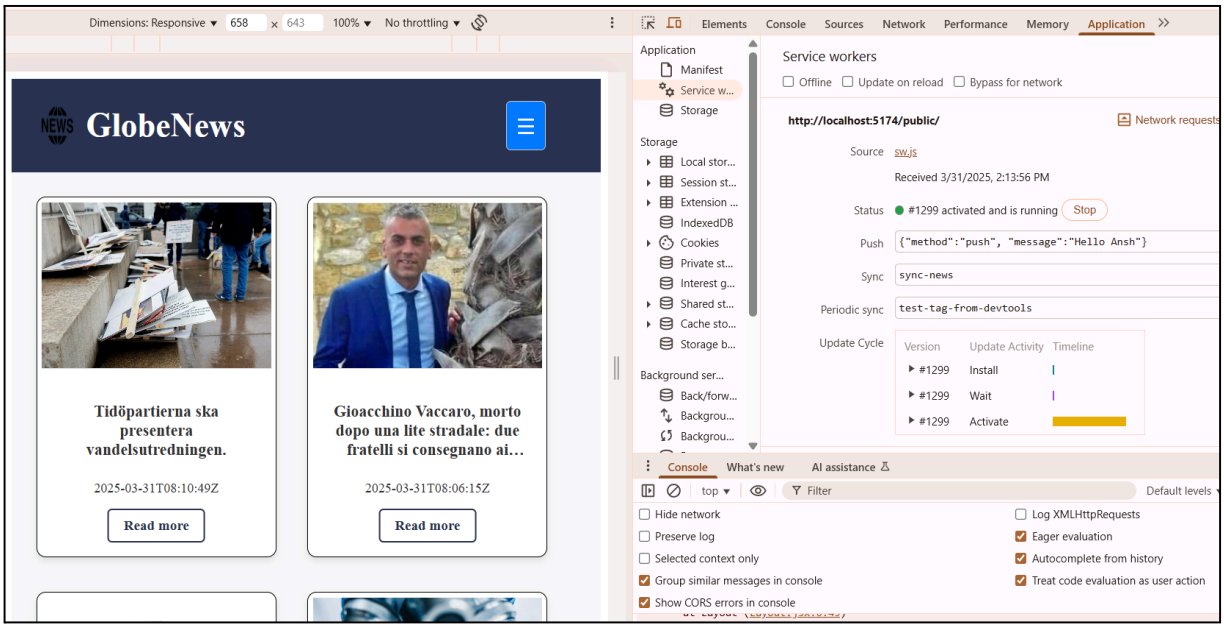
sw.js

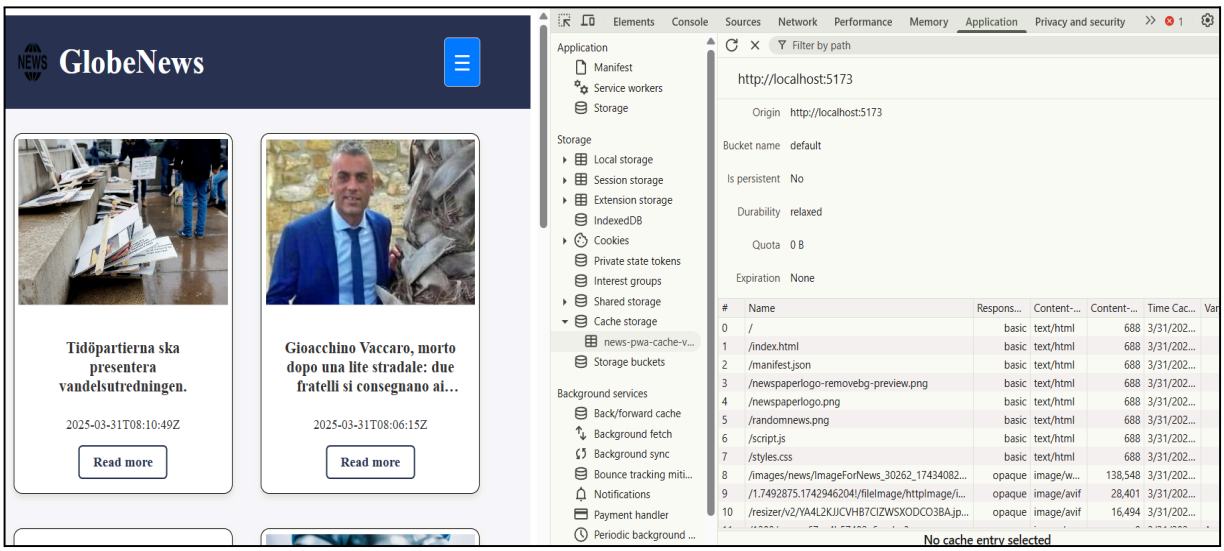
```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import App from './App.jsx'
import './index.css'

createRoot(document.getElementById('root')).render(
  <StrictMode>
```

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

Project Title: Tasty app **Roll No. 49**

Output:





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.	49
Name	Ansh Sarfare
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

Project Title: Tasty app **Roll No. 49**

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.

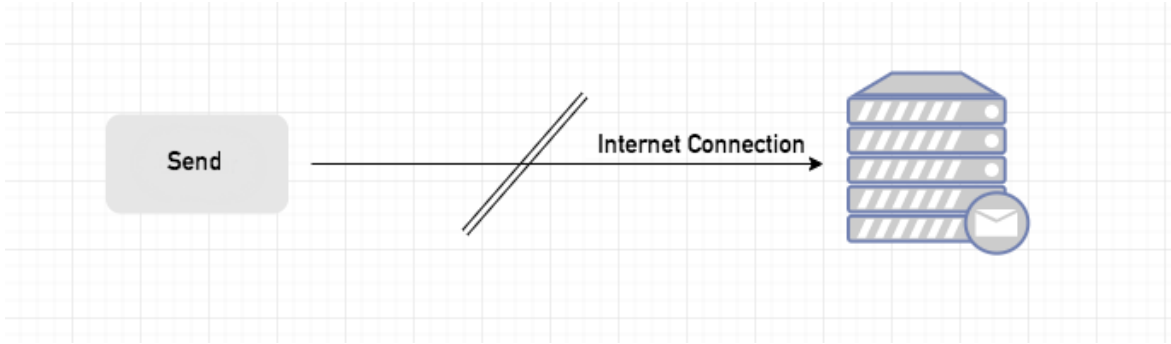
```
self.addEventListener("fetch", function (event) {
    const req = event.request;
    const url = new URL(req.url);

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

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

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

Sync Event

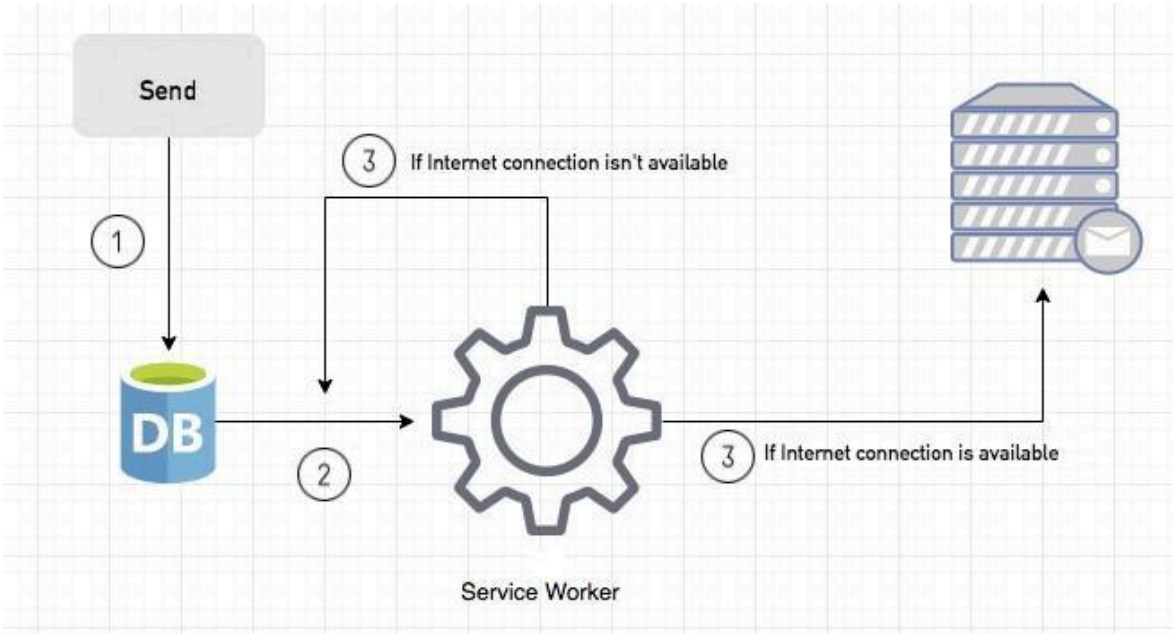
Background Sync is a Web API that is used to delay a process until the Internet connection is stable. We can adapt this definition to the real world; there is an e-mail client application that works on the browser and we want to send an email with this tool. Internet connection is broken while we are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

Here is a job for the Background Sync.

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



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



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

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

Event Listener for Background Sync Registration

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

Event Listener for sw.js

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

Push Event

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

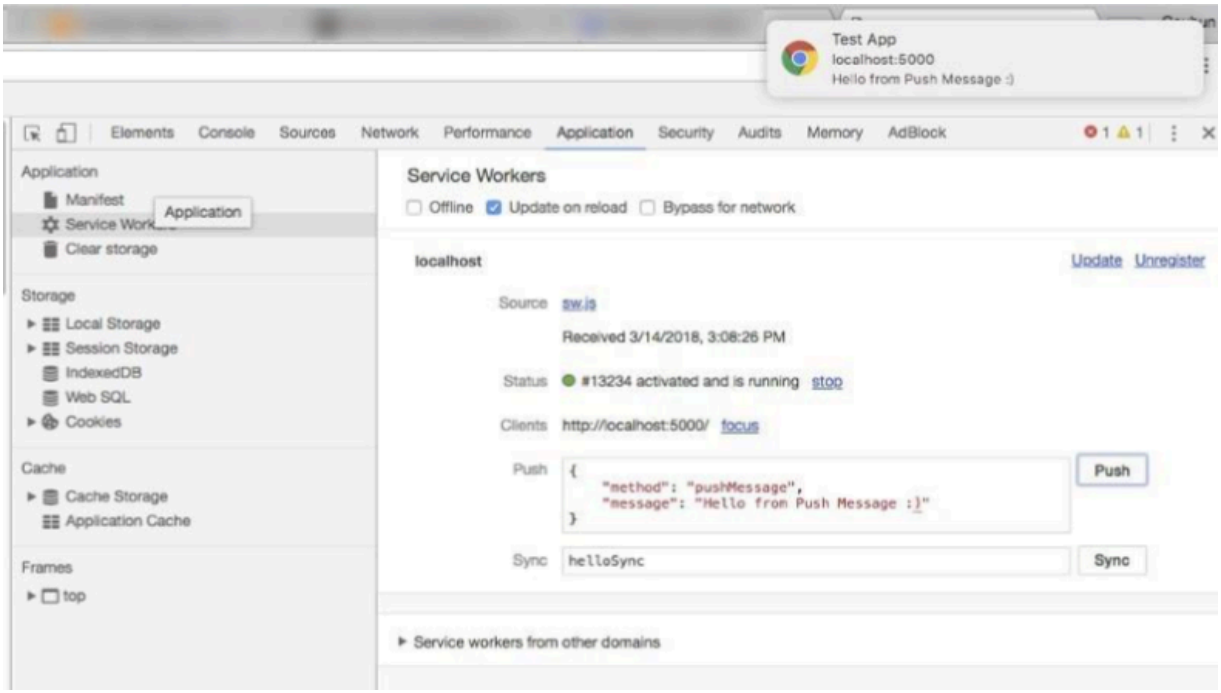
We can check in the following example.
“Notification.requestPermission();” is the necessary line to show notification to the user. If you don’t want to show any notification, you don’t need this line.

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

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

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

Project Title: Tasty app **Roll No. 49**



Code:

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

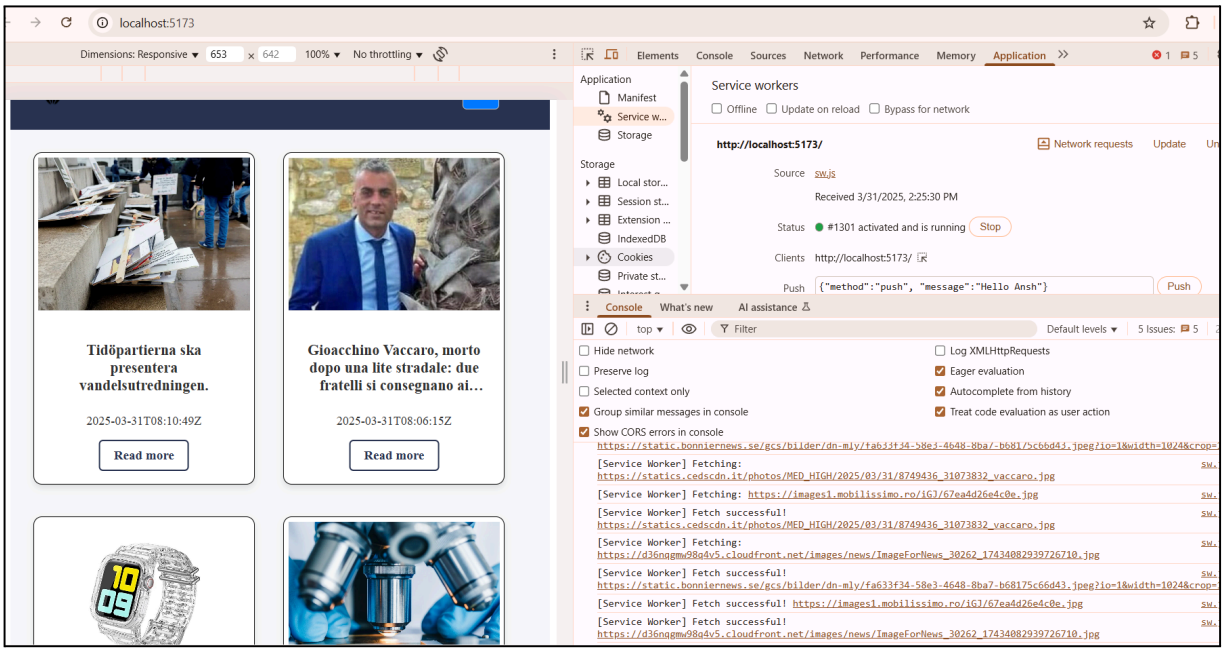
```
// Install event: Cache static assets
self.addEventListener('install', (event) => {
  console.log('[Service Worker] Installing...');
  event.waitUntil(
    caches.open(CACHE_NAME).then((cache) => {
      console.log('[Service Worker] Caching assets...');
      return cache.addAll(urlsToCache);
 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

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

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

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

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

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

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

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

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

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

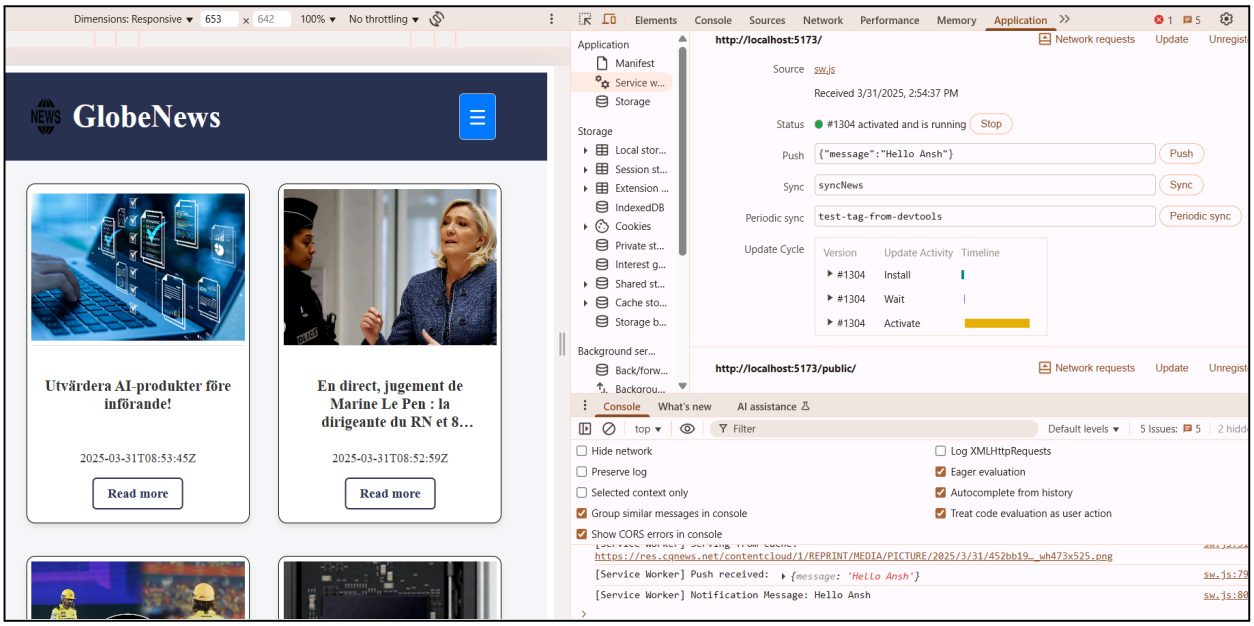
Output:

Fetch event

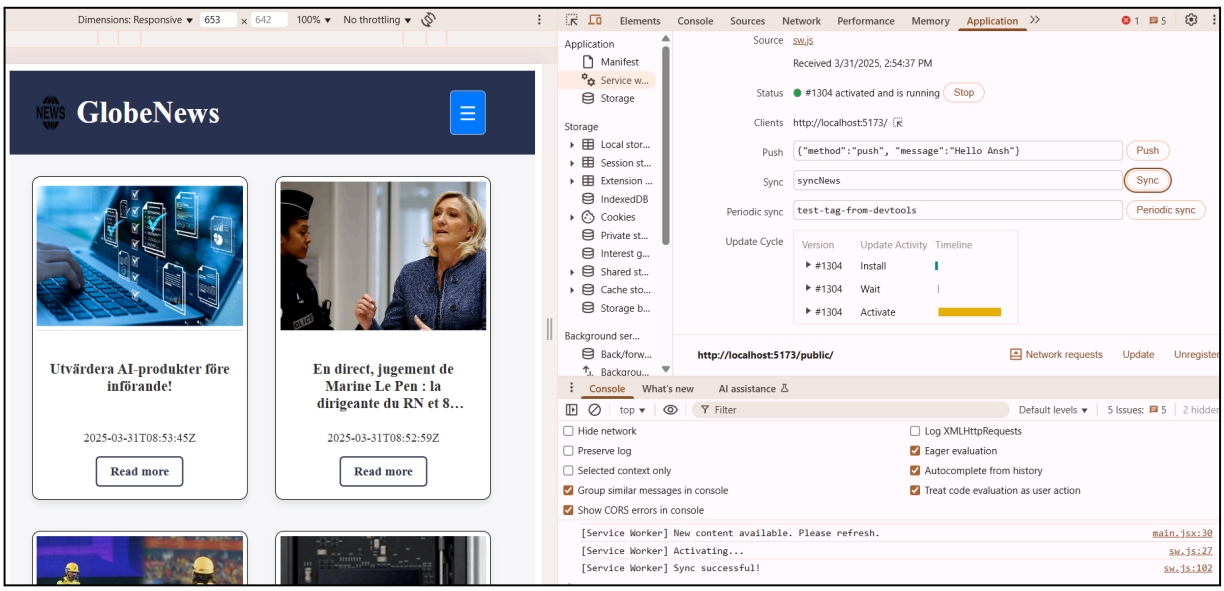


Project Title: Tasty app **Roll No. 49**

Push event



Sync event



MAD & PWA Lab
Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	49
Name	Ansh Sarfare
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

Aim:

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

Theory:

GitHub Pages

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

GitHub Pages provides the following key features:

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

Reasons for favoring this over Firebase:

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

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

Pros

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

Cons

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

Firebase

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

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

Reasons for favoring over GitHub Pages:

1. Realtime backend made easy
2. Fast and responsive

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

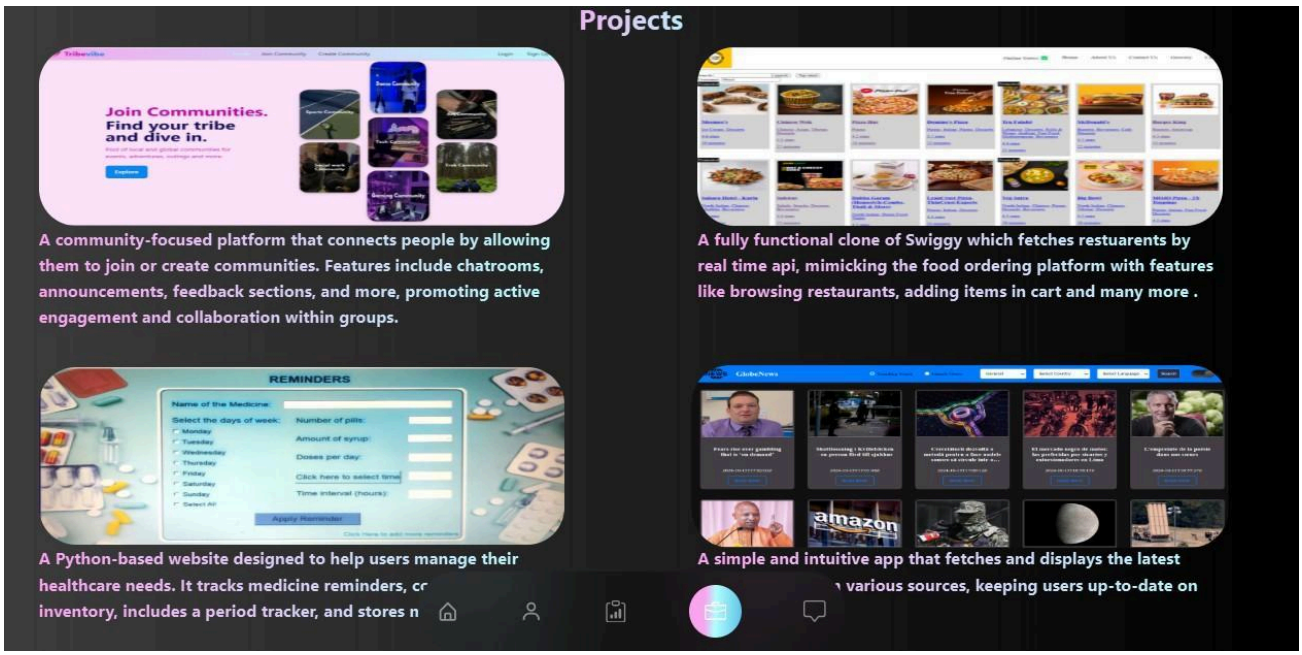
Pros

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

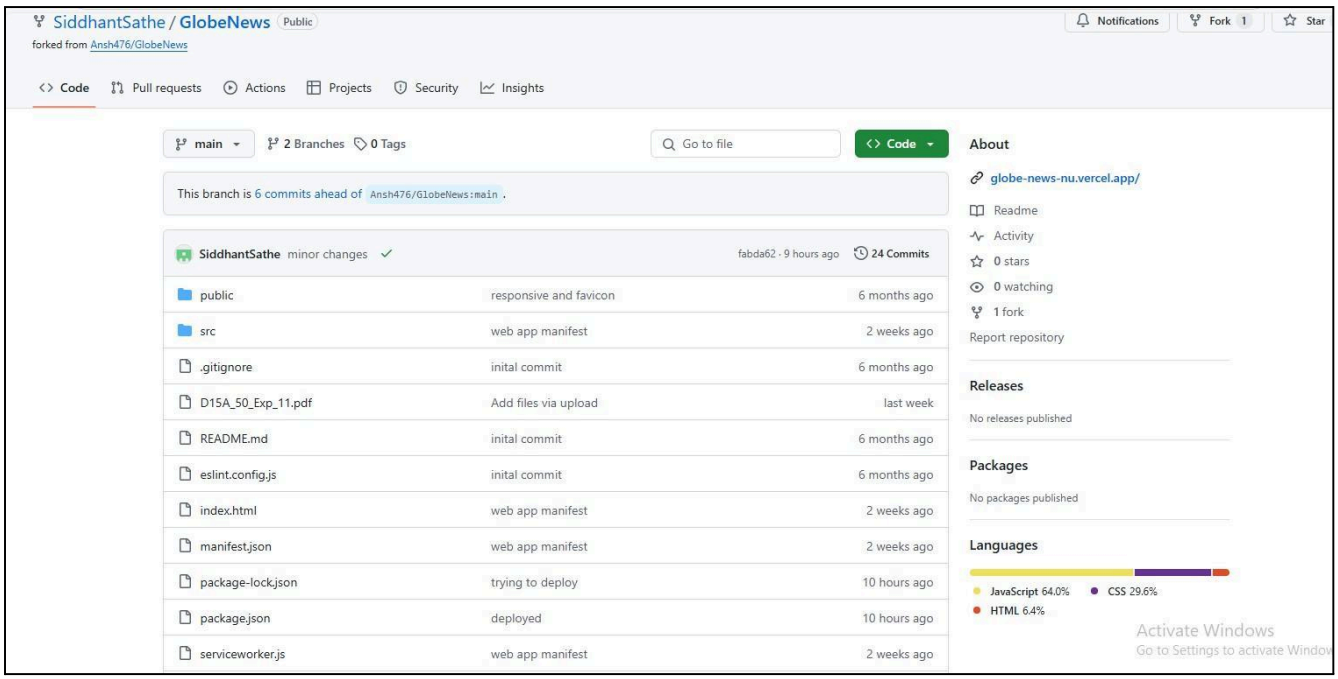
Cons

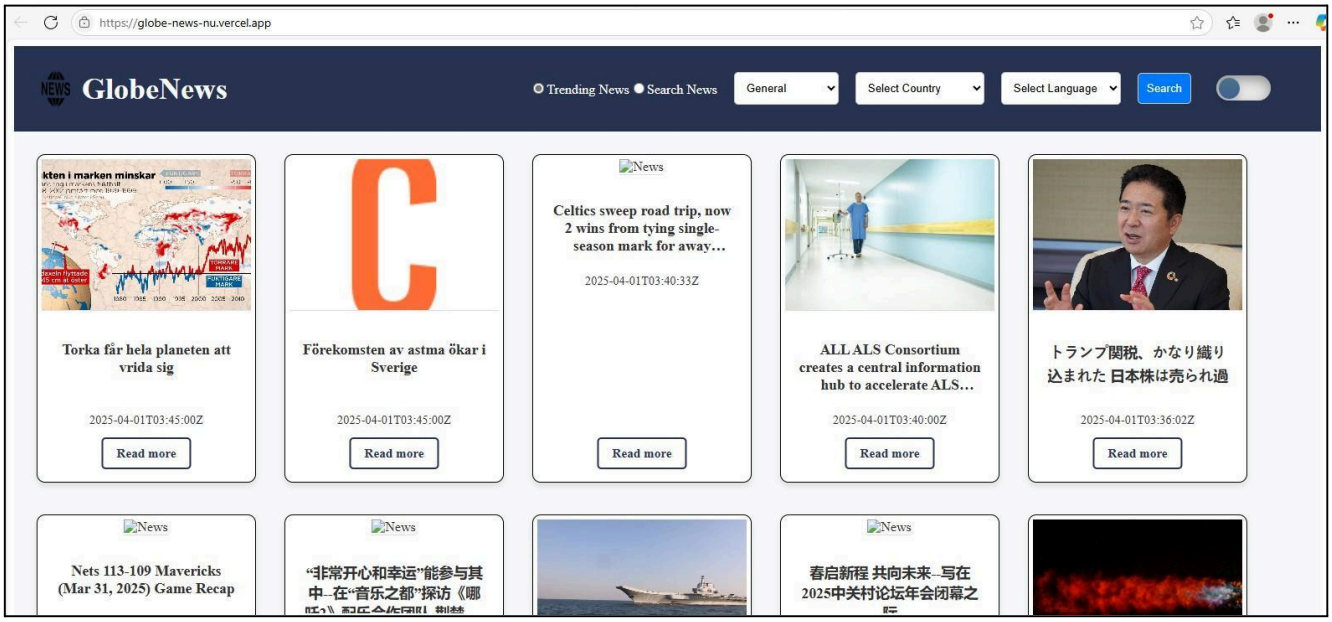
1. Only 10 GB of data transfer is allowed per month. But this is not really a big problem, if you use a CDN or AMP.
2. Command-line interface only.
3. No in-built support for any static site generator.

Github Screenshot:



PWA Website:





Github Website: [GitHub - Ansh476/GlobeNews]

Hosted Website: globe-news-nu.vercel.app/

MAD & PWA Lab
Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	49
Name	Ansh Sarfare
Class	D15A/D15B
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	

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

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

Google Lighthouse :

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

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

Key Features and Audit Metrics

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

1. **Performance:** This score is an aggregation of how the page fared in aspects such as (but not limited to) loading speed, time taken for loading for basic frame(s), displaying

meaningful content to the user, etc. To a layman, this score is indicative of how decently the site performs, with a score of 100 meaning that you figure in the 98th percentile, 50 meaning that you figure in the 75th percentile and so on.

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

points are based on the Baseline PWA checklist laid down by Google which includes

Service Worker implementation(s), viewport handling, offline functionality, performance in script-disabled environments, etc.

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

i.e. if all possible elements of the page are not screen-reader friendly (HTML5 introduced features that would make pages easy to interpret for screen readers used by visually

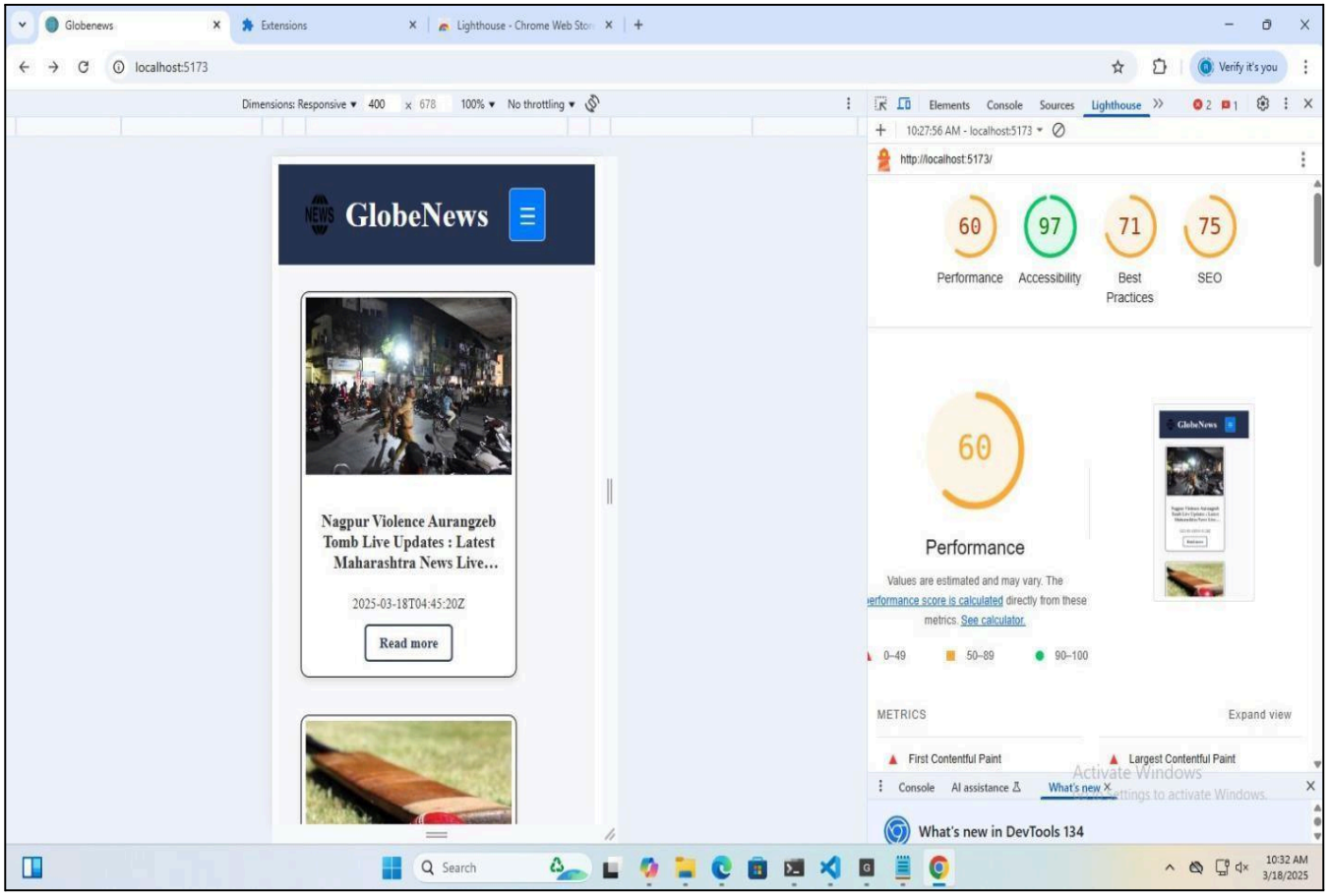
challenged people like tag names, tags such as <section>, <article>, etc.), you get a 0 on that score. The aggregate of these scores is your Accessibility metric score.

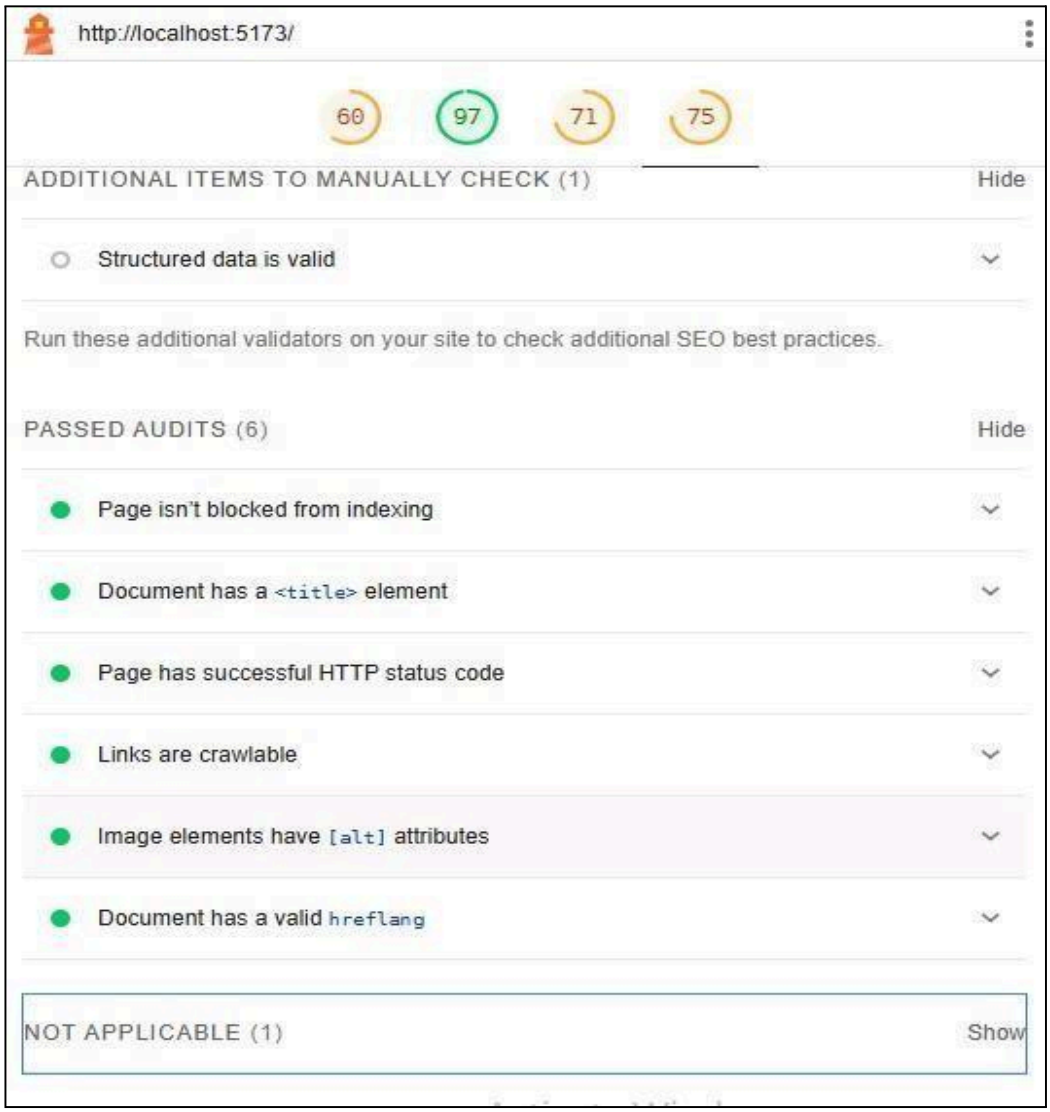
4. **Best Practices:** As any developer would know, there are a number of practices that have been deemed 'best' based on empirical data. This metric is an aggregation of many such points, including but not limited to:Use of HTTPS

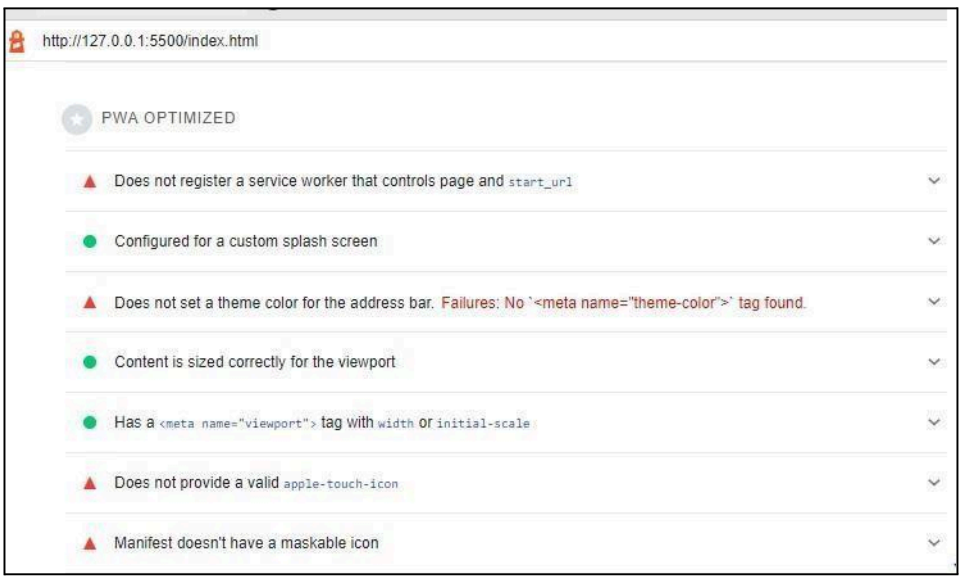
Avoiding the use of deprecated code elements like tags, directives, libraries, etc. Password input with paste-into disabled

Geo-Location and cookie usage alerts on load, etc

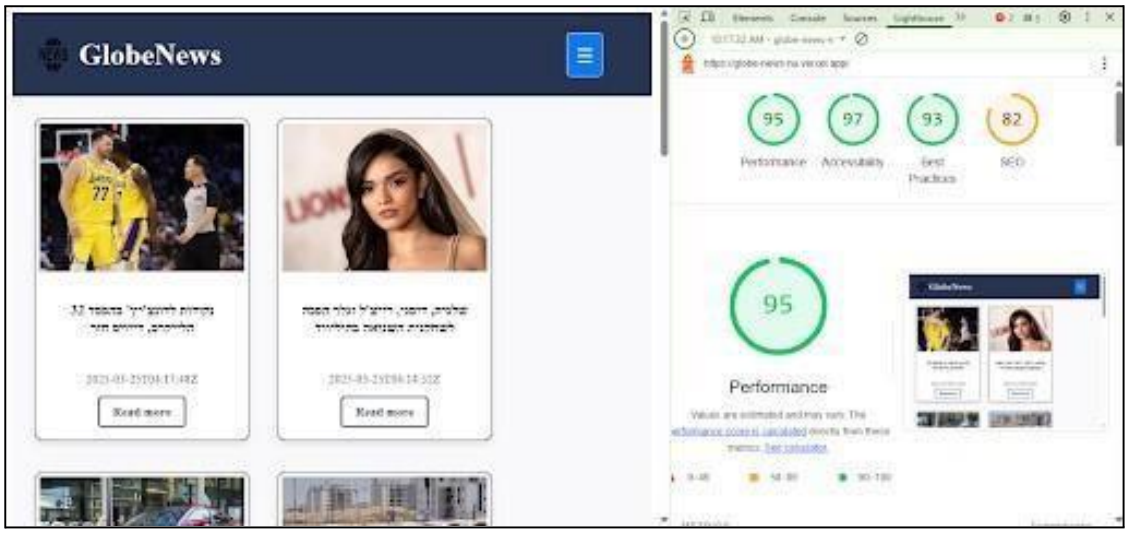
Before:


Globenews
Extensions
Lighthouse - Chrome Web Stor
localhost:5173
Verify it's you
Dimensions: Responsive 400 x 678 100% No throttling
Elements Console Sources Lighthouse
10:27:56 AM - localhost:5173
http://localhost:5173/
GlobeNews
Nagpur Violence Aurangzeb Tomb Live Updates : Latest Maharashtra News Live...
2025-03-18T04:45:20Z
Read more
60
97
71
75
Performance
Accessibility
Best Practices
SEO
60
Performance
Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.
0–49
50–89
90–100
METRICS
Expand view
First Contentful Paint
Largest Contentful Paint
Console
AI assistance
What's new
What's new in DevTools 134
Activate Windows
Search
10:32 AM
3/18/2025


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


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

After:



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