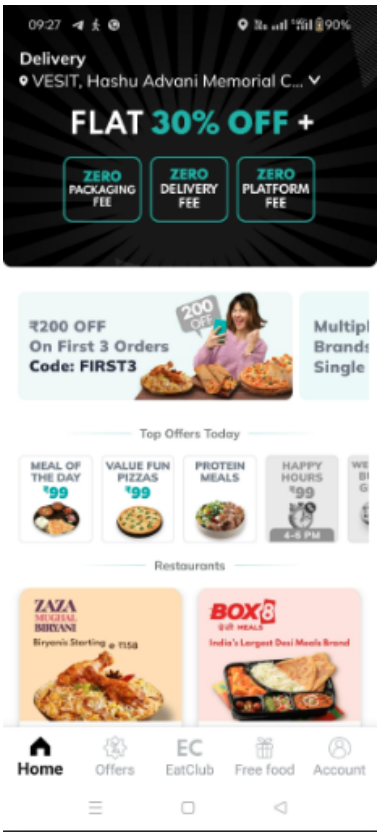
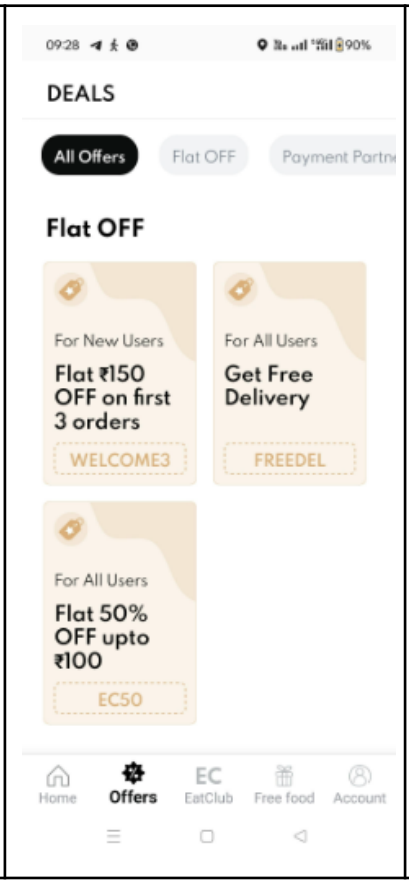
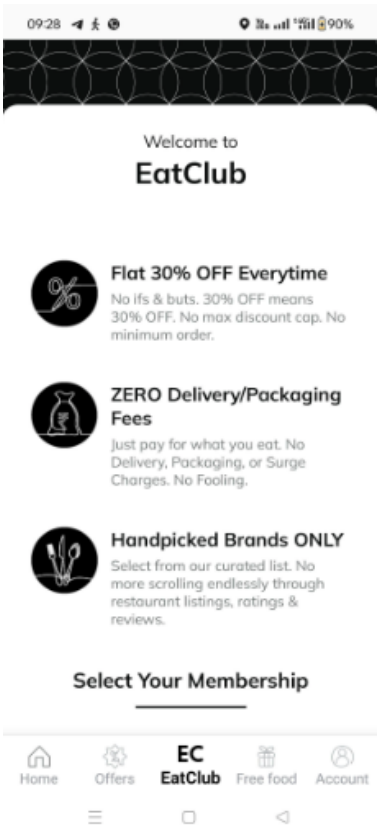
MPL Prerequisites

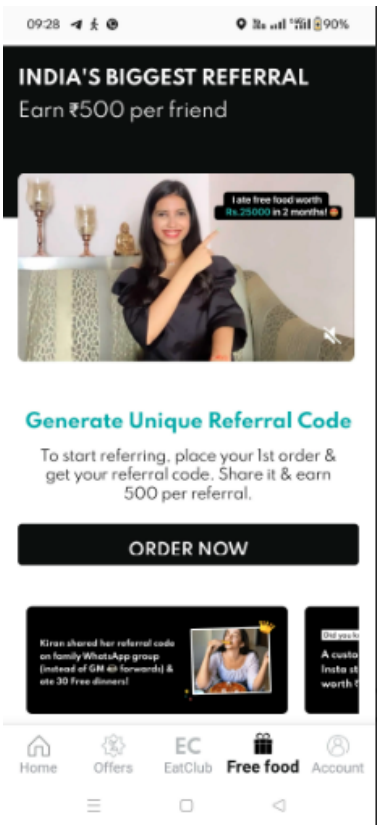
Aim: Selecting features for application development, the features should comprise of:

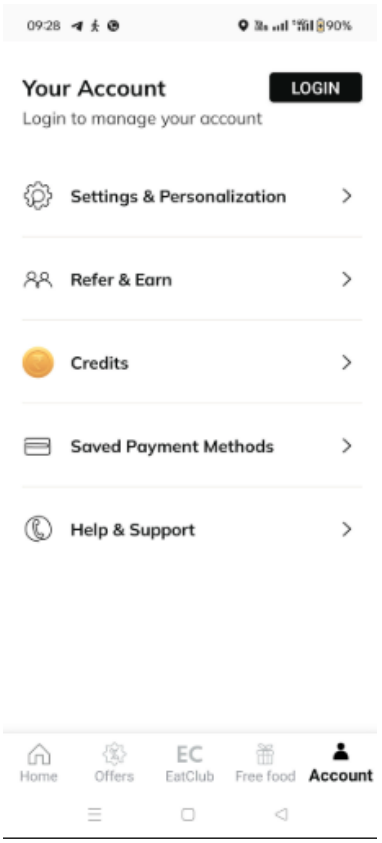
- Common widgets
- Should include icons, images, charts etc.
- Should have an interactive Form
- Should apply navigation, routing and gestures
- Should connect with FireBase database

No.	Screenshot	Features
1.		Home Page: This is also the landing page of the application. Here, we can select different brands to order our food from.

2.		Offers Page : Various coupons card for customers

3.		Membership Page: Displays 3 month or 6 month membership. Customers can select according to their needs

4.		Free food: This page describes a referral program

5.		Account: The account section typically allows users to manage their personal information, preferences, and settings

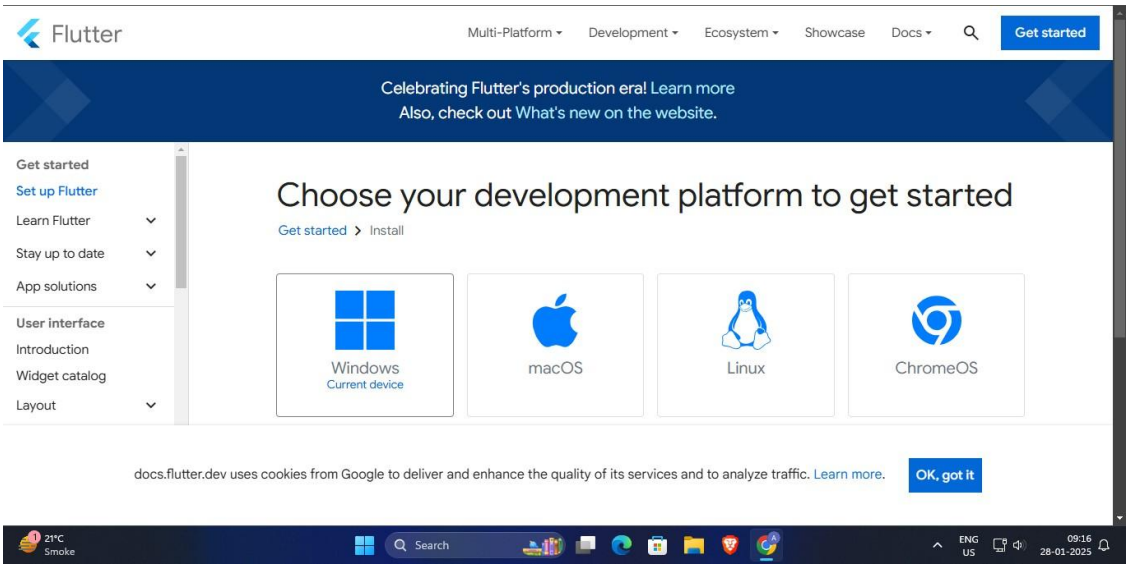
Riya Varyani 61 D15A

EXPERIMENT 1

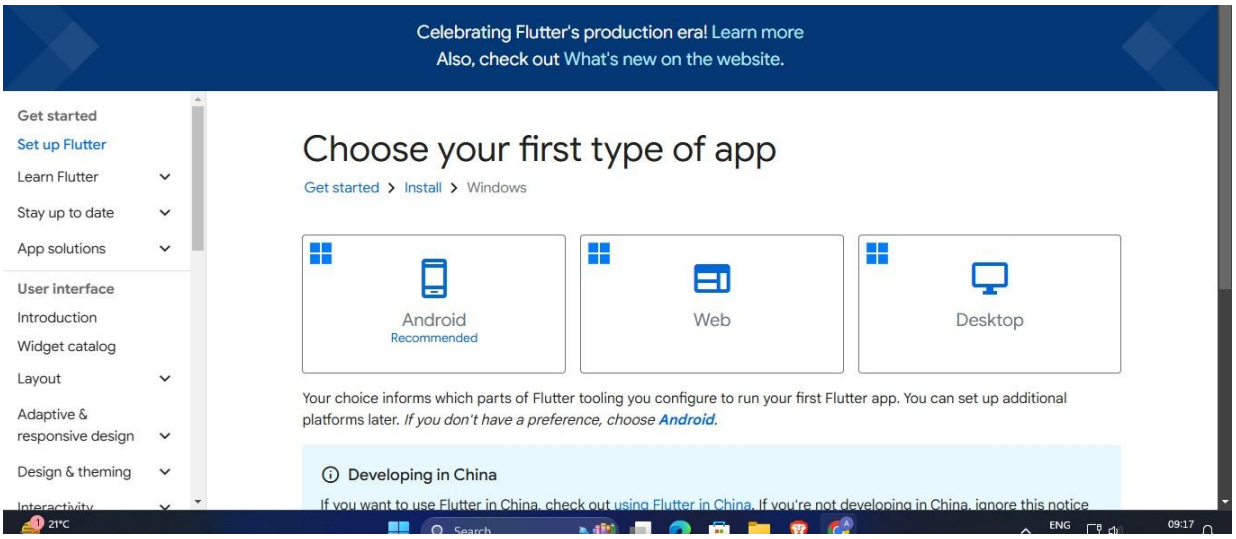
Aim : Installation and Configuration of Flutter Environment.

Steps :

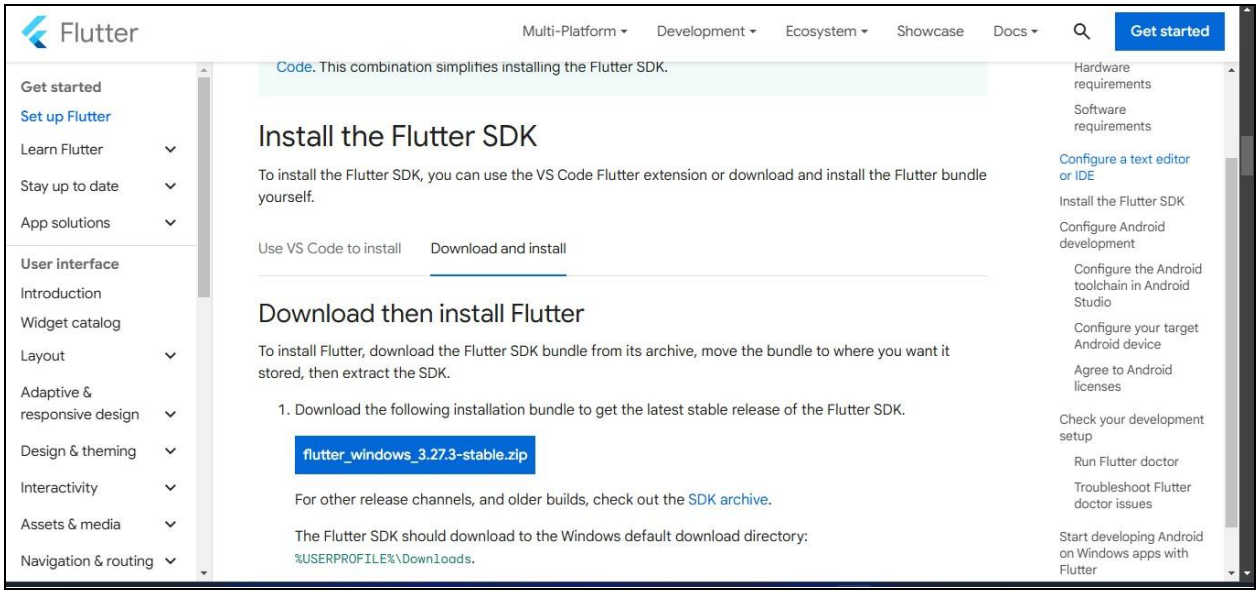
Step 1: Download the installation bundle of the Flutter Software Development Kit for windows. To download Flutter SDK, Go to its official website https://docs.flutter.dev/get-started/install , you will get the following screen.



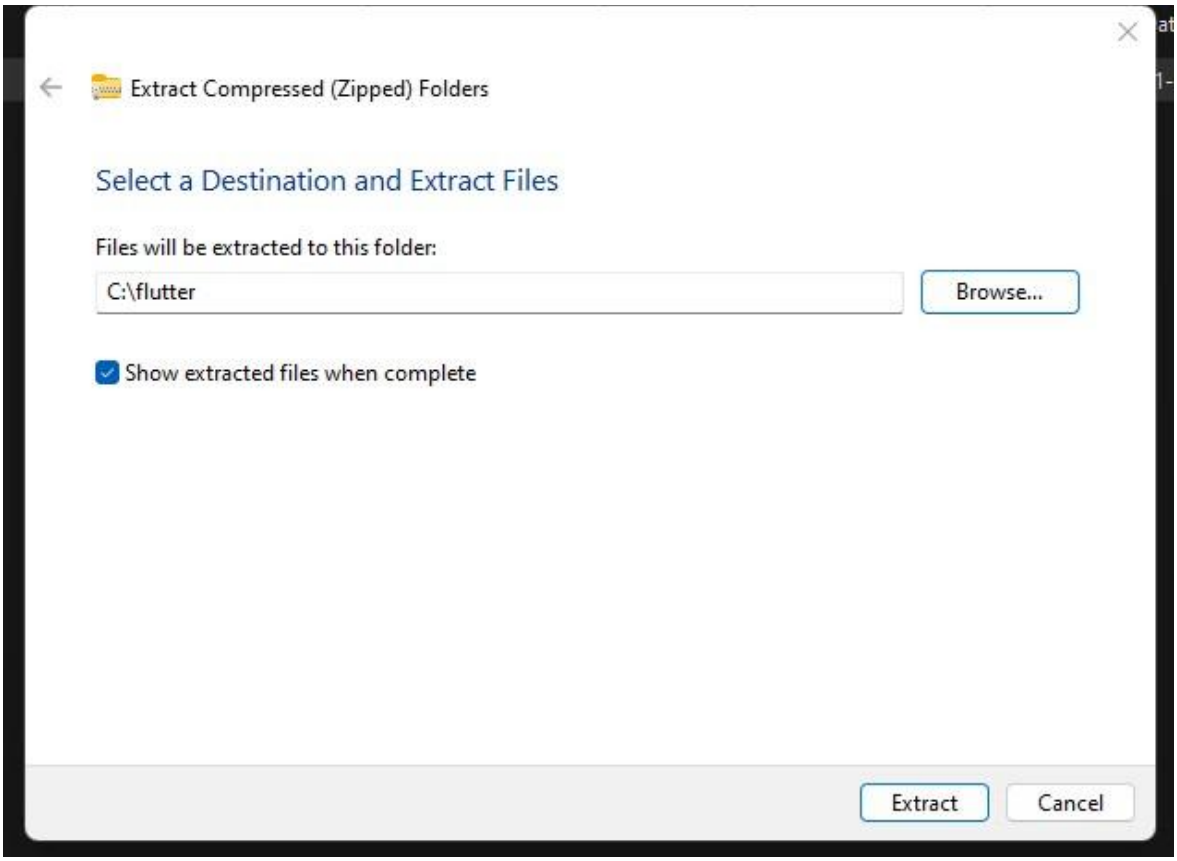
Step 2: Next, to download the latest Flutter SDK, click on the Windows icon and then select Android. Here, you will find system requirements and the download link for SDK.



Riya Varyani 61 D15A



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



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

```
Microsoft Windows [Version 10.0.26100.2894]
(c) Microsoft Corporation. All rights reserved.

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

Common commands:

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

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

Usage: flutter <command> [arguments]

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

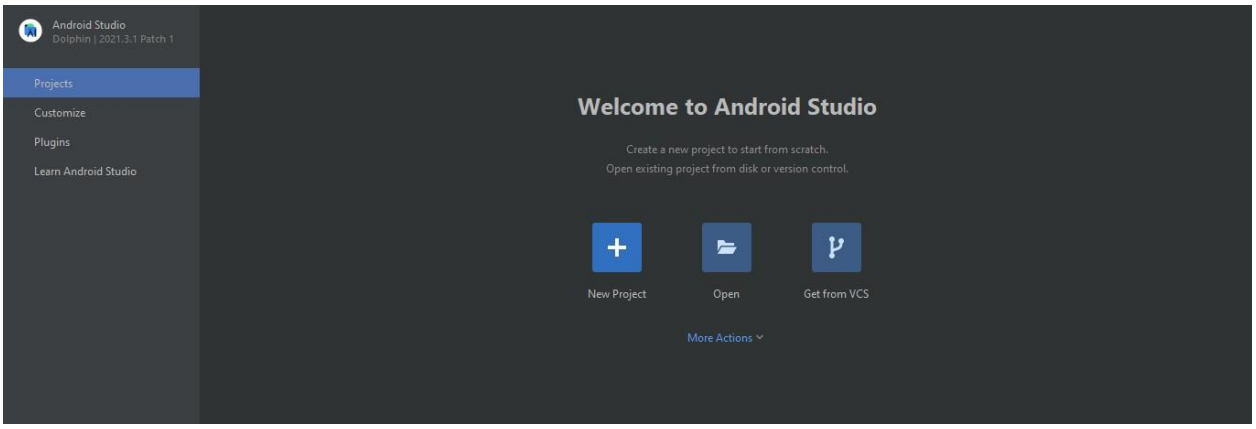
Step 5 : Now, run the $ flutter doctor command. This command checks for all the requirements of Flutter app development and displays a report of the status of your Flutter installation. Step 8: When you run the above command, it will analyze the system and show its report, as shown in the below image.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

```
C:\Users\varya>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.27.3, on Microsoft Windows [Version 10.0.26100.2894], locale en-IN)
[√] Windows Version (Installed version of Windows is version 10 or higher)
[√] Android toolchain - develop for Android devices (Android SDK version 35.0.1)
[√] Chrome - develop for the web
[√] Visual Studio - develop Windows apps (Visual Studio Community 2022 17.12.4)
[√] Android Studio (version 2024.2)
[√] VS Code (version 1.96.4)
[√] Connected device (3 available)
[√] Network resources

• No issues found!
```

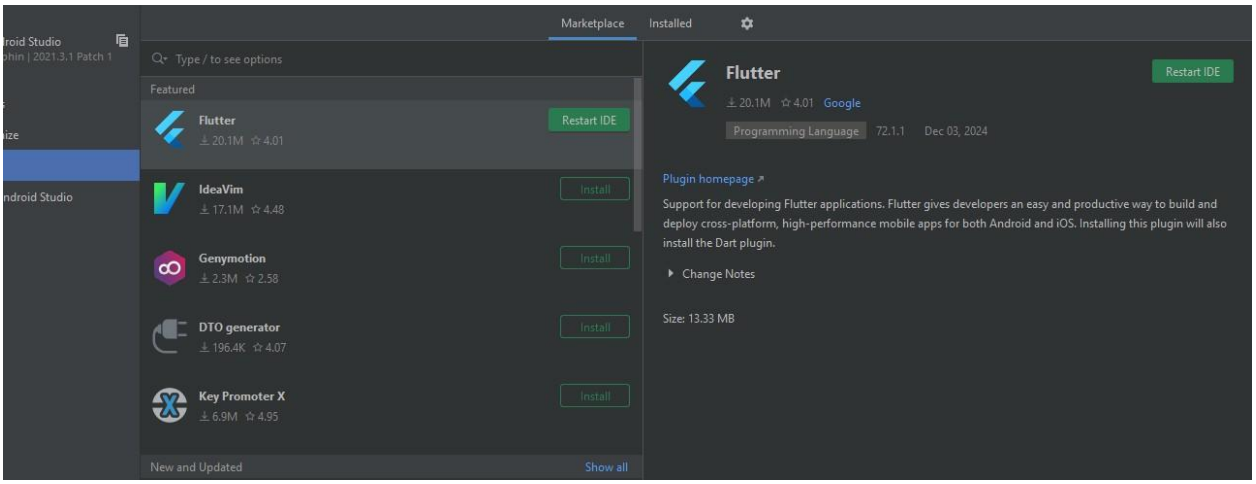
Step 6: Install the Android SDK. If the flutter doctor command does not find the Android SDK tool in your system, then you need first to install the Android Studio IDE. To install Android Studio IDE, do the following steps.

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

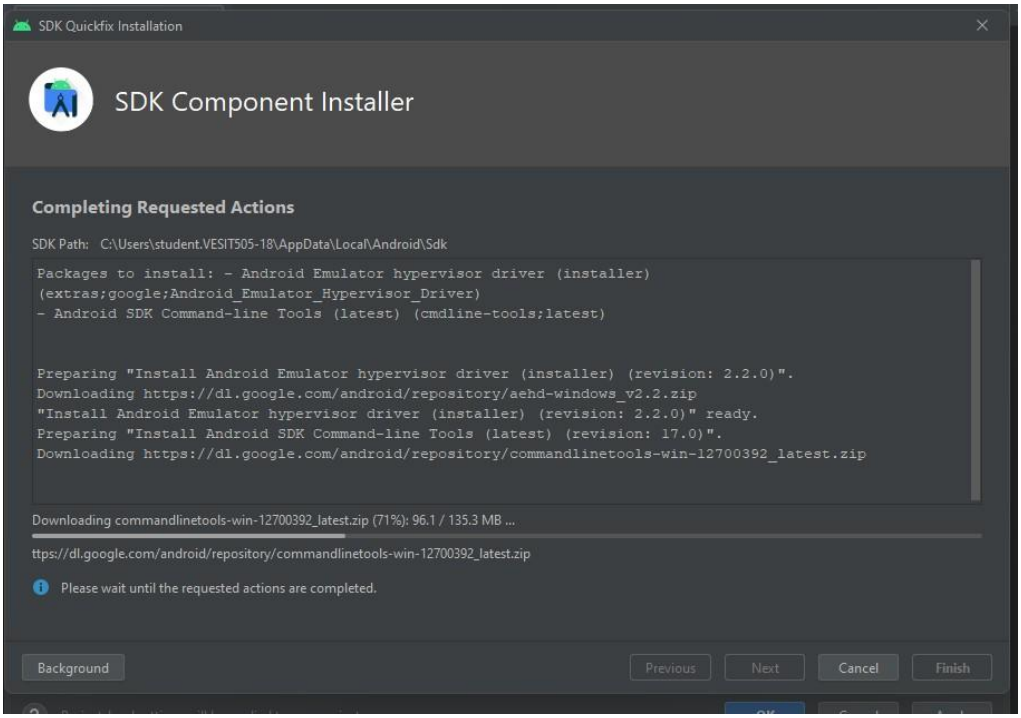
Riya Varyani 61 D15A



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

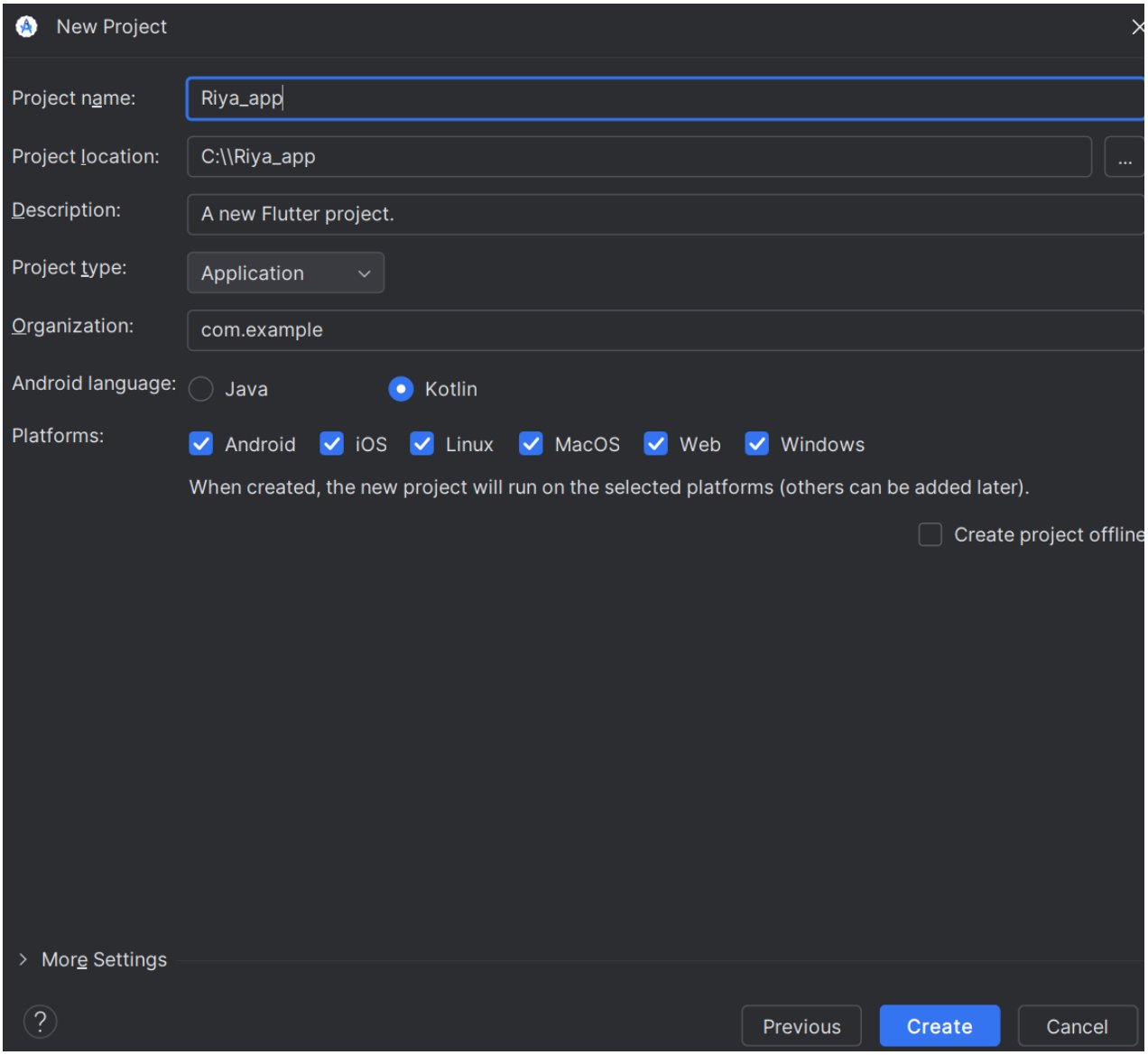


Riya Varyani 61 D15A



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

Riya Varyani 61 D15A



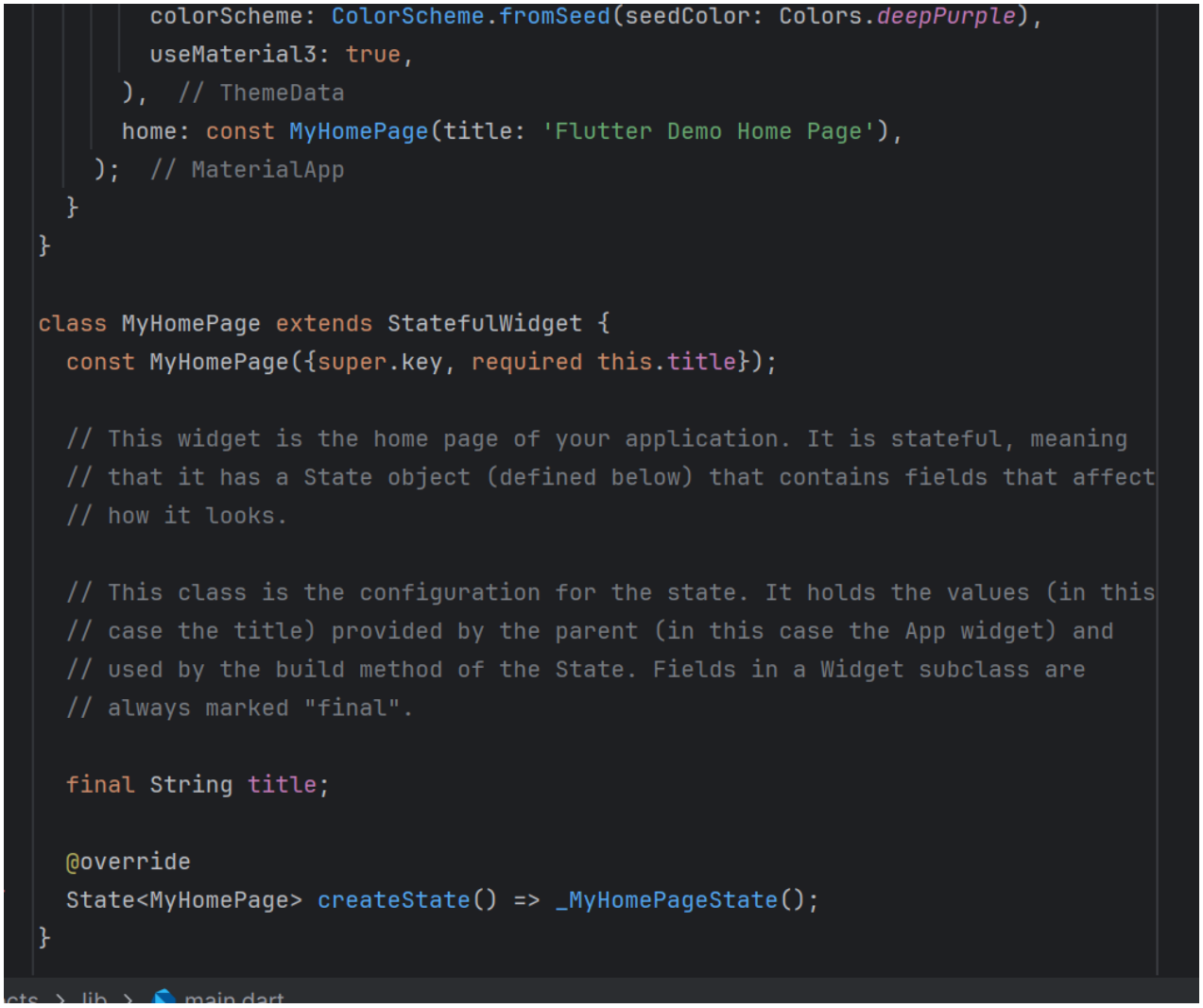
Creating project in visual studio code :

>flutter create myapp
>cd myapp
>flutter run

Riya Varyani 61 D15A

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

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

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

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        // This is the theme of your application.
        //
        // TRY THIS: Try running your application with "flutter run". You'll see
        // the application has a purple toolbar. Then, without quitting the app,
        // try changing the seedColor in the colorScheme below to Colors.green
        // and then invoke "hot reload" (save your changes or press the "hot
        // reload" button in a Flutter-supported IDE, or press "r" if you used
        // the command line to start the app).
        //
        // Notice that the counter didn't reset back to zero; the application
```

Riya Varyani 61 D15A



```
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),  // ThemeData
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    );  // MaterialApp
  }
}

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

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

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

  final String title;

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

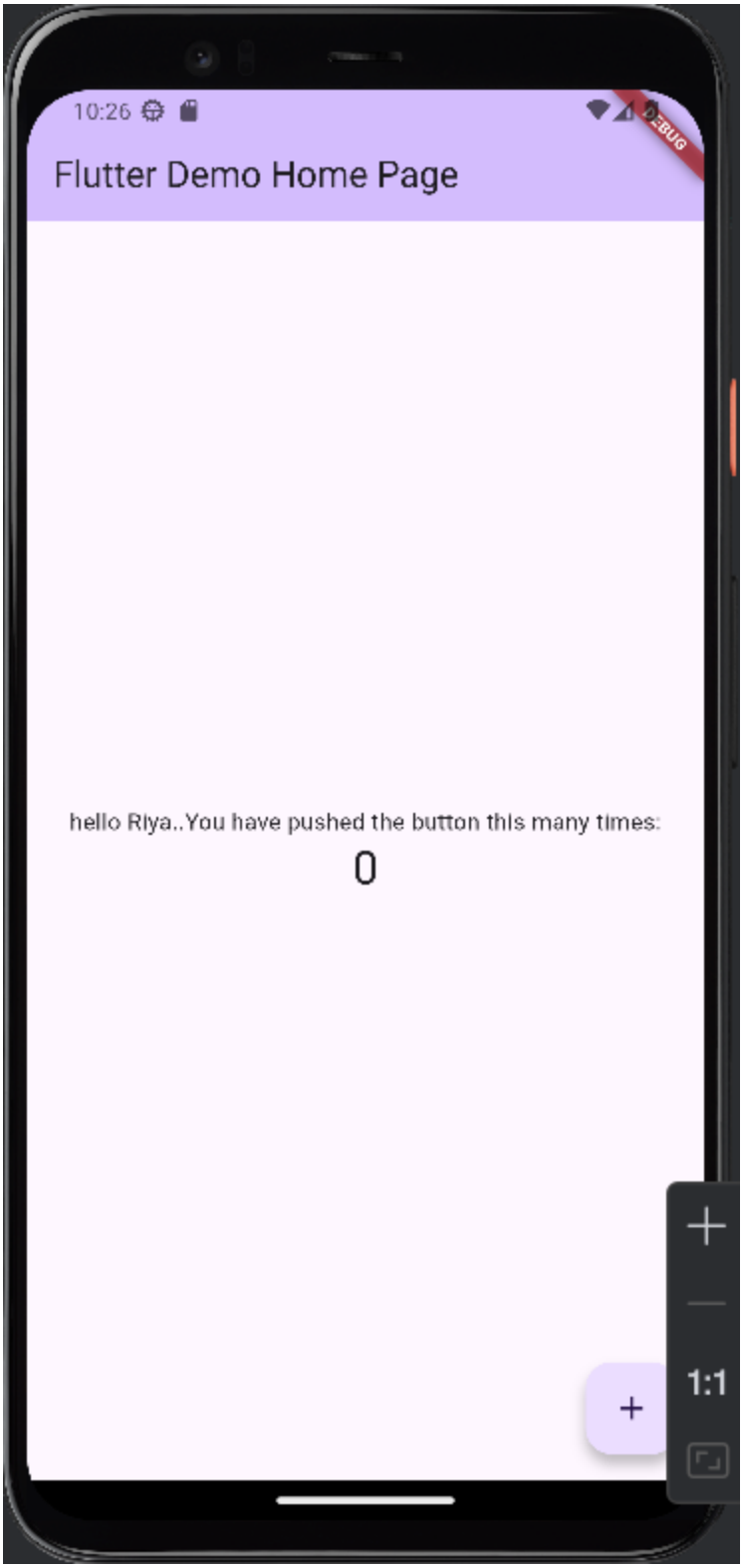
cts > lib > main.dart

Riya Varyani 61 D15A

```
            const Text(
              "hello Riya.."
              'You have pushed the button this many times:',
            ),  // Text
            Text(
              '$_counter',
              style: Theme.of(context).textTheme.headlineMedium,
            ),  // Text
          ],  // <Widget>[]
        ),  // Column
      ),  // Center
      floatingActionButton: FloatingActionButton(
        onPressed: _incrementCounter,
        tooltip: 'Increment',
        child: const Icon(Icons.add),
      ), // This trailing comma makes auto-formatting nicer for build methods.  // F
    );  // Scaffold
  }
}
```

You can view the project as follow:

Riya Varyani 61 D15A



Riya Varyani 61 D15A

Riya Varyani D15A 61

Experiment No. 2

AIM : To design Flutter UI by including common widgets

Theory:

In Flutter, designing UIs involves combining various widgets to build interactive and visually appealing applications. Here's a more detailed overview of key concepts:

1. **Widgets in Flutter**: Everything in Flutter is a widget. Widgets are the building blocks of the UI. There are two main types:
    - **Stateless Widgets**: These are immutable and don't change over time. They are responsible for displaying UI based on fixed data or input.
    - **Stateful Widgets**: These can change their state over time. They are dynamic and are used when the UI needs to update in response to user interaction or other factors.

2. **Layout Widgets**: The layout of your UI is primarily constructed using widgets like:
    - **Container**: A versatile widget used to hold other widgets and apply styling such as padding, margin, colors, and shapes.
    - **Column**: A widget that arranges its children vertically. It's useful for stacking widgets in a vertical list.
    - **Row**: A widget that arranges its children horizontally. It's useful for placing widgets side by side.
    - **Expanded**: A widget that can be used inside a Column, Row, or Flex to make child widgets flexible and fill available space.

3. **Text and Icons**:
    - **Text**: The Text widget is used to display static or dynamic text. It can be styled with custom fonts, sizes, colors, and more.
    - **Icon**: Flutter provides a large set of material design icons, and the Icon widget lets you display them in various sizes and colors.

4. **Buttons and User Interactions**:

    - Flutter provides multiple button widgets like **ElevatedButton**, **TextButton**, and **IconButton** to handle user interaction. These widgets can trigger actions when tapped.

- **TextField**: Used for user input. You can configure it to accept different types of text, such as email or password.
- **Checkbox**, **Radio**, and **Switch**: Used for boolean selections, allowing users to choose options in forms or settings.

5. **Navigation**:
   - Flutter's **Navigator** widget is responsible for managing routes or screens. You use Navigator.push to navigate to a new screen, and Navigator.pop to return to the previous one.
   - **Routes** define the pages of an app, and you can pass data between them using arguments.

6. **Displaying Lists and Grids**:
   - **ListView**: The ListView widget is used to display a list of items that can scroll. It's perfect for long lists that need to be dynamically generated.
   - **GridView**: This widget allows you to display items in a grid format, with configurable row and column layouts.

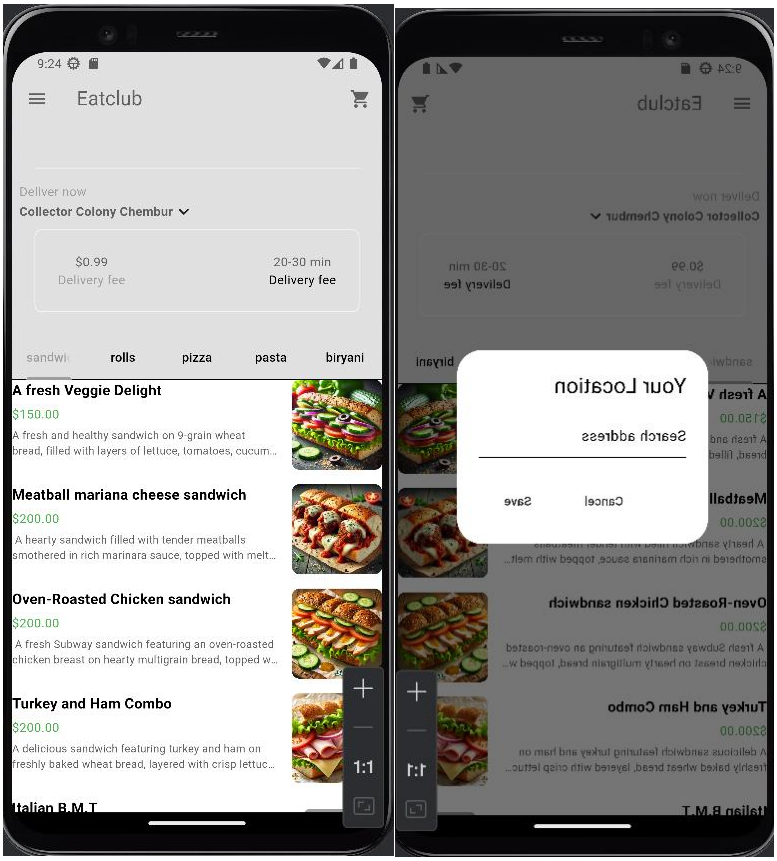
7. **State Management**:
   - Flutter provides a variety of ways to manage state. The simplest approach is using setState() to update the UI. For more complex apps, you can use state management solutions like **Provider**, **Riverpod**, or **Bloc** to separate business logic from UI code.
   - Proper state management ensures your UI stays in sync with the underlying data, especially in interactive or dynamic applications.

8. **Theming and Styling**:
   - Flutter allows you to define a global **Theme** for your app using ThemeData, which ensures consistent styling across the entire app. You can customize colors, typography, and button styles.

9. **Animations and Transitions**:
   - Flutter provides powerful animation support to create smooth and visually appealing transitions between UI states.
   - **AnimatedContainer**: A widget that animates changes in properties like width, height, or color over a given duration.

- You can also create custom animations using **AnimationController** and **Tween**.

Screenshots:



Code Snippets:

Scaffold & Column Widget

```
import 'package:flutter/material.dart';
import 'package:flutter_eats/components/my_current_location.dart';
import 'package:flutter_eats/components/my_description_box.dart';
import 'package:flutter_eats/components/my_food_tile.dart';
import 'package:flutter_eats/components/my_sliver_app_bar.dart';
import 'package:flutter_eats/models/food.dart';
import 'package:flutter_eats/models/restaurant.dart';
import 'package:flutter_eats/pages/food_page.dart';
import 'package:provider/provider.dart';
import '../components/my_drawer.dart';
```

```
import '../components/my_tab_bar.dart';

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

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> with SingleTickerProviderStateMixin {
  // Tab Controller
  late TabController _tabController;

  @override
  void initState() {
    super.initState();
    _tabController = TabController(length: FoodCategory.values.length, vsync: this);
  }

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

  // Sort out and return a list of food items that belong to a specific category
  List<Widget> getFoodInThisCategory(List<Food> fullMenu) {
    return FoodCategory.values.map((category) {
      List<Food> categoryMenu = fullMenu.where((food) => food.category == category).toList();

      return ListView.builder(
        itemCount: categoryMenu.length,  // Fixed syntax error (was itemBuilder:
categoryMenu.lenght)
        physics: const NeverScrollableScrollPhysics(),
        padding: EdgeInsets.zero,
        itemBuilder: (context, index) {
          //get individual food
          final food = categoryMenu[index];
          return FoodTile(
            food: food,
            onTap: () => Navigator.push(
              context,
              MaterialPageRoute(
```

```
              builder: (context) => FoodPage(food: food),
            ),
          ),  // Fixed incorrect indexing (was categoryMenu[food])
        );
      },
    );
  }).toList();
 }

 @override
 Widget build(BuildContext context) {
  return Scaffold(
    drawer: const MyDrawer(),
    body: NestedScrollView(
      headerSliverBuilder: (context, innerBoxIsScrolled) => [
        MySliverAppBar(
          title: MyTabBar(tabController: _tabController),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.end,
            children: [
              Divider(
                indent: 25,
                endIndent: 25,
                color: Theme.of(context).colorScheme.secondary,
              ),
              // My current location
              const MyCurrentLocation(),

              // Description box
              const MyDescriptionBox(),
            ],
          ),
        ),
      ],
      body: Consumer<Restaurant>(
         builder: (context, restaurant, child) => TabBarView(
            controller: _tabController,
            children: getFoodInThisCategory(restaurant.menu),)
      ),
    ),
  );
 }
}
```

Riya Varyani D15A 61

Experiment No. 3

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

Theory:

To include icons, images, and fonts in a Flutter app, you need to understand the following core concepts related to asset management in Flutter. Here's the theory behind including these resources:

1. Assets in Flutter:

Assets are files or resources such as images, fonts, icons, or sounds that you include in your app and bundle within the app package. In Flutter, you can include these assets in your project and then use them in your app.

2. Adding Assets to pubspec.yaml:

In Flutter, you declare assets in the pubspec.yaml file. This is where you specify which assets should be bundled with your app during the build process.

Example for adding assets:

flutter:

assets:

- assets/images/

- assets/icons/

In this example, the images are stored in the assets/images directory, and icons in the assets/icons directory. You can also specify specific files instead of directories.

3. Including Images:

Flutter provides several ways to include images in your app, including network images, asset images, and file images. To use asset images, you reference them by their file path relative to the assets directory.

Example of including an asset image:

```
Image.asset('assets/images/my_image.png')
```

For this to work, the image (my_image.png) must be listed in the pubspec.yaml file under the flutter section, like this: flutter: assets:

```
  - assets/images/my_image.png
```

4. Including Icons:

Flutter allows you to use custom icons in your app. You can add icon files (e.g., .png or .svg) to your assets folder and use them in the app. Alternatively, Flutter provides built-in icons via the Icons class.

Example of using an asset icon:

```
Image.asset('assets/icons/my_icon.png')
```

5. Including Fonts:

To include custom fonts, you place your font files (e.g., .ttf or .otf files) in a folder inside your assets directory. Then, you declare these fonts in the pubspec.yaml file and use them in your app.

Example of adding custom fonts in pubspec.yaml:

```
flutter:

fonts:

- family: CustomFont

    fonts:

- asset: assets/fonts/CustomFont-Regular.ttf

- asset: assets/fonts/CustomFont-Bold.ttf
```

Example of using the custom font in Flutter:

```
Text(
```

'Hello, World!', style: TextStyle(fontFamily:

'CustomFont', fontSize: 20),

)

6. Font Weight and Style:

When specifying fonts, you can also define specific font weights and styles (like bold, italic) to support different text styles in your app.

7. Working with Icon Libraries:

While you can use custom icon files, Flutter also supports popular icon libraries like **FontAwesome**, **MaterialIcons**, etc. For example, Flutter's built-in Icons class provides access to the Material Design icons.

Example of using a Material icon:

Icon(Icons.home)

8. Caching and Optimization:

- **Images**: Flutter caches images, but you might want to use libraries like cached_network_image for better image loading and caching.
- **Fonts**: Custom fonts are loaded from assets when the app is first started, and they remain available for the lifecycle of the app.

9. SVG Images:

If you want to use vector-based images (like SVG), you can include them using packages like flutter_svg, which allows you to display scalable vector graphics (SVG files) in Flutter.

Example of including an SVG:

import 'package:flutter_svg/flutter_svg.dart';

SvgPicture.asset('assets/icons/my_icon.svg')

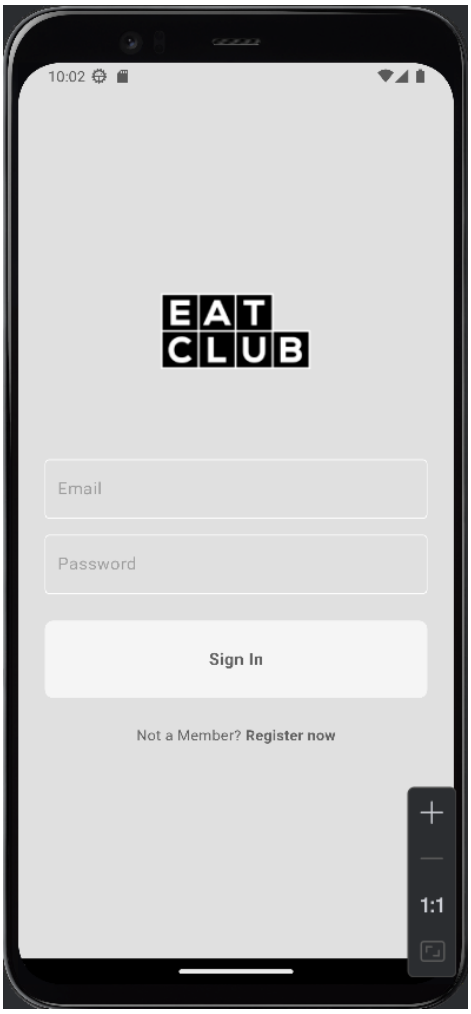
Summary of Key Steps:

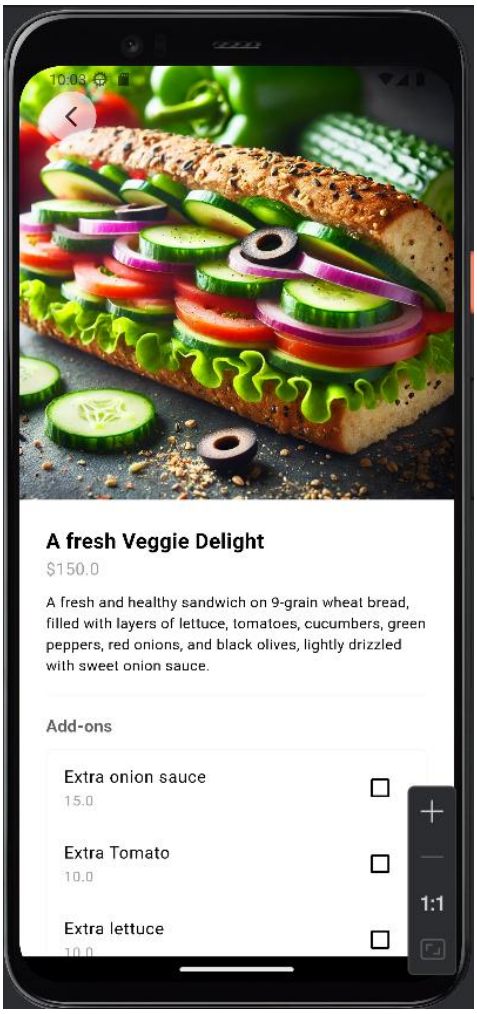
1. **Declare assets in pubspec.yaml.**
2. **For images and icons**, use Image.asset('path/to/image') and Image.asset('path/to/icon').

3. **For fonts**, declare them under the flutter section in pubspec.yaml, then use them via the TextStyle class.
4. **Use Icon Libraries**: Use Flutter's built-in Icons class or third-party icon libraries.
5. **For SVGs**, use packages like flutter_svg.

Understanding these concepts will help you effectively use images, icons, and fonts in your Flutter app.

Screenshots:





Code Snippets:

```
import 'package:collection/collection.dart';
import 'package:flutter/cupertino.dart';
```

```
import 'package:flutter_eats/models/cart_item.dart';

import 'food.dart';

class Restaurant extends ChangeNotifier {
  // list of food menu
  final List<Food> _menu = [

    //Sandwiches
    Food(
        name: "A fresh Veggie Delight",
        description:"A fresh and healthy sandwich on 9-grain wheat bread, filled with layers of lettuce, tomatoes, cucumbers, green peppers, red onions, and black olives, lightly drizzled with sweet onion sauce.",
        imagePath: "lib/images/Sandwiches/veggie_delight.webp",
        price: 150,
        category: FoodCategory.sandwiches,
        availableAddons: [
          Addon(name: "Extra onion sauce" , price: 15),
          Addon(name: "Extra Tomato" , price: 10),
          Addon(name: "Extra lettuce" , price: 10),
        ]),

    Food(
        name: "Meatball mariana cheese sandwich",
        description:" A hearty sandwich filled with tender meatballs smothered in rich marinara sauce, topped with melted mozzarella cheese, all served on Italian herbs and cheese bread.",
        imagePath: "lib/images/Sandwiches/Meatball_mariana.webp",
        price: 200,
        category: FoodCategory.sandwiches,
        availableAddons: [
          Addon(name: "Extra mariana sauce" , price: 30),
          Addon(name: "Extra cheese" , price: 20),
          Addon(name: "Extra meatballs" , price: 60),
        ]),

    Food(
        name: "Oven-Roasted Chicken sandwich",
        description:" A fresh Subway sandwich featuring an oven-roasted chicken breast on hearty multigrain bread, topped with lettuce, sliced tomatoes, cucumbers, and a drizzle of honey mustard sauce.",
        imagePath: "lib/images/Sandwiches/Roasted_Chicken.webp",
        price: 200,
        category: FoodCategory.sandwiches,
```

```
      availableAddons: [
        Addon(name: "Honey Mustard Sauce" , price: 30),
        Addon(name: "Extra Tomato" , price: 20),
        Addon(name: "Extra Cucumber" , price: 50),
      ]),

  Food(
      name: "Turkey and Ham Combo",
      description:"A delicious sandwich featuring turkey and ham on freshly baked wheat bread, layered with crisp lettuce, juicy tomatoes, red onions, Swiss cheese, and a light spread of mayonnaise and mustard.",
      imagePath: "lib/images/Sandwiches/Turkey_Ham.webp",
      price: 200,
      category: FoodCategory.sandwiches,
      availableAddons: [
        Addon(name: "Swiss Cheese" , price: 30),
        Addon(name: "Extra Mayonnise" , price: 20),
        Addon(name: "Extra Mustard" , price: 10),
      ]),

  Food(
      name: "Italian B.M.T",
      description:"A delicious Subway sandwich with layers of pepperoni, salami, and ham on freshly baked bread, topped with crisp lettuce, tomatoes, onions, and olives, and finished with a drizzle of mayonnaise and mustard",
      imagePath: "lib/images/Sandwiches/Salami_Sub .webp",
      price: 200,
      category: FoodCategory.sandwiches,
      availableAddons: [
        Addon(name: "Extra Ham" , price: 30),
        Addon(name: "Extra Salami " , price: 20),
        Addon(name: "Extra Pepperoni" , price: 30),
      ]),

  //Pasta

  Food(
      name: "Spaghetti Carbonara",
      description: "A classic Italian dish featuring al dente spaghetti coated in a creamy egg-based sauce with crispy pancetta, freshly grated Parmesan cheese, and a sprinkle of black pepper.",
      imagePath: "lib/images/Pasta/Spaghetti_Carabonara.webp",
      price: 200,
      category: FoodCategory.pasta,
      availableAddons: [
```

```
        Addon(name: "Garlic Bread" , price: 100),
        Addon(name: "Bread Sticks" , price: 20),
        Addon(name: "Extra chesse" , price: 30),
      ]),

  Food(
      name: "Alfredo Pasta",
      description: "A creamy and rich pasta dish with perfectly cooked fettuccine noodles coated in a smooth and velvety Parmesan cheese and butter sauce, garnished with freshly chopped parsley.",
      imagePath: "lib/images/Pasta/Alfredo_Pasta.webp",
      price: 200,
      category: FoodCategory.pasta,
      availableAddons: [
        Addon(name: "Garlic Bread" , price: 100),
        Addon(name: "Bread Sticks" , price: 50),
        Addon(name: "Parmesan cheese" , price: 30),
      ]),

  Food(
      name: "Basil Pesto Pasta",
      description: "A creamy and rich pasta dish with perfectly cooked fettuccine noodles coated in a smooth and velvety Parmesan cheese and butter sauce, garnished with freshly chopped parsley.",
      imagePath: "lib/images/Pasta/Basil_Pestp_Pasta.webp",
      price: 200,
      category: FoodCategory.pasta,
      availableAddons: [
        Addon(name: "Garlic Bread" , price: 100),
        Addon(name: "Bread Sticks" , price: 50),
        Addon(name: "Extra Pesto" , price: 60),
      ]),

  Food(
      name: "Lasagna",
      description: " A delicious homemade lasagna with layers of pasta sheets, rich ground beef and tomato sauce, creamy béchamel, and melted mozzarella cheese, baked to perfection and topped with fresh basil.",
      imagePath: "lib/images/Pasta/Chicken_Lasagna.webp",
      price: 200,
      category: FoodCategory.pasta,
      availableAddons: [
        Addon(name: "Garlic Bread" , price: 100),
        Addon(name: "Bread Sticks" , price: 50),
        Addon(name: "Extra Pesto" , price: 60),
```

```
    ]),

  Food(
    name: "Penne Arrabiata Pasta",
    description: "  A vibrant dish featuring al dente penne pasta tossed in a spicy tomato sauce with red chili flakes, garlic, and olive oil, garnished with fresh basil leaves and grated Parmesan cheese.",
    imagePath: "lib/images/Pasta/Penne_Arrabiata_Pasta.webp",
    price: 200,
    category: FoodCategory.pasta,
    availableAddons: [
     Addon(name: "Garlic Bread" , price: 100),
     Addon(name: "Bread Sticks" , price: 50),
     Addon(name: "Extra Pesto" , price: 60),
    ]),

  //Pizza

  Food(
    name: "BBQ Chicken Pizza",
    description: "  A vibrant dish featuring al dente penne pasta tossed in a spicy tomato sauce with red chili flakes, garlic, and olive oil, garnished with fresh basil leaves and grated Parmesan cheese.",
    imagePath: "lib/images/Pizza/BBQ_Chicken_Pizza.webp",
    price: 200,
    category: FoodCategory.pizza,
    availableAddons: [
     Addon(name: "Garlic Bread" , price: 100),
     Addon(name: "Bread Sticks" , price: 50),
     Addon(name: "Extra Pesto" , price: 60),
    ]),

  Food(
    name: "Classic Margherita Pizza",
    description: " A vibrant dish featuring al dente penne pasta tossed in a spicy tomato sauce with red chili flakes, garlic, and olive oil, garnished with fresh basil leaves and grated Parmesan cheese.",
    imagePath: "lib/images/Pizza/Classic_Margherita_pizza.webp",
    price: 200,
    category: FoodCategory.pizza,
    availableAddons: [
     Addon(name: "Garlic Bread" , price: 100),
     Addon(name: "Bread Sticks" , price: 50),
     Addon(name: "Extra Pesto" , price: 60),
    ]),
```

```
  Food(
    name: "Musshroom Pizza",
    description: "  A vibrant dish featuring al dente penne pasta tossed in a spicy tomato sauce with red chili flakes, garlic, and olive oil, garnished with fresh basil leaves and grated Parmesan cheese.",
    imagePath: "lib/images/Pizza/Musshroom_Pizza.webp",
    price: 200,
    category: FoodCategory.pizza,
    availableAddons: [
     Addon(name: "Garlic Bread" , price: 100),
     Addon(name: "Bread Sticks" , price: 50),
     Addon(name: "Extra Pesto" , price: 60),
    ]),

  Food(
    name: "Pepperoni pizza",
    description: "  A vibrant dish featuring al dente penne pasta tossed in a spicy tomato sauce with red chili flakes, garlic, and olive oil, garnished with fresh basil leaves and grated Parmesan cheese.",
    imagePath: "lib/images/Pizza/Pepperoni_pizza.webp",
    price: 200,
    category: FoodCategory.pizza,
    availableAddons: [
     Addon(name: "Garlic Bread" , price: 100),
     Addon(name: "Bread Sticks" , price: 50),
     Addon(name: "Extra Pesto" , price: 60),
    ]),

  Food(
    name: "Veggie Supreme Pizza",
    description: "  A vibrant dish featuring al dente penne pasta tossed in a spicy tomato sauce with red chili flakes, garlic, and olive oil, garnished with fresh basil leaves and grated Parmesan cheese.",
    imagePath: "lib/images/Pizza/Veggie_Supreme_Pizza.webp",
    price: 200,
    category: FoodCategory.pizza,
    availableAddons: [
     Addon(name: "Garlic Bread" , price: 100),
     Addon(name: "Bread Sticks" , price: 50),
     Addon(name: "Extra Pesto" , price: 60),
    ]),


  //rolls
```

```
  Food(
    name: "Chicken Kathi roll",
    description: "  A vibrant dish featuring al dente penne pasta tossed in a spicy tomato sauce with red chili flakes, garlic, and olive oil, garnished with fresh basil leaves and grated Parmesan cheese.",
    imagePath: "lib/images/rolls/Chicken_Kathi_roll.webp",
    price: 200,
    category: FoodCategory.rolls,
    availableAddons: [
     Addon(name: "Garlic Bread" , price: 100),
     Addon(name: "Bread Sticks" , price: 50),
     Addon(name: "Extra Pesto" , price: 60),
    ]),

  Food(
    name: "Chicken Shawarma Wrap",
    description: "  A vibrant dish featuring al dente penne pasta tossed in a spicy tomato sauce with red chili flakes, garlic, and olive oil, garnished with fresh basil leaves and grated Parmesan cheese.",
    imagePath: "lib/images/rolls/Chicken_Shawarma_Wrap.webp",
    price: 200,
    category: FoodCategory.rolls,
    availableAddons: [
     Addon(name: "Garlic Bread" , price: 100),
     Addon(name: "Bread Sticks" , price: 50),
     Addon(name: "Extra Pesto" , price: 60),
    ]),

  Food(
    name: "Tandoori Paneer roll",
    description: "  A vibrant dish featuring al dente penne pasta tossed in a spicy tomato sauce with red chili flakes, garlic, and olive oil, garnished with fresh basil leaves and grated Parmesan cheese.",
    imagePath: "lib/images/rolls/Tandoori_Paneer_roll.webp",
    price: 200,
    category: FoodCategory.rolls,
    availableAddons: [
     Addon(name: "Garlic Bread" , price: 100),
     Addon(name: "Bread Sticks" , price: 50),
     Addon(name: "Extra Pesto" , price: 60),
    ]),

  Food(
    name: "Veggie Spring roll",
```

```
      description: "  A vibrant dish featuring al dente penne pasta tossed in a spicy tomato sauce with red chili flakes, garlic, and olive oil, garnished with fresh basil leaves and grated Parmesan cheese.",
      imagePath: "lib/images/rolls/Veggie_Spring_roll.webp",
      price: 200,
      category: FoodCategory.rolls,
      availableAddons: [
       Addon(name: "Garlic Bread" , price: 100),
       Addon(name: "Bread Sticks" , price: 50),
       Addon(name: "Extra Pesto" , price: 60),
      ]),

   Food(
      name: "Mutton Roll",
      description: " A vibrant dish featuring al dente penne pasta tossed in a spicy tomato sauce with red chili flakes, garlic, and olive oil, garnished with fresh basil leaves and grated Parmesan cheese.",
      imagePath: "lib/images/rolls/Mutton_Roll.webp",
      price: 200,
      category: FoodCategory.rolls,
      availableAddons: [
       Addon(name: "Garlic Bread" , price: 100),
       Addon(name: "Bread Sticks" , price: 50),
       Addon(name: "Extra Pesto" , price: 60),
      ]),


   //biryani

   Food(
      name: "Handi Biryani",
      description: "  A vibrant dish featuring al dente penne pasta tossed in a spicy tomato sauce with red chili flakes, garlic, and olive oil, garnished with fresh basil leaves and grated Parmesan cheese.",
      imagePath: "lib/images/Biryani/Handi_Biryani.webp",
      price: 200,
      category: FoodCategory.biryani,
      availableAddons: [
       Addon(name: "Garlic Bread" , price: 100),
       Addon(name: "Bread Sticks" , price: 50),
       Addon(name: "Extra Pesto" , price: 60),
      ]),

   Food(
      name: "Hyderabadi Biryani",
```

```
      description: "  A vibrant dish featuring al dente penne pasta tossed in a spicy tomato sauce with red chili flakes, garlic, and olive oil, garnished with fresh basil leaves and grated Parmesan cheese.",
      imagePath: "lib/images/Biryani/Hyderabadi_Biryani.webp",
      price: 200,
      category: FoodCategory.biryani,
      availableAddons: [
       Addon(name: "Garlic Bread" , price: 100),
       Addon(name: "Bread Sticks" , price: 50),
       Addon(name: "Extra Pesto" , price: 60),
      ]),

   Food(
      name: "Kolkata Biryani",
      description: "  A vibrant dish featuring al dente penne pasta tossed in a spicy tomato sauce with red chili flakes, garlic, and olive oil, garnished with fresh basil leaves and grated Parmesan cheese.",
      imagePath: "lib/images/Biryani/Kolkata_Biryani.webp",
      price: 200,
      category: FoodCategory.biryani,
      availableAddons: [
       Addon(name: "Garlic Bread" , price: 100),
       Addon(name: "Bread Sticks" , price: 50),
       Addon(name: "Extra Pesto" , price: 60),
      ]),

   Food(
      name: "Lucknowi Biryani",
      description: "  A vibrant dish featuring al dente penne pasta tossed in a spicy tomato sauce with red chili flakes, garlic, and olive oil, garnished with fresh basil leaves and grated Parmesan cheese.",
      imagePath: "lib/images/Biryani/Lucknowi_Biryani.webp",
      price: 200,
      category: FoodCategory.biryani,
      availableAddons: [
       Addon(name: "Garlic Bread" , price: 100),
       Addon(name: "Bread Sticks" , price: 50),
       Addon(name: "Extra Pesto" , price: 60),
      ]),

   Food(
      name: "Malabar Biryani",
      description: "  A vibrant dish featuring al dente penne pasta tossed in a spicy tomato sauce with red chili flakes, garlic, and olive oil, garnished with fresh basil leaves and grated Parmesan cheese.",
```

```
      imagePath: "lib/images/Biryani/Malabar_Biryani.webp",
      price: 200,
      category: FoodCategory.biryani,
      availableAddons: [
       Addon(name: "Garlic Bread" , price: 100),
       Addon(name: "Bread Sticks" , price: 50),
       Addon(name: "Extra Pesto" , price: 60),
      ]),

  ];
  /*
  G E T T E R S
   */

List<Food> get menu => _menu;
List<CartItem> get cart => _cart;

/*
O P E R A T I O NS
 */

// user cart
final List<CartItem> _cart = [];


// add to cart
void addTocart(Food food, List<Addon> selectedAddons) {
  // see if there is a cart item already with the same food and selected addons

  CartItem? cartItem = _cart.firstWhereOrNull((item) {
    // check if the food items are the same
    bool isSameFood = item.food == food;


    //check if the list of selected addons are the same
    bool isSameAddons = ListEquality().equals(item.selectedAddons,
selectedAddons);

    return isSameFood && isSameAddons;
  } );

  // if item already exists, increase its quantity
  if (cartItem !=null) {
    cartItem.quantity++;
  }
```

```
  //otherwise, add a new cart item
  else {
    _cart.add(
        CartItem(
          food: food,
          selectedAddons: selectedAddons,
        ),);
  }
  notifyListeners();
}

// remove from cart

void removeFromCart(CartItem cartItem) {
  int cartIndex = _cart.indexOf(cartItem);

  if (cartIndex != -1) {
    if (_cart[cartIndex].quantity > 1 ) {
      _cart[cartIndex].quantity--;
    } else {
      _cart.removeAt(cartIndex);
    }
  }

  notifyListeners();
}

//get total price of cart

double getTotalPrice() {
  double total = 0.0;

  for (CartItem cartItem in _cart) {
    double itemTotal = cartItem.food.price;

    for (Addon addon in cartItem.selectedAddons) {
      itemTotal += addon.price;
    }
    total += itemTotal * cartItem.quantity;
  }
  return total;
}

//get total number of items
```

```
int getTotalItemCount() {
  int totalItemCount = 0;

  for (CartItem cartItem in _cart) {
    totalItemCount +=cartItem.quantity;
  }

  return totalItemCount;
}

//clear cart

void clearCart() {
  _cart.clear();
  notifyListeners();
}

/*
H E L P E R S
 */

// generate a receipt

// format double value into money

// format list of addons into a string summary
}
```

Experiment No. 4

AIM : To create an interactive Form using form widget

Theory:

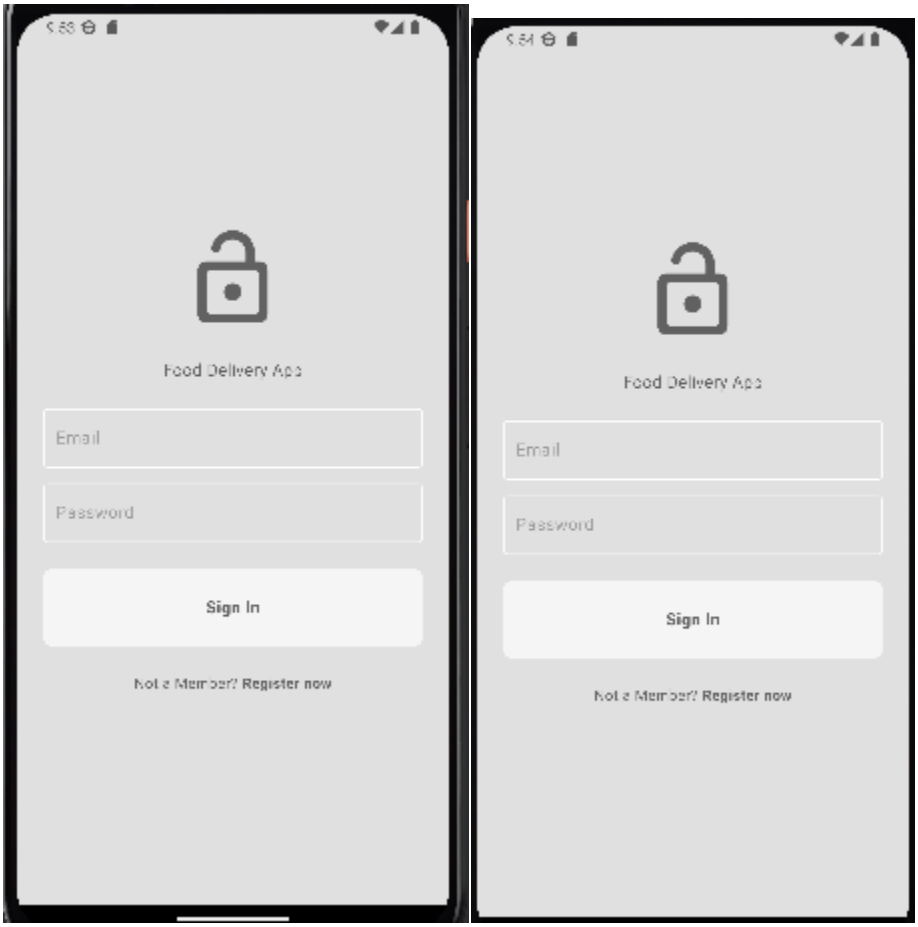
1. **Form Widget**:
    - The Form widget is a container for managing form-related interactions. It allows for validation and saving the form data.
    - It keeps track of the state of all the fields within the form (like TextFormField widgets) through a GlobalKey<FormState>.
2. **TextFormField**:
    - This is the main widget used for collecting user input (such as text). It integrates easily with form validation and submission.
    - You can apply validators to the TextFormField to ensure the input meets specific criteria (e.g., required fields, correct format).
3. **GlobalKey**:
    - A GlobalKey<FormState> is essential for managing the form's state (e.g., validating fields, saving data). It's assigned to the Form widget and can be used to trigger actions like form validation or saving the data.
4. **Validation**:
    - You can define validation rules on each form field. The TextFormField widget has a validator property, which allows you to write logic that will run whenever the form is validated.
    - A validator checks whether the input meets the required format (such as checking for valid email format or a non-empty field).
5. **Form Submission**:
    - When you're ready to submit the form, you can call formKey.currentState?.save() to trigger the save method for all fields or formKey.currentState?.validate() to check if all the fields pass the validation checks.
6. **Saving Data**:
    - After validation, data entered in the form fields can be saved to variables or used for further processing, such as sending it to a server.

Form Workflow:

1. **Create a form**: Use the Form widget to wrap all input fields.
2. **Add form fields**: Use widgets like TextFormField to collect data.

3. **Set up validation**: Add validation logic to the form fields.
4. **Submit the form**: Trigger validation and handle the form submission.
5. **Save data**: Capture the entered data once validation passes.

Flutter's form management tools are powerful, enabling you to handle complex forms with various input types, validations, and submissions effectively.



Code Snippets:
Login Page:

```
import 'package:flutter/material.dart';
import 'package:flutter_eats/components/my_button.dart';
import 'package:flutter_eats/components/my_textfield.dart';
import 'home_page.dart';

class LoginPage extends StatefulWidget {
```

```
  final void Function()? onTap;
  const LoginPage({super.key, required this.onTap});

  @override
  State<LoginPage> createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  // Text editing controllers
  final TextEditingController emailController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();

  void login() {
    /*
    fill out authentication here
     */

    // navigate to home page
    Navigator.push(
      context,
      MaterialPageRoute(
          builder: (context) => const HomePage(),
      ),
    );

  }

  @override
  void dispose() {
    emailController.dispose();
    passwordController.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Theme.of(context).colorScheme.background,
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            // Logo
            Icon(
              Icons.lock_open_rounded,
              size: 100,
              color: Theme.of(context).colorScheme.inversePrimary,
```

```
),

const SizedBox(height: 25),

// App Slogan
Text(
  "Food Delivery App",
  style: TextStyle(
    fontSize: 16,
    color: Theme.of(context).colorScheme.inversePrimary,
  ),
),

const SizedBox(height: 25),

// Email Text Field
MyTextField(
  controller: emailController,
  hintText: "Email",
  obscureText: false,
),

const SizedBox(height: 15),

// Password Text Field
MyTextField(
  controller: passwordController,
  hintText: "Password",
  obscureText: true,
),

const SizedBox(height: 25),

// Sign In Button
MyButton(
  text: "Sign In",
    onTap: login
),

const SizedBox(height: 25),

// Register Row
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    Text(
      "Not a Member?",
```

```
              style: TextStyle(color: Theme.of(context).colorScheme.inversePrimary),
            ),
            const SizedBox(width: 4),
            GestureDetector(
              onTap: widget.onTap, // Fix onTap
              child: Text(
                "Register now",
                style: TextStyle(
                  color: Theme.of(context).colorScheme.inversePrimary,
                  fontWeight: FontWeight.bold,
                ),
              ),
            ),
          ],
        ),
      ],
    ),
  ),
 );
 }
}
```

Register Page:

```
import 'package:flutter/material.dart';
import 'package:flutter_eats/components/my_button.dart';
import 'package:flutter_eats/components/my_textfield.dart';

class RegisterPage extends StatefulWidget {
 final void Function()? onTap;
 const RegisterPage({super.key, required this.onTap});

 @override
 State<RegisterPage> createState() => _RegisterPageState();
}

class _RegisterPageState extends State<RegisterPage> {
 // Text editing controllers
 final TextEditingController emailController = TextEditingController();
 final TextEditingController passwordController = TextEditingController();
 final TextEditingController confirmpasswordController = TextEditingController();


 @override
 void dispose() {
  emailController.dispose();
  passwordController.dispose();
  super.dispose();
```

```
}

@override
Widget build(BuildContext context) {
 return Scaffold(
   backgroundColor: Theme.of(context).colorScheme.background,
   body: Center(
    child: Column(
     mainAxisAlignment: MainAxisAlignment.center,
     children: [
      // Logo
      Icon(
       Icons.lock_open_rounded,
       size: 100,
       color: Theme.of(context).colorScheme.inversePrimary,
      ),

      const SizedBox(height: 25),

      // App Slogan
      Text(
       " Lets create an account for you",
       style: TextStyle(
        fontSize: 16,
        color: Theme.of(context).colorScheme.inversePrimary,
       ),
      ),

      const SizedBox(height: 25),

      // Email Text Field
      MyTextField(
       controller: emailController,
       hintText: "Email",
       obscureText: false,
      ),

      const SizedBox(height: 15),

      // Password Text Field
      MyTextField(
       controller: passwordController,
       hintText: "Password",
       obscureText: true,
      ),

      const SizedBox(height: 25),
```

```
            // Confirm Password Text Field
            MyTextField(
              controller: confirmpasswordController,
              hintText: "Confirm Password",
              obscureText: true,
            ),

            // Sign up Button
            MyButton(
              onTap: () {
                // Implement sign-in functionality
              },
              text: "Sign up",
            ),

            const SizedBox(height: 25),

            // already have an account? Login here
            Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Text(
                  "already have an account?",
                  style: TextStyle(color: Theme.of(context).colorScheme.inversePrimary),
                ),
                const SizedBox(width: 4),
                GestureDetector(
                  onTap: widget.onTap, // Fix onTap
                  child: Text(
                    "Login now",
                    style: TextStyle(
                      color: Theme.of(context).colorScheme.inversePrimary,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                ),
              ],
            ),
          ],
        ),
      ),
    );
  }
}
```

Experiment No. 5

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

Theory:

1. Navigation and Routing

In Flutter, navigation refers to moving from one screen (or "route") to another. There are several key concepts:

- **Routes**: These are the different screens or pages in your app. Every route is typically represented by a Widget in Flutter. The default route is usually the home screen of the app, but you can define multiple routes for different screens.

- **Navigator**: This is a widget that manages a stack of routes. You can "push" a new route onto the stack to navigate to another screen, or "pop" the top route off to go back.

- **Named Routes**: These are routes that are identified by a string. Instead of pushing or popping routes directly, you can refer to routes by their name (e.g., /home, /settings).

- **Custom Route Transitions**: Flutter allows you to define custom animations and transitions when navigating between routes. You can create smooth, custom page transitions using PageRouteBuilder.

- **Route Arguments**: You can pass data between routes using arguments. This is particularly useful when navigating to a screen that requires specific data (e.g., opening a product page with product details).

2. Gestures in Flutter

Gestures are interactions that a user performs with the screen, such as taps, swipes, or long presses. Flutter provides a flexible way to detect these gestures.

- **GestureDetector**: This is the most commonly used widget for detecting gestures. You can wrap it around any widget to detect gestures like tap, double tap, long press, swipe, and others.

- **Tap Gesture**: A simple touch interaction, typically detected using onTap or onLongPress callbacks.

- **Swipe Gestures**: Swiping is usually detected via onHorizontalDragUpdate, onVerticalDragUpdate, or onPanUpdate. These allow you to track the user's finger movement and respond accordingly.

- **Custom Gesture Detection**: Flutter also allows you to implement more complex gestures. For example, you can detect drag gestures to create features like a sliding menu or draggable elements.

- **Dismissible Widget**: This widget enables swipe-to-dismiss behavior, commonly used for items in a list that users can swipe left or right to remove.

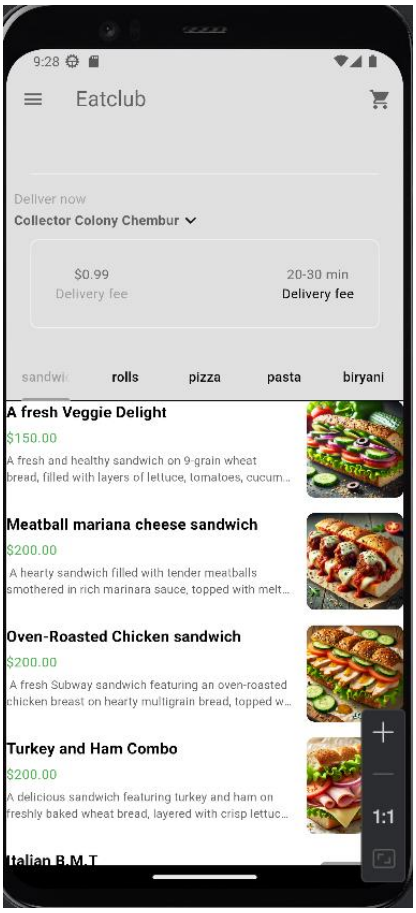
3. Managing Navigation and Gestures Together

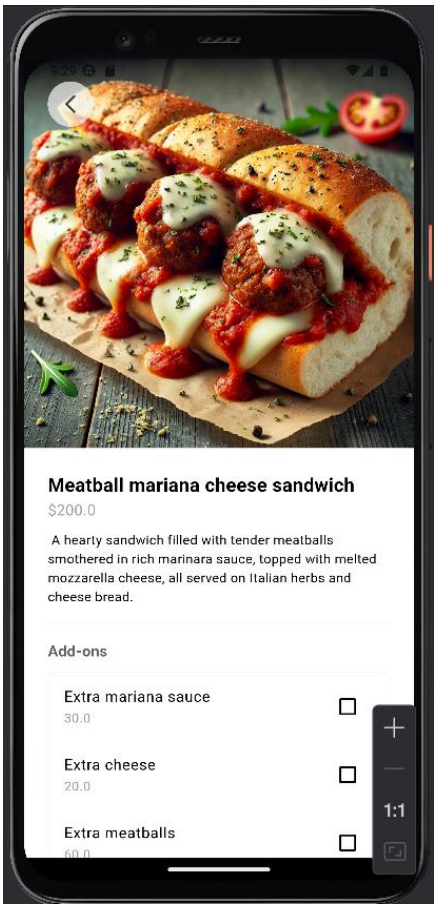
When you combine navigation with gestures, you can create more interactive and dynamic UIs. For instance, a user could swipe to navigate between screens, or tap a button that triggers navigation while performing a gesture on a different part of the screen.

4. Back Button Handling

On Android devices, there is a system-wide back button that users can press to navigate backward. Flutter provides a way to intercept and customize this behavior using WillPopScope, allowing you to decide what happens when the user tries to go back (e.g., prevent the user from leaving the current screen, show a confirmation dialog, or allow normal back navigation).

Screenshots:





Code Snippets:

Food Page:

```
import 'package:flutter/material.dart';
import 'package:flutter_eats/components/my_button.dart';
import 'package:flutter_eats/models/food.dart';
import 'package:flutter_eats/models/restaurant.dart';
import 'package:provider/provider.dart';

class FoodPage extends StatefulWidget {
  final Food food;
  final Map<Addon,bool> selectedAddons ={};

  FoodPage({
    super.key,
    required this.food,
  }) {
    // initialize selected addons to be false
    for (Addon addon in food.availableAddons) {
      selectedAddons[addon] = false;
    }
  }

  @override
  State<FoodPage> createState() => _FoodPageState();
}

class _FoodPageState extends State<FoodPage> {

  //method to add to cart
  void addTocart(Food food, Map<Addon, bool>selectedAddons) {
    // close the current food page to go back to menu
    Navigator.pop(context);

    //format the selected addon
    List<Addon> currentlySelectedAddons = [];
    for (Addon addon in widget.food.availableAddons) {
      if (widget.selectedAddons[addon] == true) {
        currentlySelectedAddons.add(addon);
      }
    }

    // add to cart
    context.read<Restaurant>().addTocart(food, currentlySelectedAddons);
  }
```

```
@override
Widget build(BuildContext context) {
  return Stack(children: [
    //scaffold UI
  Scaffold(
  body: SingleChildScrollView(
  child: Column(
  children: [
  //food image
  Image.asset(widget.food.imagePath),

  Padding(
  padding: const EdgeInsets.all(25.0),
  child: Column(
  crossAxisAlignment: CrossAxisAlignment.start,
  children: [
  // food name
  Text(
  widget.food.name,
  style: const TextStyle(
  fontWeight: FontWeight.bold,
  fontSize: 20,
  ),
  ),

  //food price
  Text(
  '\$'+ widget.food.price.toString(),
  style: TextStyle(
  fontSize: 16,
  color: Theme.of(context).colorScheme.primary,
  ),
  ),

  const SizedBox(height: 10),

  //food description
  Text(
  widget.food.description,
  ),
  const SizedBox(height: 10),

  Divider(color: Theme.of(context).colorScheme.secondary),
```

```
const SizedBox(height: 10),

//addons
Text(
"Add-ons",
style: TextStyle(
color: Theme.of(context).colorScheme.inversePrimary,
fontSize: 16,
fontWeight: FontWeight.bold,
),
),
const SizedBox(height: 10),

Container(
decoration: BoxDecoration(
border: Border.all(
color: Theme.of(context).colorScheme.secondary),
borderRadius: BorderRadius.circular(8),
),
child: ListView.builder(
shrinkWrap: true,
physics: const NeverScrollableScrollPhysics(),
padding: EdgeInsets.zero,
itemCount: widget.food.availableAddons.length,
itemBuilder: (context, index) {
//get individual addons
Addon addon = widget.food.availableAddons[index]; // Fixed syntax

//return check box UI
return CheckboxListTile(
title: Text(addon.name),
subtitle: Text(
addon.price.toString(),
style: TextStyle(
color: Theme.of(context).colorScheme.primary,
),

),
value: widget.selectedAddons[addon],
onChanged: (bool? value) {
setState(() {
widget.selectedAddons[addon] = value!;
```

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

// button add to cart
MyButton(
onTap: () => addTocart(widget.food, widget.selectedAddons),
text: "Add to cart"),

const SizedBox(height: 25),
],
),)
),



  //back button
  SafeArea(
     child: Opacity(
       opacity: 0.7,
       child: Container(
         margin: const EdgeInsets.only(left:25),
         decoration: BoxDecoration(
           color: Theme.of(context).colorScheme.secondary,
           shape: BoxShape.circle,
         ),
         child: IconButton(
            onPressed: ()=> Navigator.pop(context),
            icon: const Icon(Icons.arrow_back_ios_new_rounded)),
       ),
     ))
  ],);
 }
}
```

Food.dart file

```
class Food{
  final String name;
```

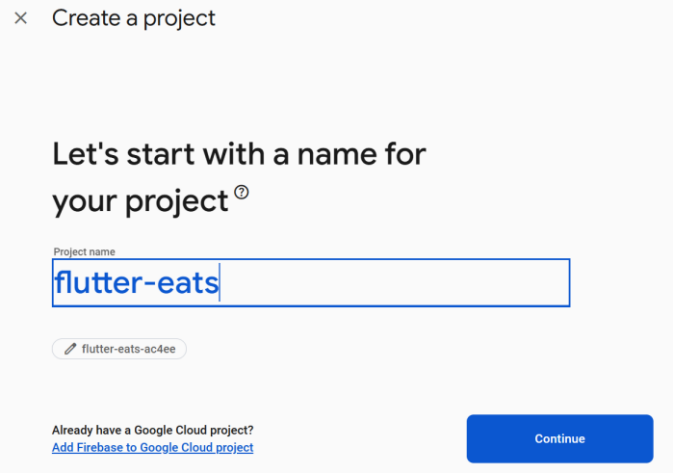
```
  final String description;
  final String imagePath;
  final double price;
  final FoodCategory category;
  List<Addon> availableAddons;

  Food( {
    required this.name,
    required this.description,
    required this.imagePath,
    required this.price,
    required this.category,
    required this.availableAddons,
  });
}

// food categories
enum FoodCategory{
  sandwiches,
  rolls,
  pizza,
  pasta,
  biryani,
}

// food addons
class Addon {
  String name;
  double price;

  Addon({
    required this.name,
    required this.price,
});
}
```

EXPERIMENT NO:- 6

Name:- Riya Varyani D15A Roll-no:-61

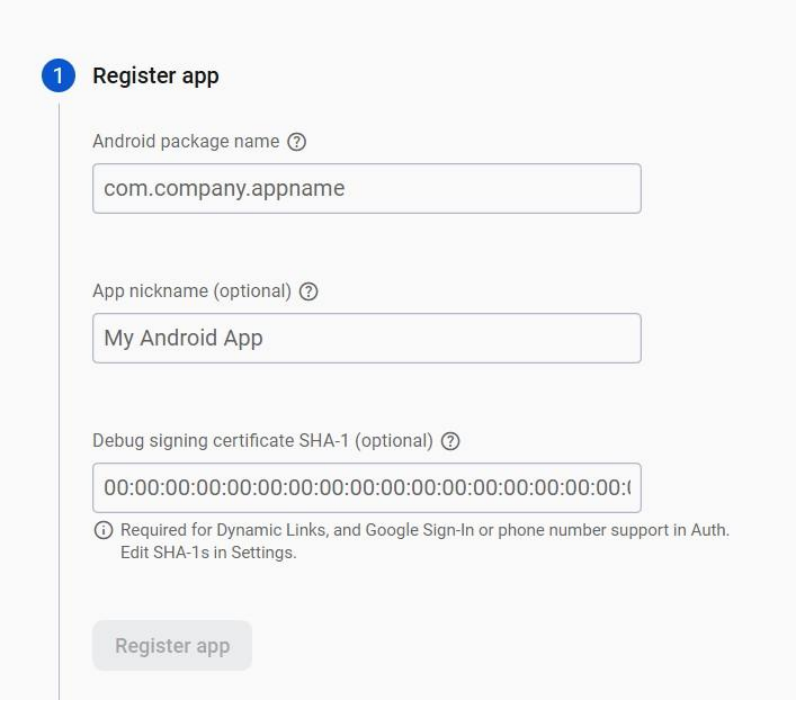
Aim:- To Connect flutter UI with firebase database

Creating a New Firebase Project



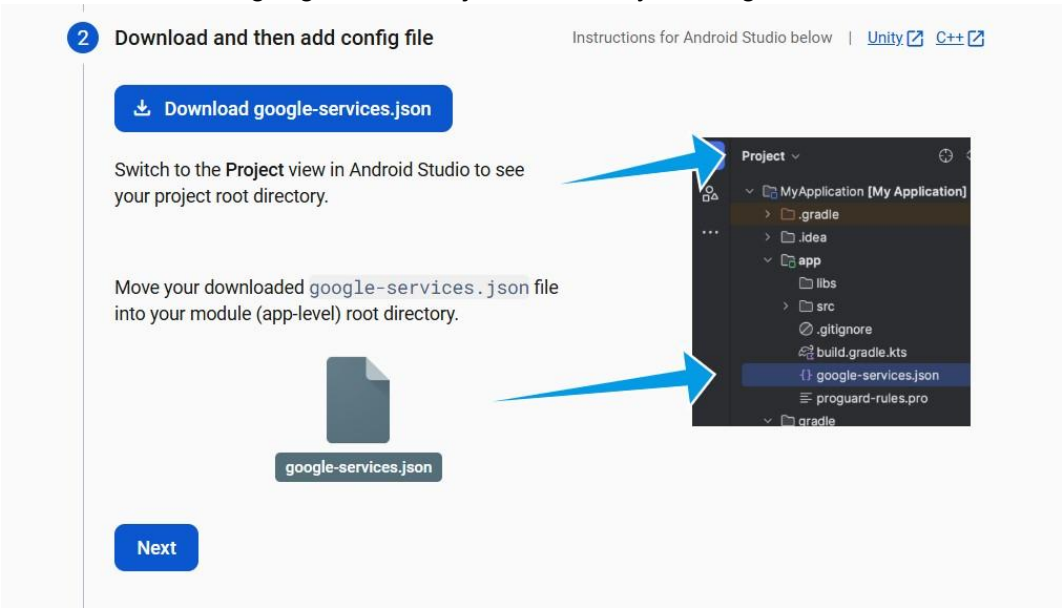
First, log in with your Google account to manage your Firebase projects. From within the Firebase dashboard, select the Create new project button and give it a name

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

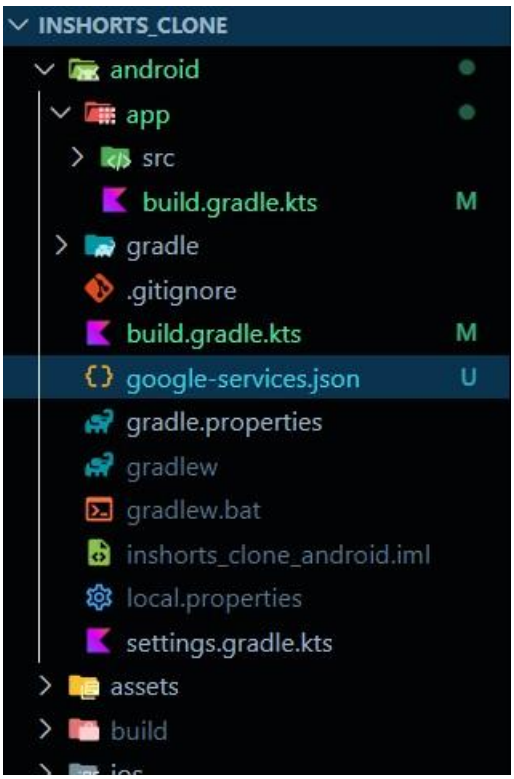


The most important thing here is to match up the Android package name that you choose here with the one inside of our application.

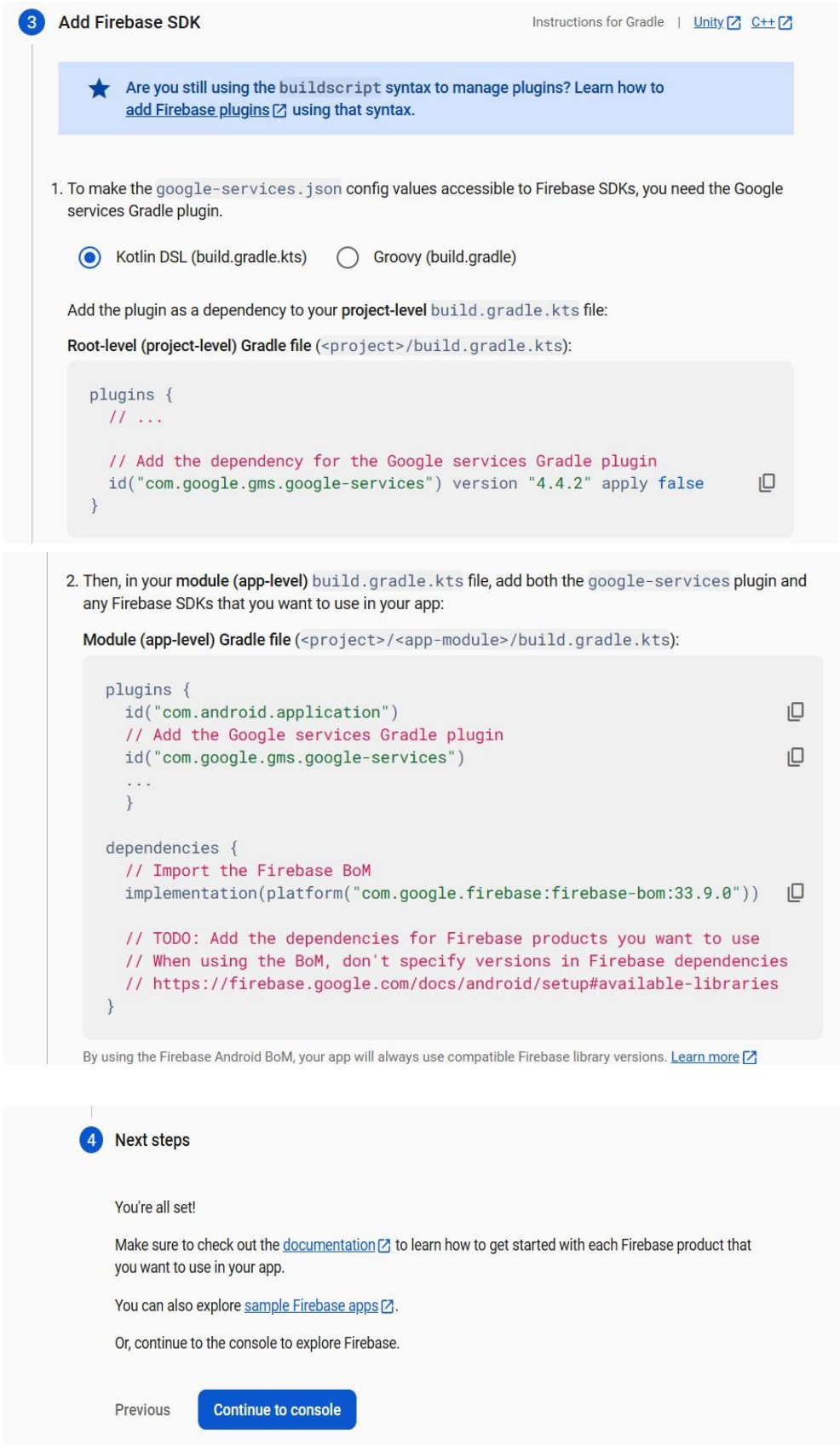
Then download the google-services.json file, that you will get.



put that file in the android folder (root level)



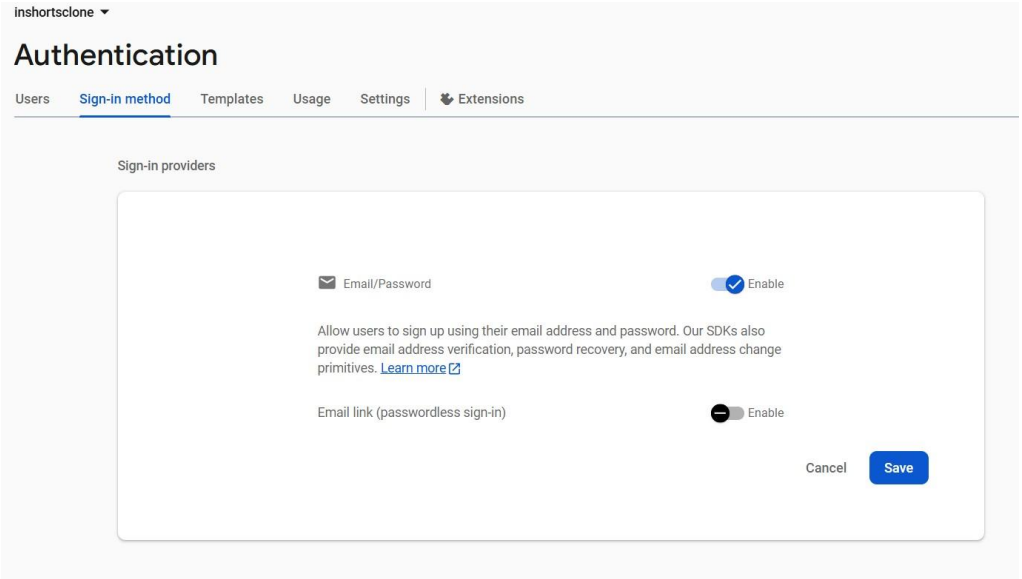
then select the build.gradle.kts (Kotlin DSL) part, and then follow the rest instructions



Generate the firebase_options.dart file, based on the google-services.json file

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

class DefaultFirebaseOptions {
  static FirebaseOptions get currentPlatform {
    return const FirebaseOptions(
      apiKey: "AIzaSyA_VIR7fj4d-b6tNZhW9qJ6GRRx5EXKqs0",
      appId: "1:388272292768:android:33180b2382688b18781ac5",
      messagingSenderId: "388272292768",
      projectId: "inshortsclone-848a9",
      storageBucket: "inshortsclone-848a9.firebasestorage.app",
      androidClientId: "1:388272292768:android:33180b2382688b18781ac5",
    );
  }
}
```

In your part select the sign-in method and enable it.



Code : auth_gate.dart

```dart
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:flutter_eats/pages/home_page.dart';
import 'package:flutter_eats/services/auth/login_or_register.dart';

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: StreamBuilder(
```

```
        stream: FirebaseAuth.instance.authStateChanges(),
        builder: (context, snapshot) {
          // user is logged in
          if(snapshot.hasData) {
            return const HomePage();
          }

          // user is Not logged in
          else {
            return const LoginOrRegister();
          }
        },
      ),
    );
  }
}
```

code : auth_service.dart

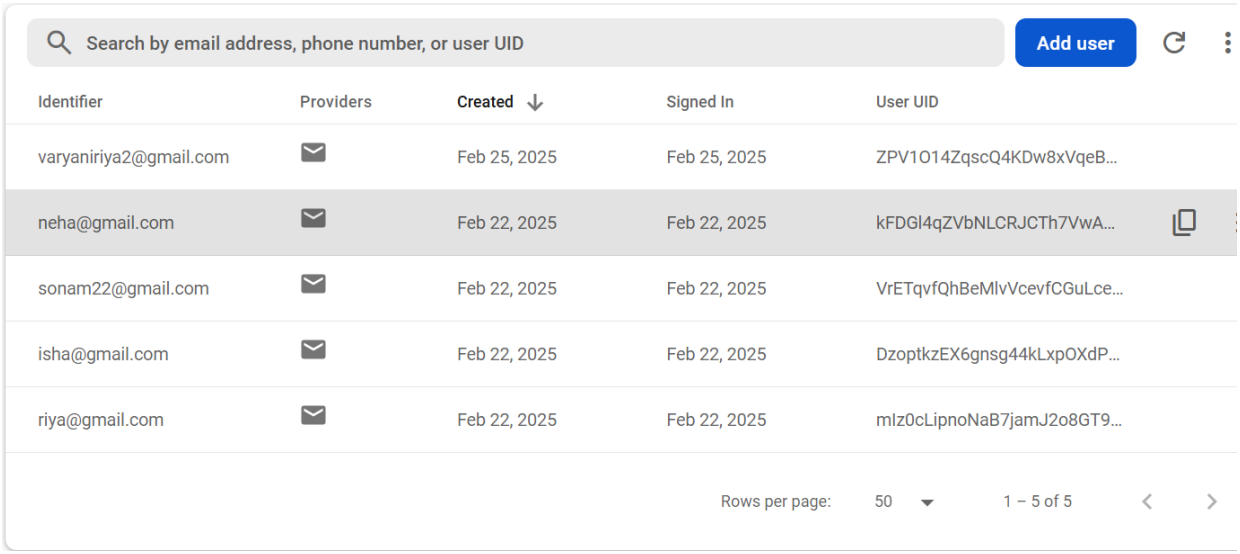
```
import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_auth/firebase_auth.dart';

class AuthService {
  // Get instance of Firebase Auth
  final FirebaseAuth _firebaseAuth = FirebaseAuth.instance;

  // Get current user
  User? getCurrentUser() {
    return _firebaseAuth.currentUser;
  }

  // Sign in
  Future<UserCredential?> signInWithEmailPassword(String email, String
password) async {
    // try sign user up
    try {
      // Sign in user
      UserCredential userCredential =
      await _firebaseAuth.signInWithEmailAndPassword(email: email,
password: password);
      return userCredential;
    }

    //catch any error
```

```
    on FirebaseAuthException catch (e) {
      throw Exception(e.code);
    }
  }


// Sign up (Placeholder for future functionality)
  Future<UserCredential?> signUpWithEmailPassword(String email, String
password) async {
  try {
  // Sign in user
  UserCredential userCredential =
  await _firebaseAuth.createUserWithEmailAndPassword(
  email: email,
  password: password);
  return userCredential;
  }

  //catch any error
  on FirebaseAuthException catch (e) {
  throw Exception(e.code);
  }
}
// Sign out (Placeholder for future functionality)

Future<void> signOut() async {
    return await _firebaseAuth.signOut();
}
}
```

code : login_or_register.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_eats/pages/login_page.dart';
import 'package:flutter_eats/pages/register_page.dart';

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

  @override
  State<LoginOrRegister> createState() => _LoginOrRegisterSate();
}

class _LoginOrRegisterSate extends State<LoginOrRegister> {
```

```
  //initially, shows login page
  bool showLoginPage = true;

  //toggle between login and register page

  void togglePages() {
    setState(() {
      showLoginPage = !showLoginPage;
    });
  }

  @override
  Widget build(BuildContext context) {
    if(showLoginPage) {
      return LoginPage(onTap: togglePages);
    } else {
      return RegisterPage(onTap: togglePages);
    }
     }
}
```



Conclusion : In this experiment, we successfully connected a Flutter UI with Firebase for authentication. We configured Firebase, integrated authentication services, and implemented user sign-in, sign-up, and sign-out functionalities. The AuthGate managed user state, while AuthService handled authentication operations. This project provides a foundation for secure user authentication in Flutter apps and can be extended with more Firebase features.

Experiment No. 7

Name:- Riya Varyani **Roll No:- 61**

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

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:-

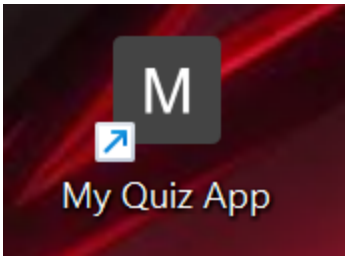
```
{
  "short_name": "Quizzy",
  "name": "My Quiz App",
  "description": "A Fun and Interactive Quiz Application",
  "icons": [
    {
      "src": "/icons/icon-192x192.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "/icons/icon-512x512.png",
      "sizes": "512x512",
```

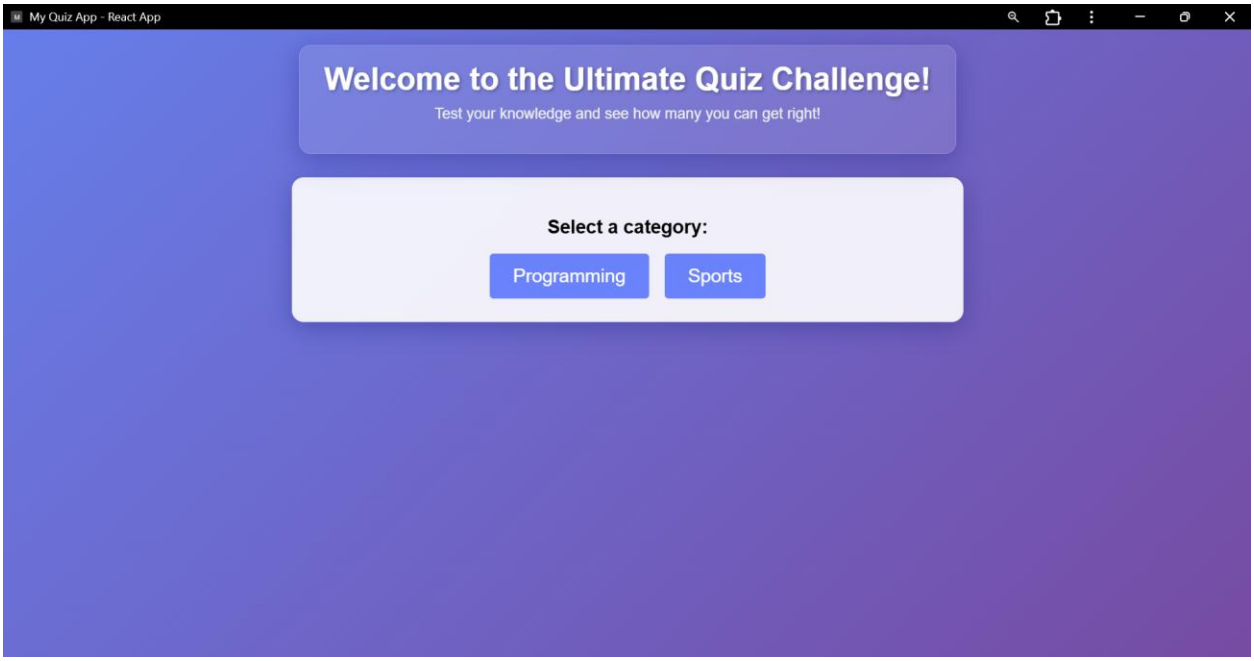
```
      "type": "image/png"
    }
  ],
  "start_url": "/index.html",
  "display": "standalone",
  "theme_color": "#4CAF50",
  "background_color": "#ffffff"
}
```

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

```
<html>
   <head>
      <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
      <script async src="https://www.googletagmanager.com/gtag/js?id=G-TWCSPJSQMW">
      </script>
```

Output:-





Conclusion:-

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

EXPERIMENT 8

Name	Roll No.
Chirag Choudhary	**10**
Riya Varyani	**61**

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

Theory:

Service Worker

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

Things to note about Service Worker:

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

What can we do with Service Workers?

- You can dominate **Network Traffic**

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

- You can **Cache**

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

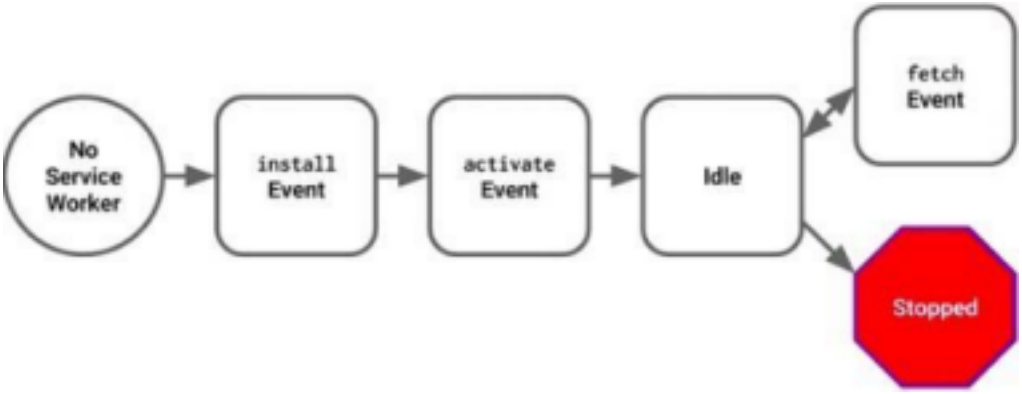
- You can manage **Push Notifications**
  You can manage push notifications with Service Worker and show any information message to the user.
- You can **Continue**
  Although Internet connection is broken, you can start any process with Background Sync of Service Worker.

What can't we do with Service Workers?

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

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

Service Worker Cycle



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

- Registration
- Installation
- Activation

Registration

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

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

```
    console.log('Service worker registration failed, error:', error);
  });
}
```

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

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

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

example: main.js

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

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

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

main.js

```
// Service Worker Script

const CACHE_NAME = 'blogbreeze-v1'; // Cache name to identify the version of cached content
const urlsToCache = [
  '/', // Home page
  'index.html', // Main HTML file
  'ani.html', // Any additional pages you want to cache
  '/src/assets/react.svg', // App icon
  '/public/icon.png', // Favicon
```

```
  '/src/main.jsx', // Main JS file for app functionality
// 'https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css' // FontAwesome for icons
];

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

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

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

// Service Worker Script
```

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

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

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

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

```
      .then((cachedResponse) => {
       // Return cached content if found, otherwise fetch from network
       return cachedResponse || fetch(event.request);
      })
  );
 });
navigator.serviceWorker.register('/app/service-worker.js',
 { scope: '/app'
 });
```

Installation

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

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

service-worker.js

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

Activation

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

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

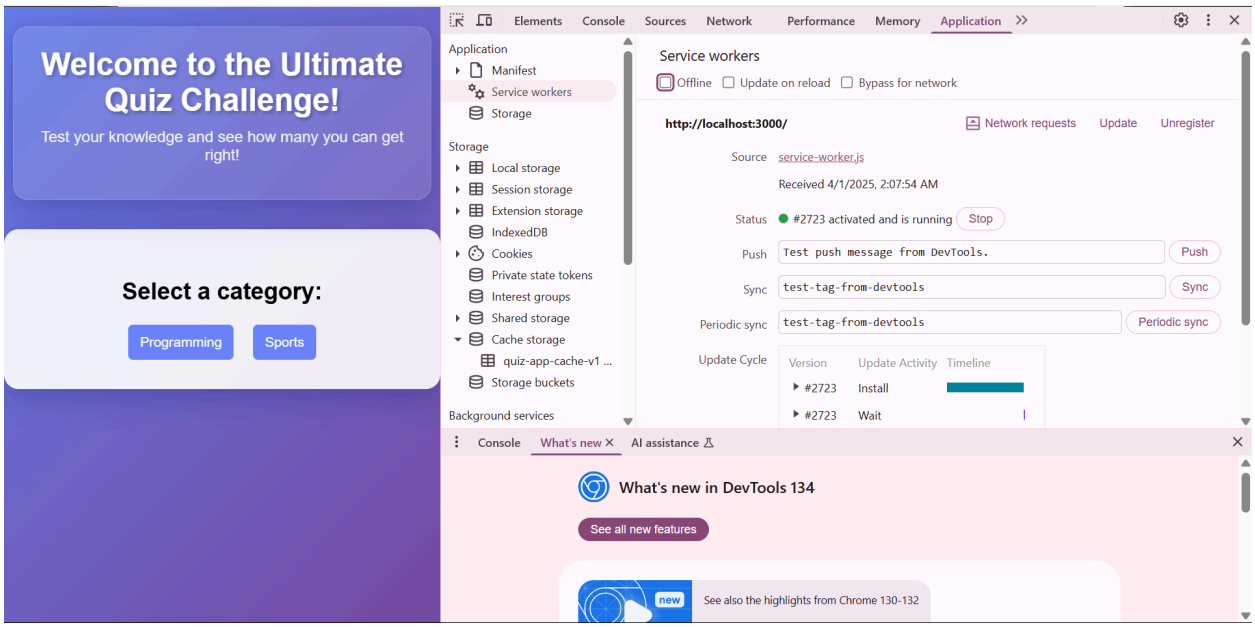
Once activated, the service worker controls all pages that load within its scope, and starts listening for

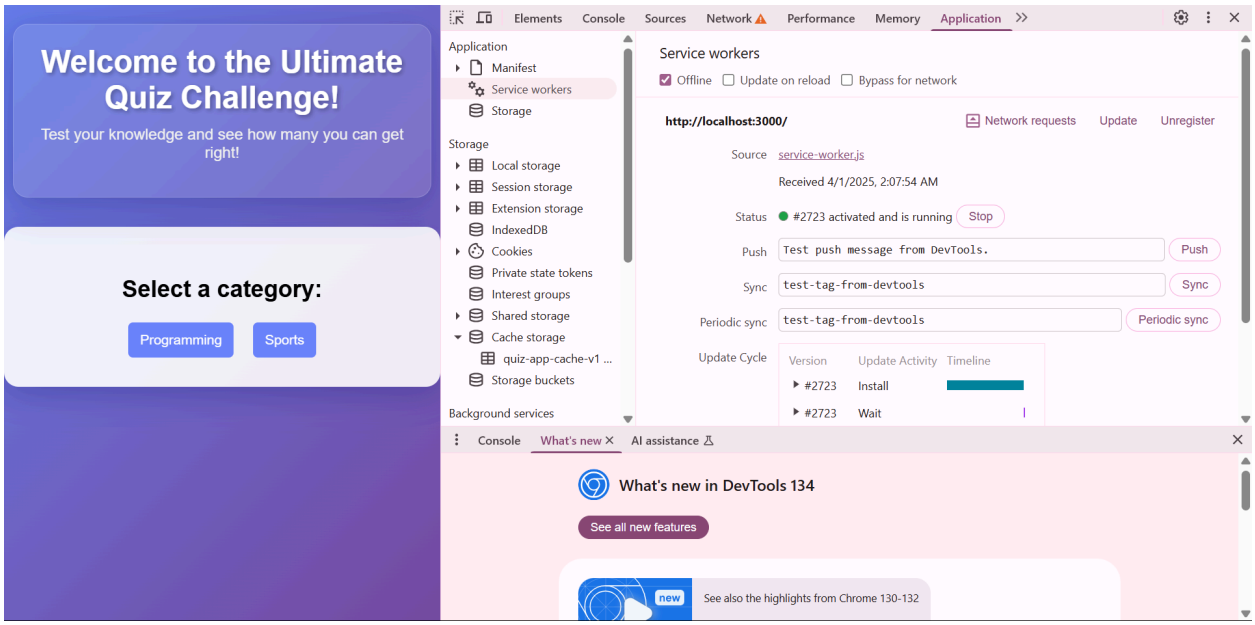
events from those pages. However, pages in your app that were loaded before the service worker activation will not be under service worker control. The new service worker will only take over when you close and reopen your app, or if the service worker calls **clients.claim()**. Until then, requests from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

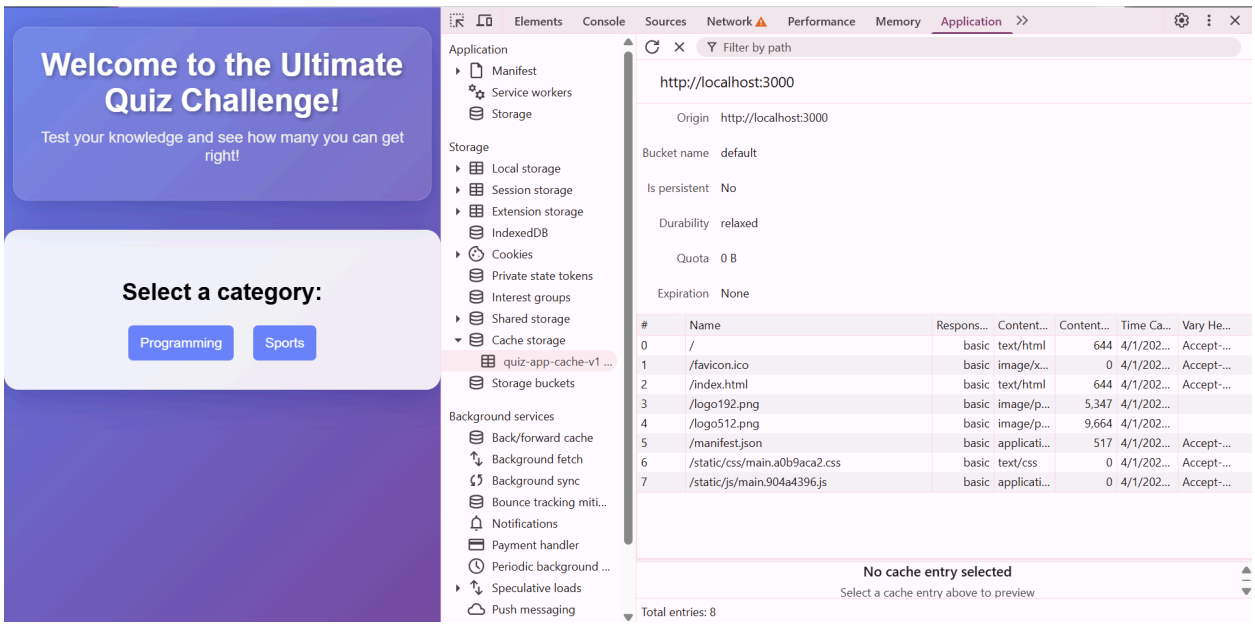
Code :

```
// Service Worker Script

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

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

// Activate Service Worker
self.addEventListener('activate', (event) => {
  console.log('Service Worker: Activated');
```

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

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


Welcome to the Ultimate Quiz Challenge!
Test your knowledge and see how many you can get right!
Select a category:
Programming
Sports
Elements
Console
Sources
Network
Performance
Memory
Application
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
quiz-app-cache-v1 ...
Storage buckets
Background services
Service workers
Offline
Update on reload
Bypass for network
http://localhost:3000/
Network requests
Update
Unregister
Source
service-worker.js
Received 4/1/2025, 2:07:54 AM
Status
#2723 activated and is running
Stop
Push
Test push message from DevTools.
Push
Sync
test-tag-from-devtools
Sync
Periodic sync
test-tag-from-devtools
Periodic sync
Update Cycle
Version
Update Activity
Timeline
#2723
Install
#2723
Wait
Console
What's new
AI assistance
What's new in DevTools 134
See all new features
new
See also the highlights from Chrome 130-132


Welcome to the Ultimate Quiz Challenge!
Test your knowledge and see how many you can get right!
Select a category:
Programming
Sports
Elements
Console
Sources
Network
Performance
Memory
Application
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
quiz-app-cache-v1 ...
Storage buckets
Background services
Service workers
Offline
Update on reload
Bypass for network
http://localhost:3000/
Network requests
Update
Unregister
Source
service-worker.js
Received 4/1/2025, 2:07:54 AM
Status
#2723 activated and is running
Stop
Push
Test push message from DevTools.
Push
Sync
test-tag-from-devtools
Sync
Periodic sync
test-tag-from-devtools
Periodic sync
Update Cycle
Version
Update Activity
Timeline
#2723
Install
#2723
Wait
Console
What's new
AI assistance
What's new in DevTools 134
See all new features
new
See also the highlights from Chrome 130-132


Welcome to the Ultimate Quiz Challenge!
Test your knowledge and see how many you can get right!
Select a category:
Programming
Sports
Elements
Console
Sources
Network
Performance
Memory
Application
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
quiz-app-cache-v1 ...
Storage buckets
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking miti...
Notifications
Payment handler
Periodic background ...
Speculative loads
Push messaging
Filter by path
http://localhost:3000
Origin
http://localhost:3000
Bucket name
default
Is persistent
No
Durability
relaxed
Quota
0 B
Expiration
None
Name Respons... Content... Content... Time Ca... Vary He...
0 / basic text/html 644 4/1/202... Accept-...
1 /favicon.ico basic image/x... 0 4/1/202... Accept-...
2 /index.html basic text/html 644 4/1/202... Accept-...
3 /logo192.png basic image/p... 5,347 4/1/202...
4 /logo512.png basic image/p... 9,664 4/1/202...
5 /manifest.json basic applicati... 517 4/1/202... Accept-...
6 /static/css/main.a0b9aca2.css basic text/css 0 4/1/202... Accept-...
7 /static/js/main.904a4396.js basic applicati... 0 4/1/202... Accept-...
No cache entry selected
Select a cache entry above to preview
Total entries: 8

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

MAD & PWA Lab
Journal

Experiment No.	09
Experiment Title.	To implement Service worker events like fetch, sync and push for E-commerce PWA
Roll No.	61
Name	Riya Varyani
Class	D15A
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

EXPERIMENT NO. 9

Aim: To implement Service worker events like fetch, sync and push for E-commerce PWA.
Theory:
Service Worker
Service Worker is a script that works on browser background without user interaction independently. Also, It resembles a proxy that works on the user side. With this script, you can track network traffic of the page, manage push notifications and develop “offline first” web applications with Cache API.

Things to note about Service Worker:

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

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

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

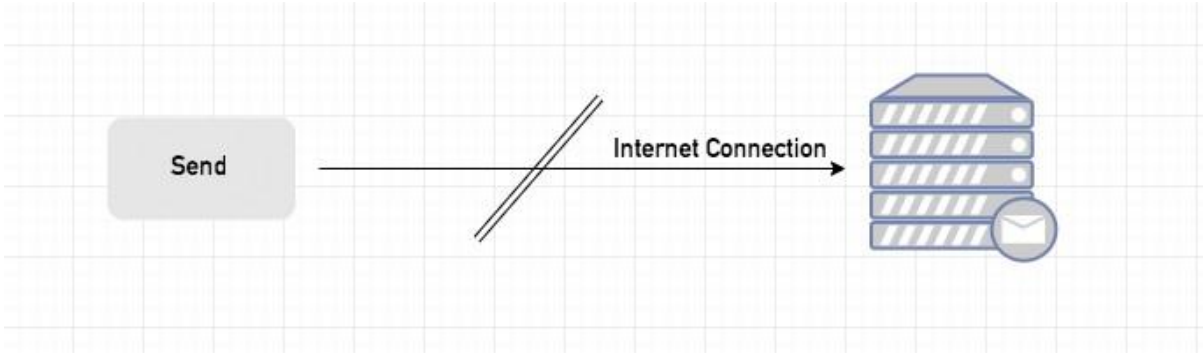
- **CacheFirst** - In this function, if the received request has cached before, the cached response is returned to the page. But if not, a new response requested from the network.
- **NetworkFirst** - In this function, firstly we can try getting an updated response from the network, if this process completed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

Sync Event

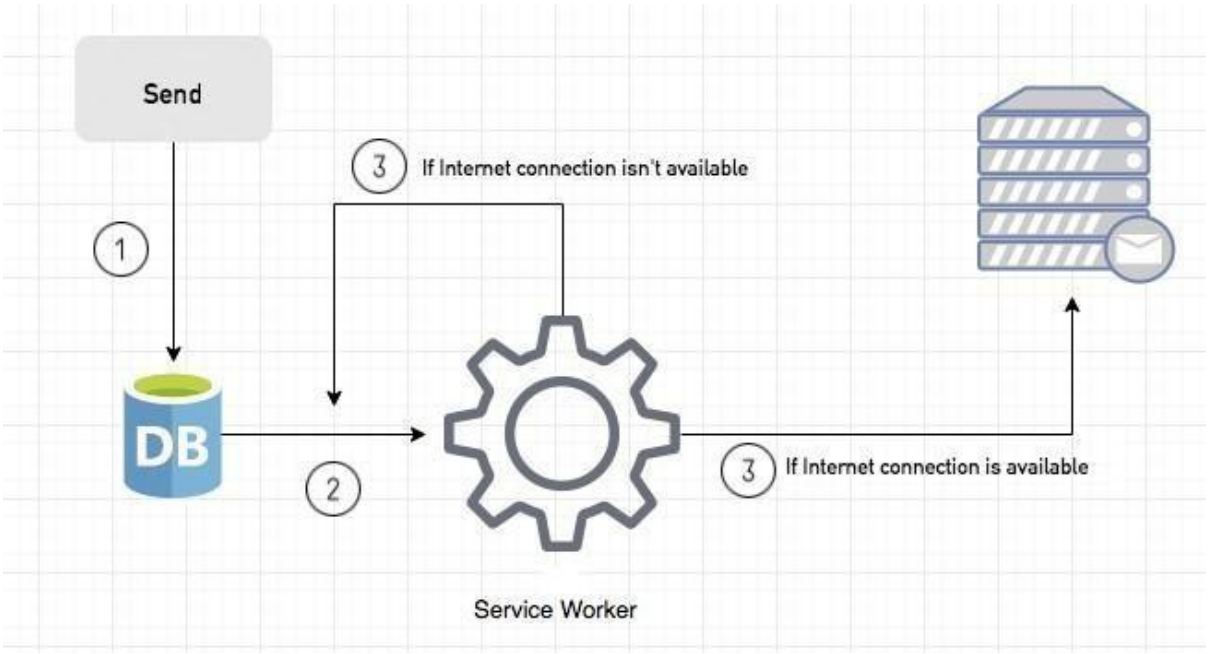
Background Sync is a Web API that is used to delay a process until the Internet connection is stable. We can adapt this definition to the real world; there is an e-mail client application that works on the browser and we want to send an email with this tool. Internet connection is broken while we are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

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

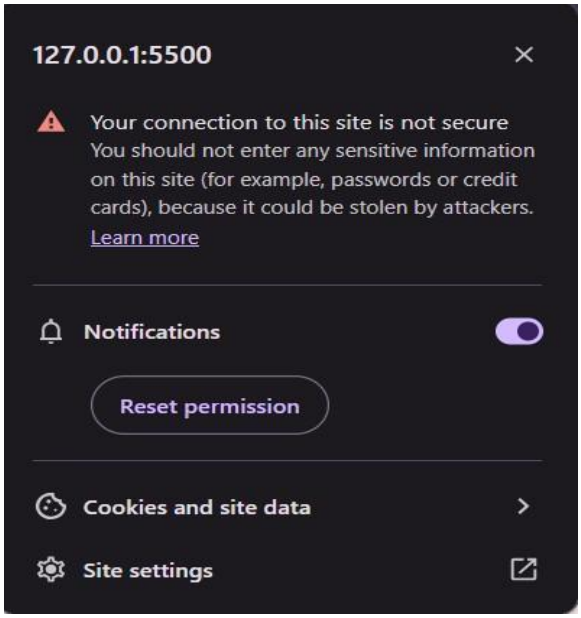
```
self.addEventListener('sync', event => {
    if (event.tag == 'helloSync') {
        console.log("helloSync [sw.js]");
    }
});
```

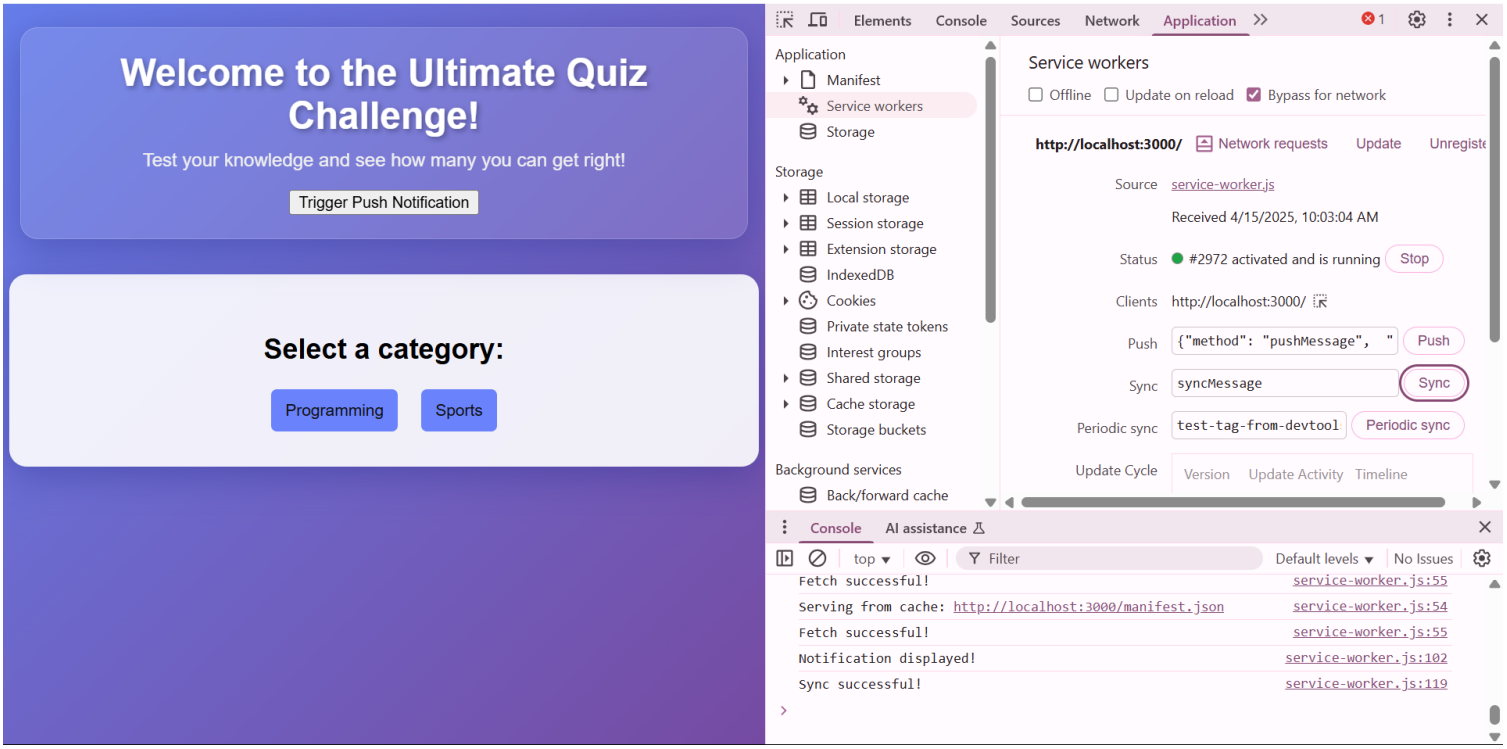
Push Event

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” proper.




Welcome to the Ultimate Quiz Challenge!
Test your knowledge and see how many you can get right!
Trigger Push Notification
Select a category:
Programming
Sports
Elements
Console
Sources
Network
Application
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Storage buckets
Background services
Back/forward cache
Service workers
Offline
Update on reload
Bypass for network
http://localhost:3000/
Network requests
Update
Source
service-worker.js
Received 4/15/2025, 10:03:04 AM
Status
#2972 activated and is running
Stop
Clients
http://localhost:3000/
Push
{"method": "pushMessage", "
Push
Sync
syncMessage
Sync
Periodic sync
test-tag-from-devtool
Periodic sync
Update Cycle
Version
Update Activity
Timeline
Console
AI assistance
top
Filter
Default levels
No Issues
Fetch successful!
service-worker.js:55
Serving from cache: http://localhost:3000/manifest.json
service-worker.js:54
Fetch successful!
service-worker.js:55
Notification displayed!
service-worker.js:102
Sync successful!
service-worker.js:119

EXPERIMENT NO: - 10

AIM: - To study and implement deployment of Ecommerce PWA to GitHub

Pages. **Theory: -**

GitHub Pages: Static Website Hosting Made Simple

GitHub Pages is a free hosting service that allows users to publish **public webpages directly from a GitHub repository**. It is particularly useful for **static websites, project documentation, and blogs**.

Key Features

- **Jekyll Integration**: Built-in support for Jekyll enables easy blogging.
- **Custom Domains**: Allows users to configure their own URLs.
- **Automatic Page Deployment**: Simply push your changes to the repository, and the updates go live.

Why Choose GitHub Pages?

- **Completely Free**: No hosting charges.
- **Seamless GitHub Integration**: Works directly with your repositories.
- **Quick Setup**: Just create a repository, push your files, and your site is live.

Who Uses GitHub Pages?

Companies like **Lyft, CircleCI, and HubSpot** use GitHub Pages for their documentation and static sites. It is widely adopted, appearing in **775 company stacks and 4,401 developer stacks**.

Pros & Cons

Pros

- Familiar interface for GitHub users.

- Simple deployment via the gh-pages branch.
- Supports custom domains with easy DNS configuration.

Cons

- Repositories need to be public unless you have a paid plan.
- Limited HTTPS support for custom domains (expected to improve).
- Jekyll plugins have limited support.

Firebase: A Full-Featured Real-Time Backend

Firebase is a cloud-based **real-time application platform** developed by Google. It enables developers to build **dynamic, collaborative applications** with ease.

Key Features

- **Real-Time Database**: Automatically syncs data across all connected clients.
- **Cloud-Based Storage**: JSON-based storage accessible via REST APIs.
- **Scalable Infrastructure**: Works well with existing services and scales automatically.
- **Authentication & Cloud Messaging**: Secure login and push notifications.

Why Choose Firebase?

- **Instant Backend Setup**: No need to build a separate backend.
- **Fast & Responsive**: Real-time data synchronization.
- **Built-in HTTPS**: Free SSL certificates for custom domains.

Who Uses Firebase?

Companies like **Instacart, 9GAG, and Twitch** rely on Firebase for their backend needs. Firebase is widely adopted, appearing in **1,215 company stacks and 4,651 developer stacks**.

Pros & Cons

Sonam chhabaidiya/ palak Chanchlani/ Mahi jodhani D15A 09/05/21

Pros

- **Hosted by Google**, ensuring reliability and security.
- **Comes with authentication, messaging, and real-time database services.**
- **Free HTTPS support** for all custom domains.

Cons

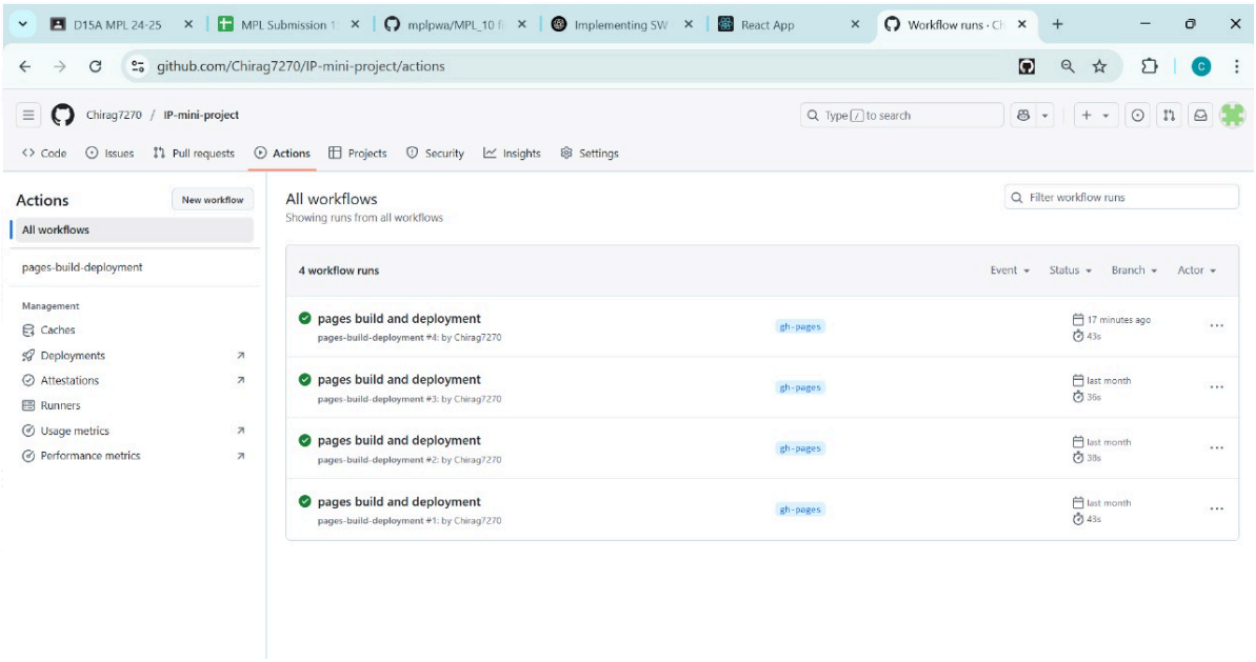
- **Limited Free Plan**: 10 GB of data transfer per month (can be mitigated with a CDN).
- **Command-Line Deployment**: No GUI for hosting.
- **No Built-in Static Site Generator Support**: Unlike GitHub Pages, Firebase doesn't natively support static site generators like Jekyll.

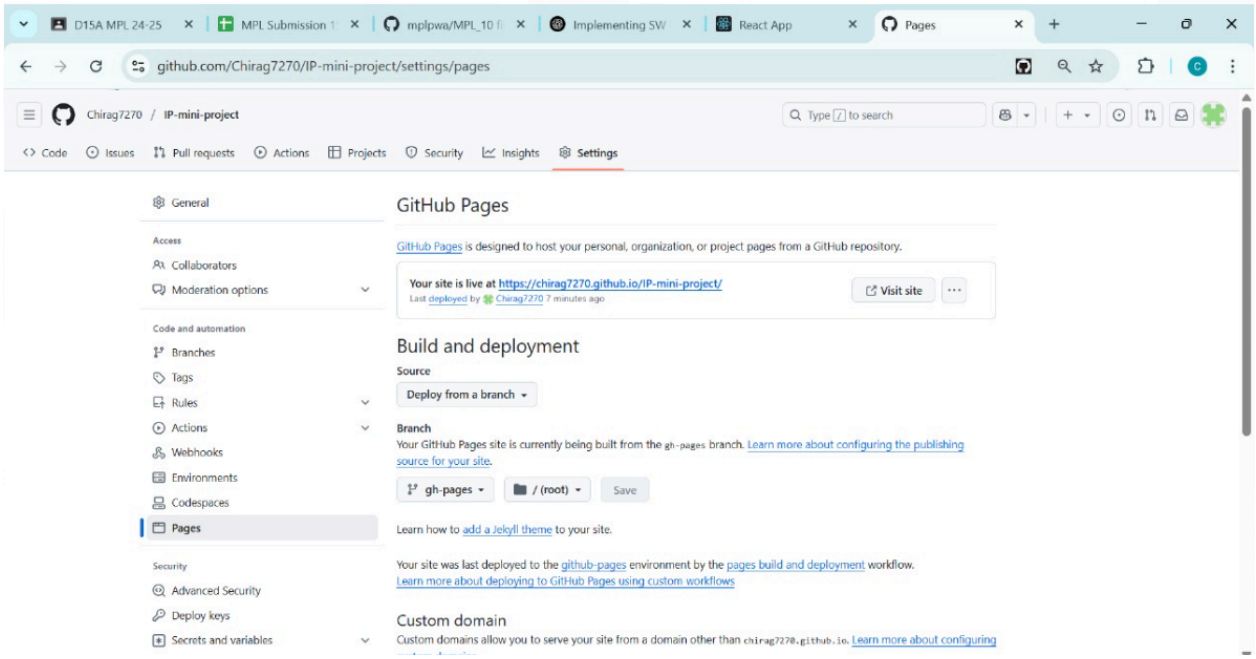
Link to our GitHub repository:

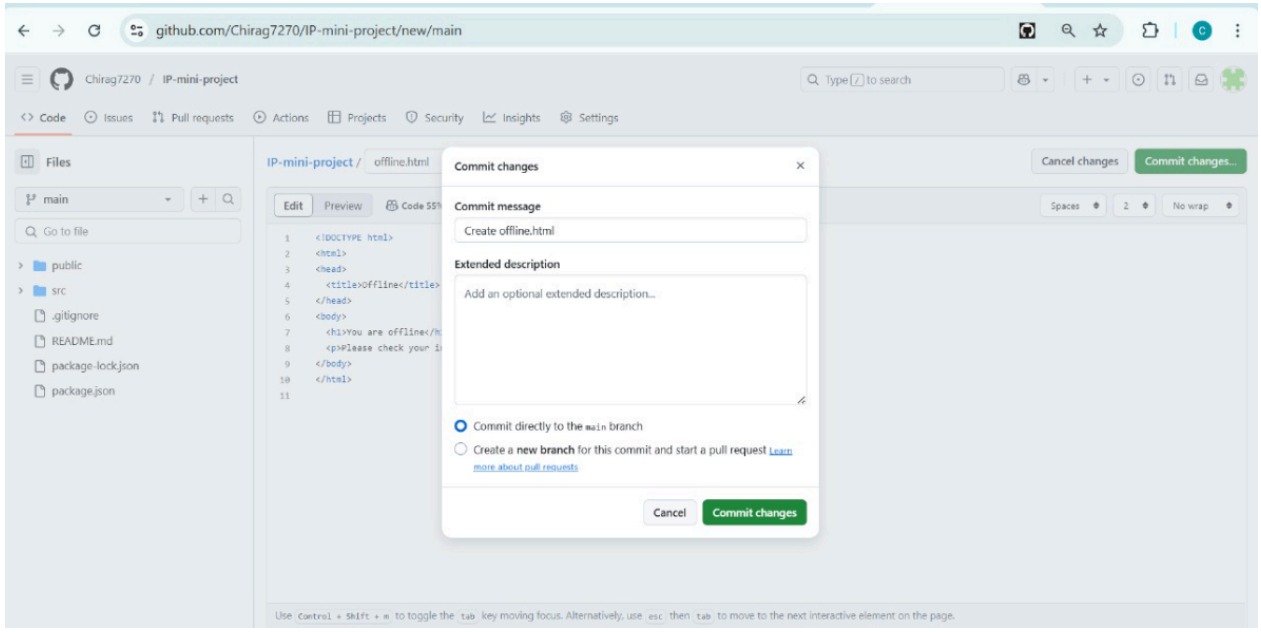
https://github.com/Chirag7270/IP-mini-project

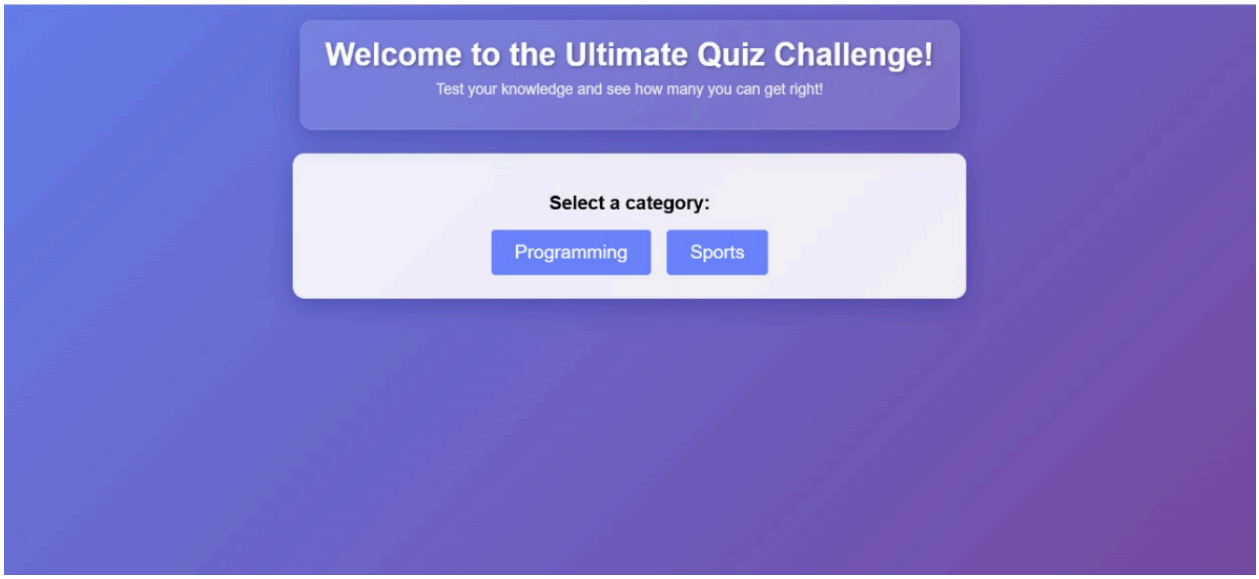
Hosted link:
https://chirag7270.github.io/IP-mini-project/

Github Screenshot:


github.com/Chirag7270/IP-mini-project/actions
Chirag7270 / IP-mini-project
Code
Issues
Pull requests
Actions
Projects
Security
Insights
Settings
Actions
New workflow
All workflows
pages-build-deployment
Management
Caches
Deployments
Attestations
Runners
Usage metrics
Performance metrics
All workflows
Showing runs from all workflows
Filter workflow runs
4 workflow runs
Event
Status
Branch
Actor
pages build and deployment
pages-build-deployment #4: by Chirag7270
gh-pages
17 minutes ago
43s
pages build and deployment
pages-build-deployment #3: by Chirag7270
gh-pages
last month
36s
pages build and deployment
pages-build-deployment #2: by Chirag7270
gh-pages
last month
38s
pages build and deployment
pages-build-deployment #1: by Chirag7270
gh-pages
last month
43s


github.com/Chirag7270/IP-mini-project/settings/pages
Chirag7270 / IP-mini-project
Code
Issues
Pull requests
Actions
Projects
Security
Insights
Settings
General
Access
Collaborators
Moderation options
Code and automation
Branches
Tags
Rules
Actions
Webhooks
Environments
Codespaces
Pages
Security
Advanced Security
Deploy keys
Secrets and variables
GitHub Pages
GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.
Your site is live at https://chirag7270.github.io/IP-mini-project/
Last deployed by Chirag7270 7 minutes ago
Visit site
Build and deployment
Source
Deploy from a branch
Branch
Your GitHub Pages site is currently being built from the gh-pages branch. Learn more about configuring the publishing source for your site.
gh-pages
/ (root)
Save
Learn how to add a Jekyll theme to your site.
Your site was last deployed to the github-pages environment by the pages build and deployment workflow.
Learn more about deploying to GitHub Pages using custom workflows
Custom domain
Custom domains allow you to serve your site from a domain other than chirag7270.github.io. Learn more about configuring


github.com/Chirag7270/IP-mini-project/new/main
Chirag7270 / IP-mini-project
Code
Issues
Pull requests
Actions
Projects
Security
Insights
Settings
Files
main
Go to file
public
src
.gitignore
README.md
package-lock.json
package.json
IP-mini-project / offline.html
Edit
Preview
Cancel changes
Commit changes...
Commit changes
Commit message
Create offline.html
Extended description
Add an optional extended description...
Commit directly to the main branch
Create a new branch for this commit and start a pull request
Learn more about pull requests
Cancel
Commit changes


Welcome to the Ultimate Quiz Challenge!
Test your knowledge and see how many you can get right!
Select a category:
Programming
Sports

PWA Experiment 11

Name	Roll No.
Chirag Choudhary	10
Riya Varyani	61

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

Theory :

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

Google Lighthouse :

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

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

Key Features and Audit Metrics

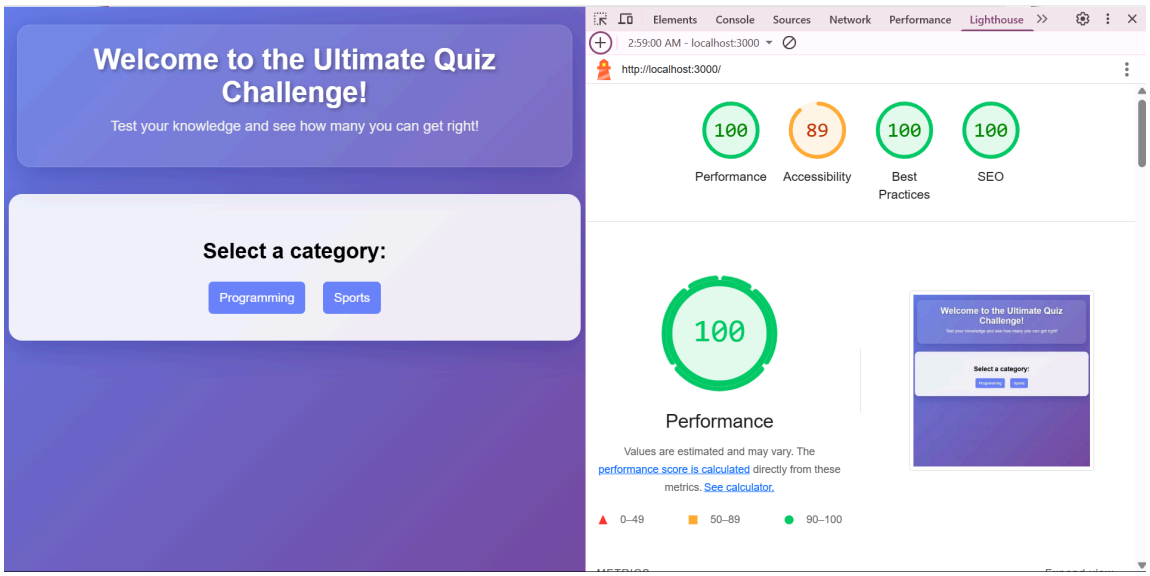
Google Lighthouse has the option of running the Audit for Desktop as well as mobile version of your page(s). The top metrics that will be measured in the Audit are:

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

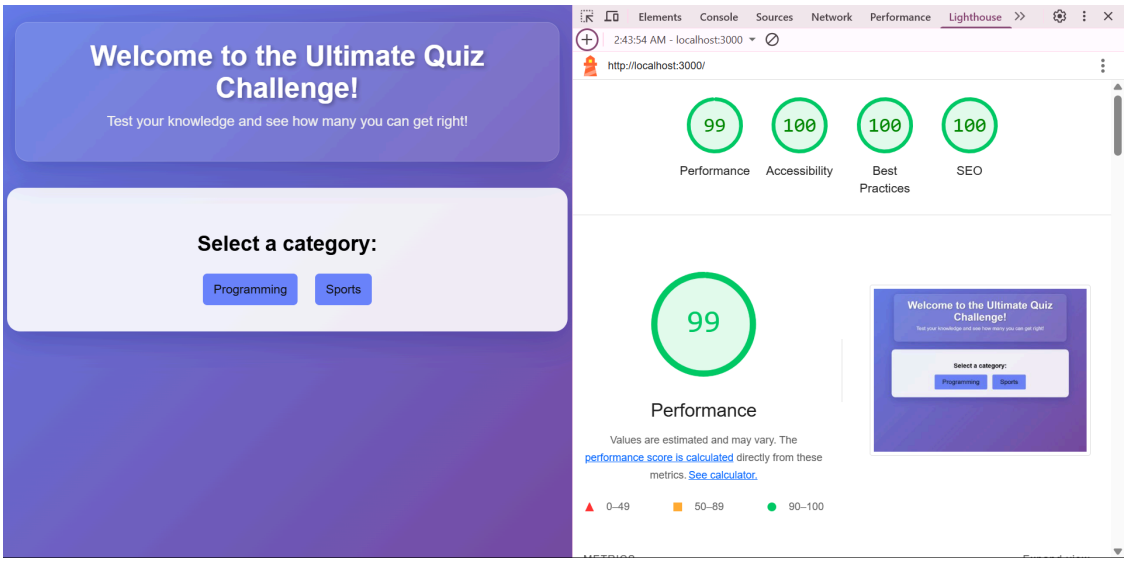
indicative of how decently the site performs, with a score of 100 meaning that you figure in the 98th percentile, 50 meaning that you figure in the 75th percentile and so on.

2. **PWA Score (Mobile):** Thanks to the rise of Service Workers, app manifests, etc., a lot of modern web applications are moving towards the PWA paradigm, where the objective is to make the application behave as close as possible to native mobile applications. Scoring points are based on the Baseline PWA checklist laid down by Google which includes Service Worker implementation(s), viewport handling, offline functionality, performance in script-disabled environments, etc.
3. **Accessibility:** As you might have guessed, this metric is a measure of how accessible your website is, across a plethora of accessibility features that can be implemented in your page (such as the 'aria-' attributes like aria-required, audio captions, button names, etc.). Unlike the other metrics though, Accessibility metrics score on a pass/fail basis i.e. if all possible elements of the page are not screen-reader friendly (HTML5 introduced features that would make pages easy to interpret for screen readers used by visually challenged people like tag names, tags such as <section>, <article>, etc.), you get a 0 on that score. The aggregate of these scores is your Accessibility metric score.
4. **Best Practices:** As any developer would know, there are a number of practices that have been deemed 'best' based on empirical data. This metric is an aggregation of many such points, including but not limited to:Use of HTTPS Avoiding the use of deprecated code elements like tags, directives, libraries, etc. Password input with paste-into disabled Geo-Location and cookie usage alerts on load, etc.

Performance before changes



Performance after changes



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