

Name: Aditya Raj

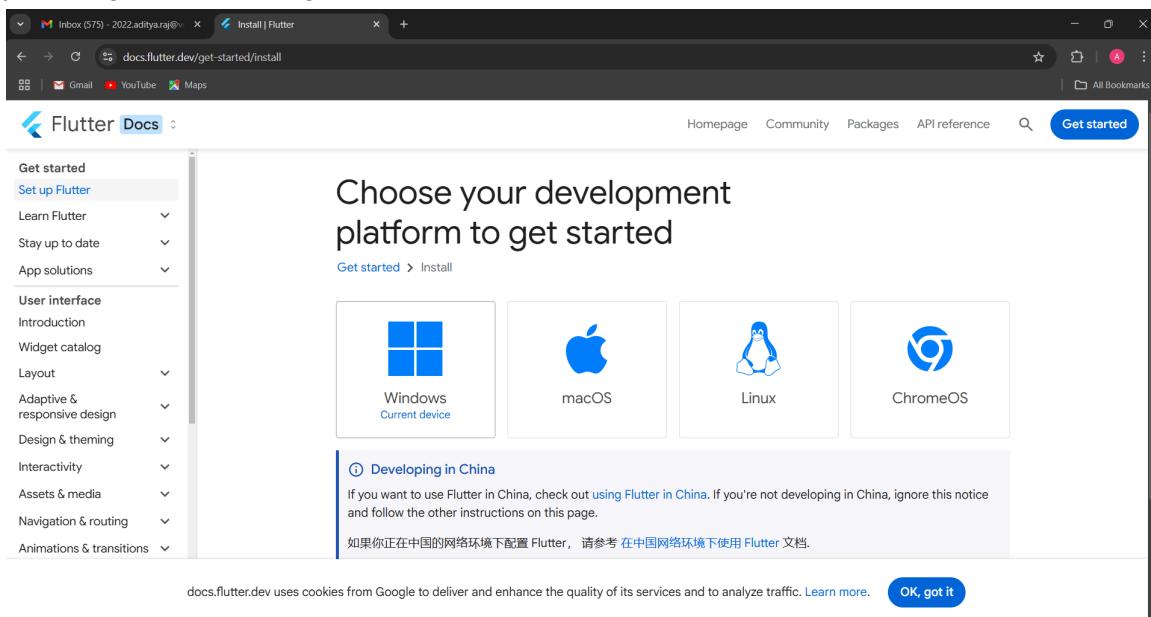
Div: D15B

Roll no: 01

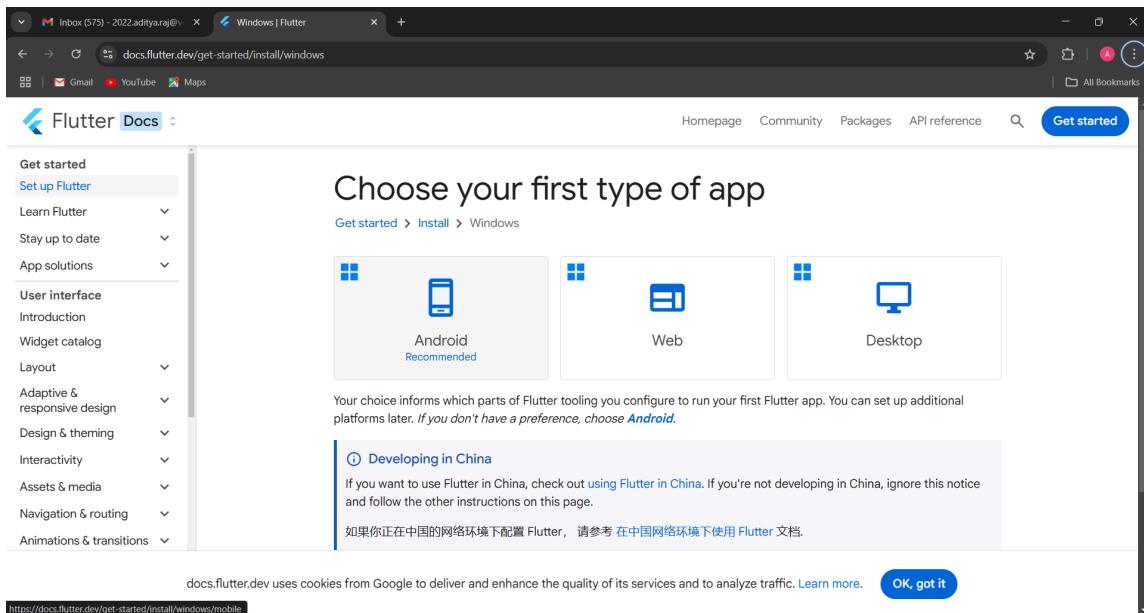
Experiment No:1

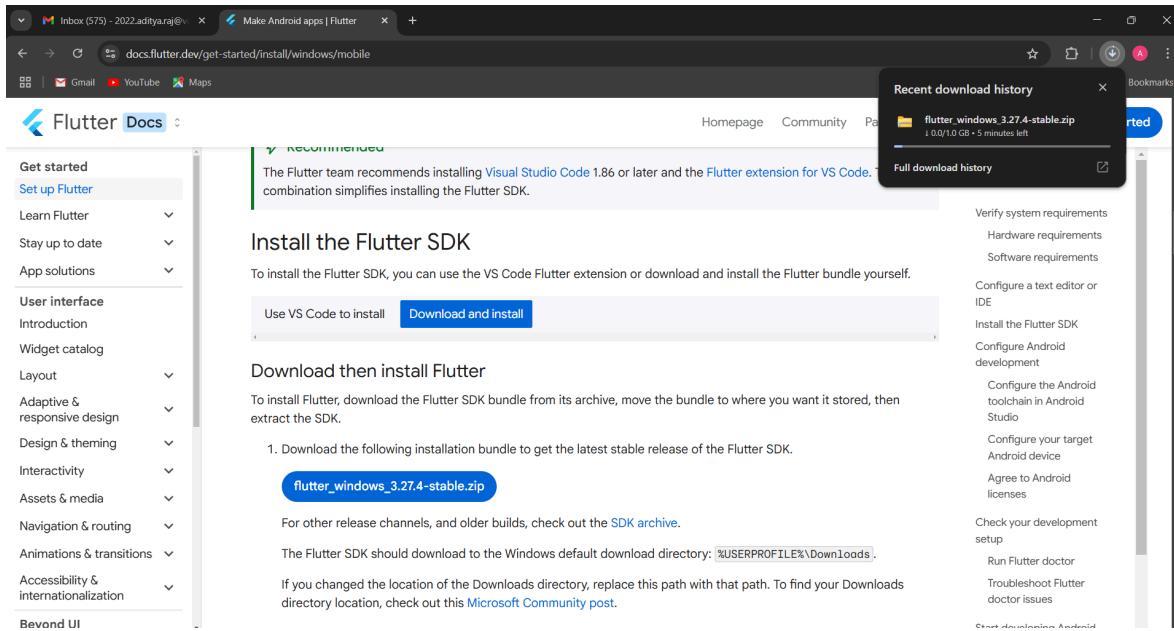
Aim:

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. Here, you will find the download link for SDK.

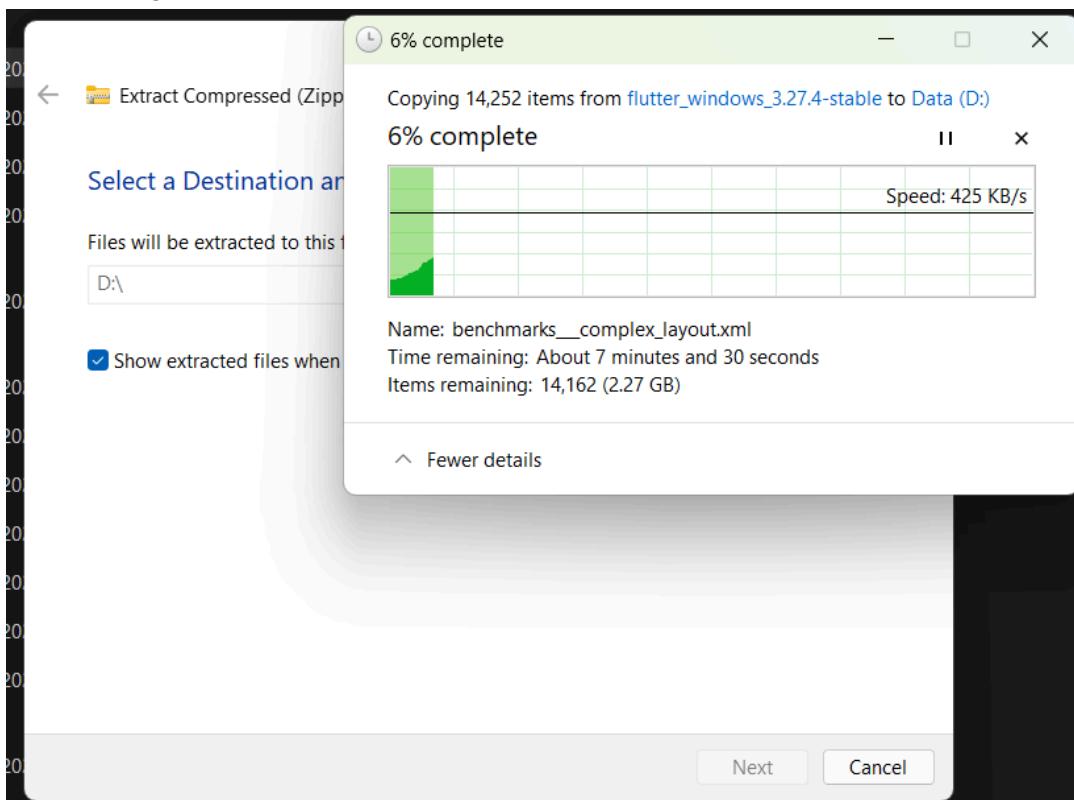




Step 3: When your download is complete, extract the zip file and place it in the desired installation folder or location, for example, C: /Flutter

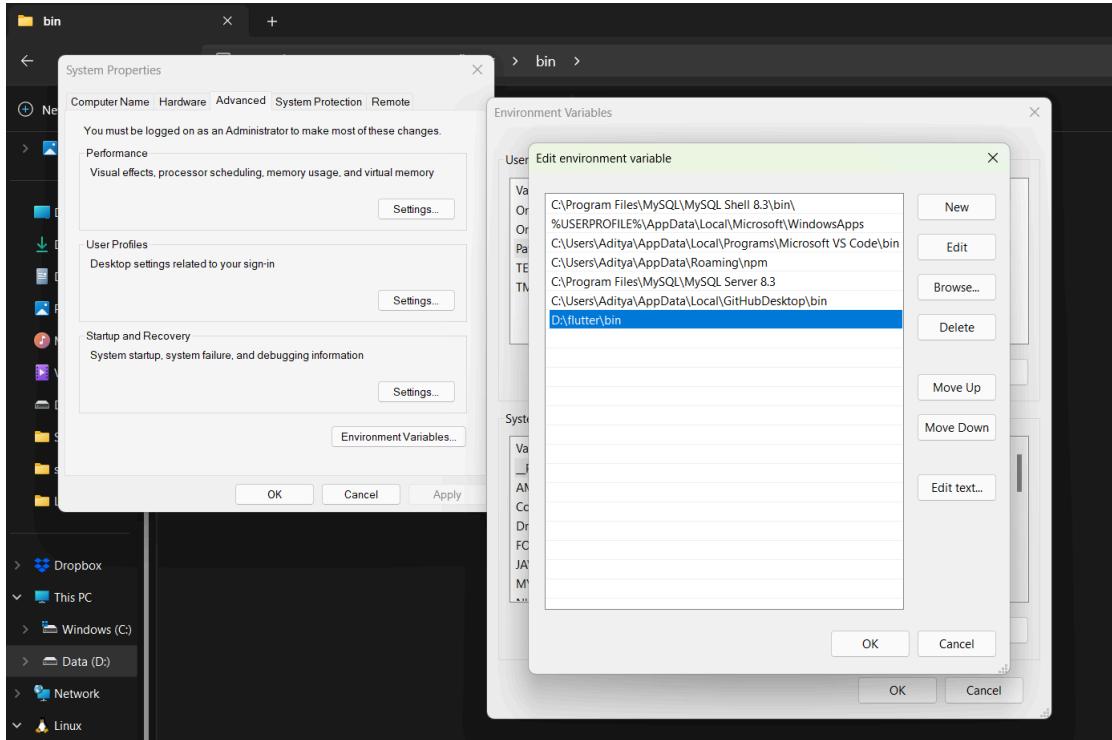
Step 4: To run the Flutter command in the regular windows console, you need to update the system path to include the flutter bin directory. The following steps are required to do this:

Step 4.1: Go to MyComputer properties -> advanced tab -> environment variables. You will get the following screen.

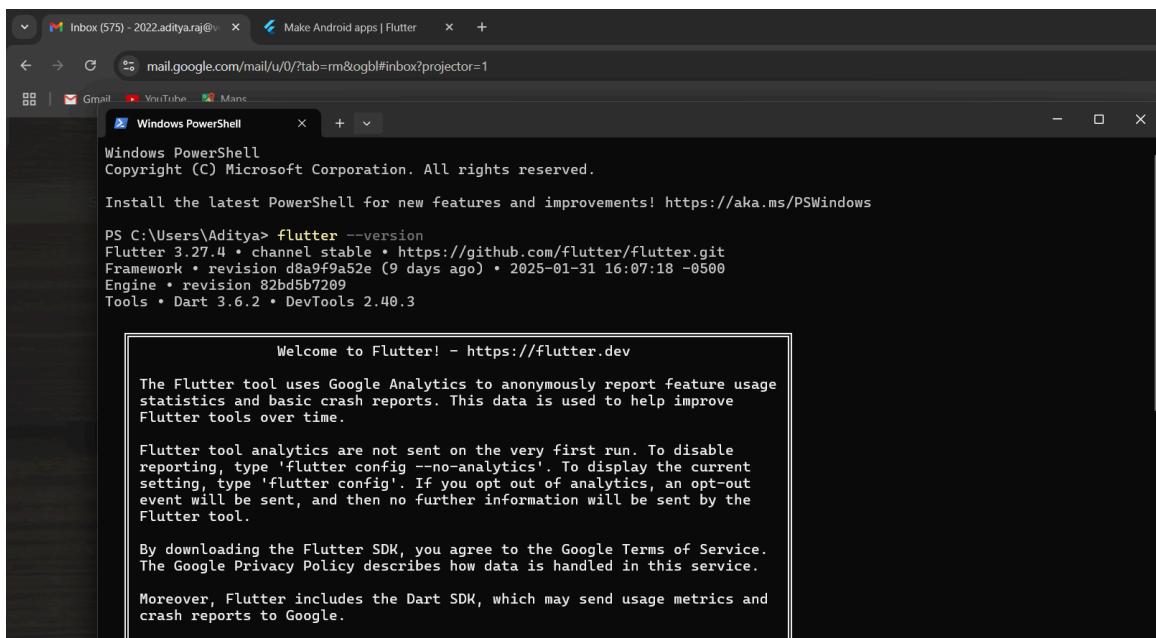


Step 4.2: Now, select path -> click on edit. The following screen appears

Step 4.3: In the above window, click on New->write path of Flutter bin folder in variable value - > ok -> ok -> ok

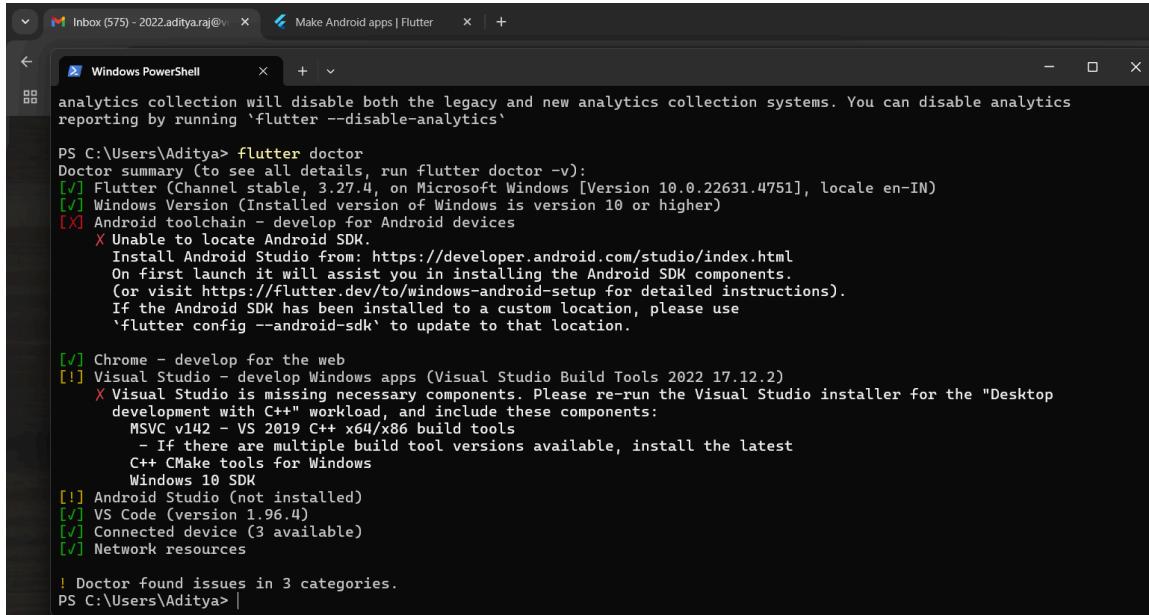


Step 5: Now, run the \$ flutter command in the command prompt.



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 6: 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 are required to run Flutter as well as the development tools that are available but not connected with the device.



```
analytics collection will disable both the legacy and new analytics collection systems. You can disable analytics reporting by running 'flutter --disable-analytics'

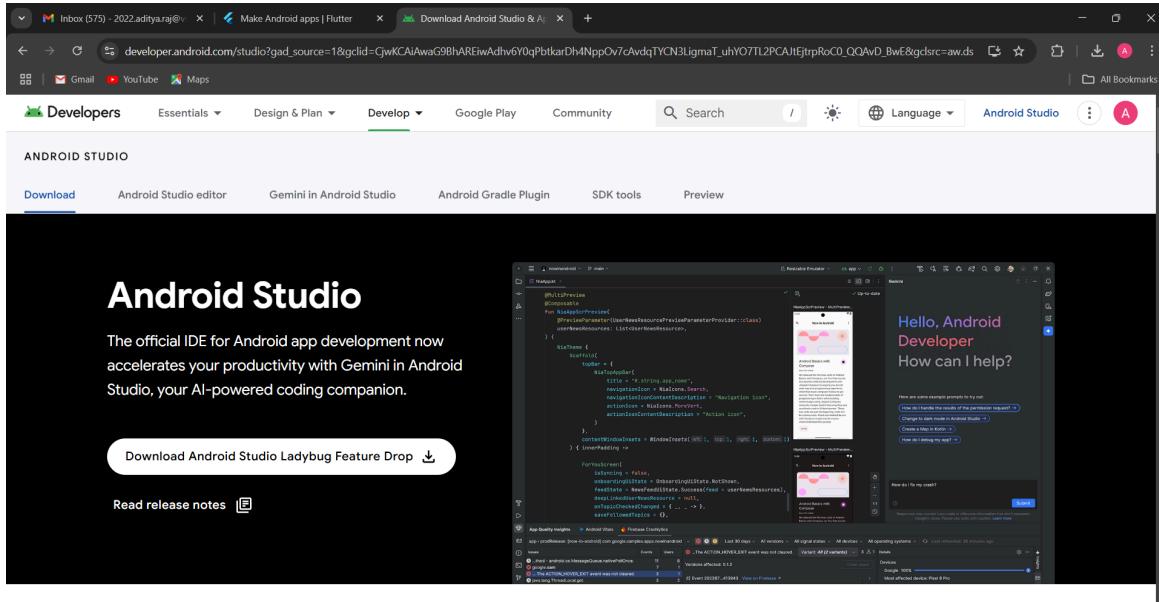
PS C:\Users\Aditya> flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.27.4, on Microsoft Windows [Version 10.0.22631.4751], locale en-IN)
[✓] Windows Version (Installed version of Windows is version 10 or higher)
[✗] Android toolchain - develop for Android devices
    ✗ Unable to locate Android SDK.
      Install Android Studio from: https://developer.android.com/studio/index.html
      On first launch it will assist you in installing the Android SDK components.
      (or visit https://flutter.dev/to/windows-android-setup for detailed instructions).
      If the Android SDK has been installed to a custom location, please use
        'flutter config --android-sdk' to update to that location.

[✓] Chrome - develop for the web
[!] Visual Studio - develop Windows apps (Visual Studio Build Tools 2022 17.12.2)
    ✗ Visual Studio is missing necessary components. Please re-run the Visual Studio installer for the "Desktop development with C++" workload, and include these components:
        MSVC v142 - VS 2019 C++ x64/x86 build tools
        - If there are multiple build tool versions available, install the latest
        C++ CMake tools for Windows
        Windows 10 SDK
[!] Android Studio (Not installed)
[✓] VS Code (version 1.96.4)
[✓] Connected device (3 available)
[✓] Network resources

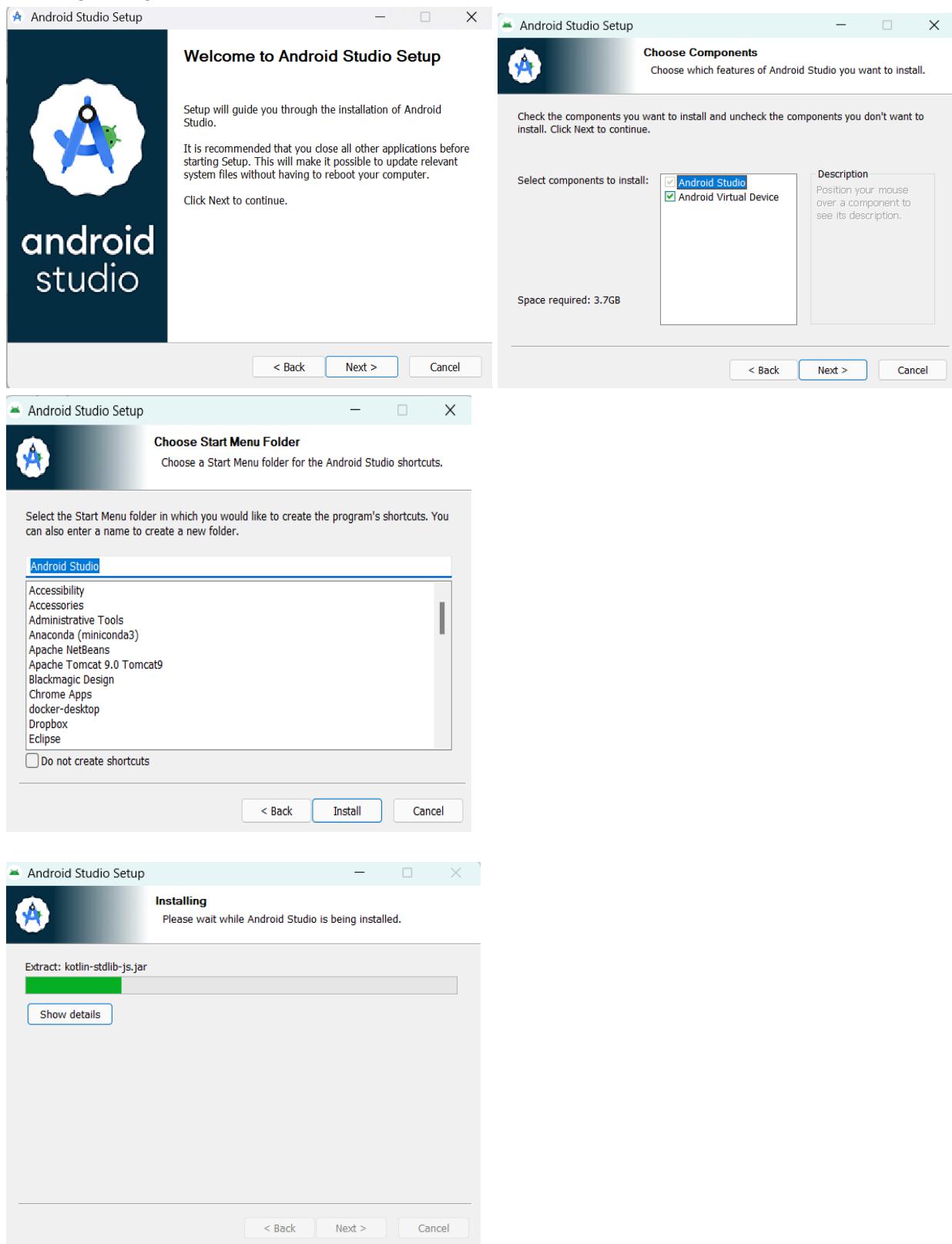
! Doctor found issues in 3 categories.
PS C:\Users\Aditya>
```

Step 7: 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 7.1: Download the latest Android Studio executable or zip file from the official site.



Step 7.2: When the download is complete, open the .exe file and run it. You will get the following dialog box

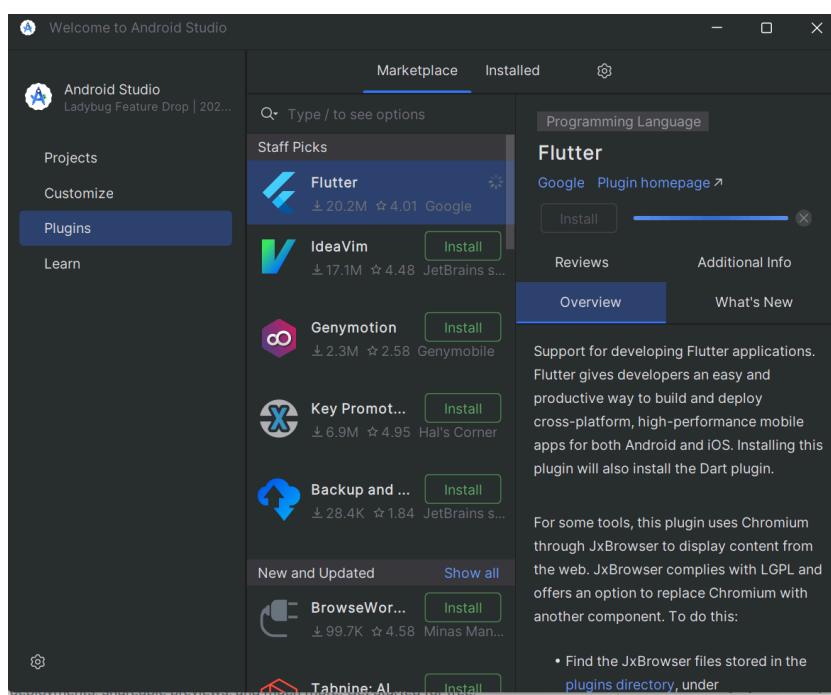
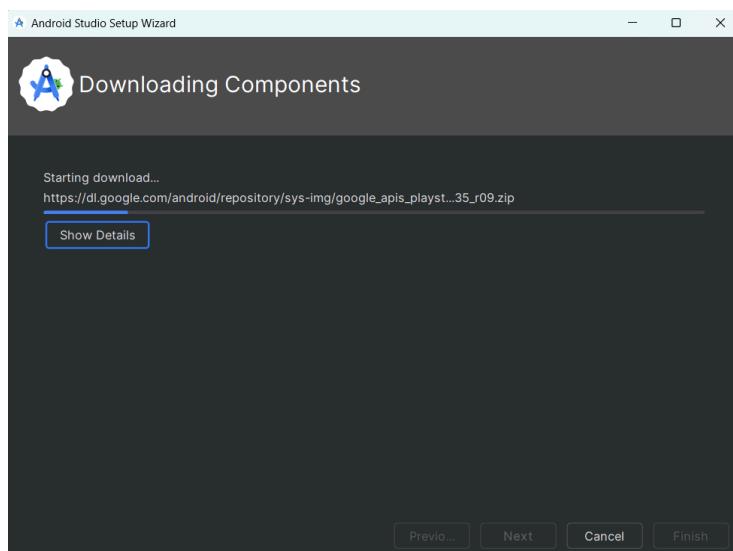


Step 7.3: Follow the steps of the installation wizard. Once the installation wizard completes, you will get the following screen.

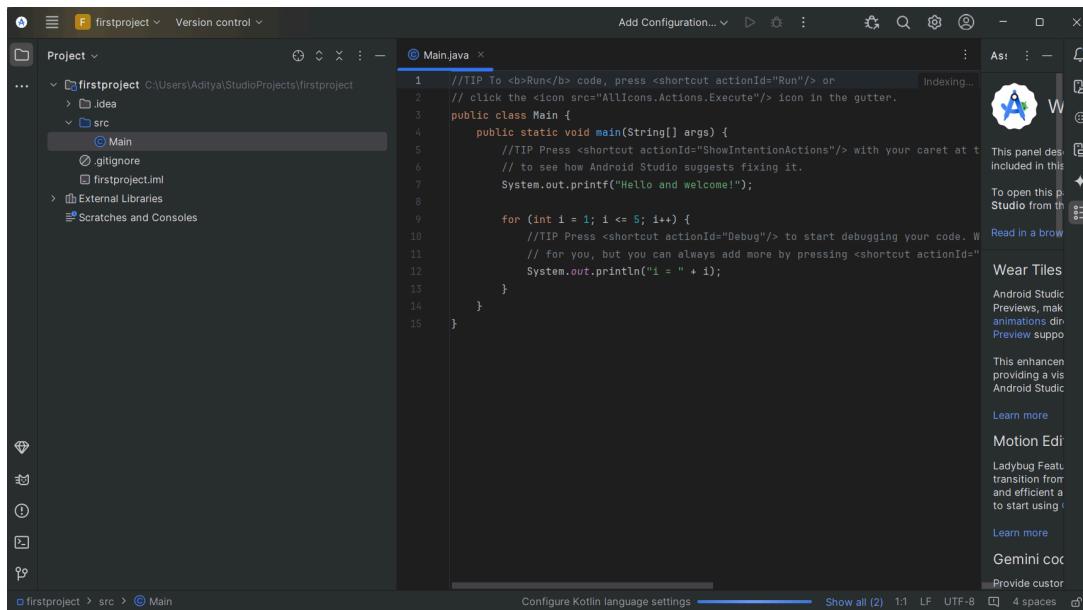
Step 7.4: In the above screen, click Next-> Finish. Once the Finish button is clicked, you need to choose the 'Don't import Settings option' and click OK. It will start the Android Studio.

Step 8: Next, you need to set up an Android emulator. It is responsible for running and testing the Flutter application.

Step 8.1: To set an Android emulator, go to Android Studio > Tools > Android > AVD Manager and select Create Virtual Device. Or, go to Help->Find Action->Type Emulator in the search box. You will get the following screen.

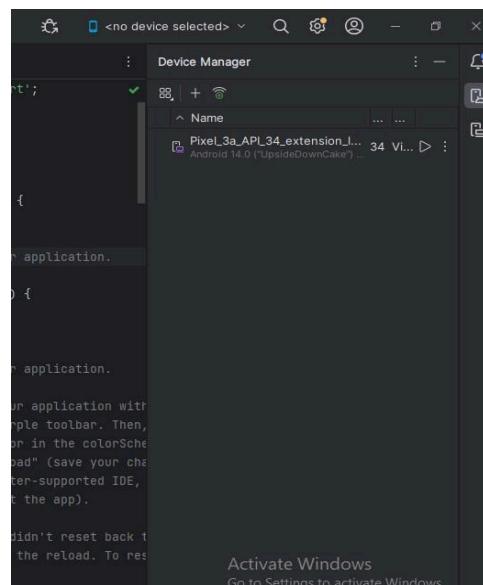


Step 8.2: Choose your device definition and click on Next.

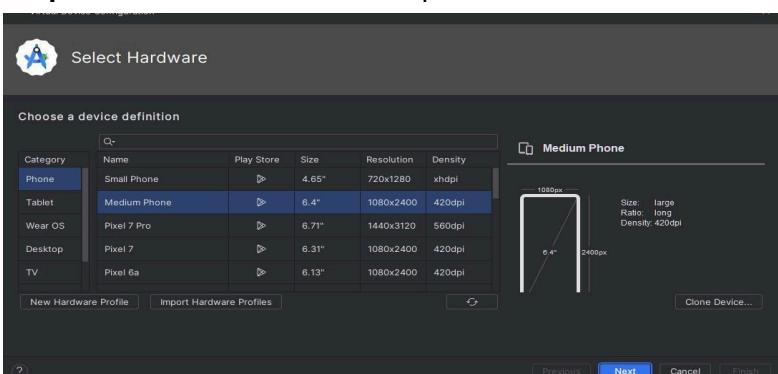


Step 8.3: Select the system image for the latest Android version and click on Next.

Step 8.4: Now, verify the all AVD configuration. If it is correct, click on Finish. The following screen appears.

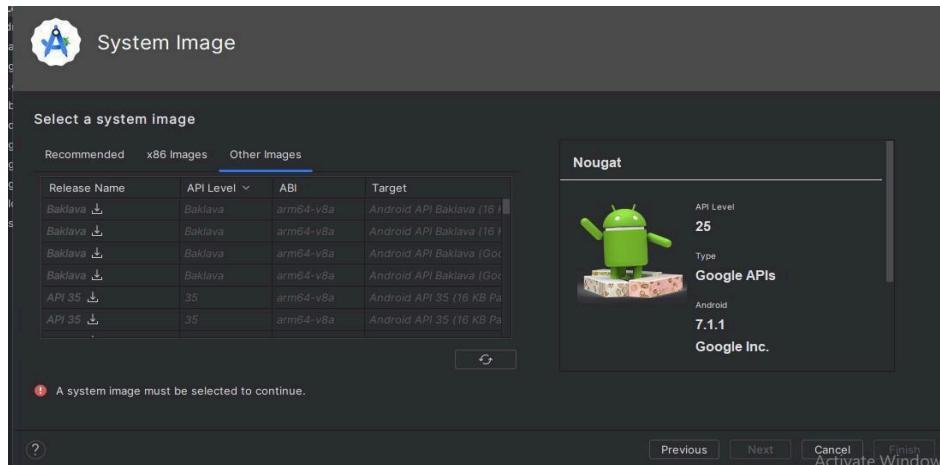


Step 8.5: Last, click on the icon pointed into the red color rectangle. The Android emulator



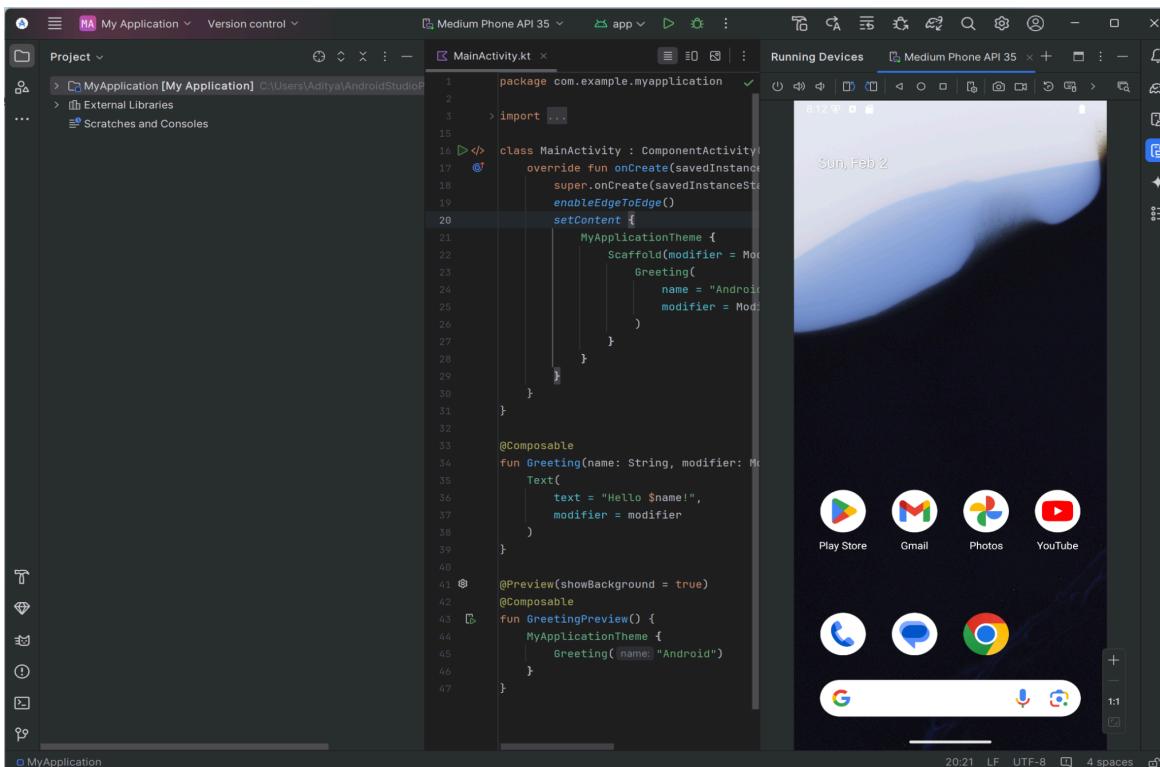
Step 9: Now, install Flutter and Dart plugin for building Flutter applications in Android Studio. These plugins provide a template to create a Flutter application, give an option to run and debug Flutter application in the Android Studio itself. Do the following steps to install these plugins.

Step 9.1: Open the Android Studio and then go to File->Settings->Plugins.



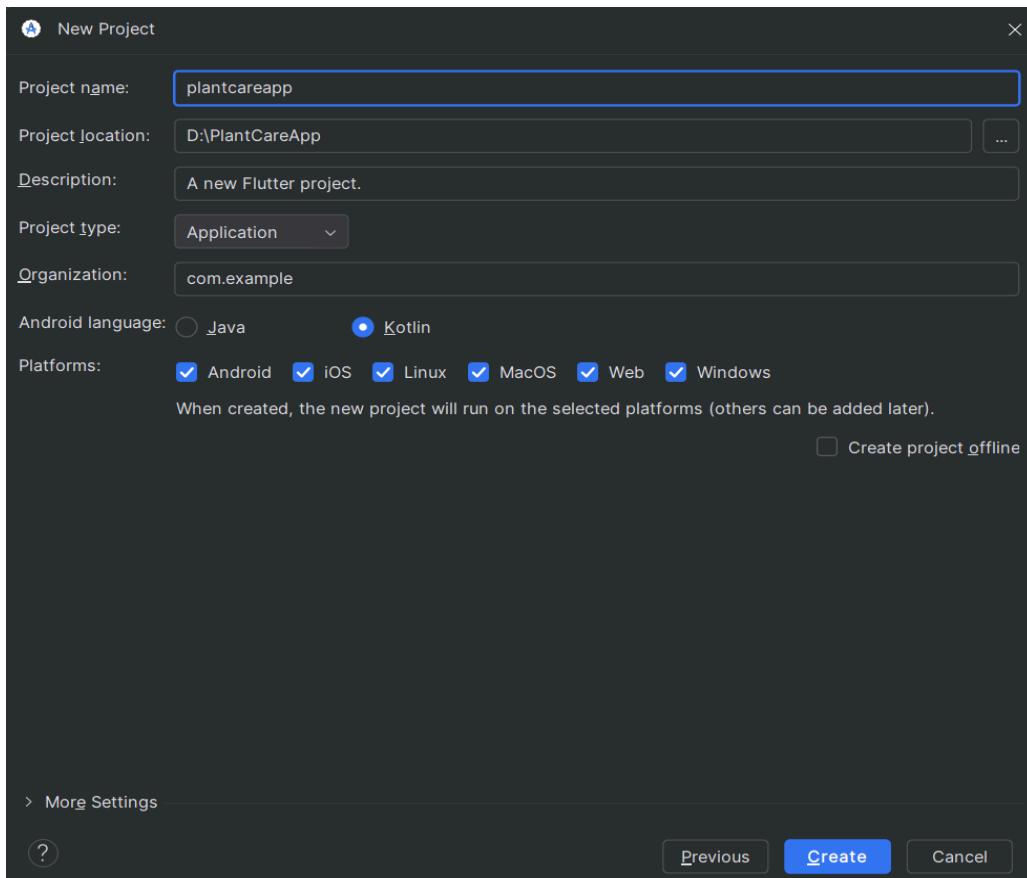
Step 9.2: Now, search the Flutter plugin. If found, select Flutter plugin and click install. When you click on install, it will ask you to install the Dart plugin as shown below screen. Click yes to proceed.

Step 9.3: Restart the Android Studio.

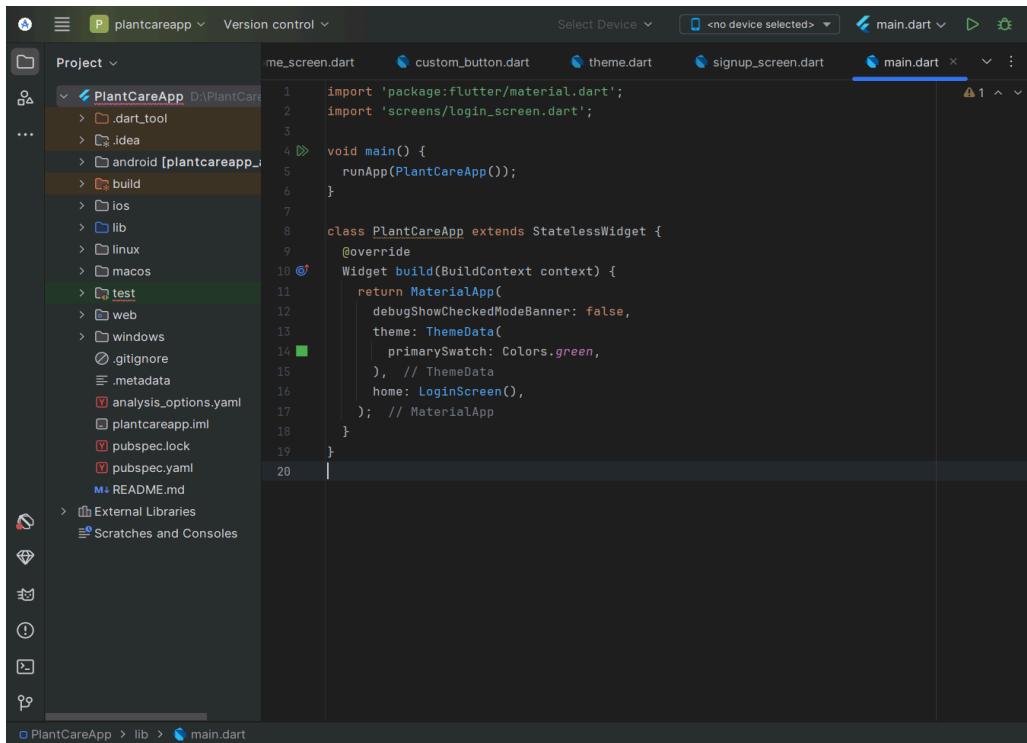


Aim : To design flutter UI by including common widgets.

Creating a new flutter project



main.dart

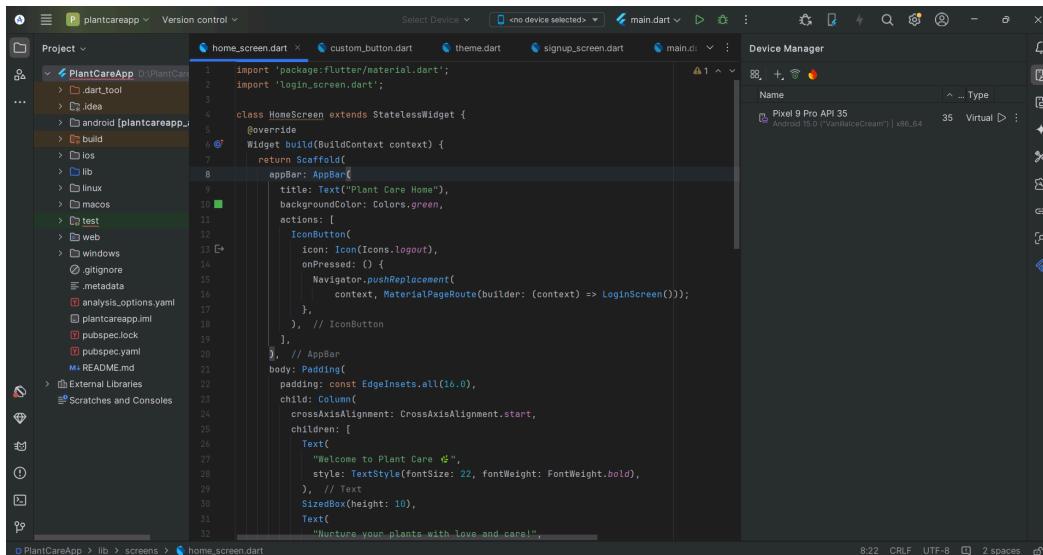


The screenshot shows the Android Studio interface with the main.dart file open in the code editor. The code defines the main application class, PlantCareApp, which extends StatelessWidget. It overrides the build method to return a MaterialApp with a green theme, a home screen of LoginScreen, and a debugShowCheckedModeBanner set to false.

```
import 'package:flutter/material.dart';
import 'screens/login_screen.dart';
void main() {
  runApp(PlantCareApp());
}

class PlantCareApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primarySwatch: Colors.green,
      ), // ThemeData
      home: LoginScreen(),
    ); // MaterialApp
}
```

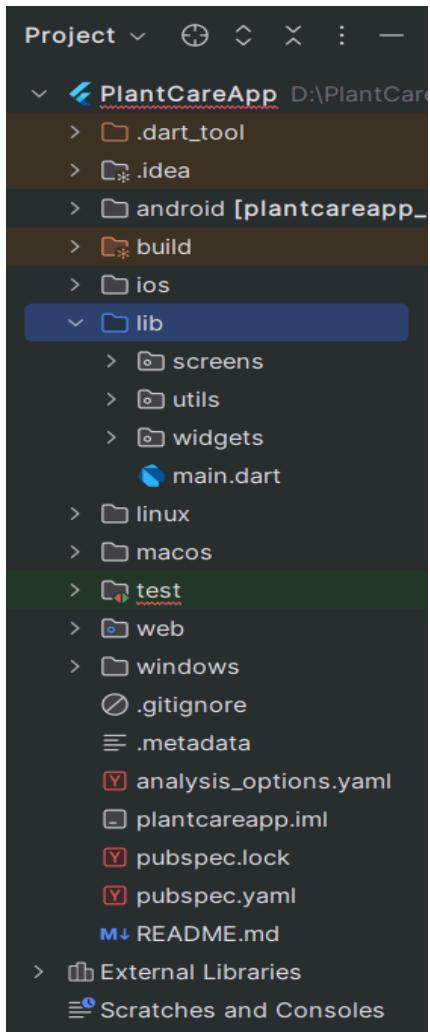
Adding virtual device: Configuring a virtual device (emulator) for testing the Flutter app.



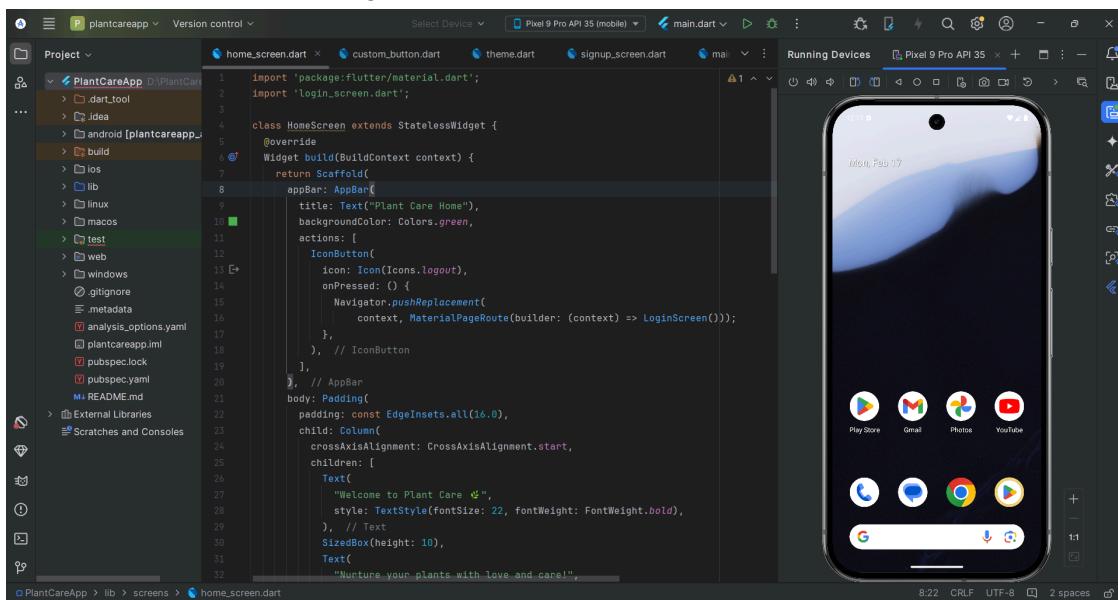
The screenshot shows the Android Studio interface with the home_screen.dart file open in the code editor. The code defines the HomeScreen StatelessWidget, which contains a Scaffold with an AppBar and a body containing a Column of Text widgets. The Device Manager panel on the right shows a single virtual device entry: Pixel 9 Pro API 35 (Android 15.0 ("VanillaIceCream") | x86_64).

```
import 'package:flutter/material.dart';
import 'login_screen.dart';
class HomeScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Plant Care Home"),
        backgroundColor: Colors.green,
        actions: [
          IconButton(
            icon: Icon(Icons.logout),
            onPressed: () {
              Navigator.pushReplacement(
                context, MaterialPageRoute(builder: (context) => LoginScreen());
              },
            ),
          ), // IconButton
        ],
      ), // AppBar
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              "Welcome to Plant Care 🌱",
              style: TextStyle(fontSize: 22, fontWeight: FontWeight.bold),
            ), // Text
            SizedBox(height: 10),
            Text(
              "Nurture your plants with love and care!",
            ),
          ],
        ),
      ),
    );
  }
}
```

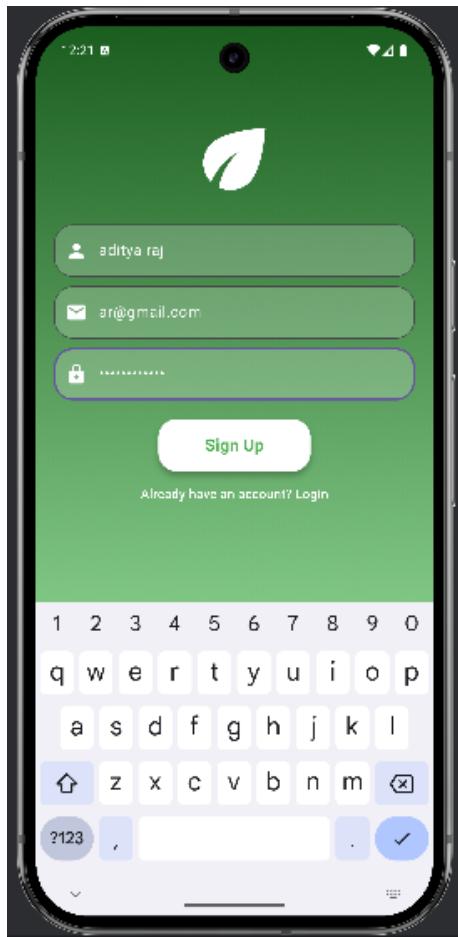
File structure: Displaying the organized directory structure of the Flutter project.



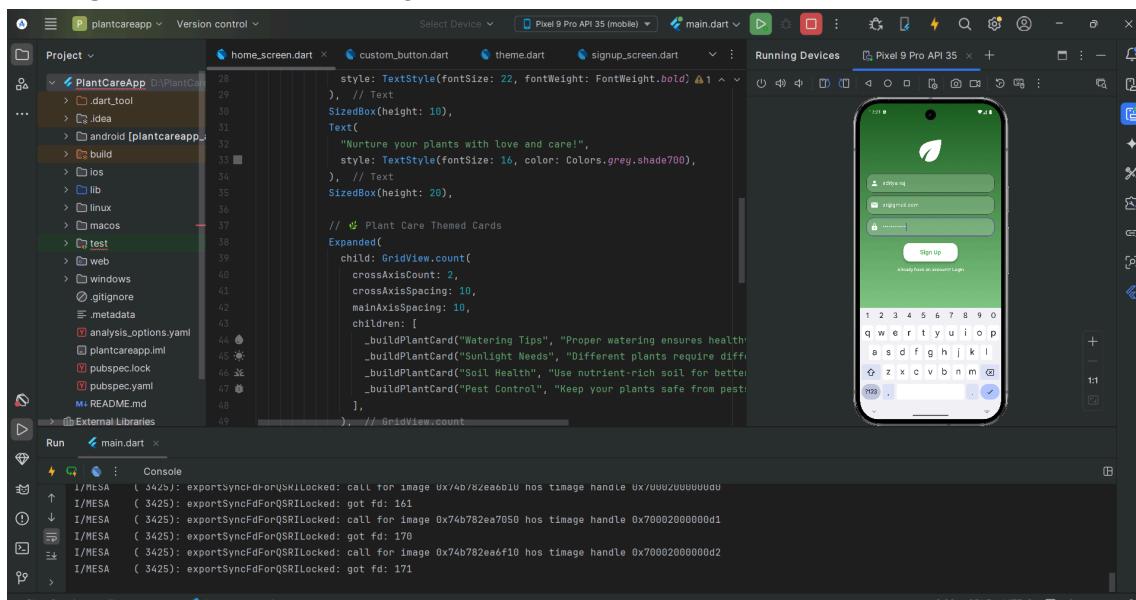
Start the emulator: Running the emulator to test and visualize the Flutter UI.



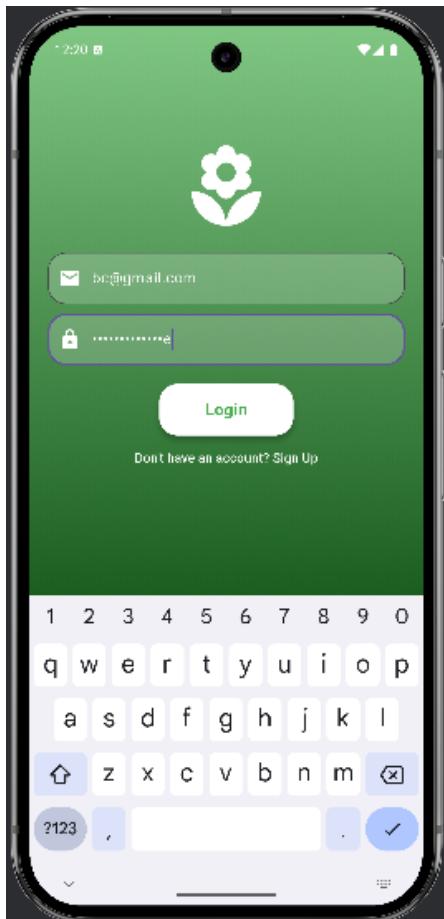
Sign-up page with widgets and buttons: User-friendly signup with text fields and buttons.



Adding icons in UI: Enhancing the interface with icons for better user experience



The **login page** features input fields, a submit button, and form validation, providing a seamless user authentication experience.



Login page code integrates **TextFields** for input, an **ElevatedButton** for submission, and form validation, ensuring a smooth user authentication experience.

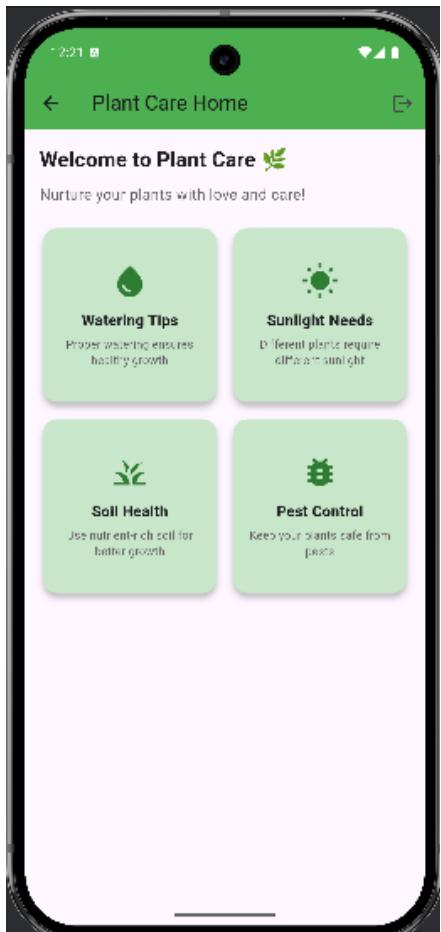
The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The code editor displays the `home_screen.dart` file, which contains Dart code for building a user interface. The interface includes an AppBar with a logo, a search bar, and a login button. Below this is a `GridView` containing four cards with plant-related information: Watering Tips, Sunlight Needs, Soil Health, and Pest Control. The bottom of the screen shows a keyboard and the Android emulator interface.

The **home page** code implements a `GridView/ListView`, `Card` widgets, and an `AppBar` with icons, creating an interactive and visually appealing plant care interface.

This screenshot shows the same setup as the previous one, but the `GridView` now displays a more refined version of the plant care interface. The cards are larger and have rounded corners. The card for "Watering Tips" includes an image of a green plant. The overall layout is cleaner and more visually appealing.

The home page of the Plant Care App features an intuitive layout with plant images, `Card` widgets for plant details, a search bar, and interactive buttons. It uses `GridView/ListView` for

plant display and a BottomNavigationBar for navigation with icons for a seamless user experience.



Conclusion : The experiment successfully demonstrated Flutter UI design by implementing common widgets, icons, images, and fonts. It provided hands-on experience in creating interactive and visually appealing mobile interfaces.

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

Code :

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

class HomeScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(
          "Plant Care Home",
          style: TextStyle(fontFamily: 'CustomFont', fontSize: 22),
        ),
        backgroundColor: Colors.green,
        actions: [
          IconButton(
            icon: Icon(Icons.logout),
            onPressed: () {
              Navigator.pushReplacement(
                context, MaterialPageRoute(builder: (context) => LoginScreen()));
            },
          ),
        ],
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            // 🌱 Welcome Section
            Text(
              "Welcome to Plant Care 🌱",
              style: TextStyle(
                fontSize: 22,
                fontWeight: FontWeight.bold,
                fontFamily: 'CustomFont',
              ),
            ),
            SizedBox(height: 10),
            Text(
              "Nurture your plants with love and care!",
            ),
          ],
        ),
      ),
    );
  }
}
```

```

        style: TextStyle(fontSize: 16, color: Colors.grey.shade700),
    ),
    SizedBox(height: 20),
}

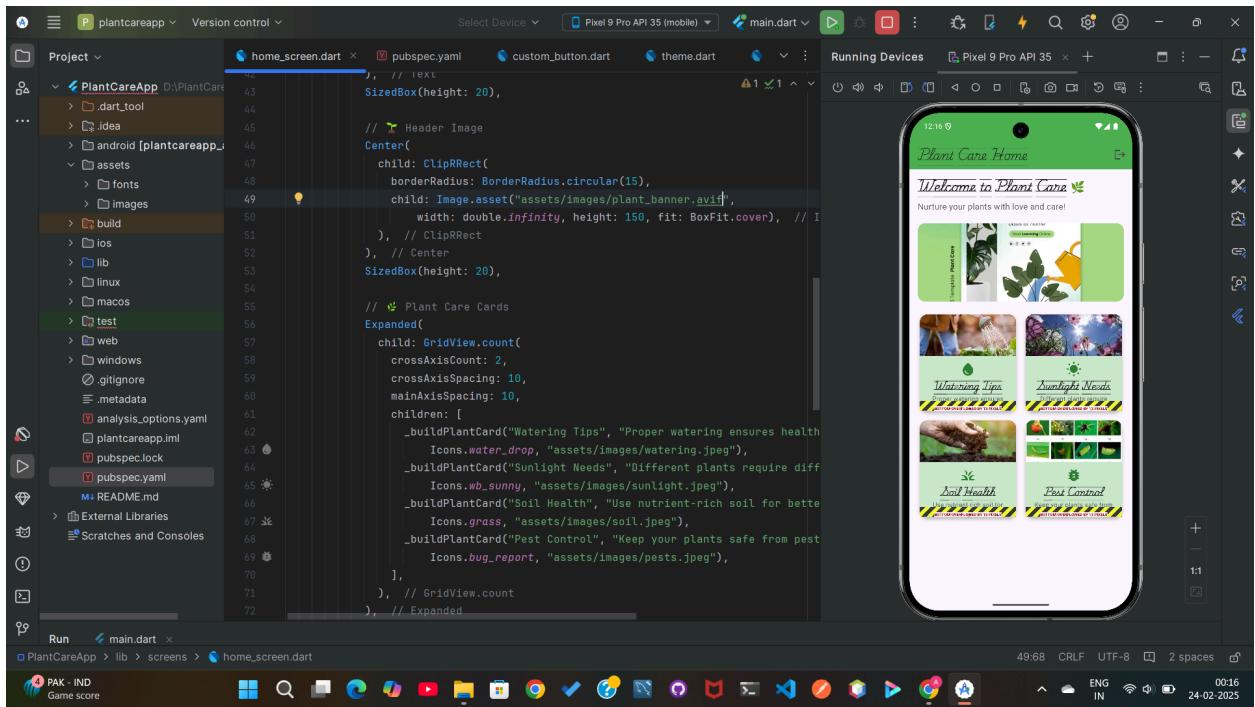
// 🌱 Header Image
Center(
    child: ClipRRect(
        borderRadius: BorderRadius.circular(15),
        child: Image.asset("assets/images/plant_banner.avif",
            width: double.infinity, height: 150, fit: BoxFit.cover),
    ),
),
SizedBox(height: 20),

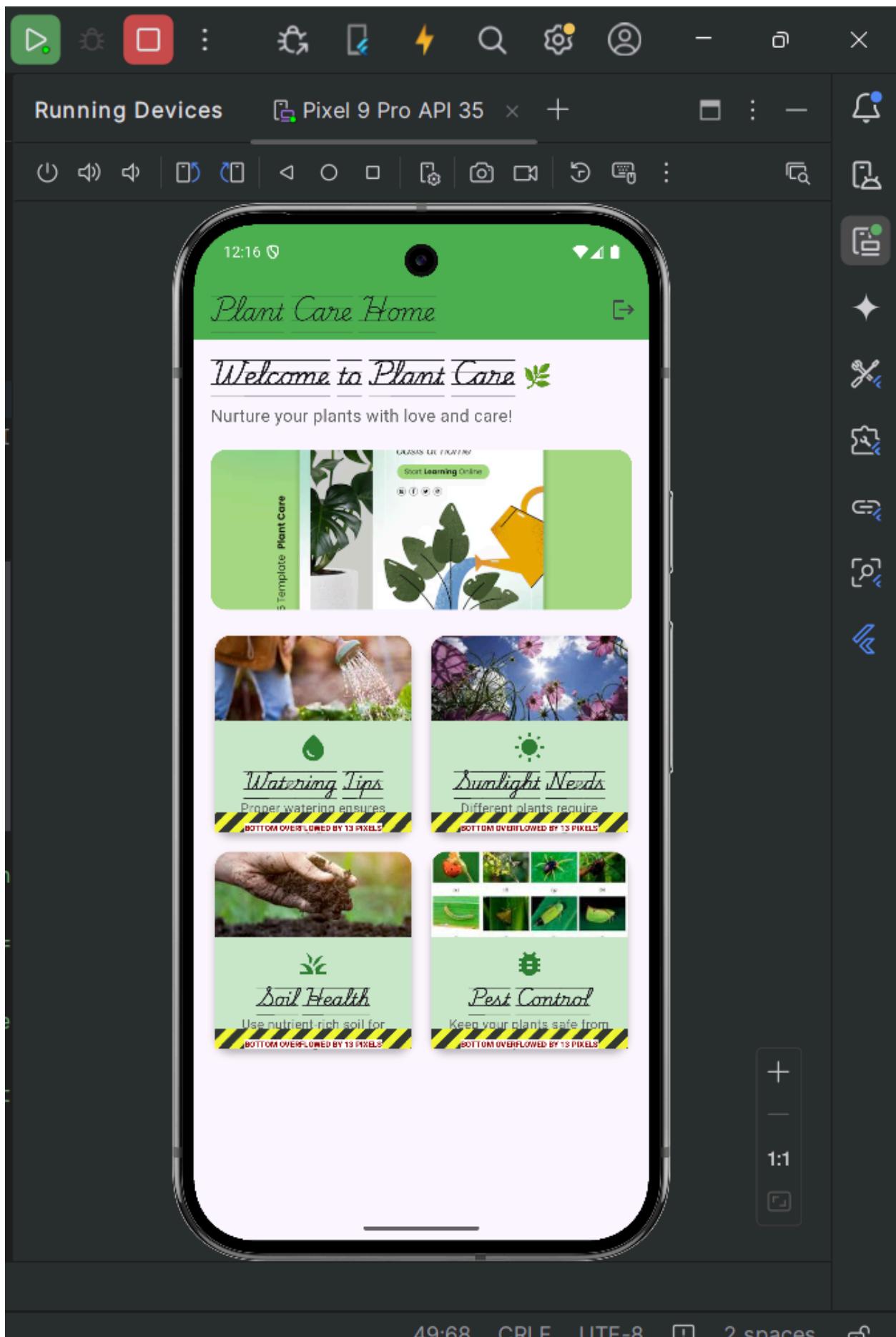
// 🌱 Plant Care Cards
Expanded(
    child: GridView.count(
        crossAxisCount: 2,
        crossAxisSpacing: 10,
        mainAxisSpacing: 10,
        children: [
            _buildPlantCard("Watering Tips", "Proper watering ensures healthy growth",
                Icons.water_drop, "assets/images/watering.jpeg"),
            _buildPlantCard("Sunlight Needs", "Different plants require different sunlight",
                Icons.wb_sunny, "assets/images/sunlight.jpeg"),
            _buildPlantCard("Soil Health", "Use nutrient-rich soil for better growth",
                Icons.grass, "assets/images/soil.jpeg"),
            _buildPlantCard("Pest Control", "Keep your plants safe from pests",
                Icons.bug_report, "assets/images/pests.jpeg"),
        ],
    ),
),
],
),
],
),
);
}

// Reusable Plant Info Card with Image
Widget _buildPlantCard(String title, String description, IconData icon, String imagePath) {
    return Card(
        shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(15)),
        elevation: 5,
        color: Colors.green.shade100,

```

```
        child: Column(
            crossAxisAlignment: CrossAxisAlignment.center,
            children: [
                ClipRRect(
                    borderRadius: BorderRadius.vertical(top: Radius.circular(15)),
                    child: Image.asset(imagePath, height: 80, width: double.infinity, fit: BoxFit.cover),
                ),
                Padding(
                    padding: EdgeInsets.all(10),
                    child: Column(
                        mainAxisAlignment: MainAxisAlignment.center,
                        children: [
                            Icon(icon, size: 30, color: Colors.green.shade800),
                            SizedBox(height: 5),
                            Text(title,
                                textAlign: TextAlign.center,
                                style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold, fontFamily: 'CustomFont')),
                            SizedBox(height: 5),
                            Text(description,
                                textAlign: TextAlign.center,
                                style: TextStyle(fontSize: 12, color: Colors.grey.shade700)),
                        ],
                    ),
                ),
            ],
        );
    }
}
```





Aim : To create an interactive Form using form widget

Signup_screen.dart

```
import 'package:flutter/material.dart';
import 'login_screen.dart';
import 'home_screen.dart';
import './widgets/custom_button.dart';

class SignupScreen extends StatelessWidget {
  final _formKey = GlobalKey<FormState>();
  final TextEditingController emailController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();
  final TextEditingController nameController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Container(
        padding: EdgeInsets.all(20),
        decoration: BoxDecoration(
          gradient: LinearGradient(
            colors: [Colors.green.shade900, Colors.green.shade300],
            begin: Alignment.topCenter,
            end: Alignment.bottomCenter,
          ),
        ),
        child: Center(
          child: SingleChildScrollView(
            child: Form(
              key: _formKey,
              child: Column(
                mainAxisSize: MainAxisSize.min,
                children: [
                  Icon(Icons.eco, size: 100, color: Colors.white),
                  SizedBox(height: 20),
                  _buildTextField(nameController, "Full Name", Icons.person, validator: _validateName),
                  SizedBox(height: 10),
                  _buildTextField(emailController, "Email", Icons.email, validator: _validateEmail),
                  SizedBox(height: 10),
                  _buildTextField(passwordController, "Password", Icons.lock, obscureText: true, validator: _validatePassword),
                  SizedBox(height: 20),
                  CustomButton(
                    text: "Sign Up",
                    onPressed: () {
                      if (_formKey.currentState!.validate()) {
                        Navigator.pushReplacement(context, MaterialPageRoute(builder: (context) =>
                          HomeScreen()));
                      }
                    },
                  ),
                  TextButton(

```

```

        onPressed: () {
            Navigator.pop(context);
        },
        child: Text("Already have an account? Login", style: TextStyle(color: Colors.white)),
    ],
),
),
),
),
),
),
);
}
}

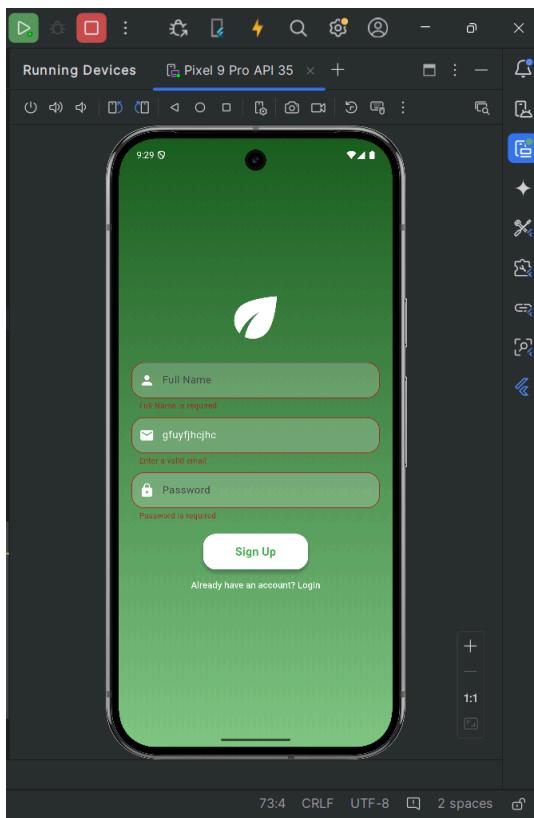
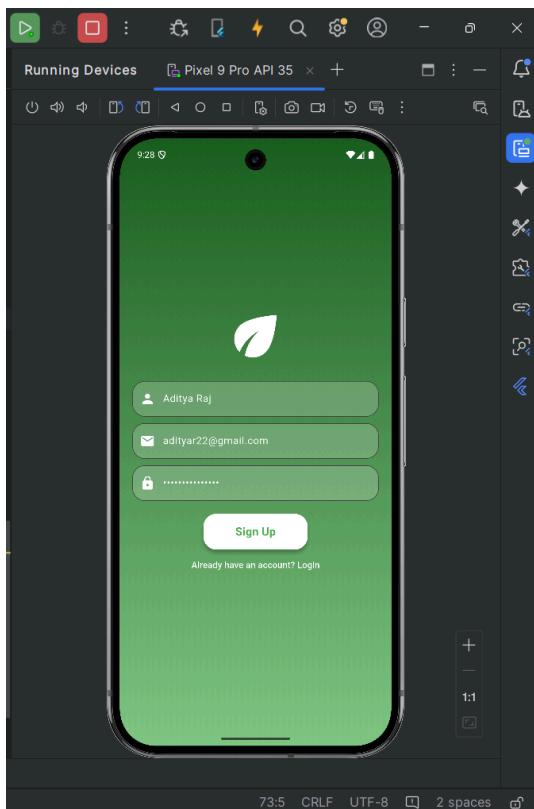
Widget _buildTextField(TextEditingController controller, String hint, IconData icon, {bool obscureText = false, String? Function(String?)? validator}) {
    return TextFormField(
        controller: controller,
        obscureText: obscureText,
        validator: validator,
        decoration: InputDecoration(
            hintText: hint,
            prefixIcon: Icon(icon, color: Colors.white),
            filled: true,
            fillColor: Colors.white.withOpacity(0.2),
            border: OutlineInputBorder(borderRadius: BorderRadius.circular(20)),
        ),
        style: TextStyle(color: Colors.white),
    );
}

String? _validateName(String? value) {
    if (value == null || value.trim().isEmpty) {
        return "Full Name is required";
    }
    return null;
}

String? _validateEmail(String? value) {
    if (value == null || value.trim().isEmpty) {
        return "Email is required";
    }
    final emailRegex = RegExp(r'^[^\s]+@[^\s]+\.[^\s]+\$');
    if (!emailRegex.hasMatch(value)) {
        return "Enter a valid email";
    }
    return null;
}

String? _validatePassword(String? value) {
    if (value == null || value.trim().isEmpty) {
        return "Password is required";
    }
    return null;
}
}

```



Login_screen.dart

```
import 'package:flutter/material.dart';
import 'dart:math'; // Used for generating random numbers
import 'signup_screen.dart';
import 'home_screen.dart';
import './widgets/custom_button.dart';

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

class _LoginScreenState extends State<LoginScreen> {
  final _formKey = GlobalKey<FormState>();
  final TextEditingController emailController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();
  final TextEditingController captchaController = TextEditingController();

  int _num1 = 0;
  int _num2 = 0;
  int _captchaAnswer = 0;

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

  void _generateCaptcha() {
    final Random random = Random();
    _num1 = random.nextInt(9) + 1; // Random number between 1-9
    _num2 = random.nextInt(9) + 1; // Random number between 1-9
    _captchaAnswer = _num1 + _num2; // Expected answer
    setState(() {});
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Container(
        padding: EdgeInsets.all(20),
        decoration: BoxDecoration(
          gradient: LinearGradient(
            colors: [Colors.green.shade300, Colors.green.shade900],
            begin: Alignment.topCenter,
            end: Alignment.bottomCenter,
          ),
        ),
        child: Center(
          child: SingleChildScrollView(
            child: Form(
              key: _formKey,

```

```

        child: Column(
            mainAxisSize: MainAxisSize.min,
            children: [
                Icon(Icons.local_florist, size: 100, color: Colors.white),
                SizedBox(height: 20),
                _buildTextField(emailController, "Email", Icons.email, validator: _validateEmail),
                SizedBox(height: 10),
                _buildTextField(passwordController, "Password", Icons.lock, obscureText: true, validator: _validatePassword),
                SizedBox(height: 10),

                // CAPTCHA Field
                _buildTextField(
                    captchaController,
                    "Solve: ${_num1} + ${_num2} = ?",
                    Icons.shield,
                    validator: _validateCaptcha,
                ),

                SizedBox(height: 20),
                CustomButton(
                    text: "Login",
                    onPressed: () {
                        if (_formKey.currentState!.validate()) {
                            Navigator.pushReplacement(context, MaterialPageRoute(builder: (context) =>
                    HomeScreen()));
                        }
                    },
                ),
                TextButton(
                    onPressed: () {
                        Navigator.push(context, MaterialPageRoute(builder: (context) => SignupScreen()));
                    },
                    child: Text("Don't have an account? Sign Up", style: TextStyle(color: Colors.white)),
                ),
            ],
        ),
    );
}

Widget _buildTextField(TextEditingController controller, String hint, IconData icon, {bool obscureText = false, String? Function(String?)? validator}) {
    return TextFormField(
        controller: controller,
        obscureText: obscureText,
        validator: validator,
        decoration: InputDecoration(
            hintText: hint,
            prefixIcon: Icon(icon, color: Colors.white),
            filled: true,
            fillColor: Colors.white.withOpacity(0.2),
            border: OutlineInputBorder(borderRadius: BorderRadius.circular(20)),
        ),
    );
}

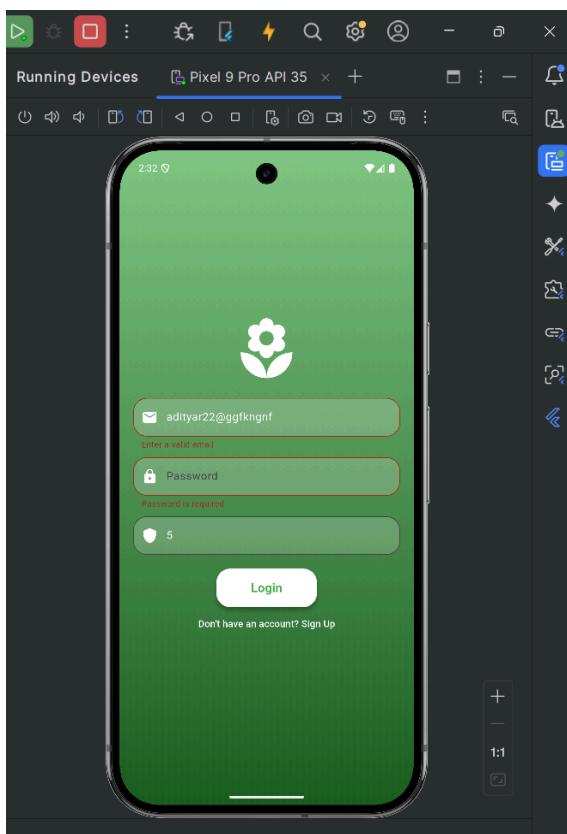
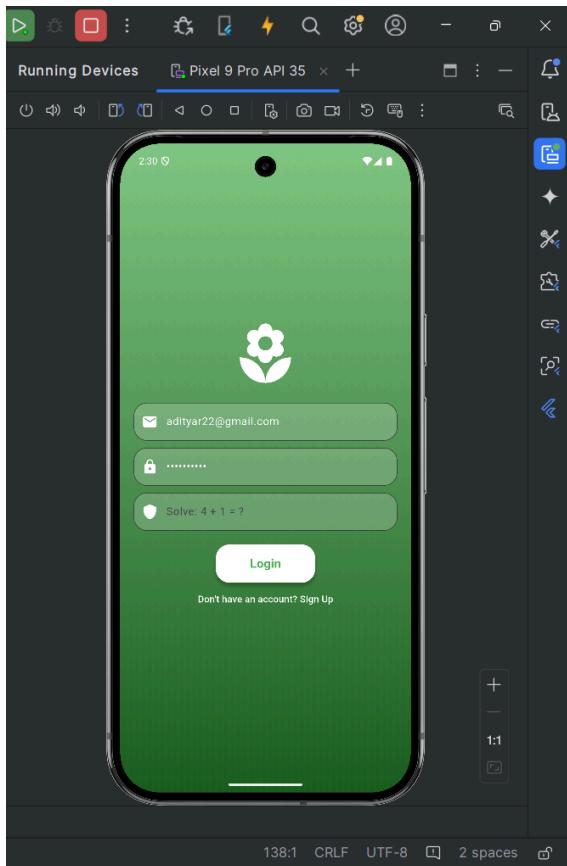
```

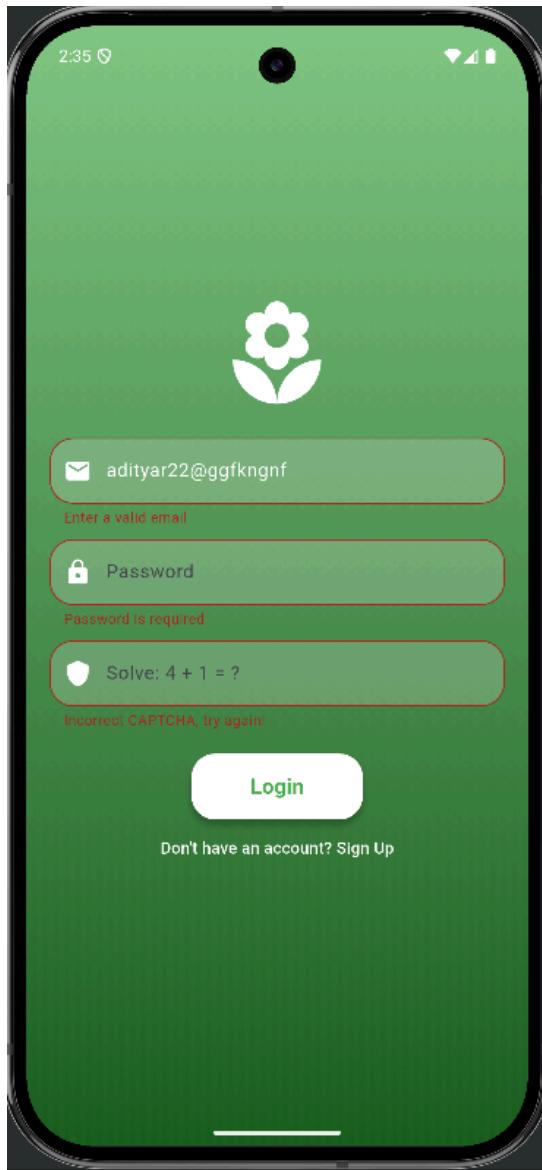
```
        ),
        style: TextStyle(color: Colors.white),
    );
}

String? _validateEmail(String? value) {
    if (value == null || value.trim().isEmpty) {
        return "Email is required";
    }
    final emailRegex = RegExp(r'^[^\@]+@[^\@]+\.[^\@]+\$');
    if (!emailRegex.hasMatch(value)) {
        return "Enter a valid email";
    }
    return null;
}

String? _validatePassword(String? value) {
    if (value == null || value.trim().isEmpty) {
        return "Password is required";
    }
    return null;
}

String? _validateCaptcha(String? value) {
    if (value == null || value.trim().isEmpty) {
        return "CAPTCHA is required";
    }
    if (int.tryParse(value) != _captchaAnswer) {
        return "Incorrect CAPTCHA, try again!";
    }
    return null;
}
```





main.dart

```
import 'package:flutter/material.dart';
import 'screens/login_screen.dart';
import 'screens/home_screen.dart';
import 'screens/details_screen.dart';

void main() {
  runApp(PlantCareApp());
}

class PlantCareApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primarySwatch: Colors.green,
      ),
      initialRoute: '/',
      routes: {
        '/': (context) => LoginScreen(),
        '/home': (context) => HomeScreen(),
        '/details': (context) => DetailsScreen(),
      },
    );
  }
}
```

home_screen.dart

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

class HomeScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(
          "Plant Care Home",
          style: TextStyle(fontFamily: 'CustomFont', fontSize: 22),
        ),
        backgroundColor: Colors.green,
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(

```

```

crossAxisAlignment: CrossAxisAlignment.start,
children: [
  Text(
    "Welcome to Plant Care 🌱",
    style: TextStyle(
      fontSize: 22,
      fontWeight: FontWeight.bold,
      fontFamily: 'CustomFont',
    ),
  ),
  SizedBox(height: 10),
  Text(
    "Nurture your plants with love and care!",
    style: TextStyle(fontSize: 16, color: Colors.grey.shade700),
  ),
  SizedBox(height: 20),
  Center(
    child: GestureDetector(
      onTap: () {
        Navigator.pushNamed(context, '/details');
      },
      child: ClipRRect(
        borderRadius: BorderRadius.circular(15),
        child: Image.asset("assets/images/plant_banner.avif",
          width: double.infinity, height: 150, fit: BoxFit.cover),
      ),
    ),
  ),
  SizedBox(height: 20),
  Expanded(
    child: GridView.count(
      crossAxisCount: 2,
      crossAxisSpacing: 10,
      mainAxisSpacing: 10,
      children: [
        _buildPlantCard(context, "Watering Tips", "Proper watering ensures healthy growth",
          Icons.water_drop, "assets/images/watering.jpeg"),
        _buildPlantCard(context, "Sunlight Needs", "Different plants require different
sunlight",
          Icons.wb_sunny, "assets/images/sunlight.jpeg"),
      ],
    ),
  );
}

```

```

Widget _buildPlantCard(BuildContext context, String title, String description, IconData icon,
String imagePath) {
  return GestureDetector(
    onTap: () {
      Navigator.pushNamed(context, '/details', arguments: {'title': title, 'description': description});
    },
    child: Card(
      shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(15)),
      elevation: 5,
      color: Colors.green.shade100,
      child: Column(
        children: [
          ClipRRect(
            borderRadius: BorderRadius.vertical(top: Radius.circular(15)),
            child: Image.asset(imagePath, height: 80, width: double.infinity, fit: BoxFit.cover),
          ),
          Padding(
            padding: EdgeInsets.all(10),
            child: Column(
              children: [
                Icon(icon, size: 30, color: Colors.green.shade800),
                SizedBox(height: 5),
                Text(title, textAlign: TextAlign.center,
                  style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold, fontFamily:
'CustomFont')),
                SizedBox(height: 5),
                Text(description, textAlign: TextAlign.center,
                  style: TextStyle(fontSize: 12, color: Colors.grey.shade700)),
              ],
            ),
          ),
        ],
      );
    );
}

```

details_screen.dart

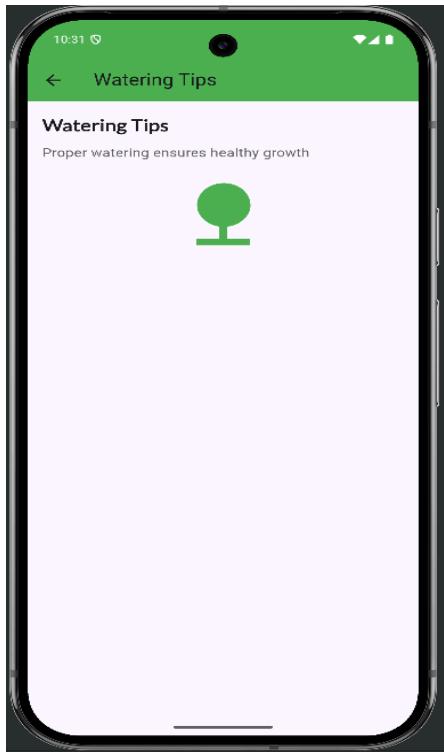
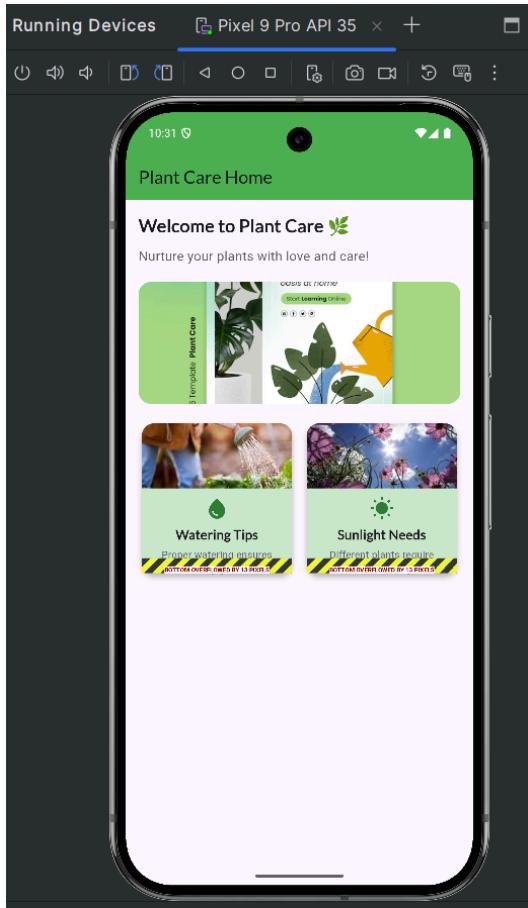
```

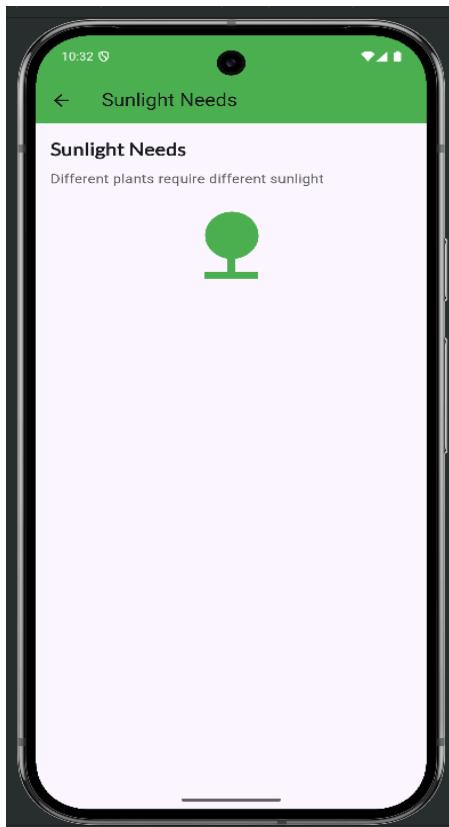
import 'package:flutter/material.dart';

class DetailsScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final Map<String, String>? args = ModalRoute.of(context)?settings.arguments as
    Map<String, String>?;
    return Scaffold(

```

```
appBar: AppBar(  
    title: Text(args?['title'] ?? 'Plant Details'),  
    backgroundColor: Colors.green,  
,  
body: GestureDetector(  
    onHorizontalDragEnd: (details) {  
        Navigator.pop(context);  
    },  
child: Padding(  
    padding: const EdgeInsets.all(16.0),  
    child: Column(  
        crossAxisAlignment: CrossAxisAlignment.start,  
        children: [  
            Text(  
                args?['title'] ?? 'Plant Details',  
                style: TextStyle(fontSize: 22, fontWeight: FontWeight.bold, fontFamily: 'CustomFont'),  
,  
            SizedBox(height: 10),  
            Text(  
                args?['description'] ?? 'No details available',  
                style: TextStyle(fontSize: 16, color: Colors.grey.shade700),  
,  
            SizedBox(height: 20),  
            Center(  
                child: Icon(Icons.nature, size: 100, color: Colors.green),  
,  
            ),  
        ],  
    ),  
),  
),  
);  
};  
}
```





The screenshot shows the Firebase Project Overview page for the project "GardenGlimpse". The left sidebar includes links for Authentication, Firestore Database, Messaging, Storage, Genkit, Vertex AI, and Product categories. The main content area displays the Analytics section, which shows "4 apps" and "GG". It features a chart for "Daily active users" with a value of 0, a "Day 1 retention" of 0%, and a timeline from Mar 30 to Apr 5. A note states "No data for the last 14 days". To the right, there's a section titled "Track your revenue!" with links to AdMob and Google Play. At the bottom, there are "This week" and "Last week" buttons.

The screenshot shows the General tab in the Project settings page for the project "GardenGlimpse". The left sidebar is identical to the Project Overview page. The main content area shows the "Your project" section with fields for Project name (GardenGlimpse), Project ID (gardenglimpse-649fd), Project number (90173116836), and Web API Key. Below this is the "Environment" section, which notes that it customizes the project for different app lifecycle stages. The "Environment type" is listed as "Unspecified".

The screenshot shows the Firebase Authentication 'Users' page. On the left, there's a sidebar with project navigation and product categories like Build, Run, Analytics, AI, Blaze, and Modify. The main content area has tabs for Users, Sign-in method, Templates, Usage, Settings, and Extensions. A message at the top states: "The following Authentication features will stop working when Firebase Dynamic Links shuts down on August 25, 2025: email link authentication for mobile apps, as well as Cordova OAuth support for web apps." Below this is a table with two rows of user data:

Identifier	Providers	Created	Signed In	User UID
prakash26@gmail.com	✉️	Apr 7, 2025		OatCKqd2KkSlo31JDxEhzMu...
adityar99@gmail.com	✉️	Mar 23, 2025	Apr 3, 2025	Fa5EEyZpC0crei9BVZGDe5UG...

At the bottom right of the table, there are buttons for "Rows per page" (50), "1 - 2 of 2", and navigation arrows. A modal window titled "Use Excel for free" offers to "Read, edit, and create documents on your desktop with free Excel." It has "Explore now" and "Later" buttons.

The screenshot shows the Firebase Authentication 'Sign-in method' page. The sidebar and navigation bar are identical to the previous screenshot. The main content area has tabs for Users, Sign-in method, Templates, Usage, Settings, and Extensions. Under the 'Sign-in method' tab, there are sections for "Sign-in providers" and "Advanced".

Sign-in providers: A table showing one provider row:

Provider	Status
✉️ Email/Password	Enabled

Advanced: A section containing a button labeled "SMS Multi-factor Authentication".

localhost:58797/#/signup

Sign Up

 Garden Glimpse 

Create Your Account

Email

Password

Solve this to prove you're human:

10 + 10 = 

[Sign Up](#)

Already have an account? [Login](#)

Signup Successful! Redirecting to Login...

localhost:58408/#/login

Login

 GardenGlimpse 

Welcome Back!

Email

Password

Solve this to prove you're human:

5 + 10 = 

[Login](#)

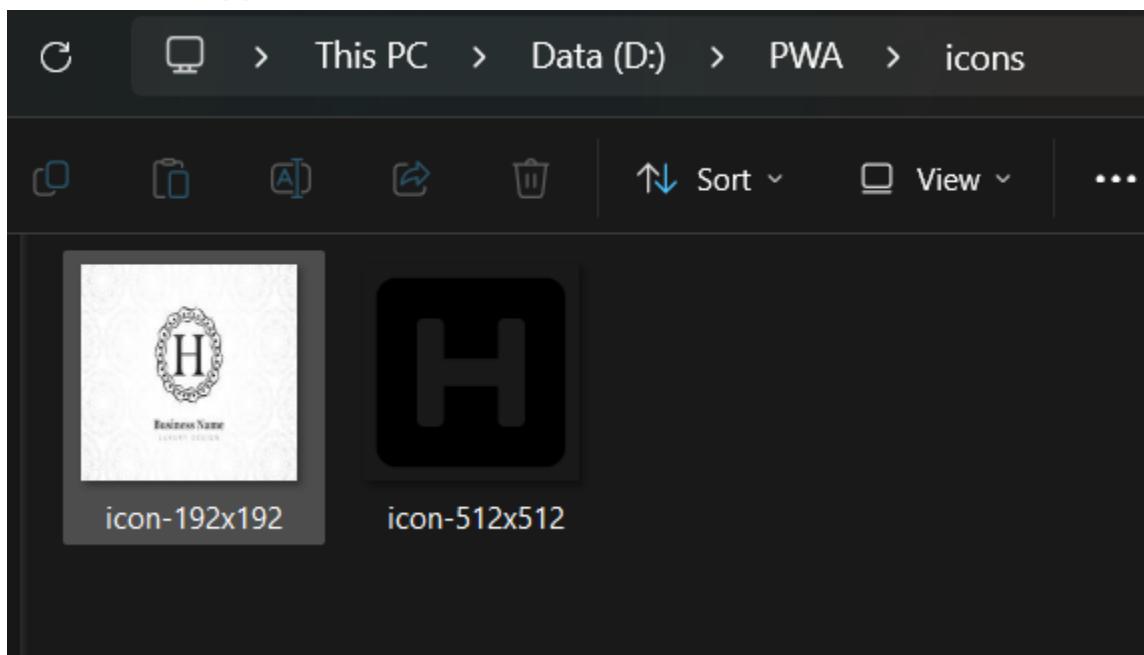
Don't have an account? [Sign up](#)

Login Successful!

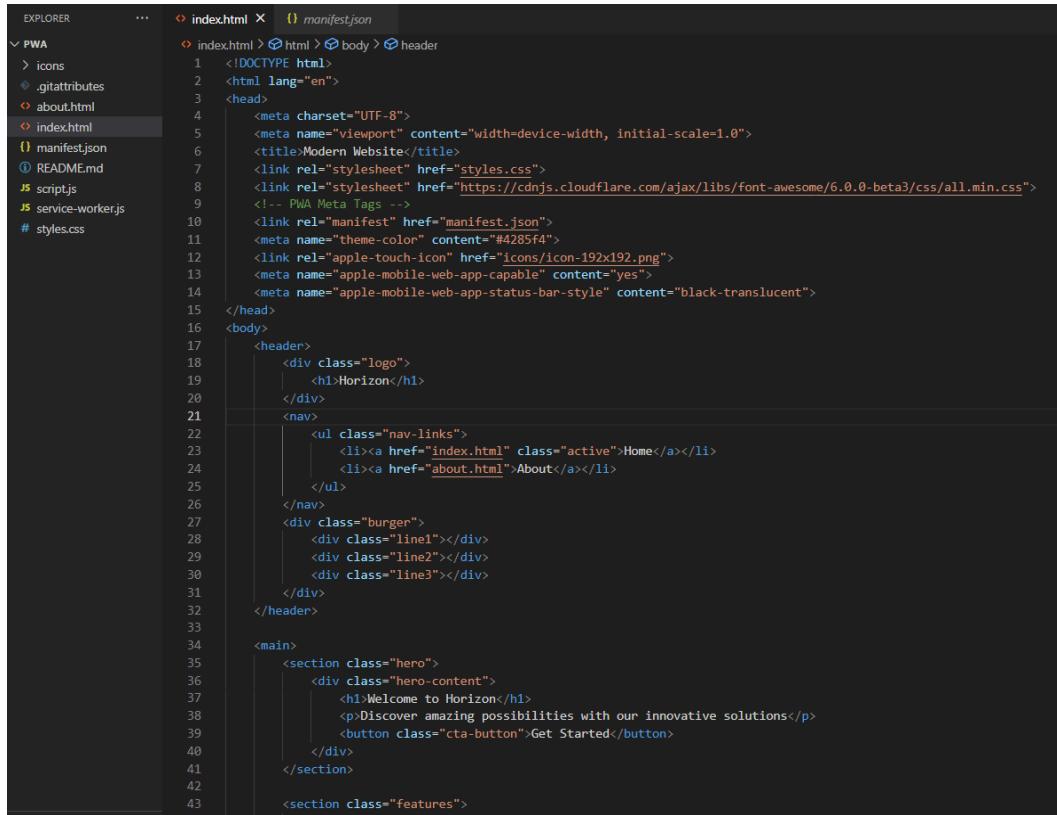
Progressive web app folder

Name	Date modified	Type	Size
icons	07-04-2025 14:07	File folder	
.gitattributes	07-04-2025 14:21	txtfile	1 KB
about	07-04-2025 12:05	Chrome HTML Do...	5 KB
index	07-04-2025 12:36	Chrome HTML Do...	5 KB
manifest	07-04-2025 14:28	JSON Source File	1 KB
README	07-04-2025 14:21	Markdown Source ...	1 KB
script	07-04-2025 12:06	JavaScript Source ...	3 KB
service-worker	07-04-2025 14:07	JavaScript Source ...	2 KB
styles	07-04-2025 12:06	CSS Source File	10 KB

Icons for the app



Index.html and manifest.json



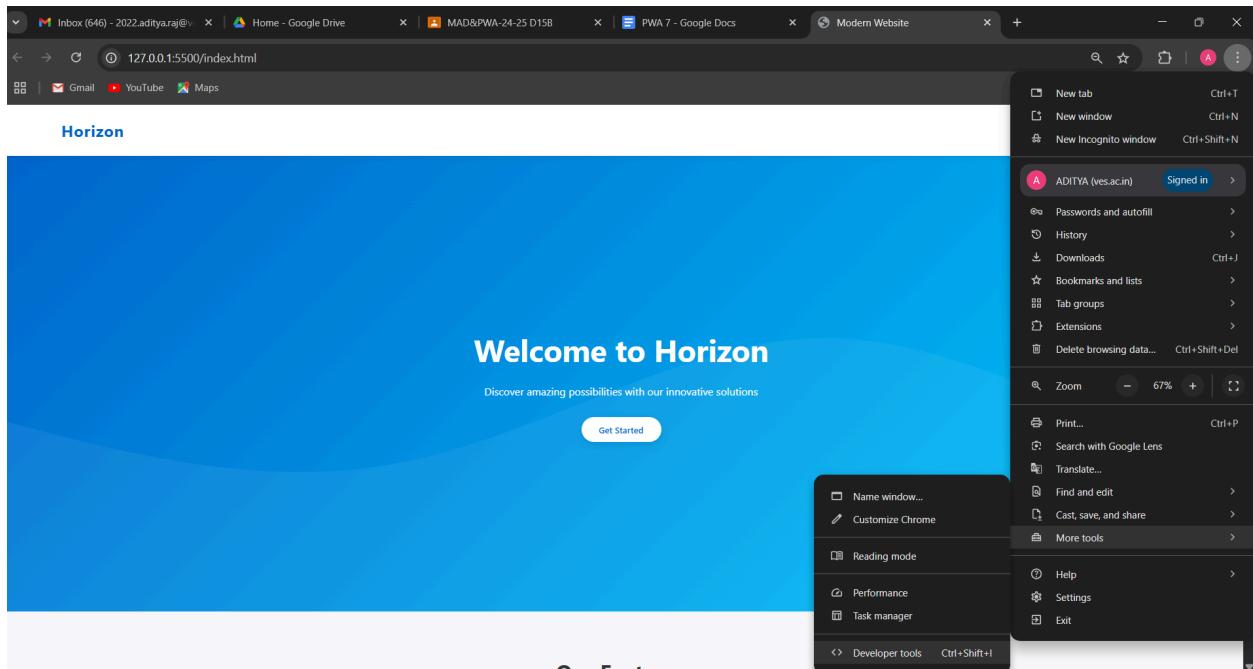
The screenshot shows a code editor interface with two files open: index.html and manifest.json. The manifest.json file is currently selected. The code editor displays the content of both files, including the manifest.json file's contents which define the application's name, theme color, and other metadata.

```
EXPLORER      ...
PWA
  > icons
  <.gitattributes>
  <about.html>
  <index.html>
  manifest.json
  README.md
  script.js
  service-worker.js
  styles.css

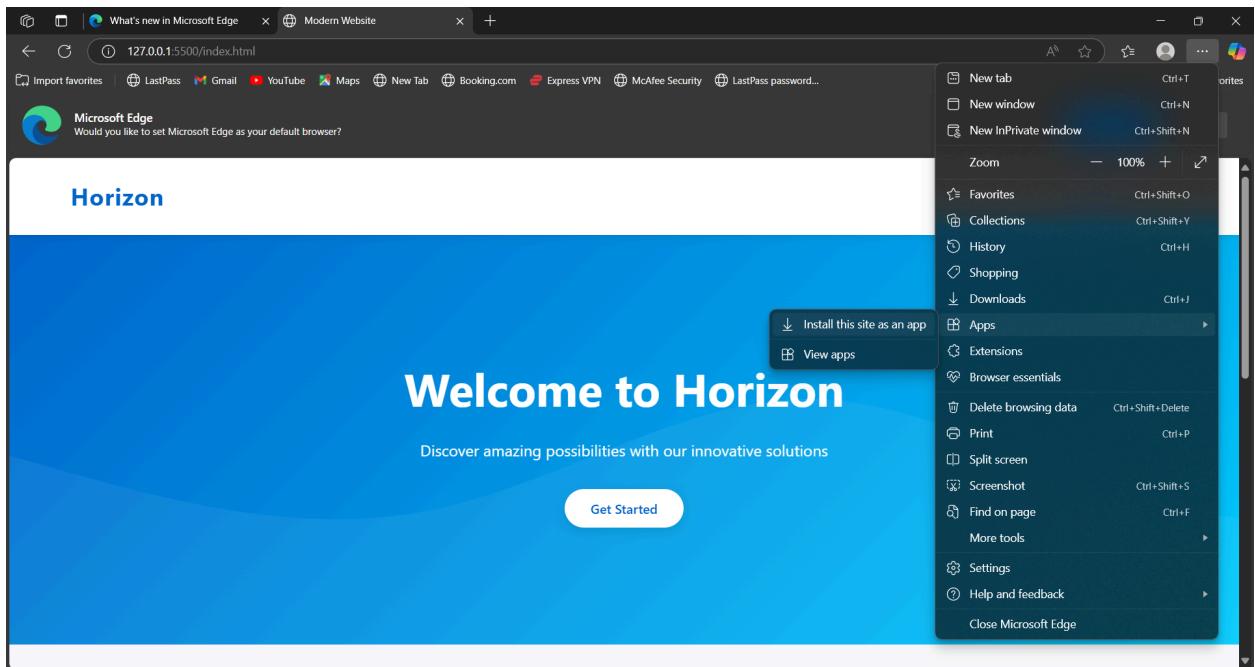
index.html x  manifest.json

index.html > html > body > header
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Modern Website</title>
  <link rel="stylesheet" href="styles.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css">
  <!-- PWA Meta Tags -->
  <link rel="manifest" href="manifest.json">
  <meta name="theme-color" content="#4285f4">
  <link rel="apple-touch-icon" href="icons/icon-192x192.png">
  <meta name="apple-mobile-web-app-capable" content="yes">
  <meta name="apple-mobile-web-app-status-bar-style" content="black-translucent">
</head>
<body>
  <header>
    <div class="logo">
      <h1>Horizon</h1>
    </div>
    <nav>
      <ul class="nav-links">
        <li><a href="#" class="active">Home</a></li>
        <li><a href="#">About</a></li>
      </ul>
    </nav>
    <div class="burger">
      <div class="line1"></div>
      <div class="line2"></div>
      <div class="line3"></div>
    </div>
  </header>
  <main>
    <section class="hero">
      <div class="hero-content">
        <h1>Welcome to Horizon</h1>
        <p>Discover amazing possibilities with our innovative solutions</p>
        <button class="cta-button">Get Started</button>
      </div>
    </section>
    <section class="features">
      <h2>Our Features</h2>
      <ul>
        <li>Feature 1</li>
        <li>Feature 2</li>
        <li>Feature 3</li>
        <li>Feature 4</li>
      </ul>
    </section>
  </main>
</body>
```

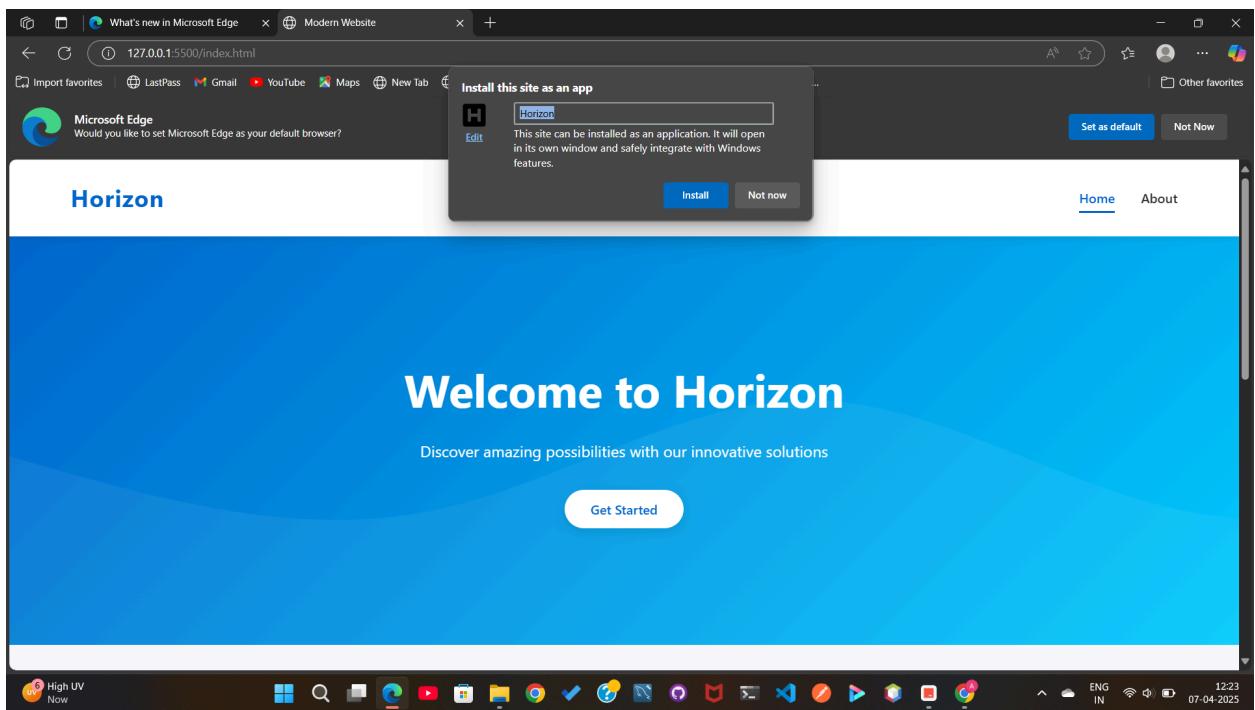
Developer tools option



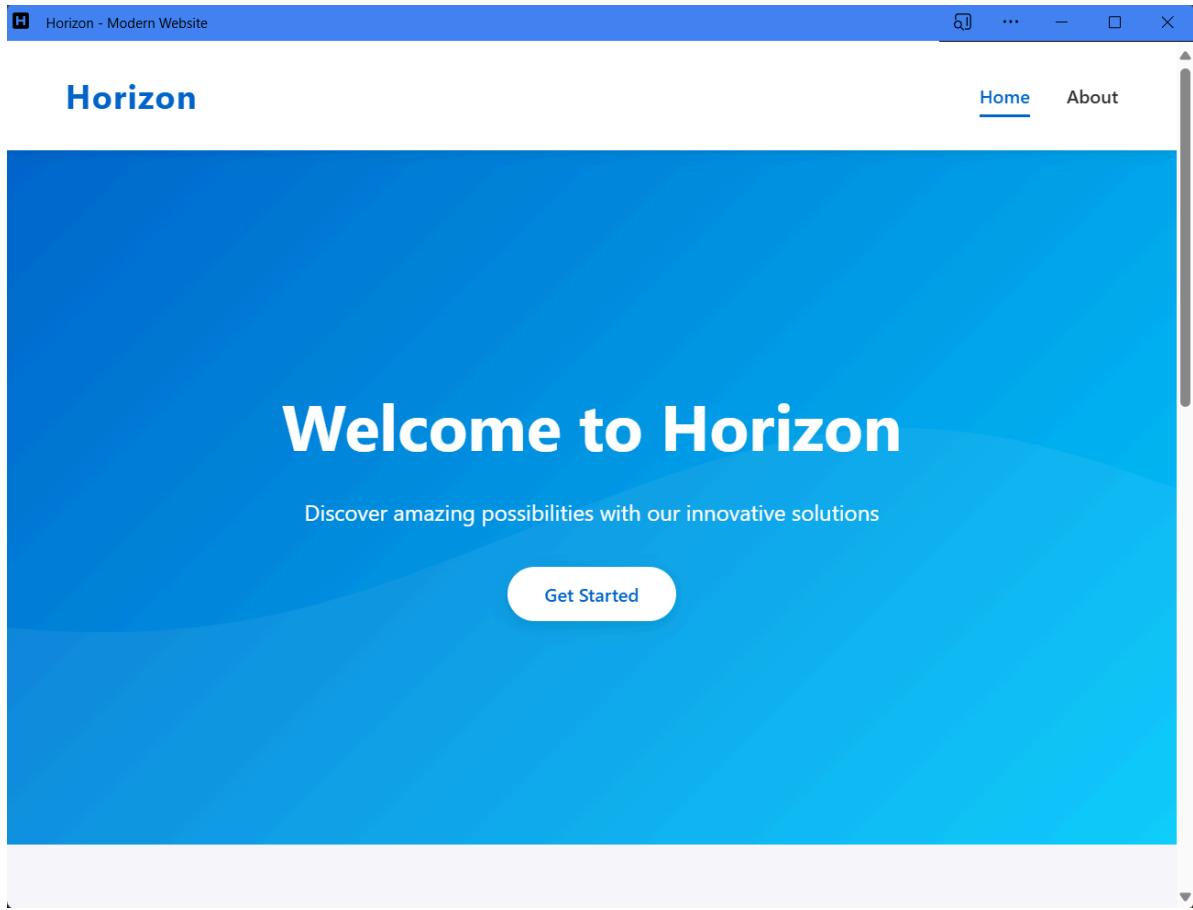
Install this site as an app



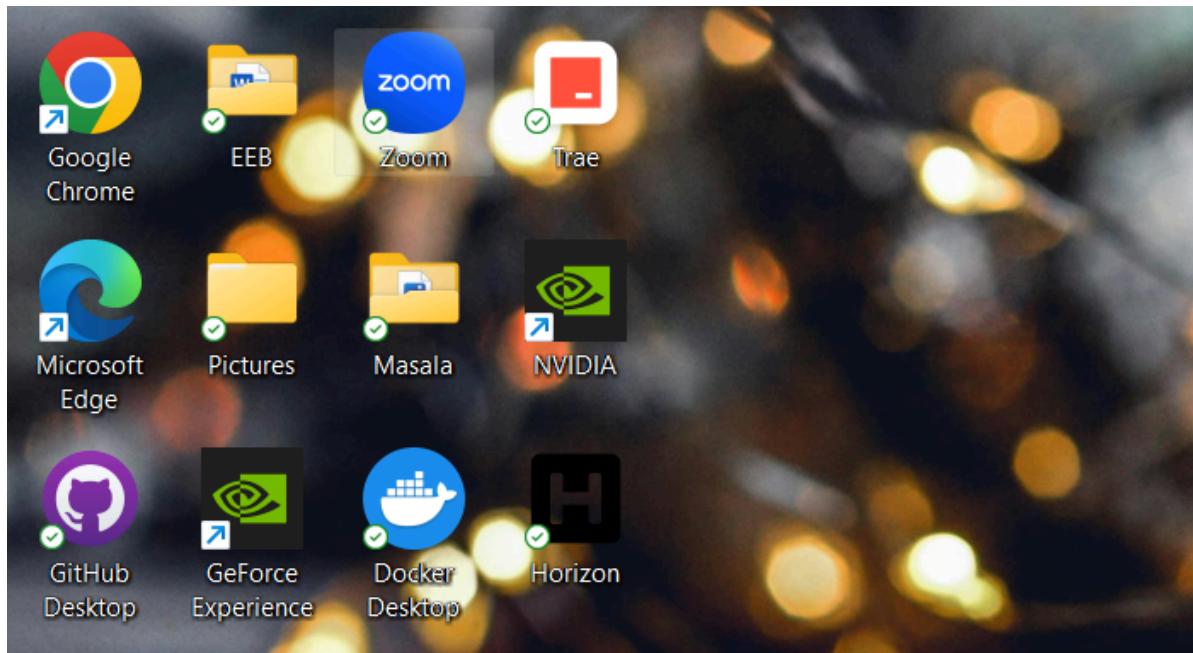
Name and shortcut



The installed progressive web app



Desktop shortcut for the app



Service-worker.js

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

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

// Activate event - clean up old caches
self.addEventListener('activate', event => {
  const cacheWhitelist = [CACHE_NAME];
  event.waitUntil(
    caches.keys().then(cacheNames => {
      return Promise.all(
        cacheNames.map(cacheName => {
          if (cacheWhitelist.indexOf(cacheName) === -1) {
            return caches.delete(cacheName);
          }
        })
      );
    })
  );
});

// Fetch event - serve from cache or network
self.addEventListener('fetch', event => {
  event.respondWith(
```

```

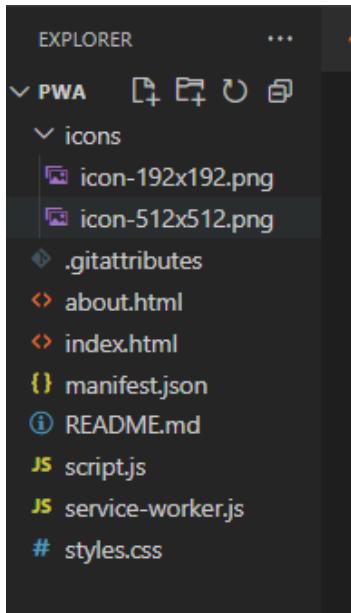
caches.match(event.request)
.then(response => {
  // Cache hit - return response
  if (response) {
    return response;
  }
  return fetch(event.request).then(
    response => {
      // Check if we received a valid response
      if (!response || response.status !== 200 || response.type !== 'basic') {
        return response;
      }

      // Clone the response
      const responseToCache = response.clone();

      caches.open(CACHE_NAME)
        .then(cache => {
          cache.put(event.request, responseToCache);
        });

      return response;
    }
  );
});
}
);
}
);
}
);

```



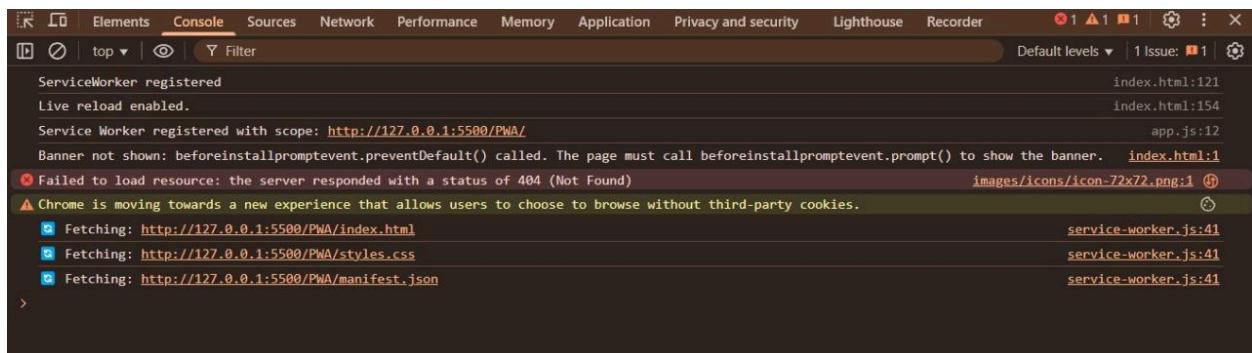
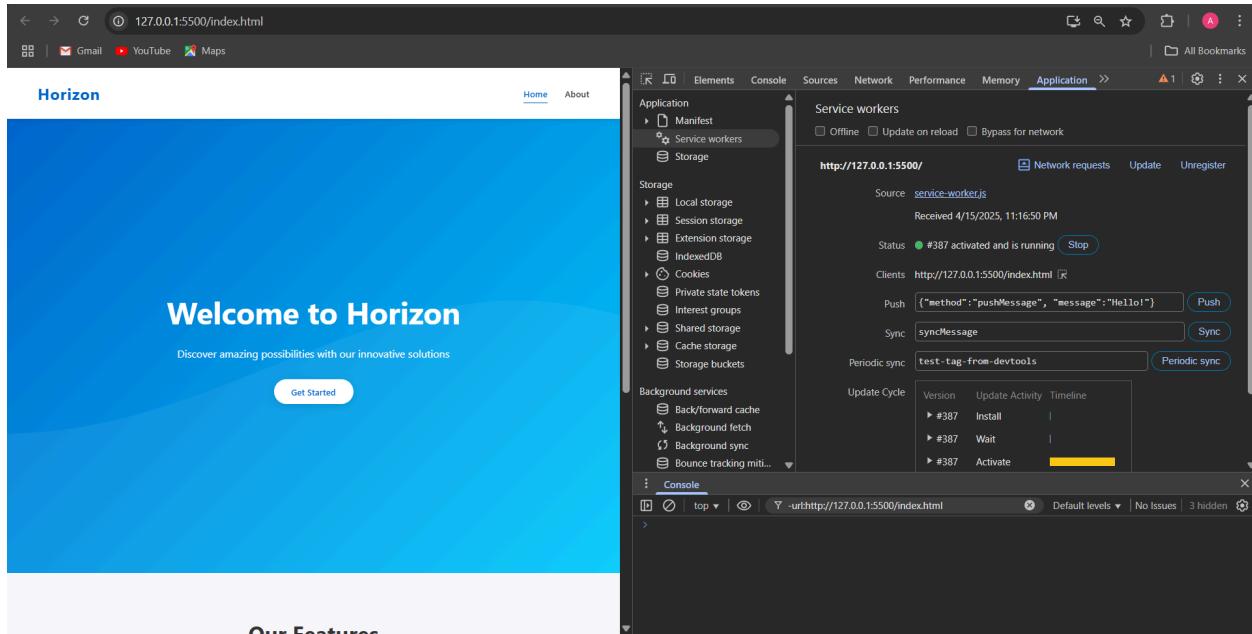
Cache storage

The screenshot shows the Chrome DevTools Application panel for the URL `http://127.0.0.1:5500`. The left sidebar lists various storage types: Manifest, Service workers, Storage, Local storage, Session storage, Extension storage, IndexedDB, Cookies, Private state tokens, Interest groups, Shared storage, and Cache storage. The 'Cache storage' section is expanded, showing a list of items including 'horizon-cache-v1 ...'. The right panel displays detailed information about the default bucket: Bucket name is 'default', Durability is 'relaxed', and Quota is '0 B'. A table lists entries with columns for Name, Response, Content-type, and Time.

Service workers

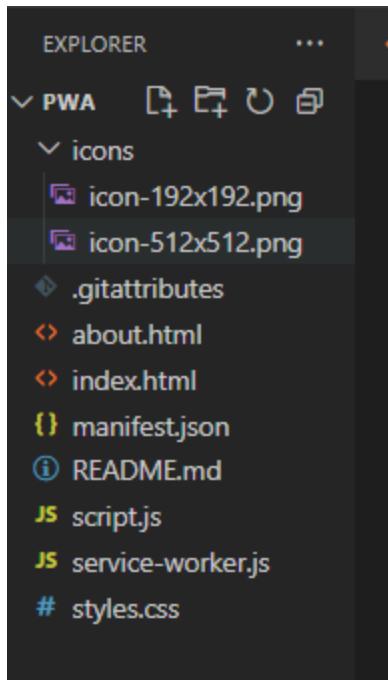
The screenshot shows the Chrome DevTools Application panel for the URL `http://127.0.0.1:5500`. The left sidebar lists Application, Manifest, Service workers, Storage, Local storage, Session storage, Extension storage, IndexedDB, Cookies, Private state tokens, Interest groups, Shared storage, Cache storage, and Storage buckets. The 'Service workers' section is selected and expanded, showing options for Offline, Update on reload, and Bypass for network. It also lists clients and periodic sync tasks. The right panel shows a list of service workers from other origins.

Aim : To implement service worker events like fetch, sync and push



The screenshot shows the Chrome DevTools Application tab. In the left sidebar, under 'Application', there are sections for Manifest, Service workers, and Storage. Under 'Service workers', it shows two workers: '#1455 activated and is running' and '#1578 waiting to activate'. The 'Clients' section lists a client at 'http://127.0.0.1:5500/PWA/index.html'. Below that, there are 'Push' and 'Sync' sections with their respective log entries. At the bottom of the sidebar, tabs for 'Console', 'What's new', 'AI assistance', and 'Issues' are visible. The main area is a log viewer with a dark background and light-colored text. It contains several log entries related to service workers, push notifications, and sync events, with file paths like 'index.html:121', 'app.js:12', and 'service-worker.js:41' indicated.

The screenshot shows a Microsoft Edge browser window with the title 'Horizon - 127.0.0.1'. The page content displays a cloud icon and the text 'Hmmm... can't reach this page'. Below that, it says '127.0.0.1 refused to connect.' and provides troubleshooting steps: 'Try: Search the web for 127.0.0.1, Checking the connection, Checking the proxy and the firewall'. At the bottom, it shows the error code 'ERR_CONNECTION_REFUSED' and a blue 'Refresh' button. The Microsoft Edge logo is at the bottom left.



Commit and push files to Github

A screenshot of a GitHub repository page for a project named "PWA". The repository has one branch ("main") and one tag ("1"). The commit history shows the initial commit of various files: icons, .gitattributes, README.md, about.html, index.html, manifest.json, script.js, service-worker.js, and styles.css. The commit message for all files is "Initial commit" and the date is "last week". The repository details page includes sections for About (no description, website, or topics provided), Activity (0 stars, 1 watching, 0 forks), Releases (no releases published), Packages (no packages published), Deployments (2 deployments, 15 minutes ago), and Languages (no languages listed).

Go to deployments and configure

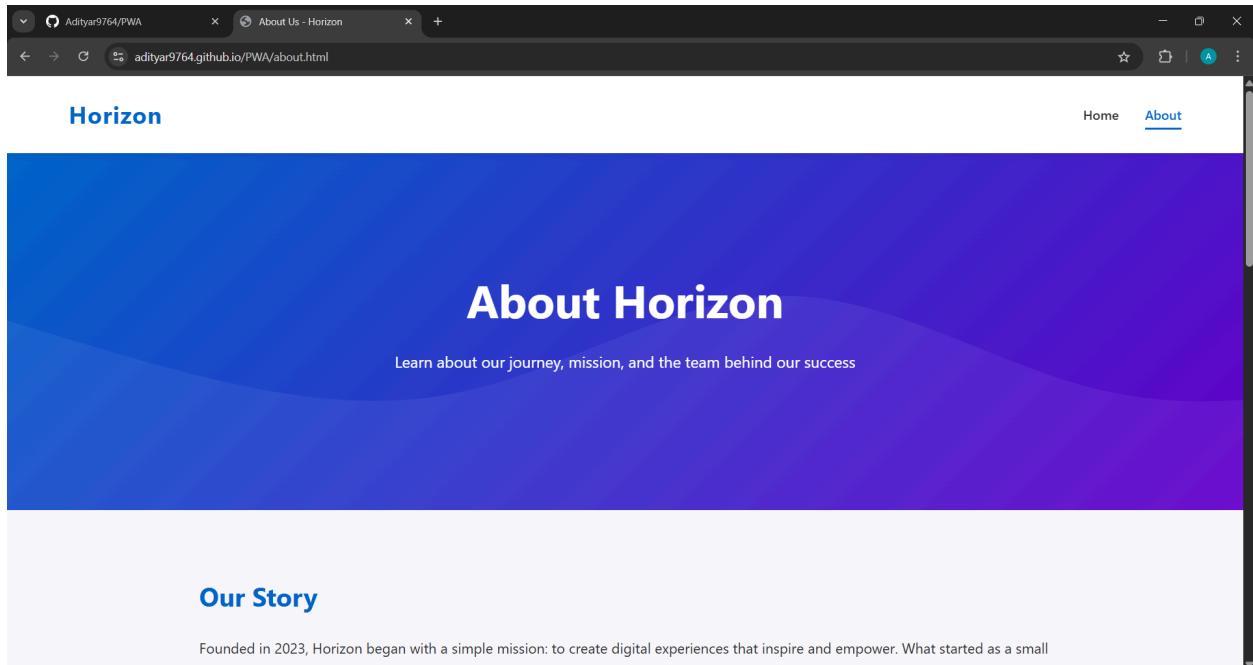
The screenshot shows the GitHub Deployments page for the repository 'Adityar9764/PWA'. The left sidebar has 'Deployments' selected under 'Environments'. The main area shows 'github-pages' as the latest deployment, followed by 'image update' and 'Initial commit'. Each deployment entry includes a link to its details.

Deployment	Status	Triggered By	Time
github-pages	Green	Adityar9764	16 minutes ago
image update	Green	Adityar9764 via pages-build-deployment #2	16 minutes ago
Initial commit	Grey	Adityar9764 via pages-build-deployment #1	last week

Visiting our deployed site

The screenshot shows the deployed website 'Horizon' at the URL 'adityar9764.github.io/PWA/'. The page features a large blue header with the text 'Welcome to Horizon' and 'Discover amazing possibilities with our innovative solutions'. It includes a 'Get Started' button and navigation links for 'Home' and 'About'.

Checking if all routes are working



DevTools → application → service workers

The screenshot shows the Chrome DevTools Application tab for the URL `http://127.0.0.1:5500/`. The left sidebar has 'Service workers' selected under 'Application'. The main panel displays information about a service worker named '#45'. It shows the source file is `service-worker.js`, received on 7/4/2025, 2:00:50 pm, and is activated and running. There are sections for 'Push' (with a button to 'Push'), 'Sync' (with a button to 'Sync'), and 'Periodic sync' (with a button to 'Periodic sync'). Below these is the 'Update Cycle' section, which shows the current state of the service worker: 'Install' (Version #45), 'Wait' (Version #45), and 'Activate' (Version #45, shown with a progress bar). At the bottom, there's a section for 'Service workers from other origins' with a link to 'See all registrations'.

Lighthouse report generation

The screenshot shows the Chrome DevTools Lighthouse tab for the URL `http://127.0.0.1:5500/about.html`. The left sidebar has 'Lighthouse' selected. A modal dialog titled 'Auditing 127.0.0.1...' is open, showing a warning message: '70% of mobile pages take nearly 7 seconds for the visual content above the fold to display on the screen. [Source: Think with Google]'. Below the message is a 'Cancel' button. To the right of the modal, under 'Categories', are four checked checkboxes: 'Performance', 'Accessibility', 'Best practices', and 'SEO'. At the top of the page, there is a note: '(i) The Lighthouse tool provides links to content hosted on third-party websites. (Don't show again)' with a 'Show more' link.

Final lighthouse report

