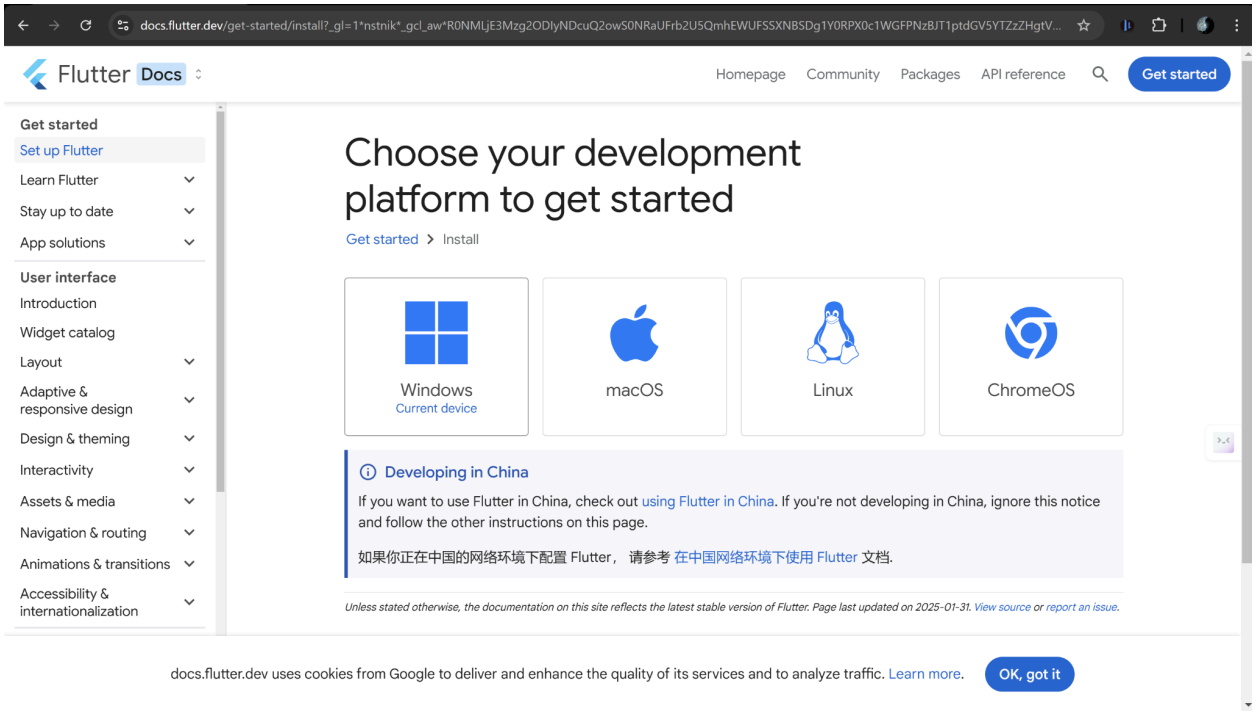
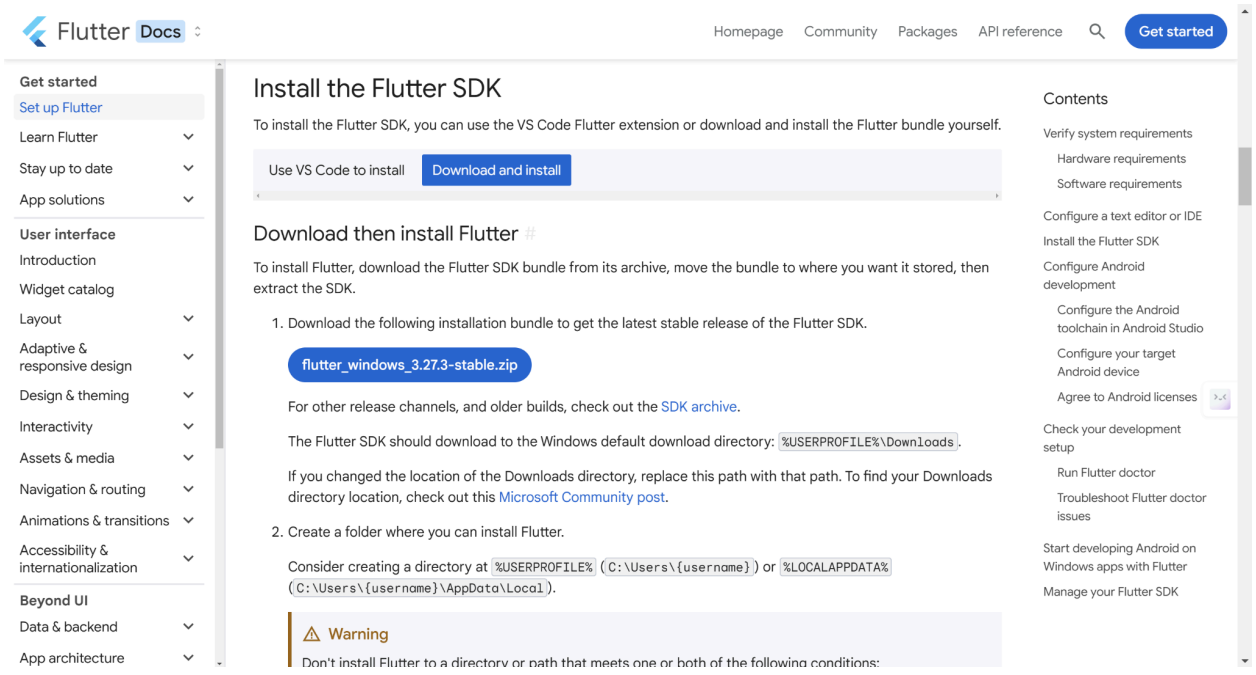
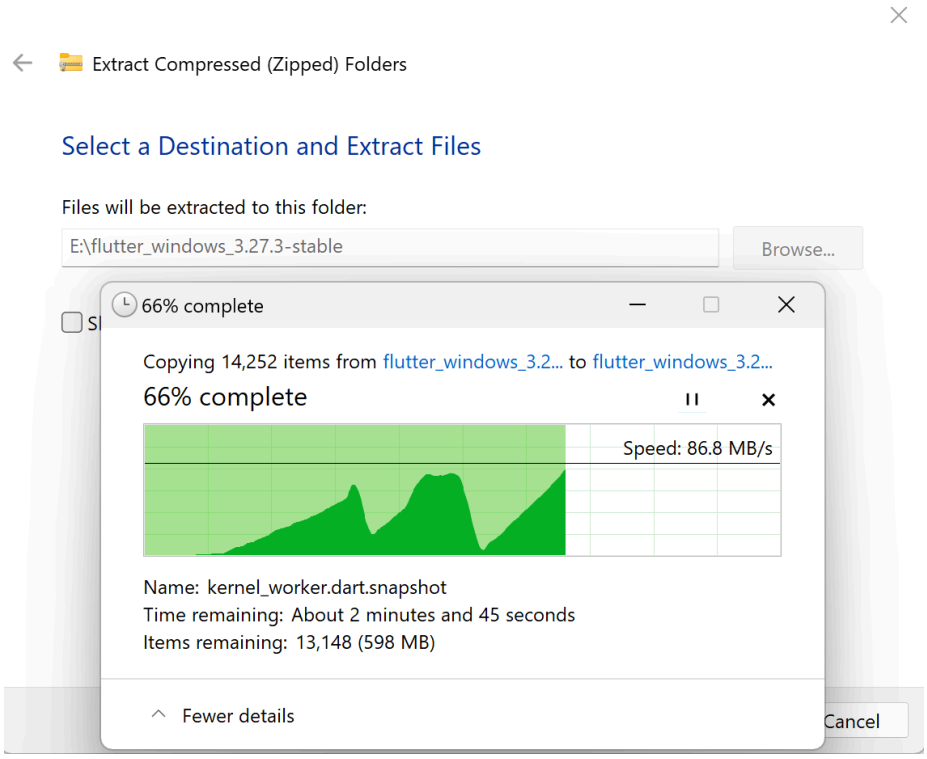
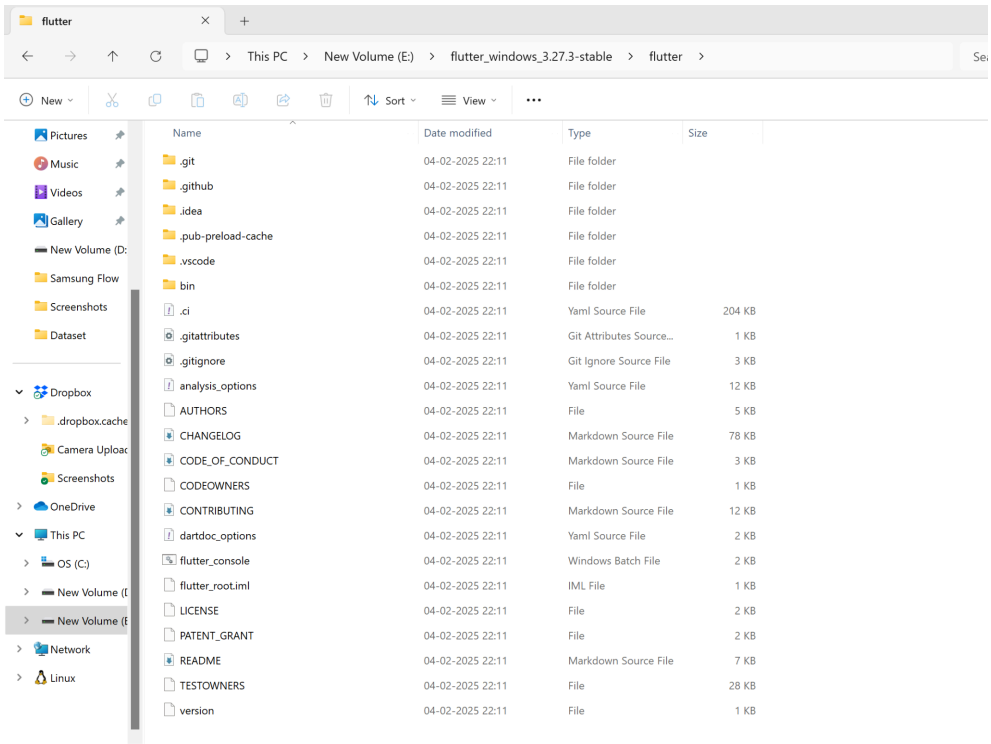
Name : Shreyas Naik
Roll no : 38
Div : D15B

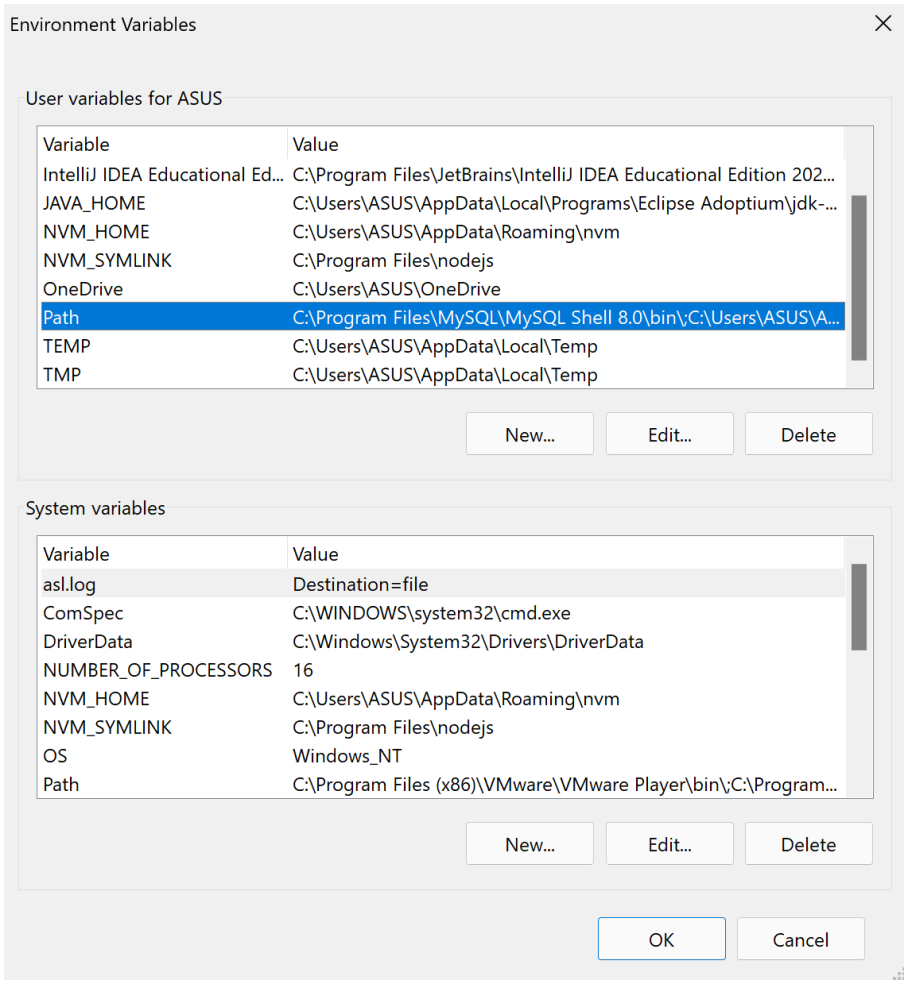
MPL Practical 1

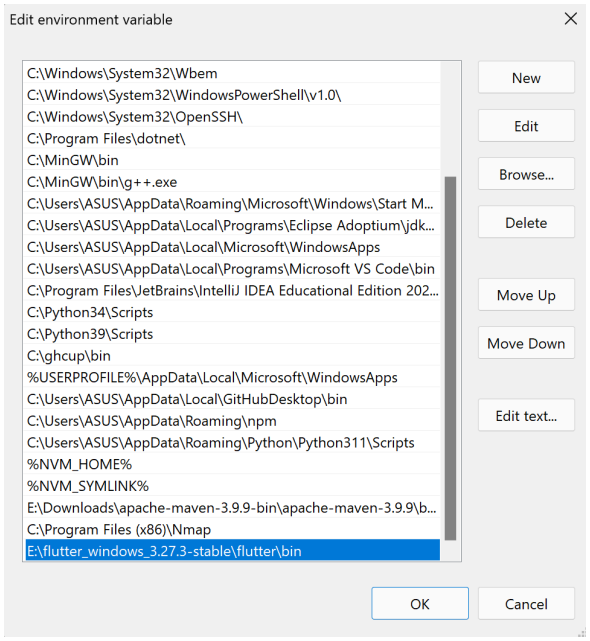


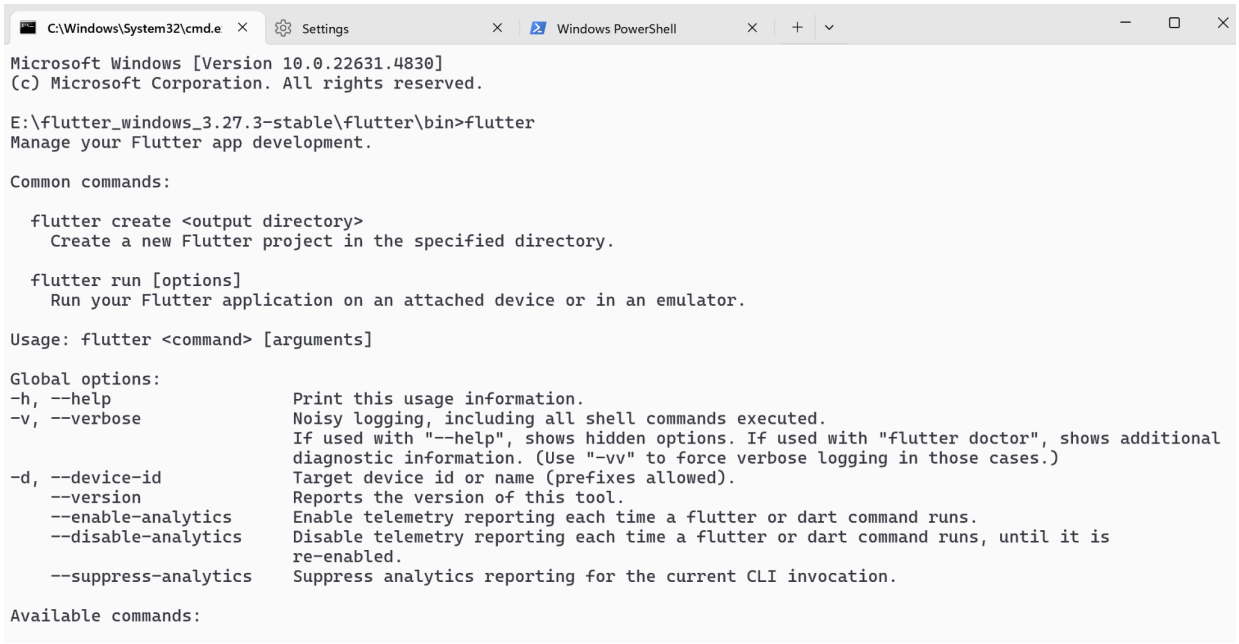


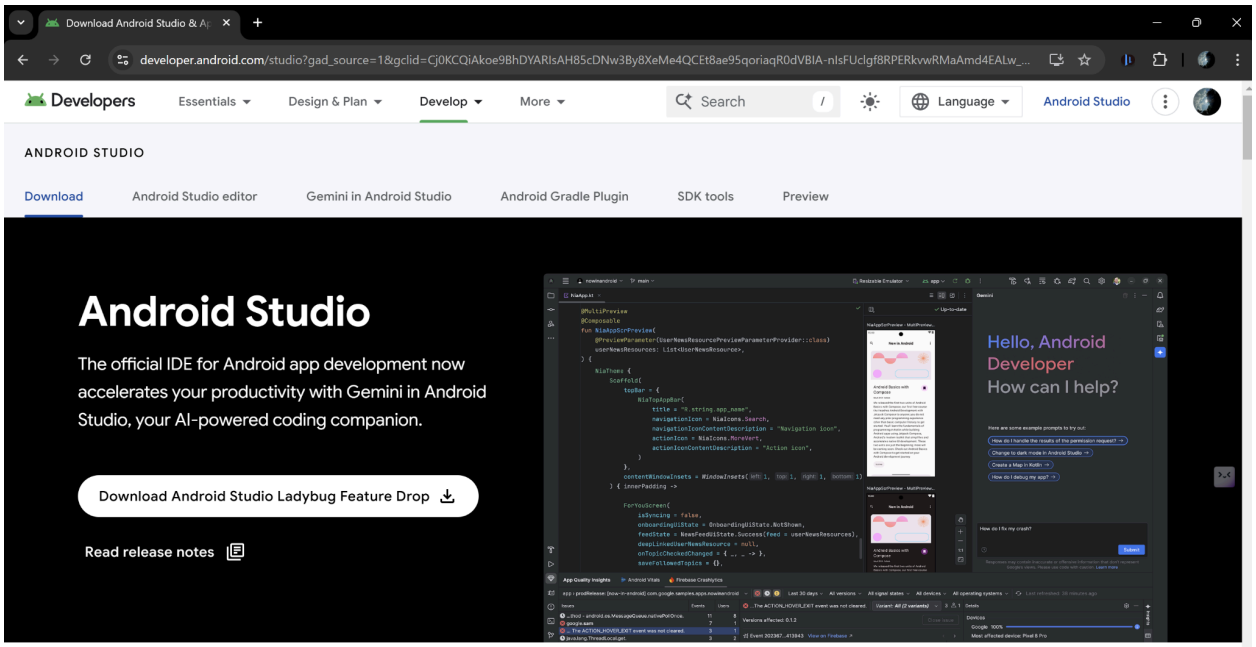

Extract Compressed (Zipped) Folders
Select a Destination and Extract Files
Files will be extracted to this folder:
E:\flutter_windows_3.27.3-stable
Browse...
66% complete
Copying 14,252 items from flutter_windows_3.2... to flutter_windows_3.2...
66% complete
Speed: 86.8 MB/s
Name: kernel_worker.dart.snapshot
Time remaining: About 2 minutes and 45 seconds
Items remaining: 13,148 (598 MB)
Fewer details
Cancel


flutter
This PC
New Volume (E:)
flutter_windows_3.27.3-stable
flutter
New
Sort
View
Name
Date modified
Type
Size
.git 04-02-2025 22:11 File folder
.github 04-02-2025 22:11 File folder
.idea 04-02-2025 22:11 File folder
.pub-preload-cache 04-02-2025 22:11 File folder
.vscode 04-02-2025 22:11 File folder
bin 04-02-2025 22:11 File folder
.ci 04-02-2025 22:11 Yaml Source File 204 KB
.gitattributes 04-02-2025 22:11 Git Attributes Source... 1 KB
.gitignore 04-02-2025 22:11 Git Ignore Source File 3 KB
analysis_options 04-02-2025 22:11 Yaml Source File 12 KB
AUTHORS 04-02-2025 22:11 File 5 KB
CHANGELOG 04-02-2025 22:11 Markdown Source File 78 KB
CODE_OF_CONDUCT 04-02-2025 22:11 Markdown Source File 3 KB
CODEOWNERS 04-02-2025 22:11 File 1 KB
CONTRIBUTING 04-02-2025 22:11 Markdown Source File 12 KB
dartdoc_options 04-02-2025 22:11 Yaml Source File 2 KB
flutter_console 04-02-2025 22:11 Windows Batch File 2 KB
flutter_root.iml 04-02-2025 22:11 IML File 1 KB
LICENSE 04-02-2025 22:11 File 2 KB
PATENT_GRANT 04-02-2025 22:11 File 2 KB
README 04-02-2025 22:11 Markdown Source File 7 KB
TESTOWNERS 04-02-2025 22:11 File 28 KB
version 04-02-2025 22:11 File 1 KB
Pictures
Music
Videos
Gallery
New Volume (D:
Samsung Flow
Screenshots
Dataset
Dropbox
Camera Upload
Screenshots
OneDrive
This PC
OS (C:)
Network
Linux

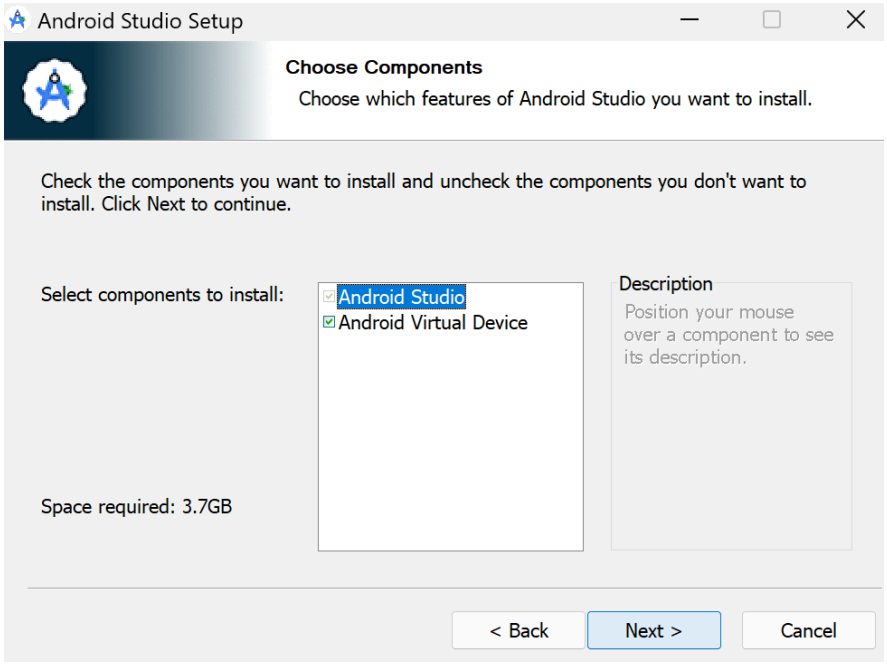

Environment Variables
User variables for ASUS
Variable
Value
IntelliJ IDEA Educational Ed...
C:\Program Files\JetBrains\IntelliJ IDEA Educational Edition 202...
JAVA_HOME
C:\Users\ASUS\AppData\Local\Programs\Eclipse Adoptium\jdk-...
NVM_HOME
C:\Users\ASUS\AppData\Roaming\nvm
NVM_SYMLINK
C:\Program Files\nodejs
OneDrive
C:\Users\ASUS\OneDrive
Path
C:\Program Files\MySQL\MySQL Shell 8.0\bin\;C:\Users\ASUS\A...
TEMP
C:\Users\ASUS\AppData\Local\Temp
TMP
C:\Users\ASUS\AppData\Local\Temp
New...
Edit...
Delete
System variables
Variable
Value
asl.log
Destination=file
ComSpec
C:\WINDOWS\system32\cmd.exe
DriverData
C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS
16
NVM_HOME
C:\Users\ASUS\AppData\Roaming\nvm
NVM_SYMLINK
C:\Program Files\nodejs
OS
Windows_NT
Path
C:\Program Files (x86)\VMware\VMware Player\bin\;C:\Program...
New...
Edit...
Delete
OK
Cancel

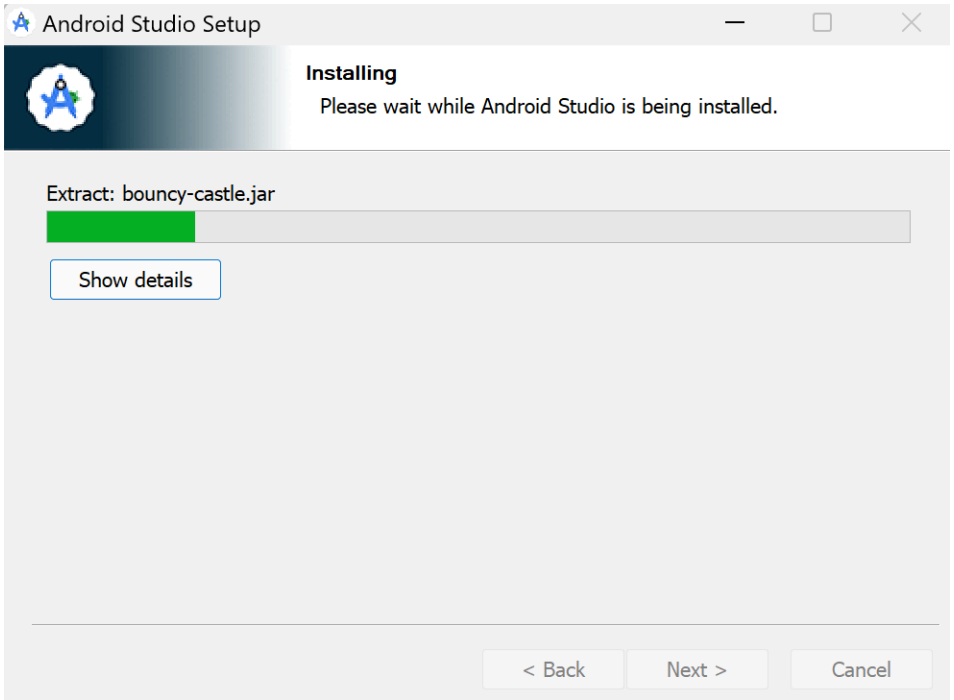

Edit environment variable
C:\Windows\System32\Wbem
C:\Windows\System32\WindowsPowerShell\v1.0\
C:\Windows\System32\OpenSSH\
C:\Program Files\dotnet\
C:\MinGW\bin
C:\MinGW\bin\g++.exe
C:\Users\ASUS\AppData\Roaming\Microsoft\Windows\Start M...
C:\Users\ASUS\AppData\Local\Programs\Eclipse Adoptium\jdk...
C:\Users\ASUS\AppData\Local\Microsoft\WindowsApps
C:\Users\ASUS\AppData\Local\Programs\Microsoft VS Code\bin
C:\Program Files\JetBrains\IntelliJ IDEA Educational Edition 202...
C:\Python34\Scripts
C:\Python39\Scripts
C:\ghcup\bin
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
C:\Users\ASUS\AppData\Local\GitHubDesktop\bin
C:\Users\ASUS\AppData\Roaming\npm
C:\Users\ASUS\AppData\Roaming\Python\Python311\Scripts
%NVM_HOME%
%NVM_SYMLINK%
E:\Downloads\apache-maven-3.9.9-bin\apache-maven-3.9.9\b...
C:\Program Files (x86)\Nmap
E:\flutter_windows_3.27.3-stable\flutter\bin
New
Edit
Browse...
Delete
Move Up
Move Down
Edit text...
OK
Cancel

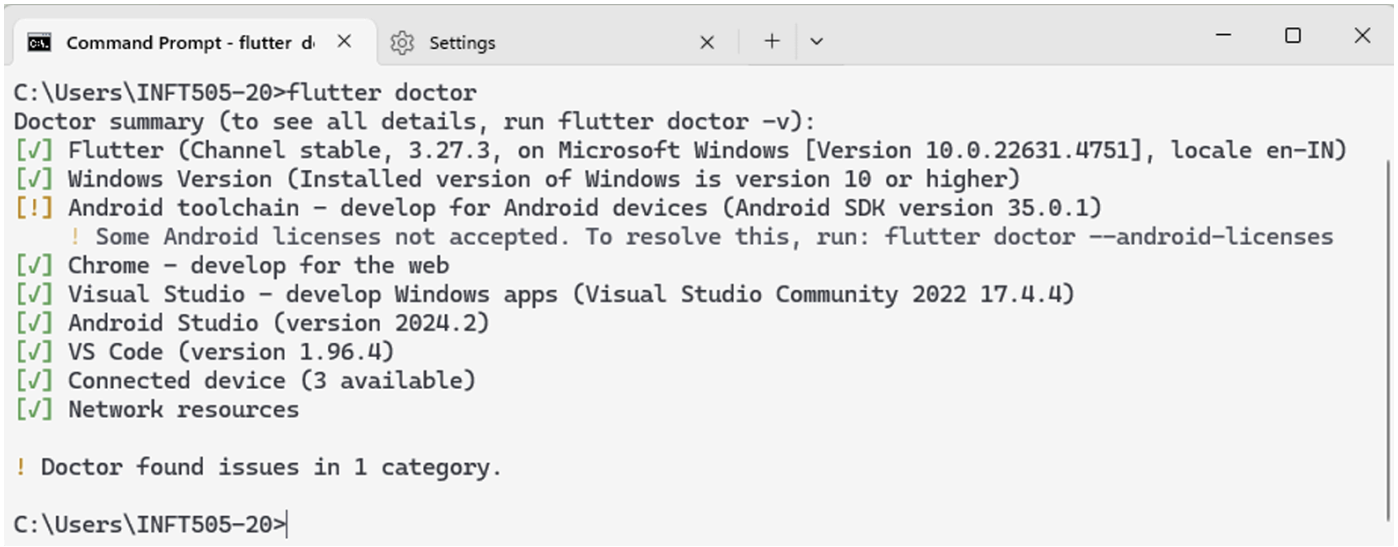


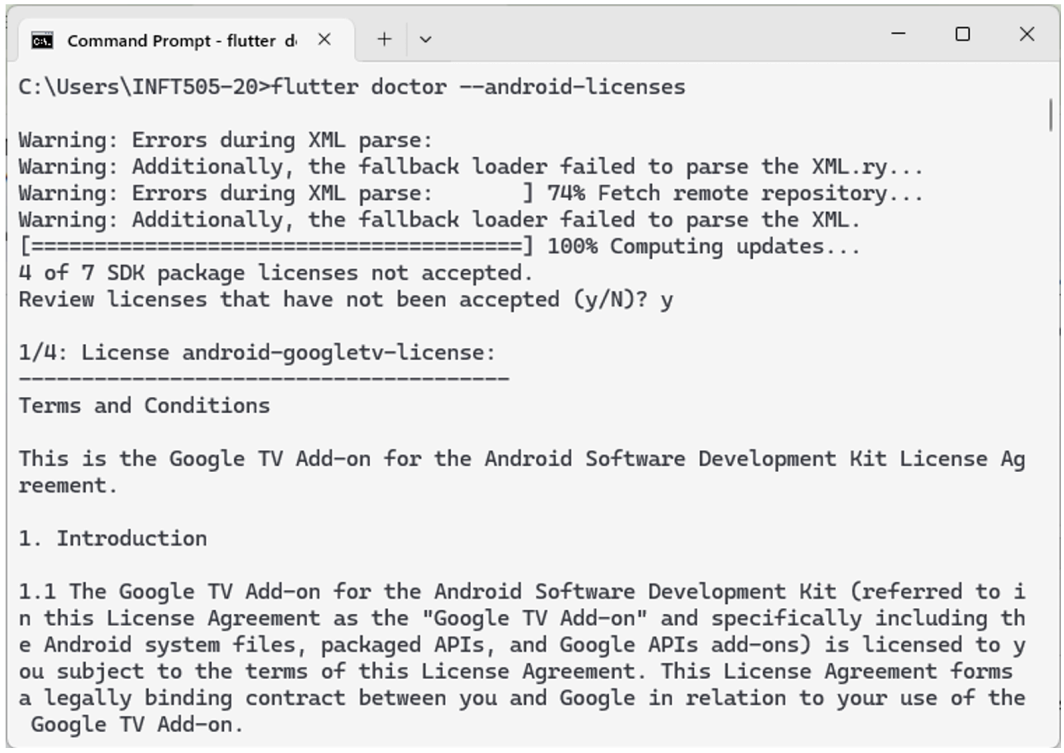


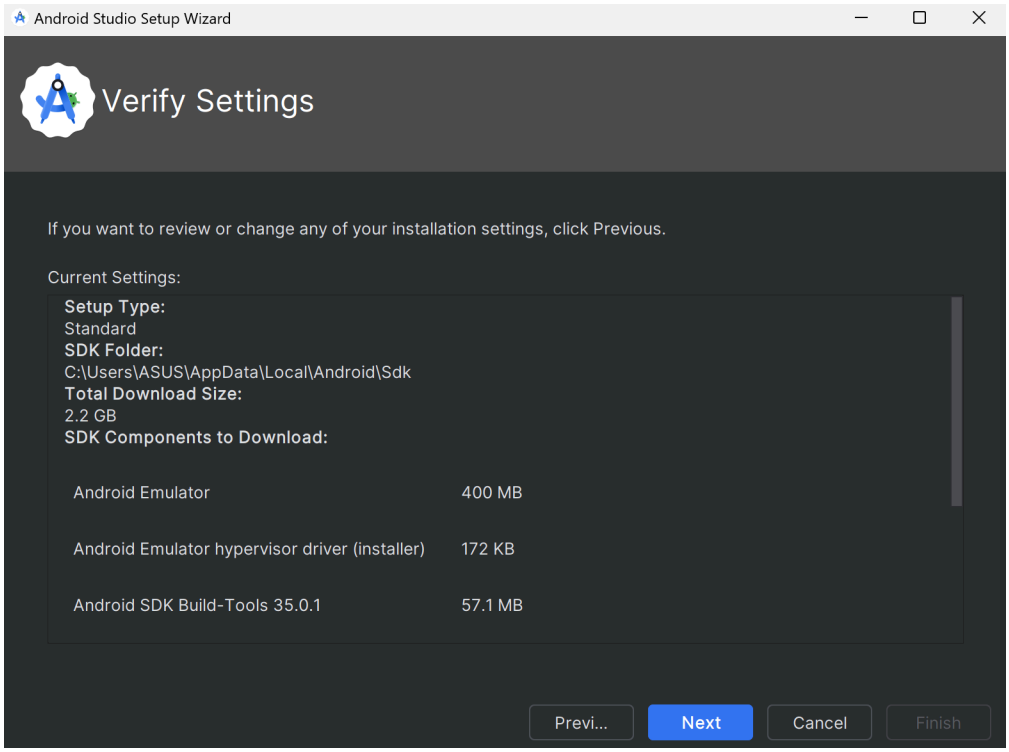
New features

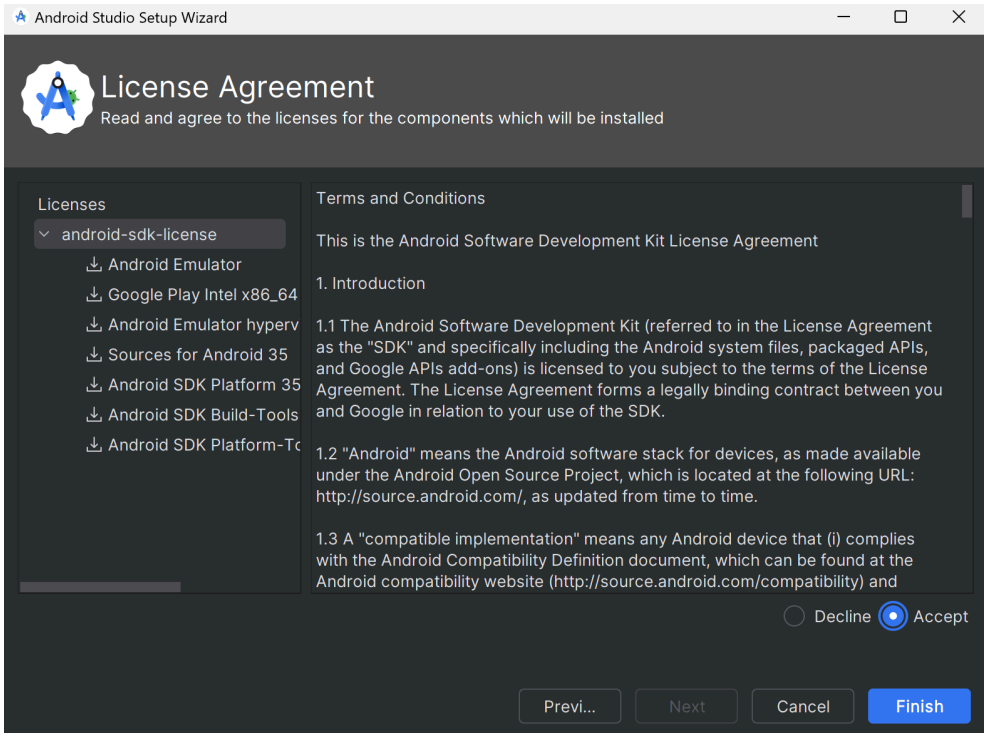

Android Studio Setup
Choose Components
Choose which features of Android Studio you want to install.
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.
Select components to install:
Android Studio
Android Virtual Device
Description
Position your mouse over a component to see its description.
Space required: 3.7GB
< Back
Next >
Cancel

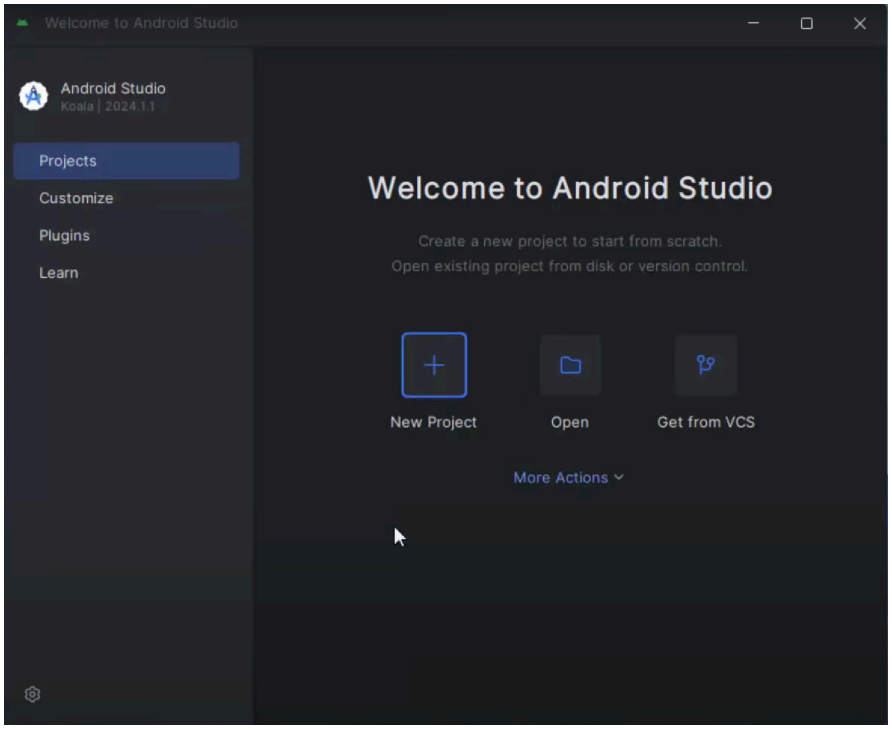

Android Studio Setup
Installing
Please wait while Android Studio is being installed.
Extract: bouncy-castle.jar
Show details
< Back
Next >
Cancel

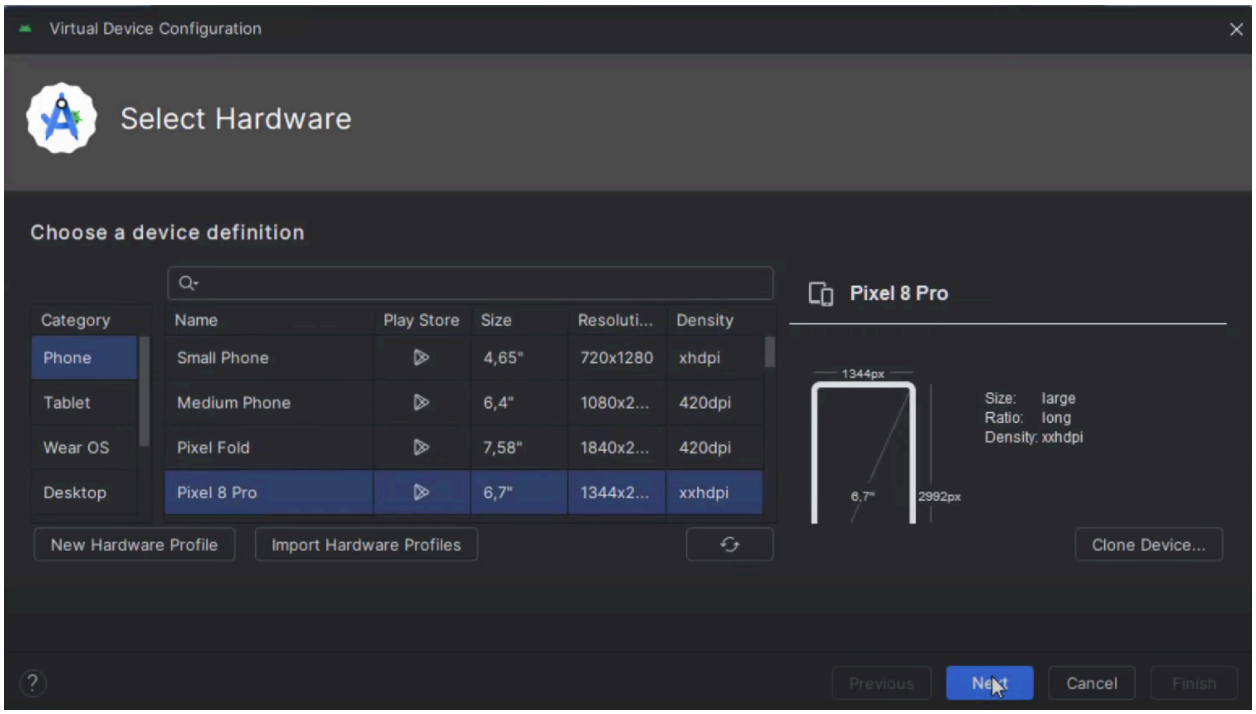

Command Prompt - flutter d
Settings
C:\Users\INFT505-20>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.27.3, on Microsoft Windows [Version 10.0.22631.4751], locale en-IN)
[✓] Windows Version (Installed version of Windows is version 10 or higher)
[!] Android toolchain - develop for Android devices (Android SDK version 35.0.1)
! Some Android licenses not accepted. To resolve this, run: flutter doctor --android-licenses
[✓] Chrome - develop for the web
[✓] Visual Studio - develop Windows apps (Visual Studio Community 2022 17.4.4)
[✓] Android Studio (version 2024.2)
[✓] VS Code (version 1.96.4)
[✓] Connected device (3 available)
[✓] Network resources
! Doctor found issues in 1 category.
C:\Users\INFT505-20>

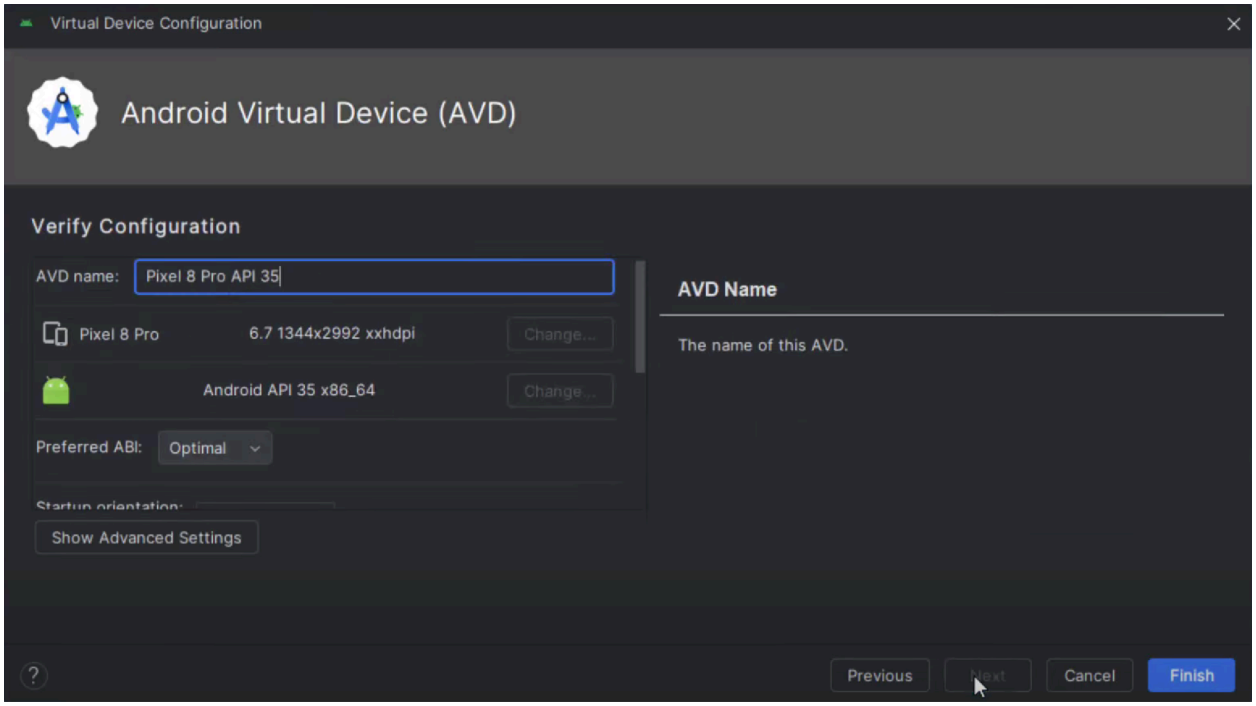

Command Prompt - flutter d
C:\Users\INFT505-20>flutter doctor --android-licenses
Warning: Errors during XML parse:
Warning: Additionally, the fallback loader failed to parse the XML.ry...
Warning: Errors during XML parse:] 74% Fetch remote repository...
Warning: Additionally, the fallback loader failed to parse the XML.
[==========] 100% Computing updates...
4 of 7 SDK package licenses not accepted.
Review licenses that have not been accepted (y/N)? y
1/4: License android-googletv-license:
Terms and Conditions
This is the Google TV Add-on for the Android Software Development Kit License Ag
reement.
1. Introduction
1.1 The Google TV Add-on for the Android Software Development Kit (referred to i
n this License Agreement as the "Google TV Add-on" and specifically including th
e Android system files, packaged APIs, and Google APIs add-ons) is licensed to y
ou subject to the terms of this License Agreement. This License Agreement forms
a legally binding contract between you and Google in relation to your use of the
Google TV Add-on.

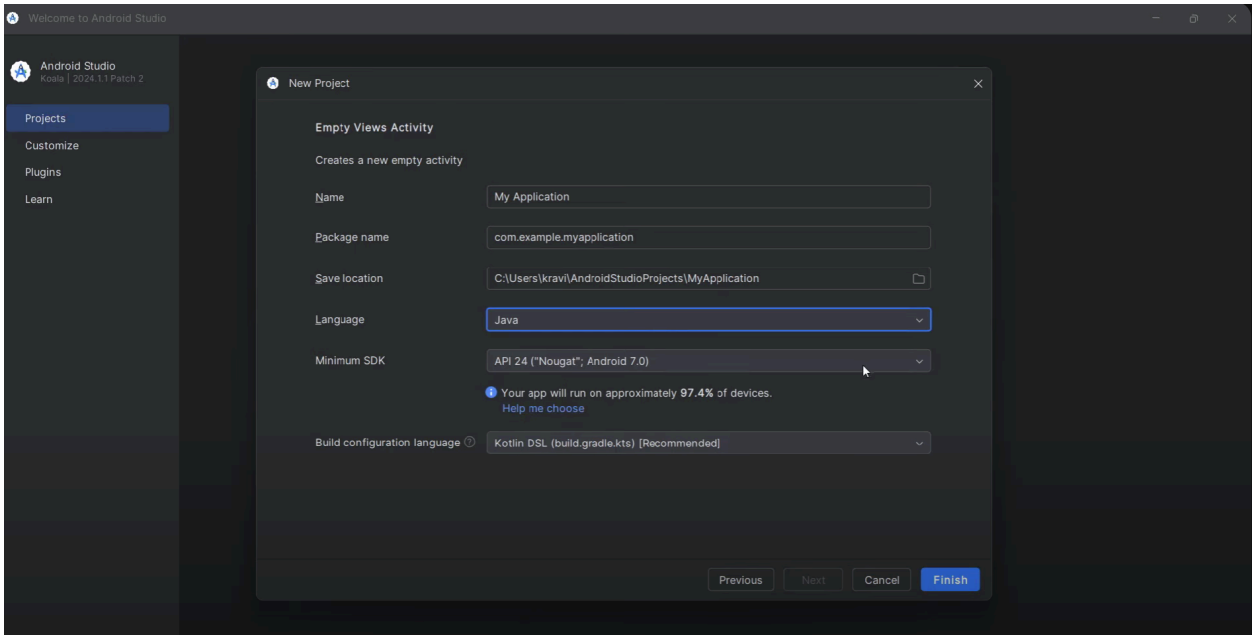

Android Studio Setup Wizard
Verify Settings
If you want to review or change any of your installation settings, click Previous.
Current Settings:
Setup Type:
Standard
SDK Folder:
C:\Users\ASUS\AppData\Local\Android\Sdk
Total Download Size:
2.2 GB
SDK Components to Download:
Android Emulator 400 MB
Android Emulator hypervisor driver (installer) 172 KB
Android SDK Build-Tools 35.0.1 57.1 MB
Previ...
Next
Cancel
Finish

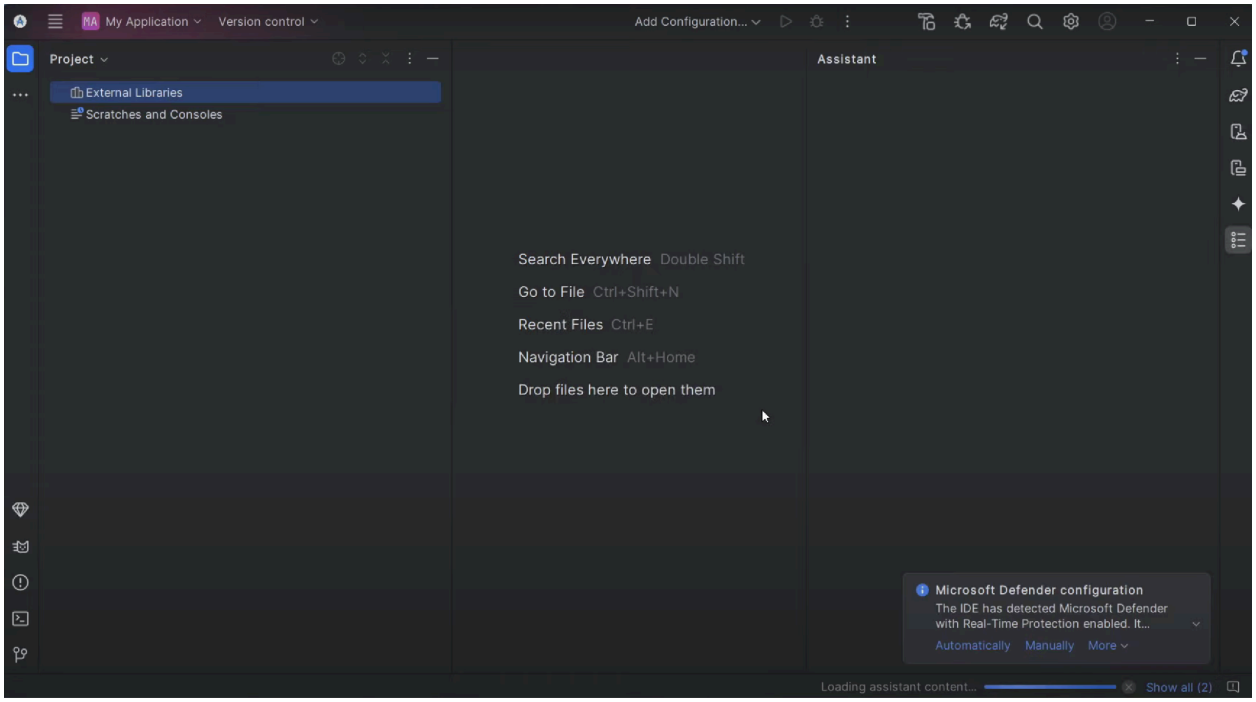

Android Studio Setup Wizard
License Agreement
Read and agree to the licenses for the components which will be installed
Licenses
android-sdk-license
Android Emulator
Google Play Intel x86_64
Android Emulator hyperv
Sources for Android 35
Android SDK Platform 35
Android SDK Build-Tools
Android SDK Platform-To
Terms and Conditions
This is the Android Software Development Kit License Agreement
1. Introduction
1.1 The Android Software Development Kit (referred to in the License Agreement as the "SDK" and specifically including the Android system files, packaged APIs, and Google APIs add-ons) is licensed to you subject to the terms of the License Agreement. The License Agreement forms a legally binding contract between you and Google in relation to your use of the SDK.
1.2 "Android" means the Android software stack for devices, as made available under the Android Open Source Project, which is located at the following URL: http://source.android.com/, as updated from time to time.
1.3 A "compatible implementation" means any Android device that (i) complies with the Android Compatibility Definition document, which can be found at the Android compatibility website (http://source.android.com/compatibility) and
Decline
Accept
Previ...
Next
Cancel
Finish

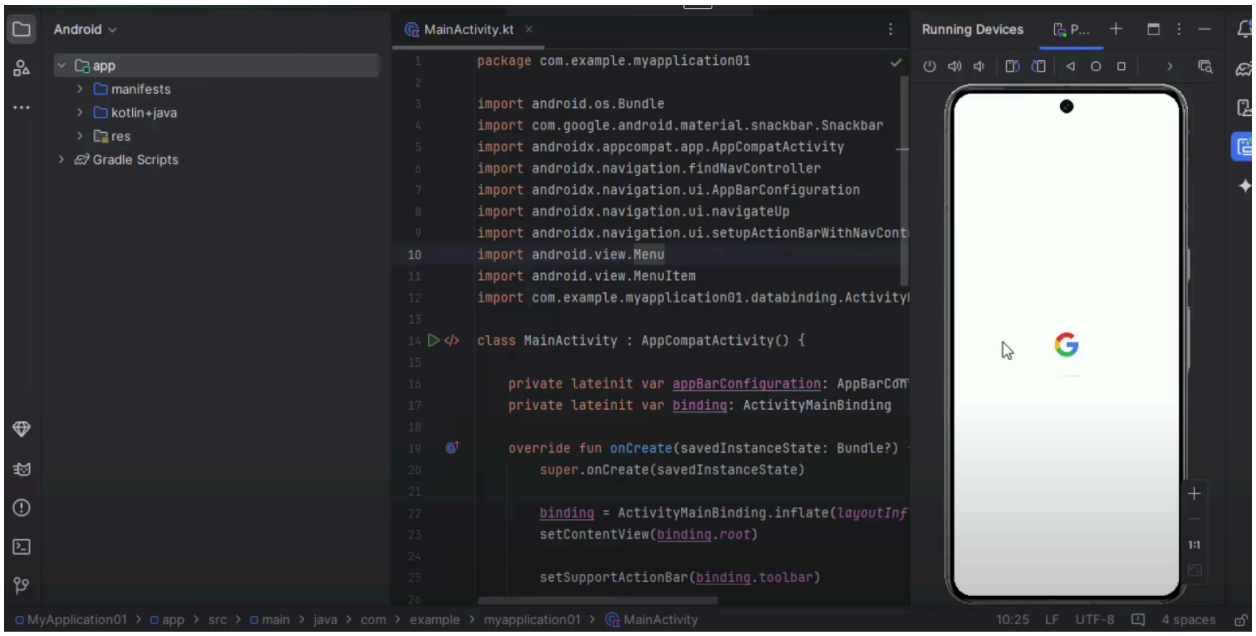

Welcome to Android Studio
Android Studio
Koala | 2024.1.1
Projects
Customize
Plugins
Learn
Welcome to Android Studio
Create a new project to start from scratch.
Open existing project from disk or version control.
New Project
Open
Get from VCS
More Actions


Virtual Device Configuration
Select Hardware
Choose a device definition
Category
Phone
Tablet
Wear OS
Desktop
Name
Play Store
Size
Resoluti...
Density
Small Phone
4,65"
720x1280
xhdpi
Medium Phone
6,4"
1080x2...
420dpi
Pixel Fold
7,58"
1840x2...
420dpi
Pixel 8 Pro
6,7"
1344x2...
xxhdpi
Pixel 8 Pro
1344px
2992px
6.7"
Size: large
Ratio: long
Density: xxhdpi
New Hardware Profile
Import Hardware Profiles
Clone Device...
Previous
Next
Cancel
Finish


Virtual Device Configuration
Android Virtual Device (AVD)
Verify Configuration
AVD name:
Pixel 8 Pro API 35
Pixel 8 Pro
6.7 1344x2992 xxhdpi
Change...
Android API 35 x86_64
Change...
Preferred ABI:
Optimal
Show Advanced Settings
AVD Name
The name of this AVD.
Previous
Next
Cancel
Finish


Welcome to Android Studio
Android Studio
Koala | 2024.1.1 Patch 2
Projects
Customize
Plugins
Learn
New Project
Empty Views Activity
Creates a new empty activity
Name
My Application
Package name
com.example.myapplication
Save location
C:\Users\kravi\AndroidStudioProjects\MyApplication
Language
Java
Minimum SDK
API 24 ("Nougat"; Android 7.0)
Your app will run on approximately 97.4% of devices.
Help me choose
Build configuration language
Kotlin DSL (build.gradle.kts) [Recommended]
Previous
Next
Cancel
Finish


My Application
Version control
Add Configuration...
Project
External Libraries
Scratches and Consoles
Assistant
Search Everywhere Double Shift
Go to File Ctrl+Shift+N
Recent Files Ctrl+E
Navigation Bar Alt+Home
Drop files here to open them
Microsoft Defender configuration
The IDE has detected Microsoft Defender with Real-Time Protection enabled. It...
Automatically
Manually
More
Loading assistant content...
Show all (2)


Android
app
manifests
kotlin+java
res
Gradle Scripts
MainActivity.kt
package com.example.myapplication01
import android.os.Bundle
import com.google.android.material.snackbar.Snackbar
import androidx.appcompat.app.AppCompatActivity
import androidx.navigation.findNavController
import androidx.navigation.ui.AppBarConfiguration
import androidx.navigation.ui.navigateUp
import androidx.navigation.ui.setupActionBarWithNavCont
import android.view.Menu
import android.view.MenuItem
import com.example.myapplication01.databinding.Activity
class MainActivity : AppCompatActivity() {
private lateinit var appBarConfiguration: AppBarCon
private lateinit var binding: ActivityMainBinding
override fun onCreate(savedInstanceState: Bundle?)
super.onCreate(savedInstanceState)
binding = ActivityMainBinding.inflate(layoutInf
setContentView(binding.root)
setSupportActionBar(binding.toolbar)
Running Devices
MyApplication01 › app › src › main › java › com › example › myapplication01 › MainActivity
10:25 LF UTF-8 4 spaces

Experiment 02 : To design Flutter UI by including common widgets.

Sign Up Screen

Code:

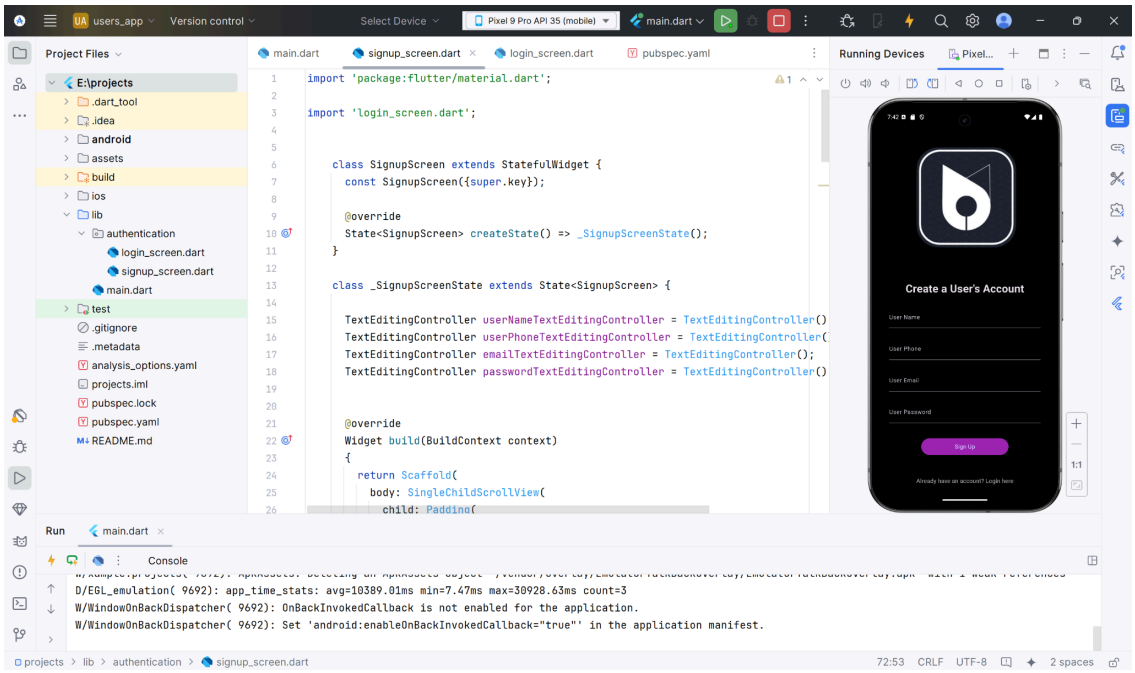
```
import 'package:flutter/material.dart';
import 'package:projects/methods/common_methods.dart';

import 'login_screen.dart';


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

    @override
    State<SignupScreen> createState() => _SignupScreenState();
  }

  class _SignupScreenState extends State<SignupScreen> {

    TextEditingController userNameTextEditingController = TextEditingController();
    TextEditingController userPhoneTextEditingController = TextEditingController();
    TextEditingController emailTextEditingController = TextEditingController();
    TextEditingController passwordTextEditingController = TextEditingController();
    CommonMethods cMethods = CommonMethods();

    checkIfNetworkIsAvailable()
    {
      cMethods.checkConnectivity(context);
      signUpFormValidation();
    }


    @override
    Widget build(BuildContext context)
    {
      return Scaffold(
        body: SingleChildScrollView(
          child: Padding(
              padding: const EdgeInsets.all(10),
              child: Column(
                children: [
                  Image.asset(
                    "assets/images/logo.png"
```

```
),

Text(
  "Create a User\'s Account",
  style: TextStyle(
    fontSize: 26,
    fontWeight: FontWeight.bold,
    fontFamily: 'Montserrat',
  ),
),

// text fields
Padding(
  padding: const EdgeInsets.all(22),
  child: Column(
    children: [

      TextField(
        controller: userNameTextEditingController,
        keyboardType: TextInputType.text,
        decoration: const InputDecoration(
          labelText: "User Name",
          labelStyle: TextStyle(
            fontSize: 14,
          ),
        ),
        style: const TextStyle(
          color: Colors.grey,
          fontSize: 15
        ),
      ),


      const SizedBox(
        height: 22,
      ),

      TextField(
        controller: userPhoneTextEditingController,
        keyboardType: TextInputType.number,
        decoration: const InputDecoration(
          labelText: "User Phone",
          prefixIcon: Icon(Icons.phone, color: Colors.purple),
          labelStyle: TextStyle(
```

```
      fontSize: 14,
    ),
  ),
  style: const TextStyle(
      color: Colors.grey,
      fontSize: 15
  ),
),


const SizedBox(
  height: 22,
),

TextField(
  controller: emailTextEditingController,
  keyboardType: TextInputType.emailAddress,
  decoration: const InputDecoration(
    labelText: "User Email",
    prefixIcon: Icon(Icons.email, color: Colors.purple),
    labelStyle: TextStyle(
      fontSize: 14,
    ),
  ),
  style: const TextStyle(
      color: Colors.grey,
      fontSize: 15
  ),
),

const SizedBox(
  height: 22,
),

TextField(
  controller: passwordTextEditingController,
  obscureText: true,
  keyboardType: TextInputType.text,
  decoration: const InputDecoration(
    labelText: "User Password",
    labelStyle: TextStyle(
      fontSize: 14,
    ),
  ),
```

```
      style: const TextStyle(
          color: Colors.grey,
          fontSize: 15
      ),
    ),

    const SizedBox(
      height: 32,
    ),

    ElevatedButton(
      onPressed: ()
      {
        checkIfNetworkIsAvailable();
      },
      style: ElevatedButton.styleFrom(
        backgroundColor: Colors.purple,
        padding: const EdgeInsets.symmetric(horizontal: 80,vertical: 10)
      ),
      child: const Text(
        "Sign Up"
      ),
    ),

  ],
 ),
),

const SizedBox(
 height: 12,
),

// textButton
TextButton(
 onPressed: ()
 {
    Navigator.push(context, MaterialPageRoute(builder: (c) => LoginScreen()));
 },
 child: const Text(
  "Already have an account? Login here",
  style: TextStyle(
    color: Colors.grey,
  ),
 )
```

```
            )



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



Login Screen

Code:

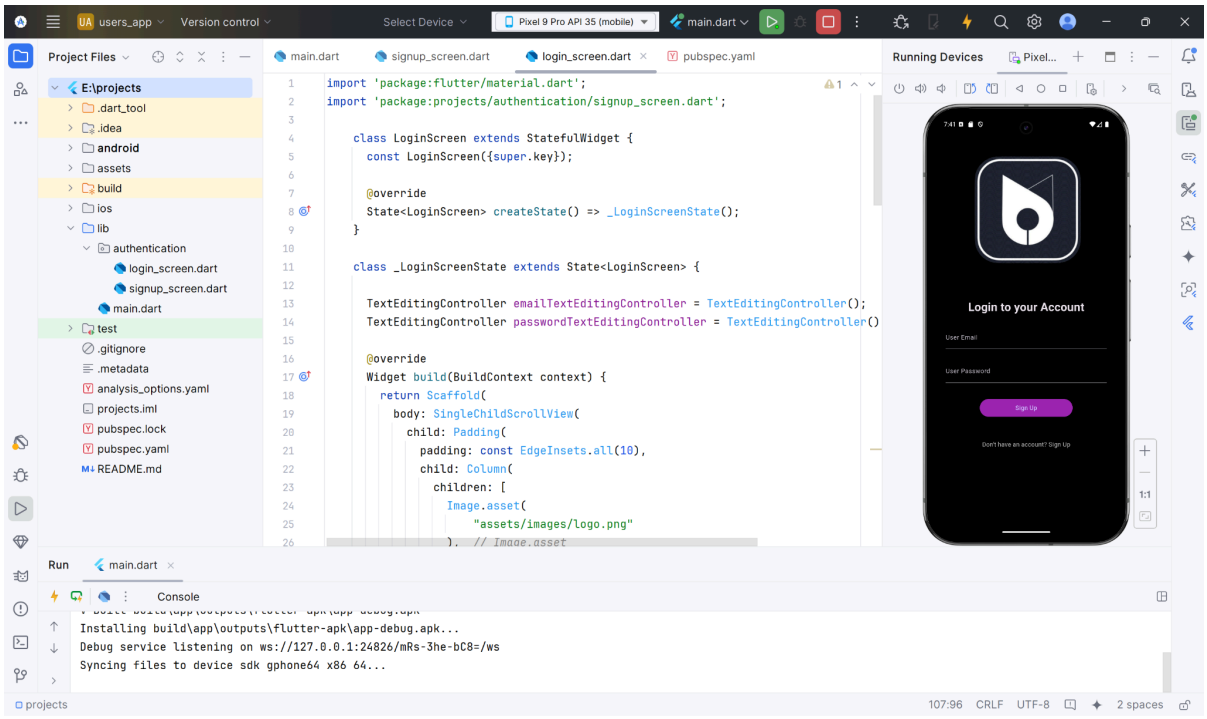
```
import 'package:flutter/material.dart';
import 'package:projects/authentication/signup_screen.dart';

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

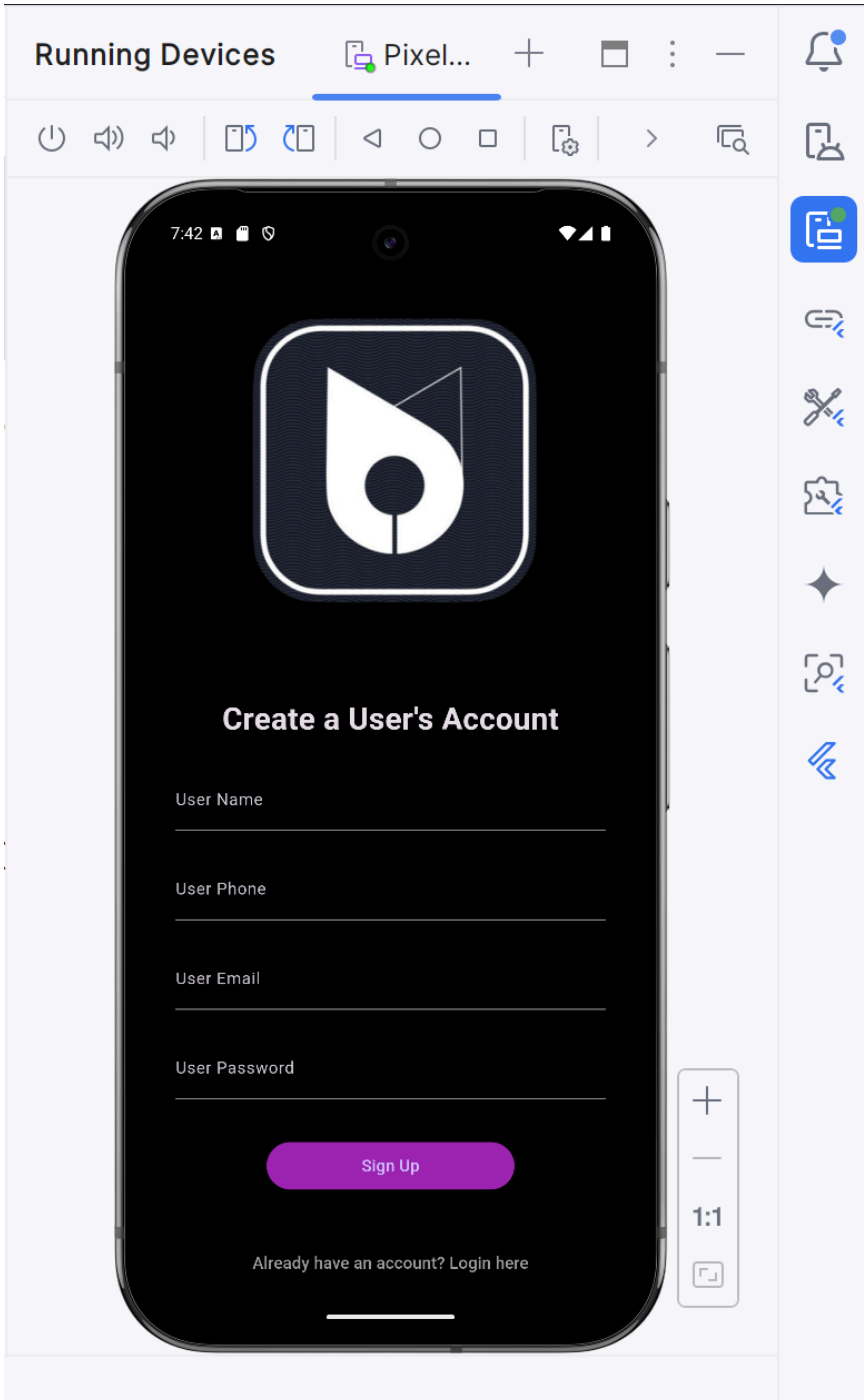
```
  @override
  State<LoginScreen> createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {

  TextEditingController emailTextEditingController = TextEditingController();
  TextEditingController passwordTextEditingController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SingleChildScrollView(
        child: Padding(
          padding: const EdgeInsets.all(10),
          child: Column(
            children: [
              Image.asset(
                  "assets/images/logo.png"
              ),

              Text(
                "Login to your Account",
                style: TextStyle(
                  fontSize: 26,
                  fontWeight: FontWeight.bold,
                  fontFamily: 'Montserrat',
                ),
              ),

              // text fields
              Padding(
                padding: const EdgeInsets.all(22),
                child: Column(
                  children: [

                    TextField(
                      controller: emailTextEditingController,
                      keyboardType: TextInputType.emailAddress,
                      decoration: const InputDecoration(
```

```
    labelText: "User Email",
    labelStyle: TextStyle(
      fontSize: 14,
    ),
  ),
  style: const TextStyle(
      color: Colors.grey,
      fontSize: 15
  ),
),

const SizedBox(
  height: 22,
),

TextField(
  controller: passwordTextEditingController,
  obscureText: true,
  keyboardType: TextInputType.text,
  decoration: const InputDecoration(
    labelText: "User Password",
    labelStyle: TextStyle(
      fontSize: 14,
    ),
  ),
  style: const TextStyle(
      color: Colors.grey,
      fontSize: 15
  ),
),

const SizedBox(
  height: 32,
),

ElevatedButton(
  onPressed: ()
  {

  },
```

```
                    style: ElevatedButton.styleFrom(
                        backgroundColor: Colors.purple,
                        padding: const EdgeInsets.symmetric(horizontal: 80,vertical: 10)
                    ),
                    child: const Text(
                        "Sign Up"
                    ),
                  ),

                ],
              ),
            ),

            const SizedBox(
              height: 12,
            ),

            // textButton
            TextButton(
                onPressed: ()
                {
                  Navigator.push(context, MaterialPageRoute(builder: (c) =>
SignupScreen()));
                },
                child: const Text(
                  "Don\'t have an account? Sign Up",
                  style: TextStyle(
                    color: Colors.grey,
                  ),
                )
            )



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

```
}
```


users_app
Version control
Select Device
Pixel 9 Pro API 35 (mobile)
main.dart
Project Files
E:\projects
.dart_tool
.idea
android
assets
build
ios
lib
authentication
login_screen.dart
signup_screen.dart
main.dart
test
.gitignore
.metadata
analysis_options.yaml
projects.iml
pubspec.lock
pubspec.yaml
README.md
main.dart
signup_screen.dart
login_screen.dart
pubspec.yaml
import 'package:flutter/material.dart';
import 'package:projects/authentication/signup_screen.dart';
class LoginScreen extends StatefulWidget {
const LoginScreen({super.key});
@override
State<LoginScreen> createState() => _LoginScreenState();
}
class _LoginScreenState extends State<LoginScreen> {
TextEditingController emailTextEditingController = TextEditingController();
TextEditingController passwordTextEditingController = TextEditingController()
@override
Widget build(BuildContext context) {
return Scaffold(
body: SingleChildScrollView(
child: Padding(
padding: const EdgeInsets.all(10),
child: Column(
children: [
Image.asset(
"assets/images/logo.png"
), // Image.asset
Running Devices
Pixel...
Login to your Account
User Email
User Password
Sign Up
Don't have an account? Sign Up
1:1
Run
main.dart
Console
Installing build\app\outputs\flutter-apk\app-debug.apk...
Debug service listening on ws://127.0.0.1:24826/mRs-3he-bC8=/ws
Syncing files to device sdk gphone64 x86 64...
projects
107:96
CRLF
UTF-8
2 spaces


Running Devices
Pixel...
7:42
Create a User's Account
User Name
User Phone
User Email
User Password
Sign Up
Already have an account? Login here
1:1

Experiment 03 : To include icons, images, fonts in Flutter app

SignUp Screen includes icons,fonts and Images Code is highlighted by green colour

Code:

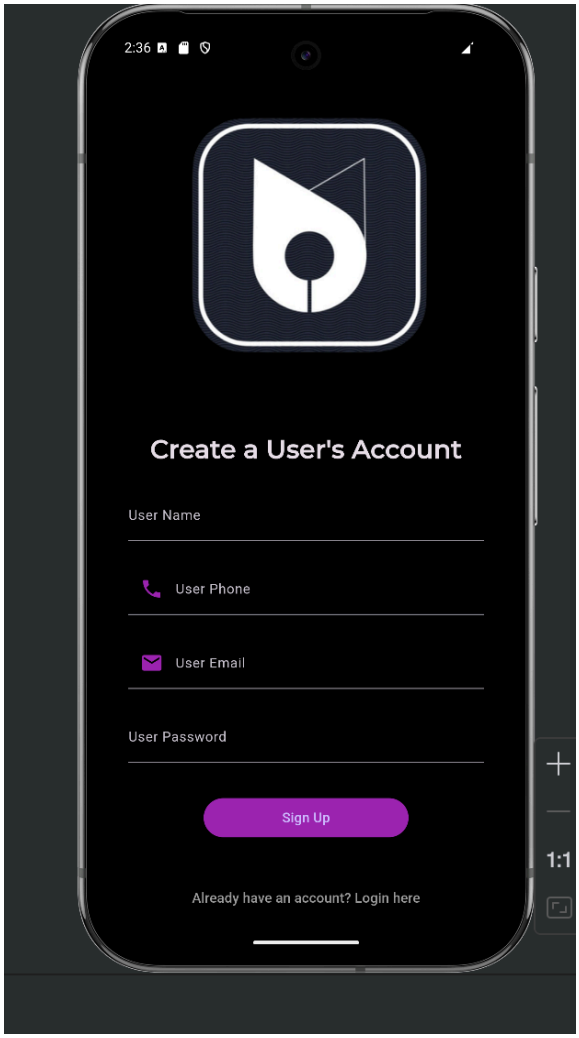
```
import 'package:flutter/material.dart';
import 'package:projects/methods/common_methods.dart';

import 'login_screen.dart';


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

    @override
    State<SignupScreen> createState() => _SignupScreenState();
  }

  class _SignupScreenState extends State<SignupScreen> {

    TextEditingController userNameTextEditingController = TextEditingController();
    TextEditingController userPhoneTextEditingController = TextEditingController();
    TextEditingController emailTextEditingController = TextEditingController();
    TextEditingController passwordTextEditingController = TextEditingController();
    CommonMethods cMethods = CommonMethods();

    checkIfNetworkIsAvailable()
    {
      cMethods.checkConnectivity(context);
      signUpFormValidation();
    }

        @override
    Widget build(BuildContext context)
    {
      return Scaffold(
        body: SingleChildScrollView(
          child: Padding(
              padding: const EdgeInsets.all(10),
              child: Column(
```

```
children: [
  Image.asset(
    "assets/images/logo.png"
  ),

  Text(
    "Create a User\'s Account",
    style: TextStyle(
      fontSize: 26,
      fontWeight: FontWeight.bold,
      fontFamily: 'Montserrat',
    ),
  ),

  // text fields
  Padding(
    padding: const EdgeInsets.all(22),
    child: Column(
      children: [

        TextField(
          controller: userNameTextEditingController,
          keyboardType: TextInputType.text,
          decoration: const InputDecoration(
            labelText: "User Name",
            labelStyle: TextStyle(
              fontSize: 14,
            ),
          ),
          style: const TextStyle(
            color: Colors.grey,
            fontSize: 15
          ),
        ),


        const SizedBox(
          height: 22,
        ),
```

```
TextField(
  controller: userPhoneTextEditingController,
  keyboardType: TextInputType.number,
  decoration: const InputDecoration(
    labelText: "User Phone",
    prefixIcon: Icon(Icons.phone, color: Colors.purple),
    labelStyle: TextStyle(
      fontSize: 14,
    ),
  ),
  style: const TextStyle(
      color: Colors.grey,
      fontSize: 15
  ),
),


const SizedBox(
  height: 22,
),

TextField(
  controller: emailTextEditingController,
  keyboardType: TextInputType.emailAddress,
  decoration: const InputDecoration(
    labelText: "User Email",
    prefixIcon: Icon(Icons.email, color: Colors.purple),
    labelStyle: TextStyle(
      fontSize: 14,
    ),
  ),
  style: const TextStyle(
      color: Colors.grey,
      fontSize: 15
  ),
),

const SizedBox(
  height: 22,
),
```

```
      TextField(
        controller: passwordTextEditingController,
        obscureText: true,
        keyboardType: TextInputType.text,
        decoration: const InputDecoration(
          labelText: "User Password",
          labelStyle: TextStyle(
            fontSize: 14,
          ),
        ),
        style: const TextStyle(
            color: Colors.grey,
            fontSize: 15
        ),
      ),

      const SizedBox(
        height: 32,
      ),

      ElevatedButton(
        onPressed: ()
        {
          checkIfNetworkIsAvailable();
        },
        style: ElevatedButton.styleFrom(
          backgroundColor: Colors.purple,
          padding: const EdgeInsets.symmetric(horizontal: 80,vertical: 10)
        ),
        child: const Text(
          "Sign Up"
        ),
      ),

    ],
  ),
),

const SizedBox(
```

```
                  height: 12,
                ),

                // textButton
                TextButton(
                  onPressed: ()
                  {
                      Navigator.push(context, MaterialPageRoute(builder: (c) =>
LoginScreen()));
                  },
                  child: const Text(
                    "Already have an account? Login here",
                    style: TextStyle(
                      color: Colors.grey,
                    ),
                  )
                )



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


2:36
Create a User's Account
User Name
User Phone
User Email
User Password
Sign Up
Already have an account? Login here
1:1

Exp 4 To create an interactive Form using form widget

SignUp Form Validation Code highlighted by green

Code:

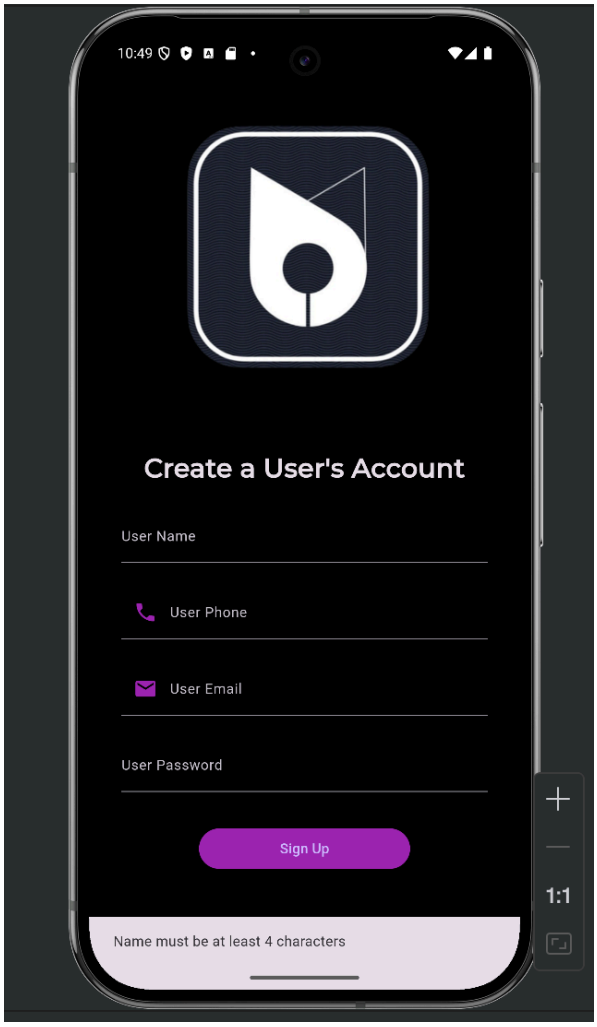
```
import 'package:flutter/material.dart';
import 'package:projects/methods/common_methods.dart';

import 'login_screen.dart';


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

    @override
    State<SignupScreen> createState() => _SignupScreenState();
  }

  class _SignupScreenState extends State<SignupScreen> {

    TextEditingController userNameTextEditingController = TextEditingController();
    TextEditingController userPhoneTextEditingController = TextEditingController();
    TextEditingController emailTextEditingController = TextEditingController();
    TextEditingController passwordTextEditingController = TextEditingController();
    CommonMethods cMethods = CommonMethods();

    checkIfNetworkIsAvailable()
    {
      cMethods.checkConnectivity(context);
      signUpFormValidation();
    }

    signUpFormValidation(){
      if(userNameTextEditingController.text.trim().length < 3)
        {
          cMethods.displaySnackBar("Name must be at least 4 characters", context);
        }
      else if(userPhoneTextEditingController.text.trim().length < 10)
        {
          cMethods.displaySnackBar("Phone number must be at least 10 characters", context);
        }
      else if(!emailTextEditingController.text.contains("@"))
```

```
    {
      cMethods.displaySnackBar("Email address is not valid", context);
    }
  else if(passwordTextEditingController.text.trim().length < 6)
    {
      cMethods.displaySnackBar("Password must be at least 6 characters", context);
    }
  else
    {
      // register User
    }
}

@override
Widget build(BuildContext context)
{
  return Scaffold(
    body: SingleChildScrollView(
      child: Padding(
          padding: const EdgeInsets.all(10),
          child: Column(
            children: [
              Image.asset(
                "assets/images/logo.png"
              ),

              Text(
                "Create a User\'s Account",
                style: TextStyle(
                  fontSize: 26,
                  fontWeight: FontWeight.bold,
                  fontFamily: 'Montserrat',
                ),
              ),

              // text fields
              Padding(
                padding: const EdgeInsets.all(22),
                child: Column(
                  children: [

                    TextField(
                      controller: userNameTextEditingController,
                      keyboardType: TextInputType.text,
```

```
  decoration: const InputDecoration(
    labelText: "User Name",
    labelStyle: TextStyle(
      fontSize: 14,
    ),
  ),
  style: const TextStyle(
    color: Colors.grey,
    fontSize: 15
  ),
),


const SizedBox(
  height: 22,
),

TextField(
  controller: userPhoneTextEditingController,
  keyboardType: TextInputType.number,
  decoration: const InputDecoration(
    labelText: "User Phone",
    prefixIcon: Icon(Icons.phone, color: Colors.purple),
    labelStyle: TextStyle(
      fontSize: 14,
    ),
  ),
  style: const TextStyle(
      color: Colors.grey,
      fontSize: 15
  ),
),


const SizedBox(
  height: 22,
),

TextField(
  controller: emailTextEditingController,
  keyboardType: TextInputType.emailAddress,
  decoration: const InputDecoration(
    labelText: "User Email",
    prefixIcon: Icon(Icons.email, color: Colors.purple),
```

```
    labelStyle: TextStyle(
      fontSize: 14,
    ),
  ),
  style: const TextStyle(
      color: Colors.grey,
      fontSize: 15
  ),
),

const SizedBox(
  height: 22,
),

TextField(
  controller: passwordTextEditingController,
  obscureText: true,
  keyboardType: TextInputType.text,
  decoration: const InputDecoration(
    labelText: "User Password",
    labelStyle: TextStyle(
      fontSize: 14,
    ),
  ),
  style: const TextStyle(
      color: Colors.grey,
      fontSize: 15
  ),
),

const SizedBox(
  height: 32,
),

ElevatedButton(
  onPressed: ()
  {
    checkIfNetworkIsAvailable();
  },
  style: ElevatedButton.styleFrom(
    backgroundColor: Colors.purple,
    padding: const EdgeInsets.symmetric(horizontal: 80,vertical: 10)
  ),
  child: const Text(
```

```
                    "Sign Up"
                  ),
                ),

              ],
            ),
          ),

          const SizedBox(
            height: 12,
          ),

          // textButton
          TextButton(
            onPressed: ()
            {
               Navigator.push(context, MaterialPageRoute(builder: (c) => LoginScreen()));
            },
            child: const Text(
              "Already have an account? Login here",
              style: TextStyle(
                color: Colors.grey,
              ),
            )
          )



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


10:49
Create a User's Account
User Name
User Phone
User Email
User Password
Sign Up
Name must be at least 4 characters
1:1

Name : Shreyas Naik
Roll no : 38
Div : D15B

Experiment 5

AIM : Exp 5 To apply navigation, routing and gestures in Flutter App

Code :

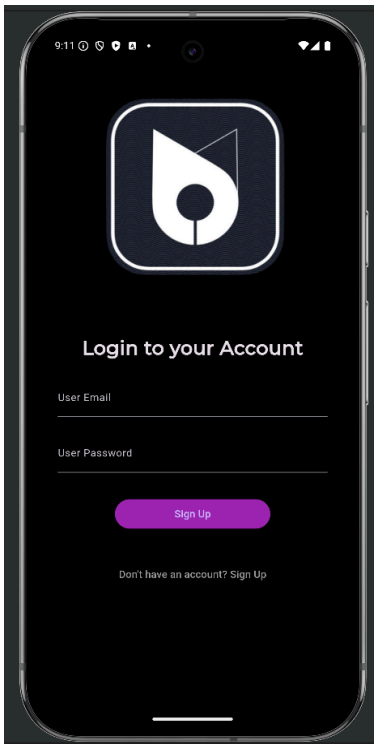
login_screen.dart

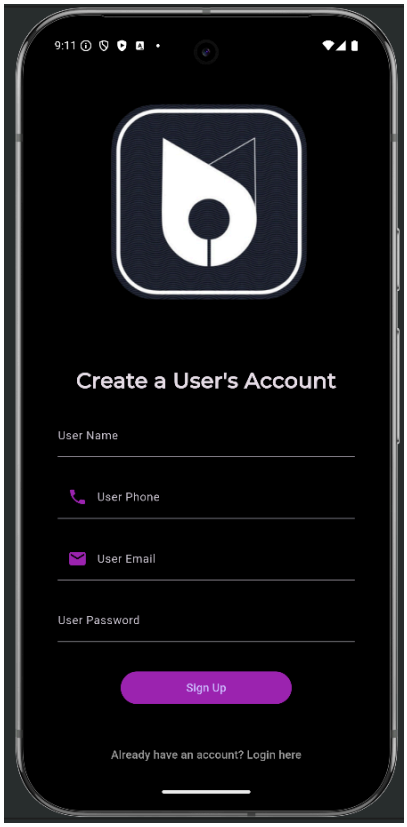
```
TextButton(
        onPressed: ()
        {
          Navigator.push(context, MaterialPageRoute(builder: (c) => SignupScreen()));
        },
        child: const Text(
          "Don\'t have an account? Sign Up",
          style: TextStyle(
            color: Colors.grey,
          ),
        )
    )
```

signup_screen.dart

```
TextButton(
    onPressed: () {
      Navigator.push(context, MaterialPageRoute(builder: (c) => LoginScreen()));
    },
    child: const Text(
      "Already have an account? Login here",
      style: TextStyle(color: Colors.grey),
    ),
  ),
```

OUTPUT


9:11
Login to your Account
User Email
User Password
Sign Up
Don't have an account? Sign Up


9:11
Create a User's Account
User Name
User Phone
User Email
User Password
Sign Up
Already have an account? Login here

Name : Shreyas Naik
Roll no : 38
Div : D15B

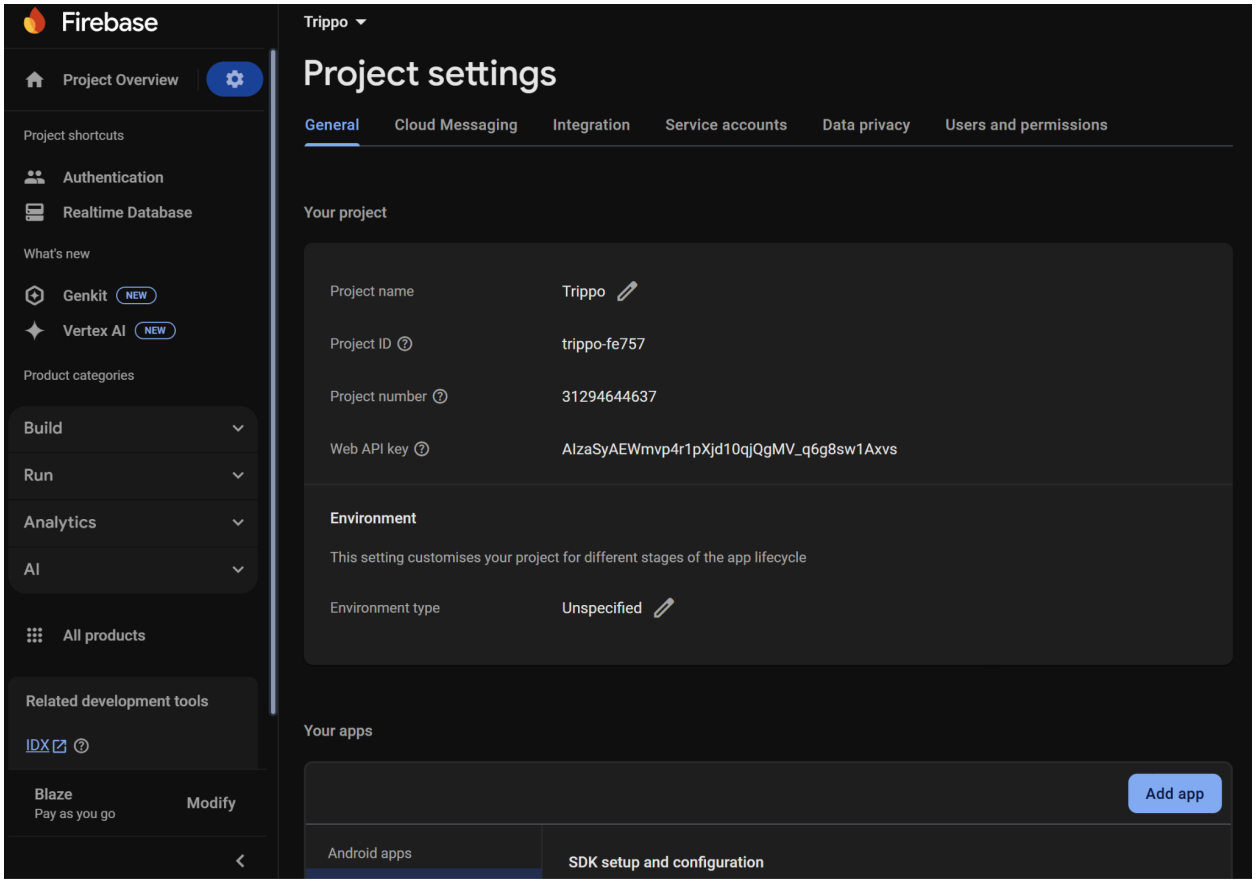
Experiment 6

AIM : How To Set Up Firebase with Flutter for iOS and Android Apps

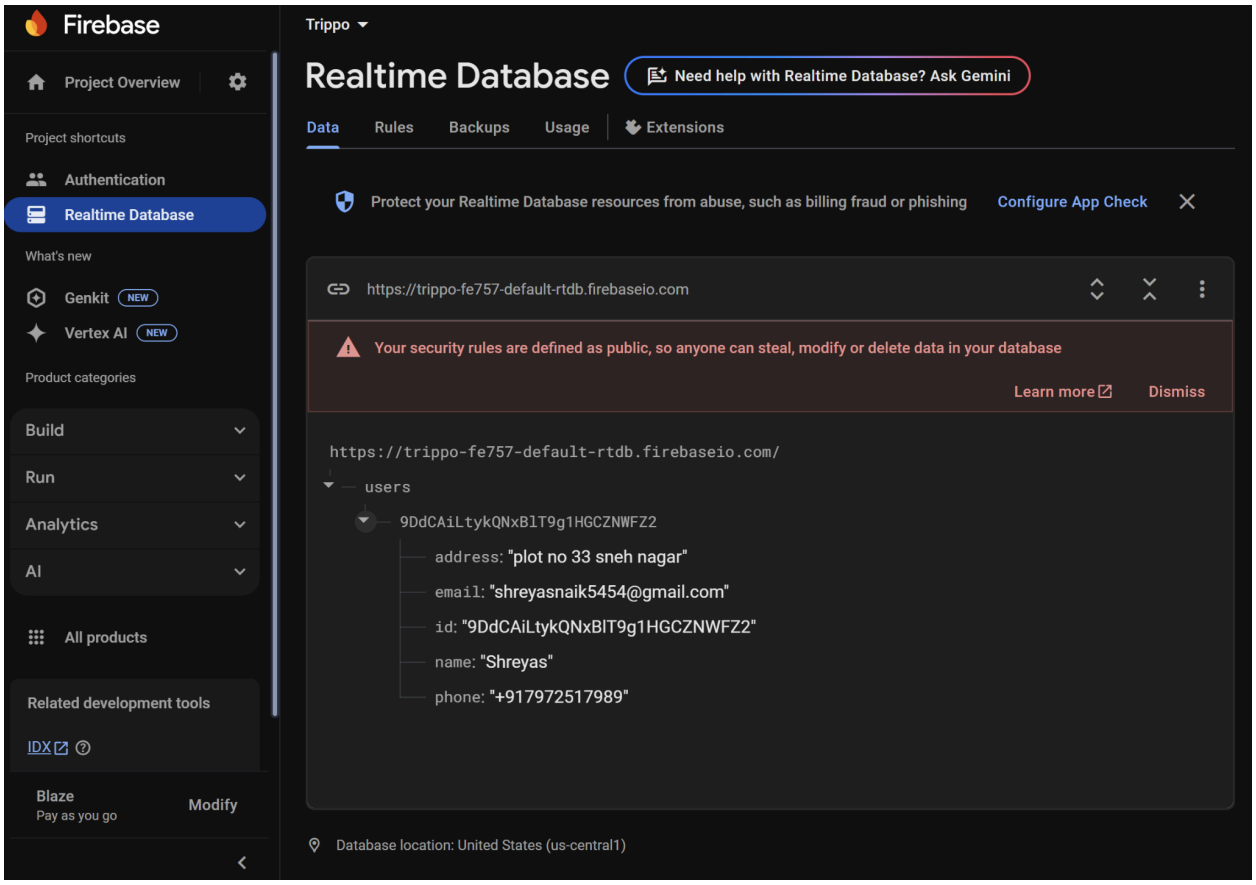
Code :
Add below code in main.dart file

```
void main() async {
  WidgetsFlutterBinding.ensureInitialized(); // Ensure Flutter is ready
  await Firebase.initializeApp(); // Initialize Firebase before runApp
  runApp(const MyApp()); // Only start app after Firebase is ready
}
```

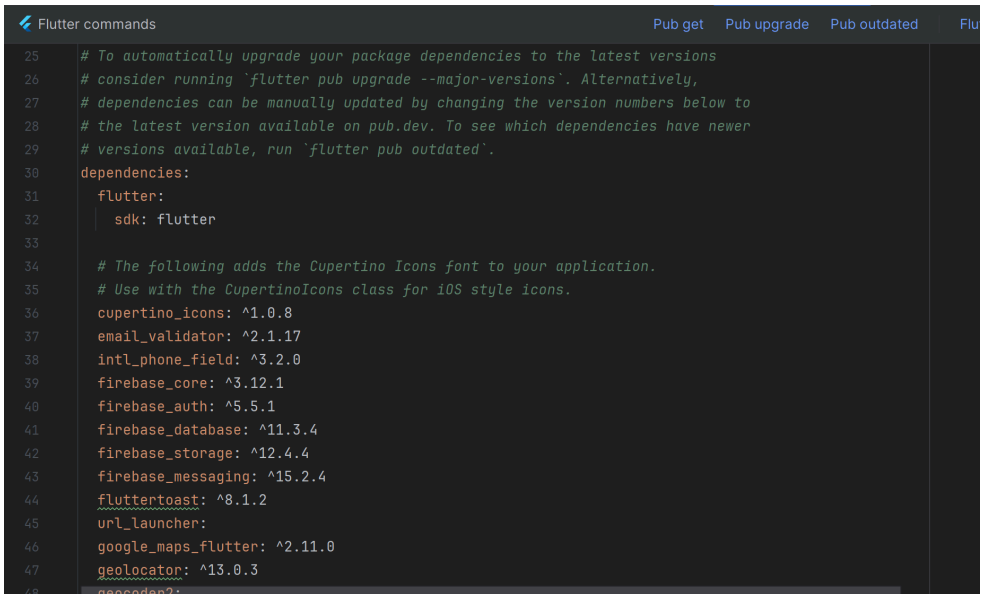
Create a realtime database

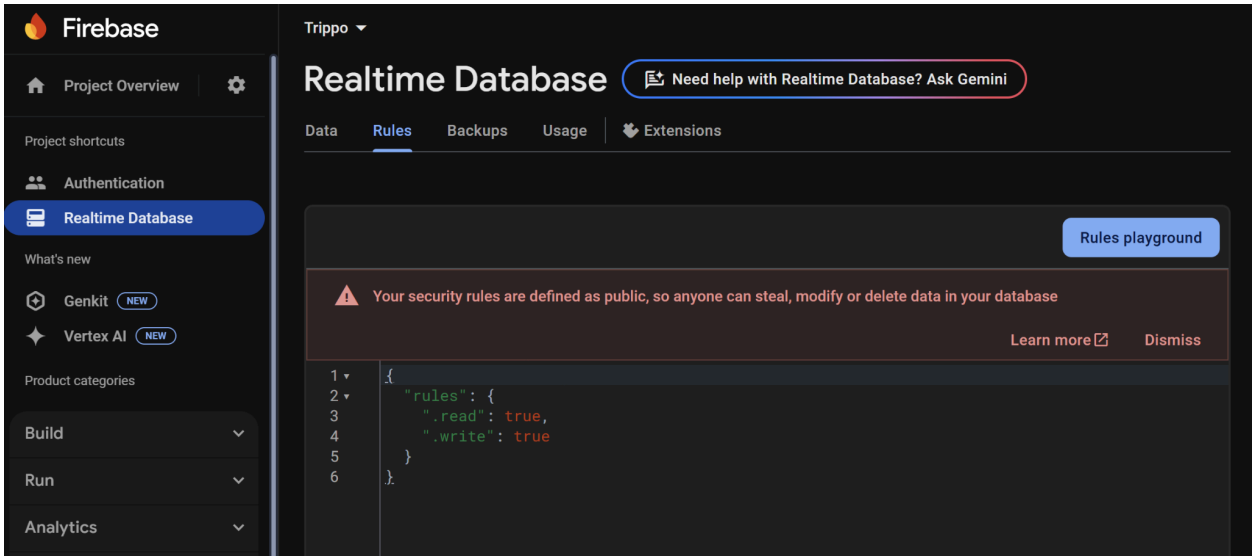


Download google-services.json file and place it in android/app folder in flutter app



Add required dependencies in pubspec.yaml file and run flutter pub get





Make the read and write permissions to true

Name : Shreyas Naik
Roll no : 38
Div: D15B

MPL Experiment 7

AIM : Implement add to home screen feature

Code :

Run npm install vite-plugin-pwa --save-devi in your project folder to install pwa plugins

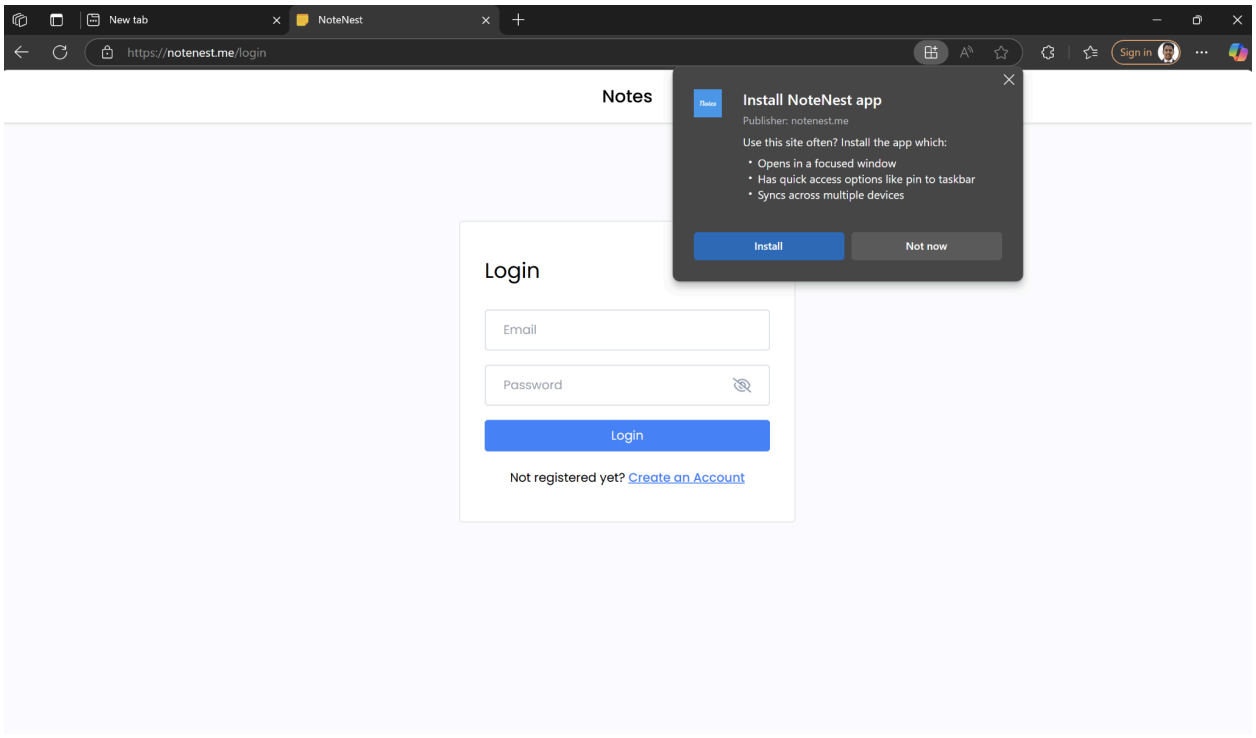
Add below code in Plugins[ ] inside vite.config.js file

```
VitePWA({
    registerType: 'autoUpdate',
    devOptions: {
      enabled: true,
    },
    manifest: {
      name: 'NoteNest',
      short_name: 'Notes',
      description: 'A Note Taking App',
      theme_color: '#ffffff',
      background_color: '#ffffff',
      display: 'standalone',
      start_url: '/',
      icons: [
        {
          src: '/android-chrome-192x192.png',
          sizes: '192x192',
          type: 'image/png',
        },
        {
          src: '/android-chrome-512x512.png',
          sizes: '512x512',
          type: 'image/png',
        },
      ],
    },
    workbox: {
      globPatterns: ['**/*.{js,css,html,png,jpg,svg}'],
    },
  }),
```

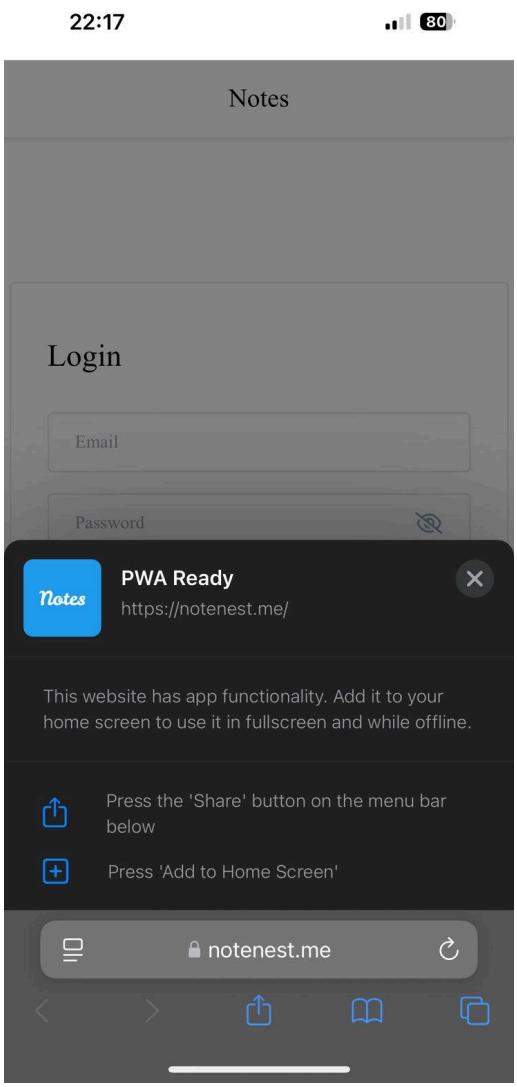
Add this is in head tag of index.html

<link rel="manifest" href="/manifest.webmanifest">

Install option on windows



Install option on IOS


22:17
80
Notes
Login
Email
Password
PWA Ready
https://notenest.me/
This website has app functionality. Add it to your home screen to use it in fullscreen and while offline.
Press the 'Share' button on the menu bar below
Press 'Add to Home Screen'
notenest.me

Name : Shreyas Naik
Roll no : 38
Div : D15B

MPL Experiment 8

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

Code :

Add this in your ***index.html*** file

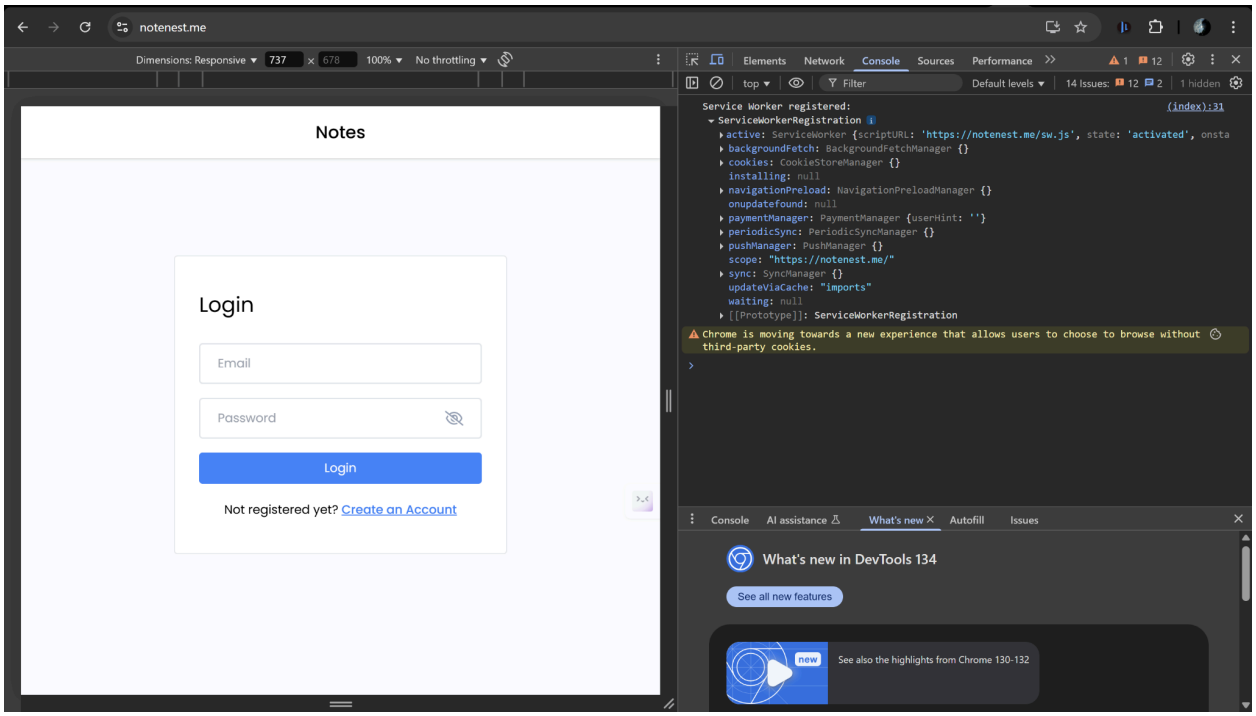
```
<!-- Register Service Worker -->
<script>
  if ('serviceWorker' in navigator) {
    navigator.serviceWorker.register('/sw.js').then((registration) => {
      console.log('Service Worker registered:', registration);
    }).catch((error) => {
      console.log('Service Worker registration failed:', error);
    });
  }
</script>
```

sw.js

```
if (!self.define) {
  let registry = {};

  // Used for `eval` and `importScripts` where we can't get script URL by other means.
  // In both cases, it's safe to use a global var because those functions are synchronous.
  let nextDefineUri;

  const singleRequire = (uri, parentUri) => {
    uri = new URL(uri + ".js", parentUri).href;
    return registry[uri] || (

      new Promise(resolve => {
        if ("document" in self) {
          const script = document.createElement("script");
          script.src = uri;
          script.onload = resolve;
          document.head.appendChild(script);
        } else {
```

```
            nextDefineUri = uri;
            importScripts(uri);
            resolve();
          }
        })

      .then(() => {
        let promise = registry[uri];
        if (!promise) {
          throw new Error(`Module ${uri} didn't register its module`);
        }
        return promise;
      })
    );
  };

  self.define = (depsNames, factory) => {
    const uri = nextDefineUri || ("document" in self ? document.currentScript.src : "") ||
location.href;
    if (registry[uri]) {
      // Module is already loading or loaded.
      return;
    }
    let exports = {};
    const require = depUri => singleRequire(depUri, uri);
    const specialDeps = {
      module: { uri },
      exports,
      require
    };
    registry[uri] = Promise.all(depsNames.map(
      depName => specialDeps[depName] || require(depName)
    )).then(deps => {
      factory(...deps);
      return exports;
    });
  };
}
define(['./workbox-54d0af47'], (function (workbox) { 'use strict';

  self.skipWaiting();
  workbox.clientsClaim();

  /**
```

```
  * The precacheAndRoute() method efficiently caches and responds to
  * requests for URLs in the manifest.
  * See https://goo.gl/S9QRab
  */
 workbox.precacheAndRoute([{
  "url": "registerSW.js",
  "revision": "3ca0b8505b4bec776b69afdba2768812"
 }, {
  "url": "index.html",
  "revision": "0.dtaucfr1o8"
 }], {});
 workbox.cleanupOutdatedCaches();
 workbox.registerRoute(new
workbox.NavigationRoute(workbox.createHandlerBoundToURL("index.html"), {
  allowlist: [/^\/$/]
 }));

}));
```



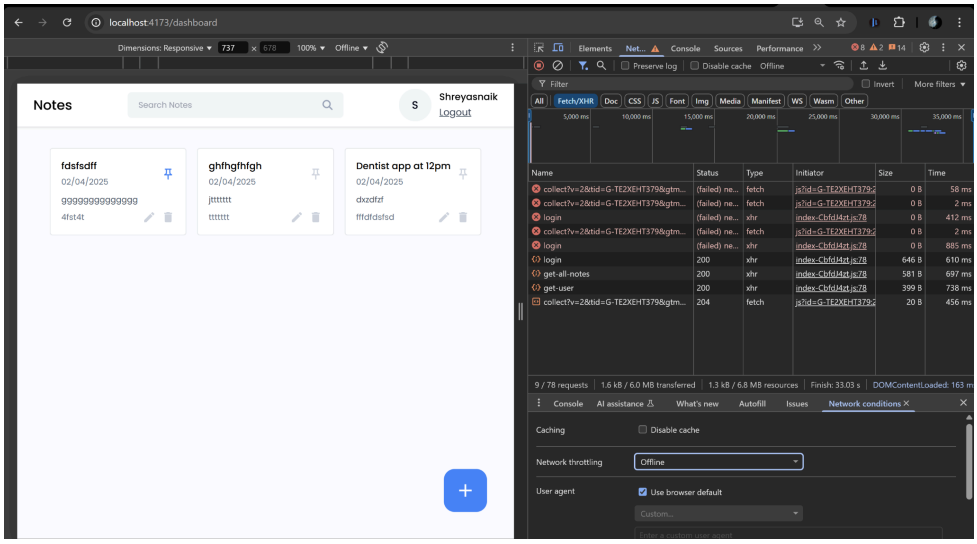
Service Workers are powerful scripts that run in the background and enable rich offline experiences, background sync, and push notifications. In this project, we implemented three core events: **fetch**, **sync**, and **push**.

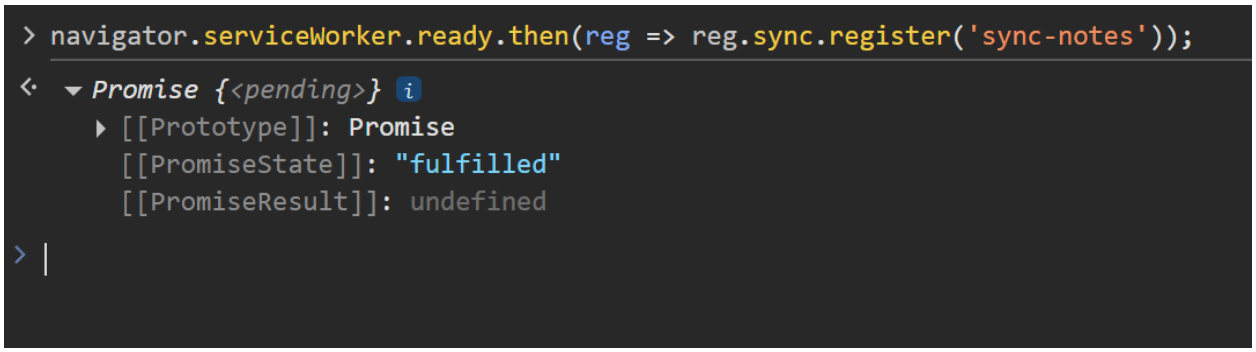
1. `fetch` Event – For Offline Caching

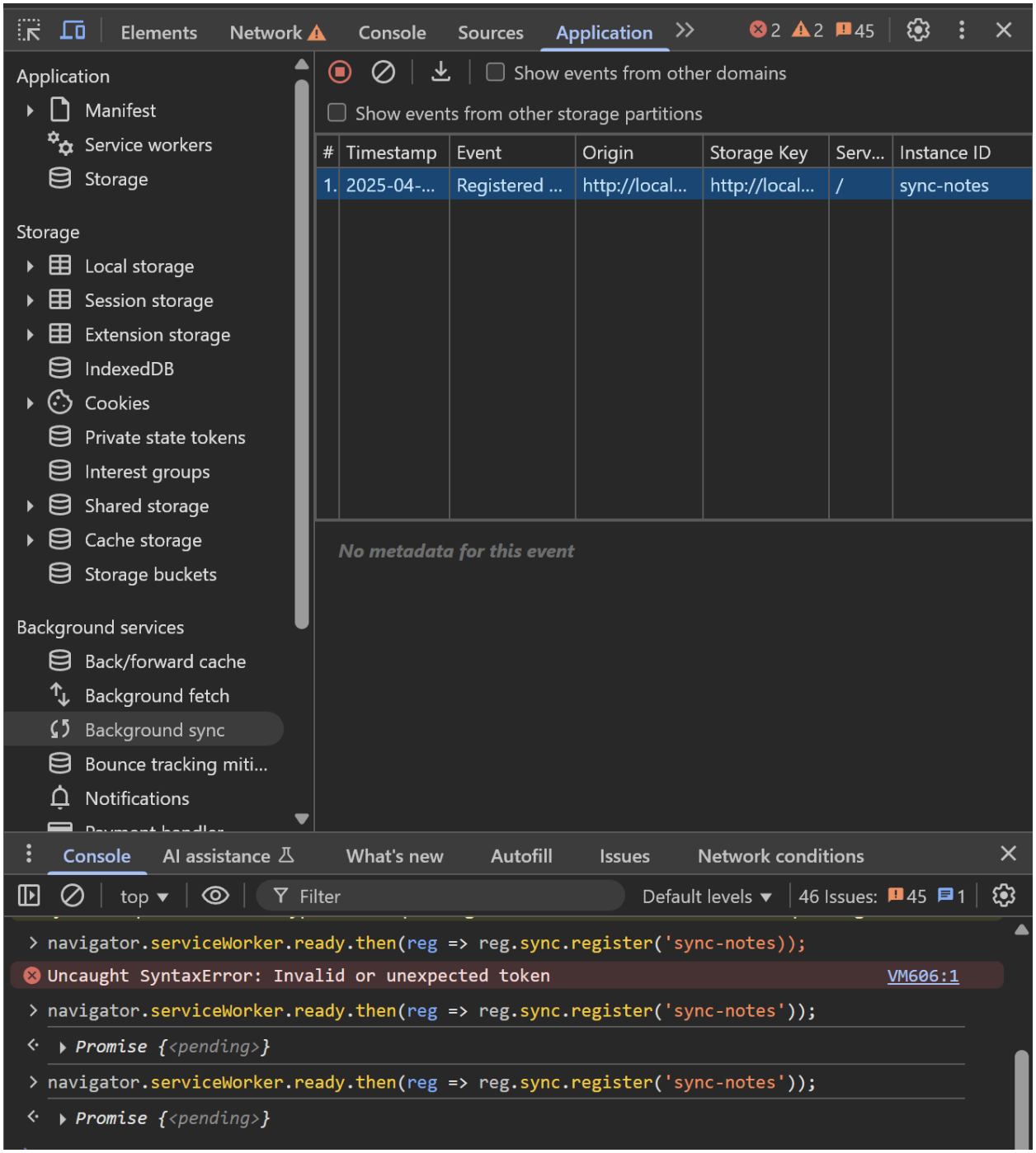
What it does:
Intercepts network requests made by the app and allows the Service Worker to serve cached content when offline.

Use case in app:
Ensures that notes and assets like images or scripts can still load when the user is offline.

```
self.addEventListener('fetch', (event) => {
  console.log('[SW] Fetching:', event.request.url);
  event.respondWith(
    caches.match(event.request).then((cachedResponse) => {
      return cachedResponse || fetch(event.request);
    })
```







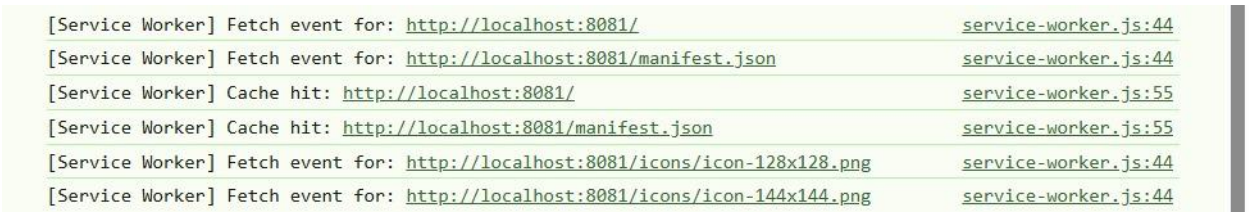

```
);
});
```

2. sync Event – Background Sync

What it does:
Runs tasks in the background (when internet reconnects), like syncing offline-created data with the server.

Use case in app:
After creating or editing a note offline, sync it with the backend when the user is online again.

```
self.addEventListener('sync', (event) => {
  if (event.tag === 'sync-notes') {
    console.log('[SW] Background sync triggered!');
    event.waitUntil(syncNotesWithServer());
  }
});

async function syncNotesWithServer() {
  console.log('[SW] Syncing notes to server...');
  // Your sync logic here
}
```



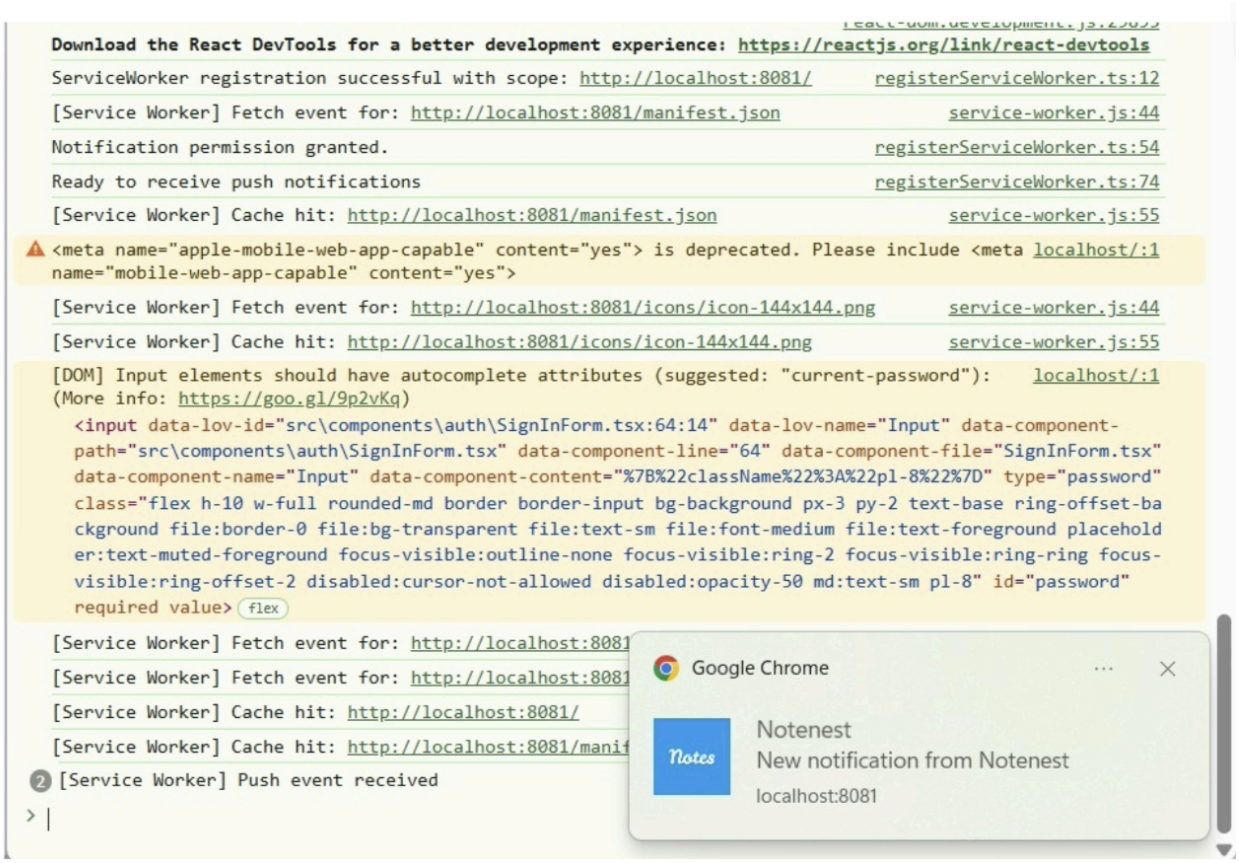
3. push Event – Push Notifications

What it does:
Handles incoming push messages from the server and shows notifications to the user.

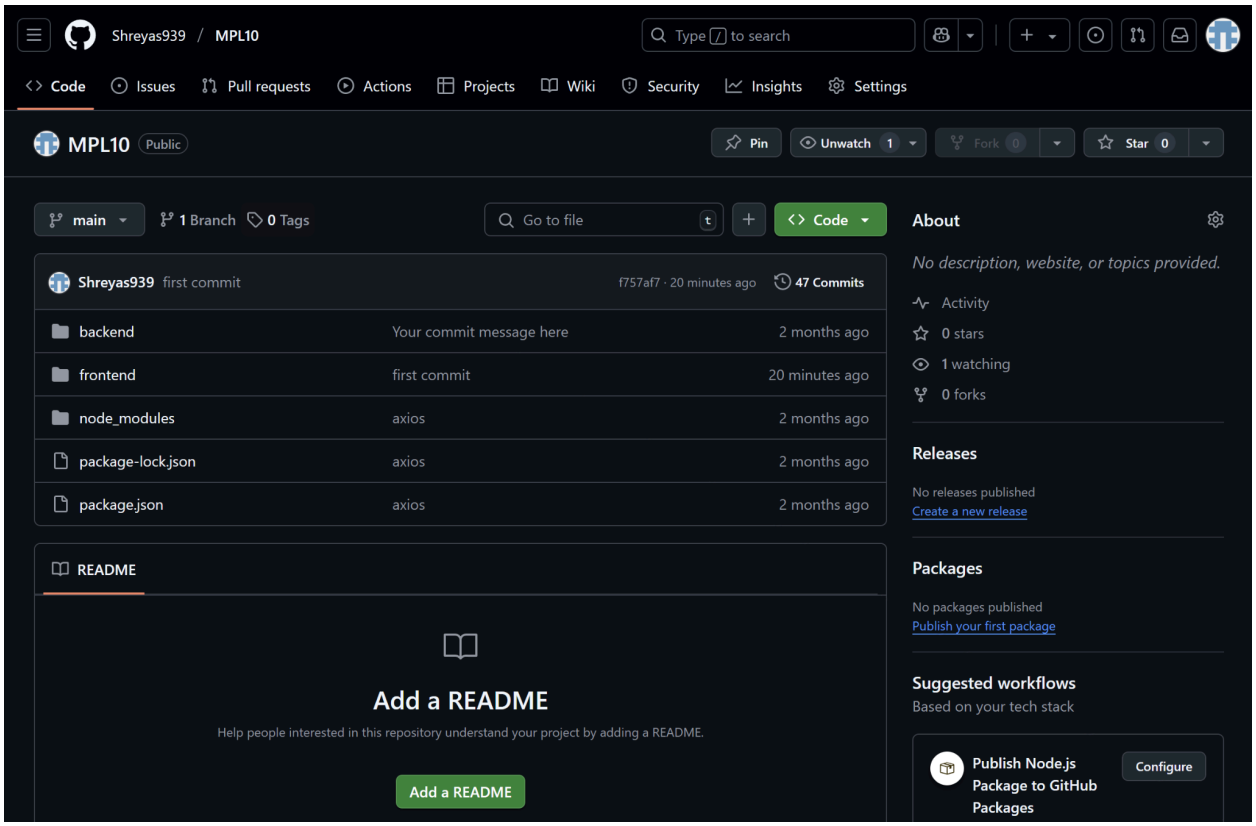
Use case in app:
Send a push notification when a new note is shared or updated.

```
self.addEventListener('push', (event) => {
```

```
  console.log('[SW] Push received:', event);

  const data = event.data?.json() || {
    title: 'NoteNest',
    message: 'You got a new note update!',
  };

  const options = {
    body: data.message,
    icon: '/android-chrome-192x192.png',
    badge: '/android-chrome-192x192.png',
  };

  event.waitUntil(
    self.registration.showNotification(data.title, options)
  );
});
```

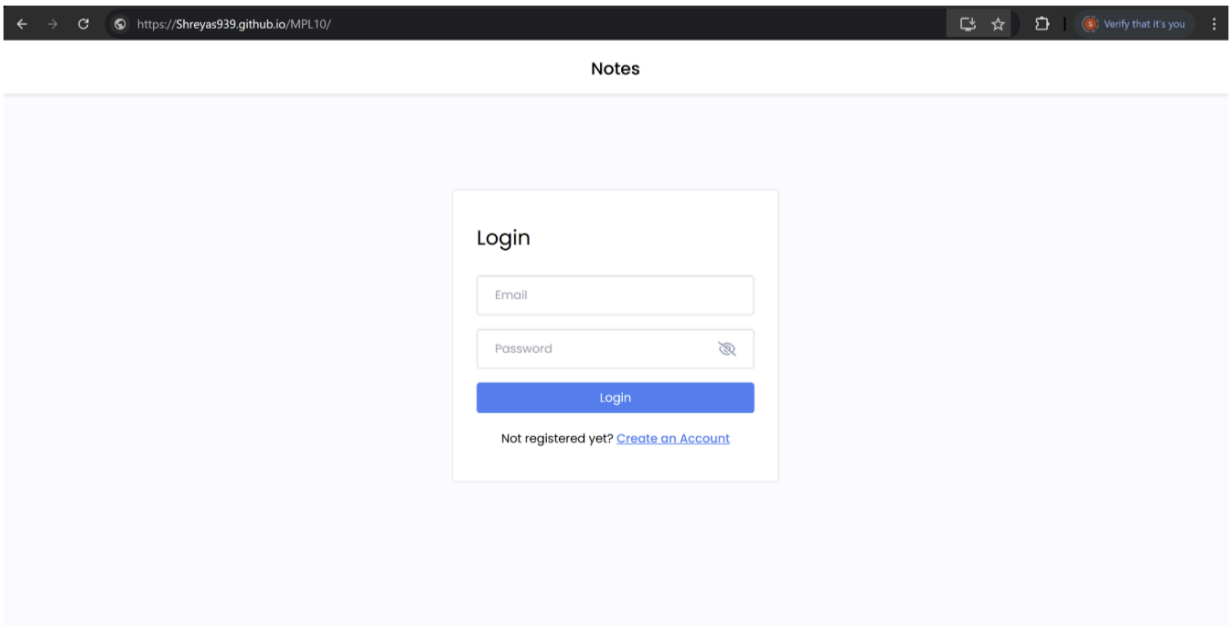


Create a new Repository and push code to that repository



- Add homepage in package.json file of your project
- Add deploy and predeploy also
- Install gh-pages using npm gh-pages -d dist command

```
"name": "frontend",
  "homepage": "https://Shreyas939.github.io/MPL10/",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "lint": "eslint .",
    "preview": "vite preview",
    "predeploy": "npm run build",
    "deploy": "gh-pages -d dist"
  },
```



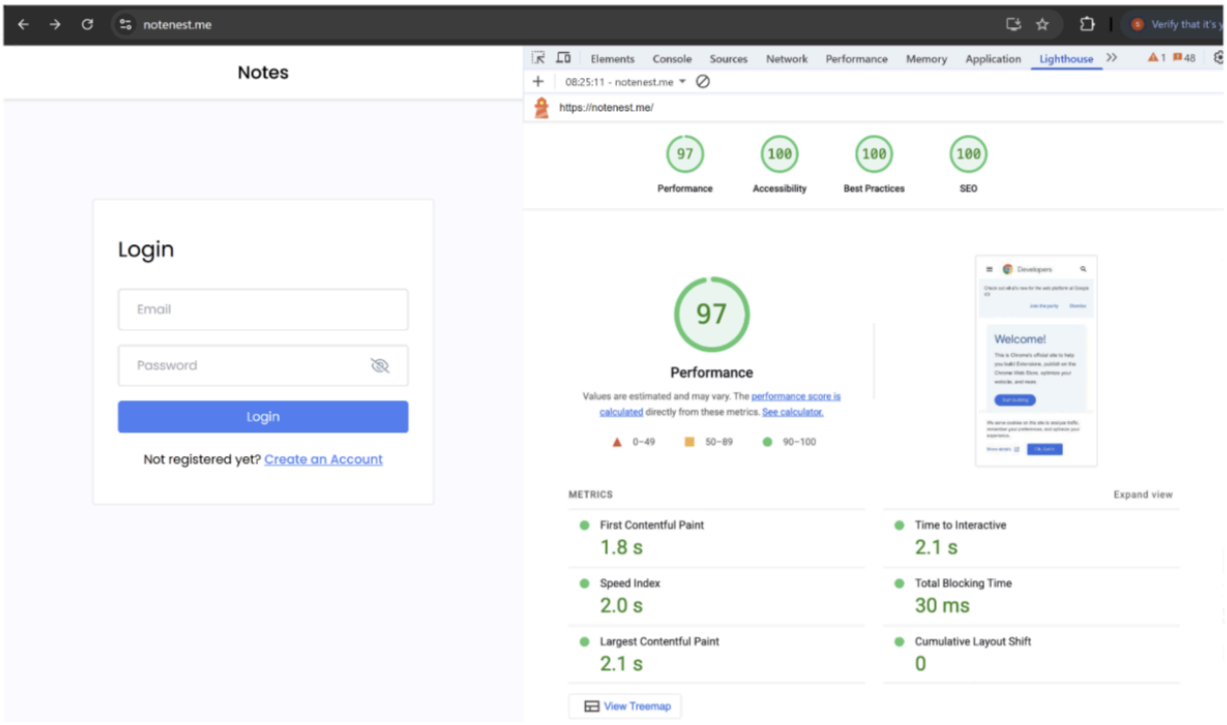
vite.config.js

```
import { defineConfig } from 'vite';
import react from '@vitejs/plugin-react';
import { VitePWA } from 'vite-plugin-pwa';

export default defineConfig({
  plugins: [
    react(),
    VitePWA({
      registerType: 'autoUpdate',
      devOptions: {
        enabled: true,
      },
      manifest: {
        name: 'NoteNest',
        short_name: 'Notes',
        description: 'A Note Taking App',
        theme_color: '#ffffff',
        background_color: '#ffffff',
        display: 'standalone',
        start_url: '/',
        icons: [
          {
            src: '/android-chrome-192x192.png',
            sizes: '192x192',
```

```
            type: 'image/png',
          },
          {
            src: '/android-chrome-512x512.png',
            sizes: '512x512',
            type: 'image/png',
          },
        ],
      },
      workbox: {
        globPatterns: ['**/*.{js,css,html,png,jpg,svg}'],
      },
    }),
  ],
  base:"/MPL10",
  server: {
    port: 3000,
  },
  build: {
    outDir: 'dist',
    rollupOptions: {
      external: [],
    },
  },
});
```

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.

- Open devtools and navigate to lighthouse and select Performance,Seo,Accessibility and Best Practices
- Click on Analyse Load



NAME: STD.: DIV.: DATE: PAGE: 09

Name: SHREYAS NAIK
Rollno: 38
PN: D15B

MPL Assignment 01

1a Explain the key features and advantages of using Flutter for mobile app development.

Key features of flutter
1) Single codebase - Write one code for both Android and IOS
2) Fast Performance - Uses dart language and high performance rendering engine.
3) Hot reload: See change instantly without restarting the app
4) Rich UI components - comes with customizable widgets for smooth UI design
5) Native like experience - Provides high quality animations and fast execution.

Advantages of using Flutter:
1) Saves time & effort - Single codebase for multiple platform
2) High speed Development - Hot reload feature speeds up coding
3) Cost effective - Reduces development cost & time
4) Attractive UI - Provides beautifull and customizable widgets
5) Good Performance - Uses dart and Skia for fast and smooth rendering.

NAME:_____ STD.:_____ DIV.:_____

DATE :
PAGE :

(b) Discuss how the Flutter framework differs from traditional approaches and why it has gained popularity in the dev community.

Soln. How flutter differs from Traditional Approaches

1) Single Codebase – Traditional methods need separate code Android and IOS, but flutter uses one code for both
2) Hot Reload : - Traditional apps requires full restart after changes, but flutter updates instately.
3) UI Rendering : Traditional apps use native components, u flutter has its own rendering engine (Skia) for faster performance.
4) Performance - Flutter compiles directly to native machine making it faster than frameworks that use a bridge
5) Customization - Traditional UI design depends on platform specific components, but flutter provides fully customiza widgets.

Why flutter is Popular Among Developers

1) Fast Development - Hot Reload and Single codebase save
2) Cross - Platform Support - Works on mobile, web and des
3) Beautiful UI - Rich, customizable widgets for modern de
4) High performance: Runs smoothly without a bridge like Native.
5) Active Community & Google Support - Regular updates and strong community help developers

NAME:_____ STD.:_____ DIV.:_____

DATE :
PAGE :

Describe the concept of the widget tree in flutter. Explain how widget composition is used to build complex user interfaces.

Concept of Widget Tree in Flutter

In flutter, everything is a widget. Widgets are arranged in a tree structure, called the widget tree. This tree represents the UI of the app, where parent widgets contain child widgets

For example: a scaffold widget can have a column widget which contains Text and Button widgets. Changes in widgets update the tree dynamically.

Widgets Composition for complex UI

Flutter uses small, reusable widgets to build complex UI Instead of creating a single large UI block, developers combine multiple small widgets like rows, column. Containers, buttons.

For example:

A list view can contain multiple Card widgets

A Column can hold Text, Images and buttons.

This modular approach makes the UI flexible, readable and easy to manage.

P.T.O

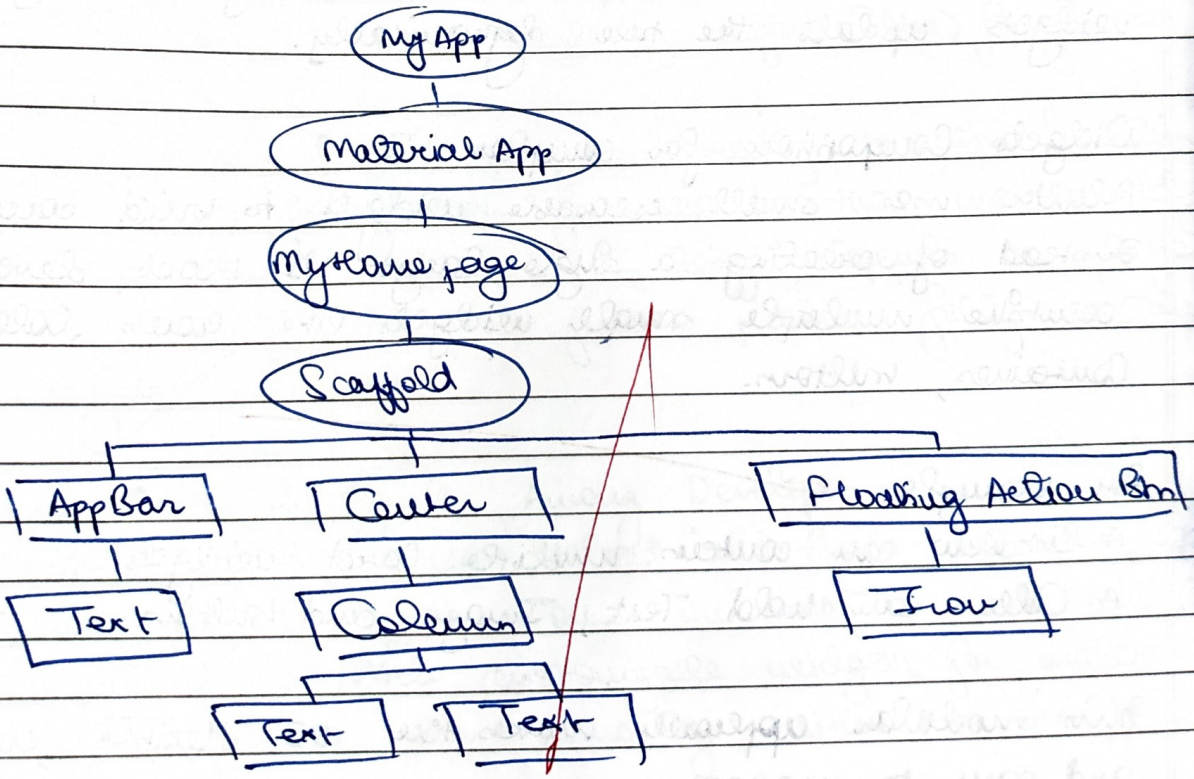
NAME:_____ STD.:_____ DIV.:_____

DATE :
PAGE :

b) Provide examples of commonly used widgets and their in creating a widgets tree.

Soln Commonly Used Widgets and Their Roles in a Widget tree

1) Scaffold: Provides the basic layout structure (Appbar,
2) AppBar: Display the top navigation bar with a title
3) Text: Displays style text on screen
4) Image: Shows images from assets or URLs
5) Container: Used for styling (Bg color, padding, margin)
6) Row: Arranges child widgets horizontally.



NAME:_____ STD.:_____ DIV.:_____

DATE :
PAGE :

Discuss the importance of state management in Flutter applications.

Importance of State Management in Flutter Application
State management is important because it controls how the app stores, updates and displays data when user interacts with it.

Why State Management is Needed?
Keeps UI Updated - Ensures that the app reflects changes (e.g. button clicks, text inputs)
Improves Performance - Updates only necessary parts of the UI instead of reloading everything.
Manages complex data: - Helps handle user inputs API data and navigation efficiently.

Types of States in Flutter:
1) local state - managed within a single widget using StatefulWidget
2) Global state - Shared across multiple screens using Provider, Riverpod, Bloc or Redux.

Compare and contrast the different state management approaches available in Flutter such as setState Provider and Riverpod. Provide scenarios where each approach is suitable

P.T.O

NAME: STD.: DIV.: DATE : PAGE :

Soln

Approach	How it Works	When to Use
setState	Updates UI by calling setState() in a StatefulWidget	Best for small apps managing state within single widget ex: To... a...
Provider	Uses InheritedWidget to share state across widgets efficiently	Suitable for medium apps where data ne... to be shared acro... multiple widgets. E... managing user authe...
Riverpod	An improved version of Provider with better performance and simpler syntax.	Best for large apps need complex state management with ... isolation ex: Handli... data and app-wid... themes.

Choosing the Right Approach

- Use setState for simple UI updates
- Use Provider for moderate state sharing across wid...
- Use Riverpod for scalable well-structured applicati...

Q.4) a) Explain the process of integrating Firebase with Flutter application. Discuss the benefits using Firebase as backend soln.

Steps of Integrating Firebase with Flutter Application

Create a firebase project - Go to [Firebase Console] (https://console.firebase.google.com/), create a new project.
Add firebase to flutter app - Register the app and download the google services.json or Google Service-Info plist
Install Firebase Packages - Add dependencies like 'firebase_core' and 'firebase_auth' in 'pubspec.yaml'
Initialize Firebase - Import Firebase in 'main.dart' and call 'Firebase.initializeApp()'.
Use Firebase Services - Implement authentication, database or cloud functions as needed.

Benefits of using Firebase as a Backend Solution
Real time database - Syncs data instantly across devices
Authentication - Provides ready to use sign-in options (Google, Email etc)
Cloud Firestore - Stores structured data efficiently
Scalability - Handles large user bases without managing servers.

Highlight the Firebase services commonly used in flutter development and provide a brief overview of how data synchronization is achieved.

Common Firebase Services Used in Flutter Development

1) Firebase Authentication - Provides user sign in method (Google, Email, Facebook, etc)
2) Cloud Firestore: A NoSQL database that stores and sy
3) Firebase Realtime Database - Stores and updates data instantly across all connected devices
4) Firebase Cloud Messaging (FCM) - Sends push notific to users
5) Firebase Hosting - Deploys web apps with fast and secure hosting.

How Data Synchronization is Achieved

1) Real-time Updates - Firestore and Realtime Database sy data across devices
2) Listeners & Stream - Widgets listen for changes a updates the UI automatically
3) Offline Support - Firebase caches data, allowing apps to work offline & sync when online.

05/05

Name: SHREYAS NAIK
Roll no: 38
Div: D15B

MPL Assignment (2)

1) Define Progressive Web App and explain its significance in modern web development. Discuss the key characteristics that differentiate PWAs from traditional mobile apps

→ A progressive web App (PWA) is a type of web application that leverages modern web technologies to provide an app like experience on web. PWAs combine the best features of traditional web apps and native mobile applications, offering improved performance offline capabilities, enhanced user engagement.

Significance in Modern Web Development

- Cross-platform compatibility: Service workers allow caching enabling PWAs to function offline or in low-network conditions
- Improved Performance: PWAs utilize caching strategies to load faster than traditional web pages.
- SEO friendly: Unlike native apps, PWAs are indexable by search engines.

Key Characteristics of PWA vs Traditional mobile Apps.

Feature	PWA	Traditional mobile App
Installation	No app store required add to home screen	Installed via App Store
Offline support	Yes, using service workers	Yes via local storage and background sync
Performance	Fast due to caching	Optimised but may use more resources
Updates	Automatic updates	manual updates via app store

Q.2) Define responsive web design and explain its importance in the context of PWAs. Compare and contrast responsive, fluid, and adaptive web design approaches.

Sol^n Responsive Web Design is a design approach that ensures a website's layout adjusts dynamically to different screen sizes and resolutions.

Importance in PWAs

- PWAs need to function seamlessly across multiple devices (mobile, tablet, desktop)
- Ensures a consistent user experience regardless of scr

sizes and resolutions
Enhances accessibility and usability
Improves SEO rankings as mobile friendly websites are favoured by search engines.

Comparison of Responsive, fluid and Adaptive Web Design

Design Type	Description	Pros
Responsive	Uses flexible grids & media queries to adjust layout dynamically	Flexible, consistent user experience
Fluid	Uses percentage-based widths to scale elements proportionally	Smooth scalability
Adaptive	Uses predefined layouts for specific screen sizes	Optimised for different devices

Describe the lifecycle of Service Workers, including registration, installation and activation phases.

Services Workers are background scripts that enable offline support, caching and background sync in PWAs their lifecycle consists of three main phases

1) Registration:
- The service worker is registered in the browser using navigator.serviceWorker.register()
- This process occurs in the main JavaScript thread of application

2) Installation:
- Run when a new service worker is detected
- Can cache assets for offline use
- If the installation fails, the service worker is discarded.

3) Activation:
- After installation, it moves to the active phase
- Can clear old caches and manage background sync.
- Becomes fully functional after activation

```
if ('serviceWorker' in navigator) {
  navigator.serviceWorker.register('/sw.js')
  .then(reg => console.log('service Worker registered
  .catch(err => console.log('serviceWorker registrationfailed', er
}
```

Q.4) Explain the use of IndexedDB in service Worker for data storage

IndexedDB is client-side NoSQL database used for storing structured data in browsers. It allows PWA to store

huge amts of data persistently

How IndexedDB works with service workers
- stores data locally, enabling offline access
- supports complex querying unlike localStorage
- Asynchronous operations prevent UI blocking
- Data persistence even when the browser is closed

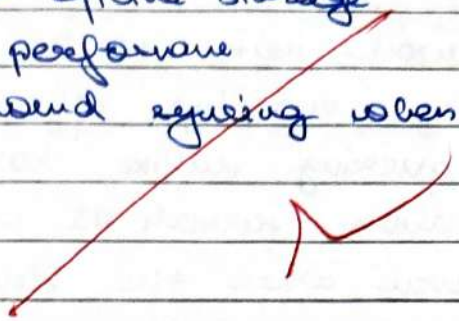
eg.

```
let db;
let request = indexedDB.open("PWA_DB", 1);

request.onupgradeneeded = function (event) {
    db = event.target.result;
    let store = db.createObjectStore("users", {keyPath: "id"});
};
request.onsuccess = function (event){
    db = event.target.result;
};
function addUser (user){
    let transaction = db.transaction(["users"], "readwrite");
    let store = transaction.objectStore("users");
    store.add(user);
}
self.addEventListener("fetch", event => {
    event.respondWith(
      caches.match(event.request).then(response => {
      return response || fetch(event.request);
    }));
});
```

Teacher's Sign.: _____

Benefits of using Indexed DB in PWAs

- enables efficient offline storage
- Improves app performance
- Allows background syncing when the connection is restored.



Teacher's Sign.: _____