

Name:Dhruv Maurya

Div:D15B

Roll No:31

Exp No.1

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>,

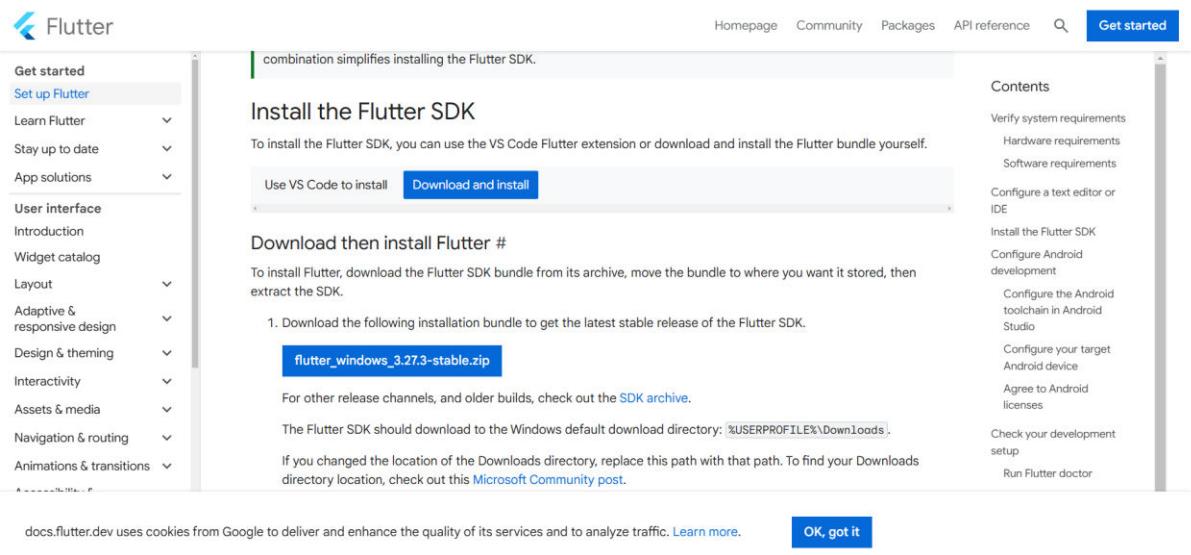
you will get the following screen.

The screenshot shows the Flutter documentation homepage at docs.flutter.dev/get-started/install?gclid=CjwKCAiAqfe8BhBwEiwAsne6gRbhEjYn0yWRQjeHiohoZDQE09cB7uohoukE4rrCgHeBDxgfRuwChoCSCIQAvD_BwE&gclsrc=aw.ds. The page title is "platform to get started". It features a sidebar with navigation links like "Stay up to date", "App solutions", and various UI design sections. The main content area has four platform icons: Windows (current device), macOS, Linux, and ChromeOS. A note about developing in China is present, and a footer indicates the documentation is the latest stable version (2025-01-31).

The screenshot shows the "Choose your first type of app" page at docs.flutter.dev/get-started/install/windows. The sidebar includes "Get started", "Set up Flutter", and other UI sections. The main content asks to choose between "Android (Recommended)", "Web", and "Desktop". A note about developing in China is shown, along with a "OK, got it" button at the bottom.

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.

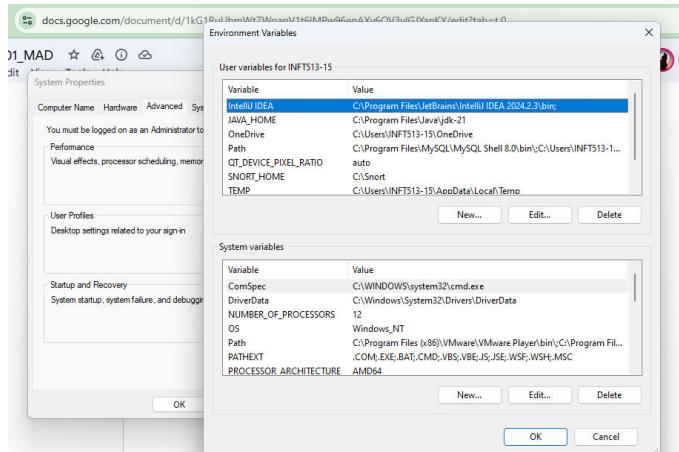


The screenshot shows the Flutter website's 'Get started' page. On the left, there's a sidebar with links like 'Get started', 'Set up Flutter', 'Learn Flutter', 'Stay up to date', 'App solutions', 'User interface', 'Introduction', 'Widget catalog', 'Layout', 'Adaptive & responsive design', 'Design & theming', 'Interactivity', 'Assets & media', 'Navigation & routing', and 'Animations & transitions'. The main content area has a heading 'Install the Flutter SDK' with a sub-section 'Download then install Flutter #'. It includes instructions for downloading the Flutter SDK bundle from its archive and moving it to the desired location. A blue button labeled 'flutter_windows_3.27.3-stable.zip' is highlighted. To the right, there's a 'Contents' sidebar with links such as 'Verify system requirements', 'Hardware requirements', 'Software requirements', 'Configure a text editor or IDE', 'Install the Flutter SDK', 'Configure Android development', 'Configure the Android toolchain in Android Studio', 'Configure your target Android device', 'Agree to Android licenses', 'Check your development setup', and 'Run Flutter doctor'. At the bottom, there's a note about cookie usage and a 'OK, got it' button.

Step 4: To run the Flutter command in regular windows console, you need to update the system

path to include the flutter bin directory. The following steps are required to do this:

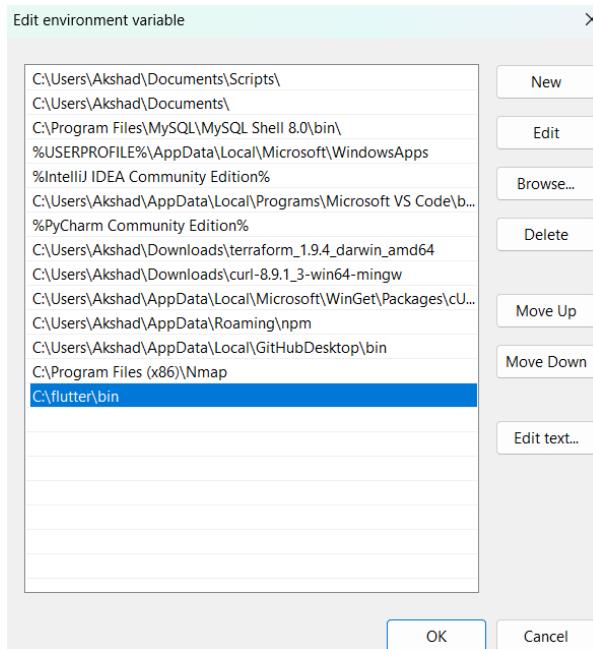
Step 4.1: Go to MyComputer properties -> advanced tab -> environment variables. You will get the following screen.



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 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.

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

C:\Users\Akshad>flutter --version
Flutter 3.27.3 • channel stable • https://github.com/flutter/flutter.git
Framework • revision c519ee916e (11 days ago) • 2025-01-21 10:32:23 -0800
Engine • revision e672b006cb
Tools • Dart 3.6.1 • DevTools 2.40.2

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

Common commands:

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

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

Usage: flutter <command> [arguments]

Global options:
  -h, --help                  Print this usage information.
  -v, --verbose                Noisy logging, including all shell commands executed.
                                If used with "--help", shows hidden options. If used with "flutter doc", shows diagnostic information. (Use "-vv" to force verbose logging in those cases.)
  -d, --device-id              Target device id or name (prefixes allowed).
  --version                   Reports the version of this tool.
  --enable-analytics          Enable telemetry reporting each time a flutter or dart command runs.
```

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 required to

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

```
C:\Users\Akshad>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.27.3, on Microsoft Windows [Version 10.0.22621.198])
[✗] Windows Version (the doctor check crashed)
  X Due to an error, the doctor check did not complete. If the error message
    https://github.com/flutter/flutter/issues.
  X ProcessException: Failed to find "powershell" in the search path.
    Command: powershell
[✗] Android toolchain - develop for Android devices
  X 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.

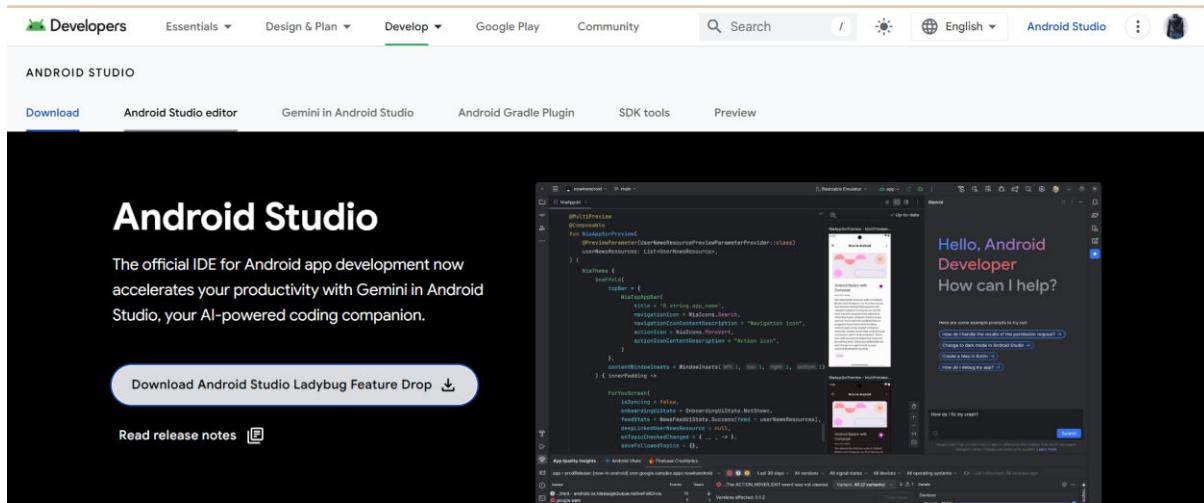
[✓] Chrome - develop for the web
[✗] Visual Studio - develop Windows apps
  X Visual Studio not installed; this is necessary to develop Windows apps.
    Download at https://visualstudio.microsoft.com/downloads/.
    Please install the "Desktop development with C++" workload, including
    Microsoft C/C++ Build Tools.
[!] Android Studio (not installed)
[✓] VS Code (version 1.96.4)
[✓] Connected device (3 available)
[✓] Network resources

! Doctor found issues in 4 categories.

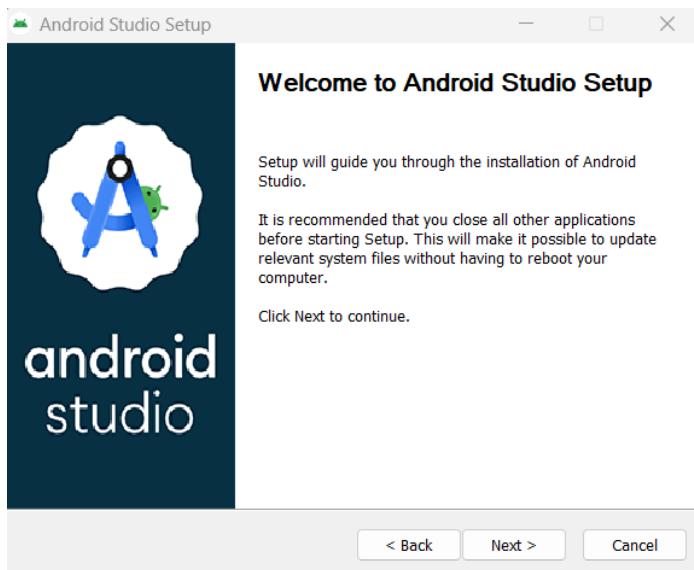
C:\Users\Akshad>
```

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.

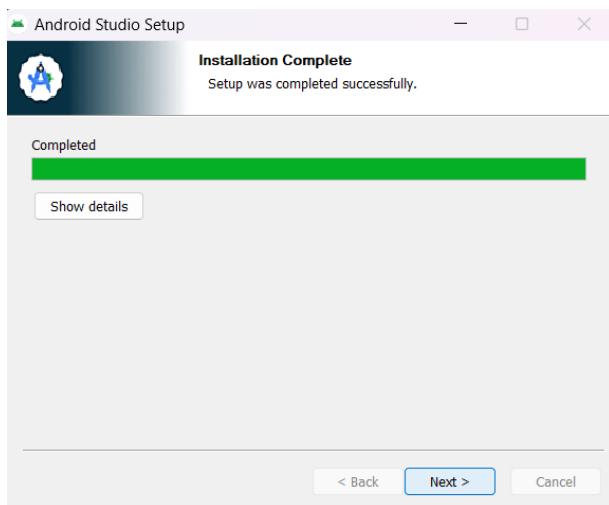
Step 7.1: Download the latest Android Studio executable or zip file from the official site.



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.

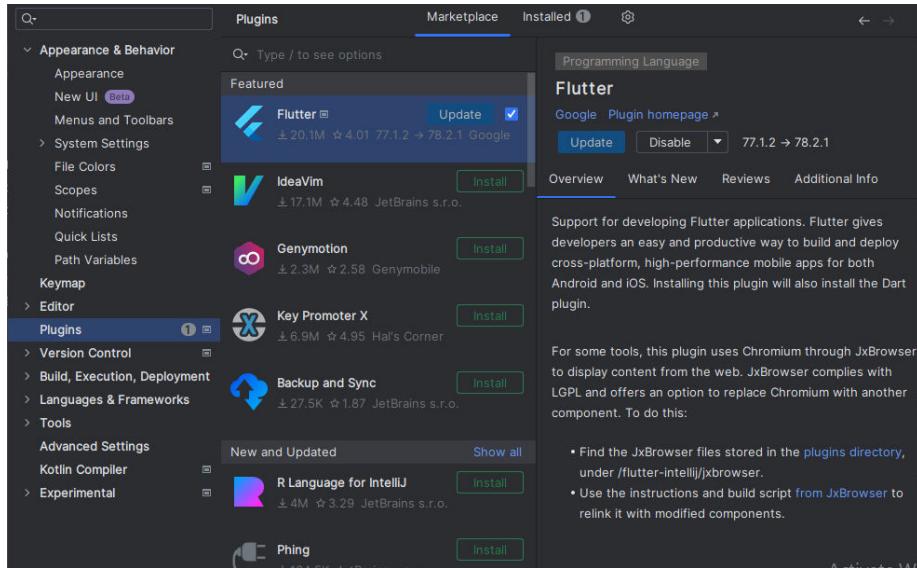
Step 8: Next, you need to set up an Android emulator. It is responsible for running and testing

the Flutter application.

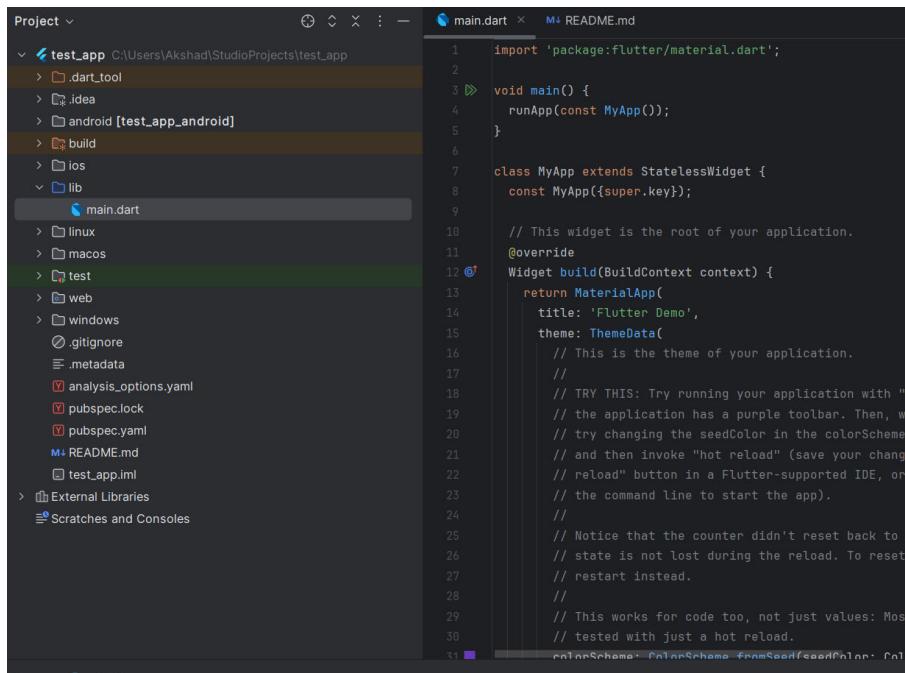
Step 8.1: To set an Android emulator, go to Android Studio > Tools > Android > AVD Manager

and select Create Virtual Device. Or, go to Help->Find Action->Type Emulator in the search

box. You will get the following screen.

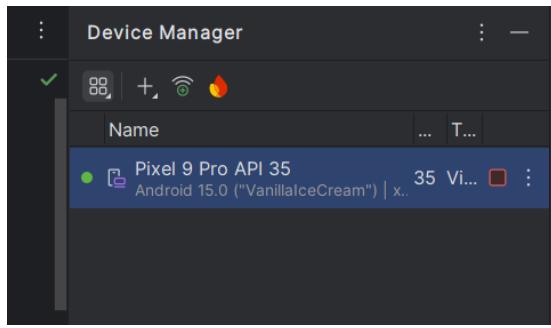


Step 8.2: Choose your device definition and click on Next.

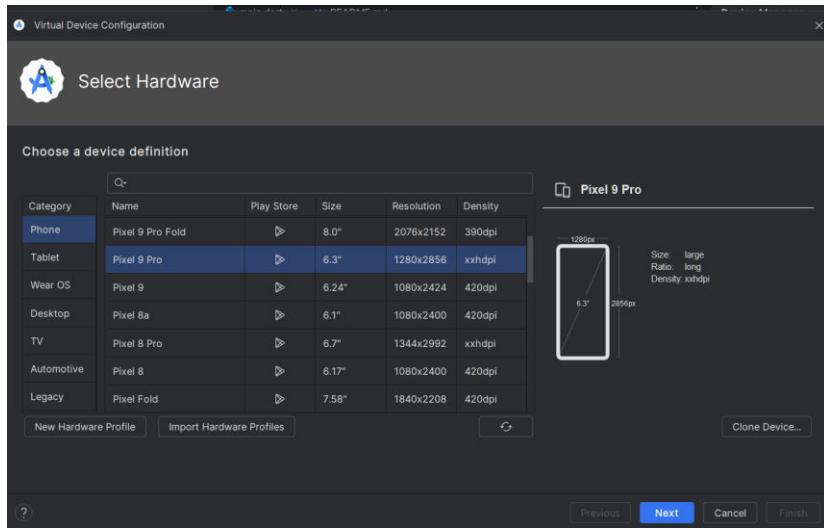


Step 8.3: Select the system image for the latest Android version and click on Next.

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

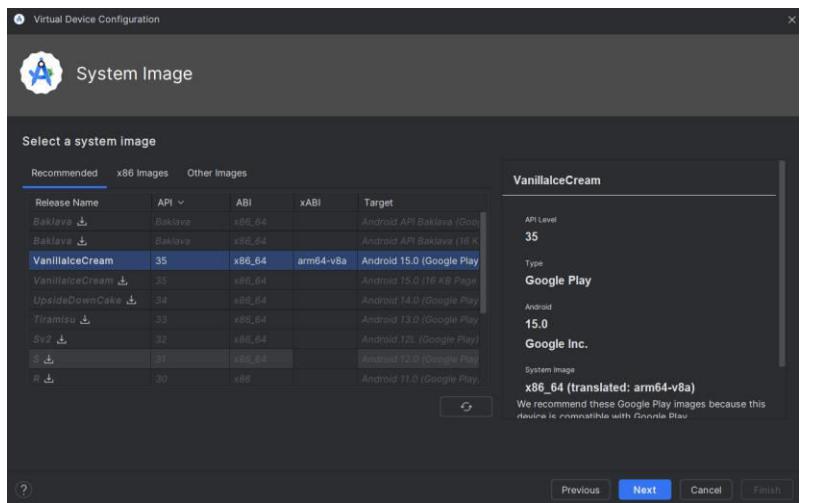


Step 9: Now, install Flutter and Dart plugin for building Flutter application in Android Studio.

These plugins provide a template to create a Flutter application, give an option to run and debug

Flutter application in the Android Studio itself. 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 Dart plugin as below screen. Click yes to proceed.

Step 9.3: Restart the Android Studio.

NAME - Dhruv Maurya

DIV - D15B

ROLL NO - 31

MPL EXP 2

```
import 'package:flutter/material.dart';
import 'auth_service.dart';
import 'home_screen.dart';
import 'register_screen.dart';
```

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

  @override
  _LoginScreenState createState() => _LoginScreenState();
}
```

```
class _LoginScreenState extends State<LoginScreen> {
  final TextEditingController emailController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();
  final AuthService _auth = AuthService();
```

```
void login() async {
  var user =
    await _auth.signIn(emailController.text, passwordController.text);
  if (user != null) {
    Navigator.pushReplacement(
      context, MaterialPageRoute(builder: (context) => HomeScreen()));
  } else {
    ScaffoldMessenger.of(context)
      .showSnackBar(SnackBar(content: Text('Login Failed')));
  }
}
```

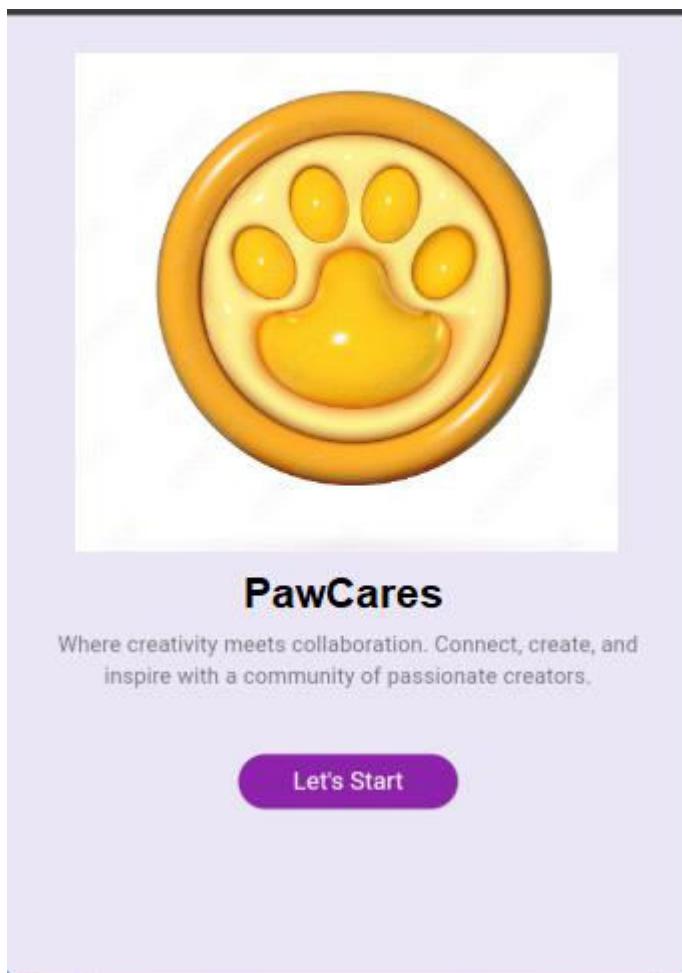
```
@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Color(0xFFFF3E5F5),
    body: Center(
      child: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Container(
          decoration: BoxDecoration(
            color: Colors.white,
            borderRadius: BorderRadius.circular(20),
            boxShadow: [
              BoxShadow(
                color: Colors.deepPurple.withOpacity(0.3),
                blurRadius: 10,
                offset: Offset(0, 5),
              ),
            ],
          ),
          padding: EdgeInsets.all(20),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.min,
            children: [
              Text(
                "Login",
                style: TextStyle(
                  fontSize: 28,
                  fontWeight: FontWeight.bold,
                  color: Colors.deepPurple,
                ),
              ),
              SizedBox(height: 20),
```

```
TextField(  
    controller: emailController,  
    decoration: InputDecoration(  
        labelText: "Email",  
        prefixIcon: Icon(Icons.email, color: Colors.deepPurple),  
        border: OutlineInputBorder(  
            borderRadius: BorderRadius.circular(10),  
        ),  
    ),  
,  
SizedBox(height: 15),  
TextField(  
    controller: passwordController,  
    decoration: InputDecoration(  
        labelText: "Password",  
        prefixIcon: Icon(Icons.lock, color: Colors.deepPurple),  
        border: OutlineInputBorder(  
            borderRadius: BorderRadius.circular(10),  
        ),  
    ),  
    obscureText: true,  
,  
SizedBox(height: 20),  
ElevatedButton(  
    onPressed: login,  
    style: ElevatedButton.styleFrom(  
        backgroundColor: Colors.deepPurple,  
        padding: EdgeInsets.symmetric(horizontal: 40, vertical: 15),  
        shape: RoundedRectangleBorder(  
            borderRadius: BorderRadius.circular(10),  
        ),  
    ),
```

```
        child: Text("Log In",
            style: TextStyle(fontSize: 16, color: Colors.white)),
        ),
        SizedBox(height: 10),
        Text("Or continue with"),
        SizedBox(height: 10),
        Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
                IconButton(
                    icon: Icon(Icons.g_mobiledata,
                        size: 30, color: Colors.deepPurple),
                    onPressed: () {}),
                ),
                IconButton(
                    icon: Icon(Icons.facebook,
                        size: 30, color: Colors.deepPurple),
                    onPressed: () {}),
                ),
            ],
        ),
        SizedBox(height: 10),
        TextButton(
            onPressed: () => Navigator.push(
                context,
                MaterialPageRoute(builder: (context) => RegisterScreen())),
        ),
        child: Text(
            "Don't have an account? Create now",
            style: TextStyle(color: Colors.deepPurple),
        ),
    ),
```

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

}
```



Login

 Email

 Password

Log In

Or continue with



Don't have an account? [Create now](#)

11:06

19.0 KB/S

PawCare

Find Your New Best Friend

7 pets available for adoption

Search pets by name, breed...

All Dogs Cats Others

Rocky ♂
Rabbit
🎂 1 year

Daisy ♀
Poodle
🎂 1 year

Luna ♀
Cat
🎂 1 year

Charlie ♂
Dog
🎂 1 year

My Pets

Shop

Cart

Profile

11:06

29.0 KB/S



Rocky

♂ Male

Rabbit

1 years

4-8 kg

Rabbit

San Francisco

Ready for adoption

About Rocky

Rocky is a 1 year old Rabbit other looking for a loving forever home. He is friendly, playful, and gets along well with other cats. He enjoys chasing toys and cuddling. Rocky would make a perfect companion for a calm household.

Characteristics

Energy



Friendliness

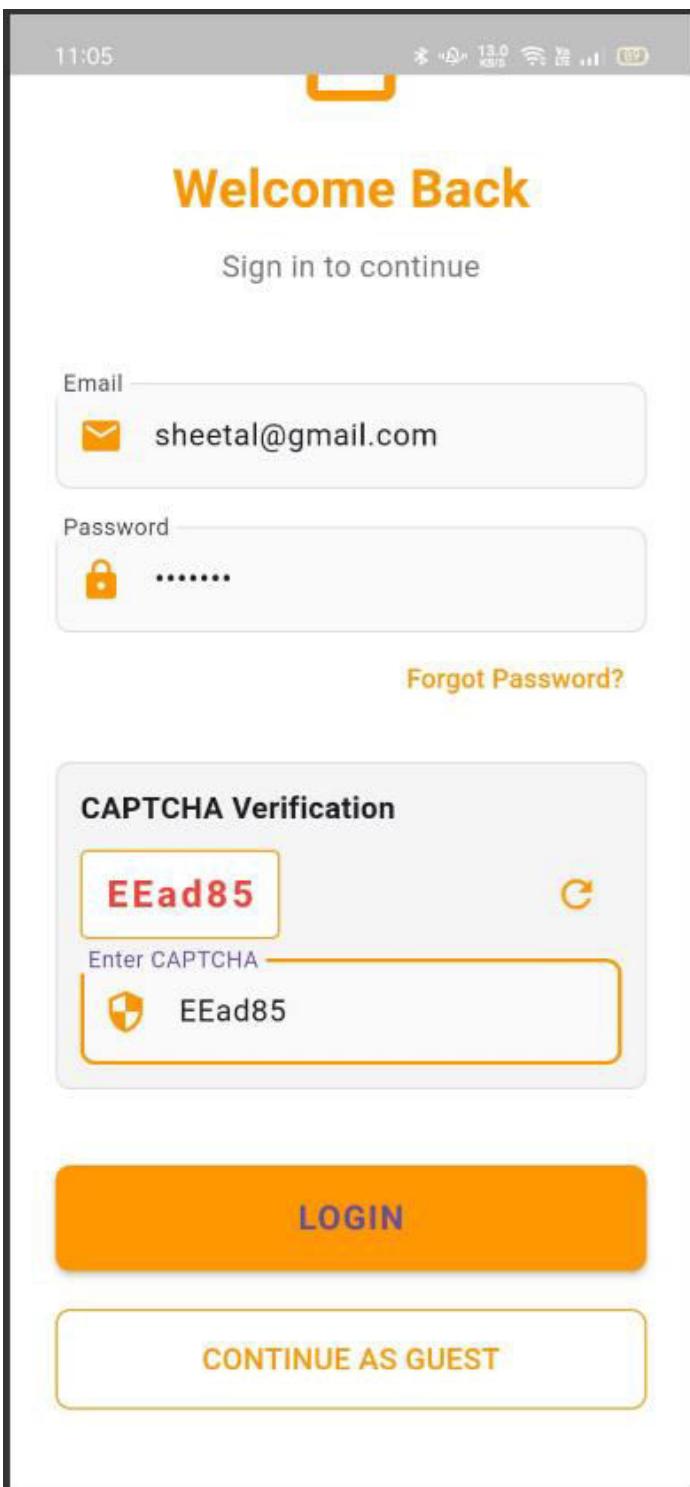


Trainability



Grooming Needs

[Adopt Rocky](#)[Health & Care](#)



This screenshot shows the Firebase Authentication console on a mobile device. The top status bar indicates the time is 07:30 and battery level is 86%. The main interface shows the "Authentication" tab selected, with "Users", "Sign-in method", and "Templates" tabs available. A prominent warning message in a box states: "The following authentication features will stop working when Firebase Dynamic Links shuts down on 25 August 2025: email link authentication for mobile apps, as well as Cordova OAuth support for web apps." Below the message is a table listing two users: "sheetal@gmail.com" and "akshad@gmail.com". The bottom of the screen shows standard iOS navigation icons.



Pet Services

Grooming

Vet Checkup

Adoption

Book an Appointment

Your Name

Pet Type

Book Appointment

Home

Store

Profile



\$800

\$600



German Shepherd

\$750



Siberian Husky

\$900



British Shorthair

\$650



Labrador Retriever

\$850



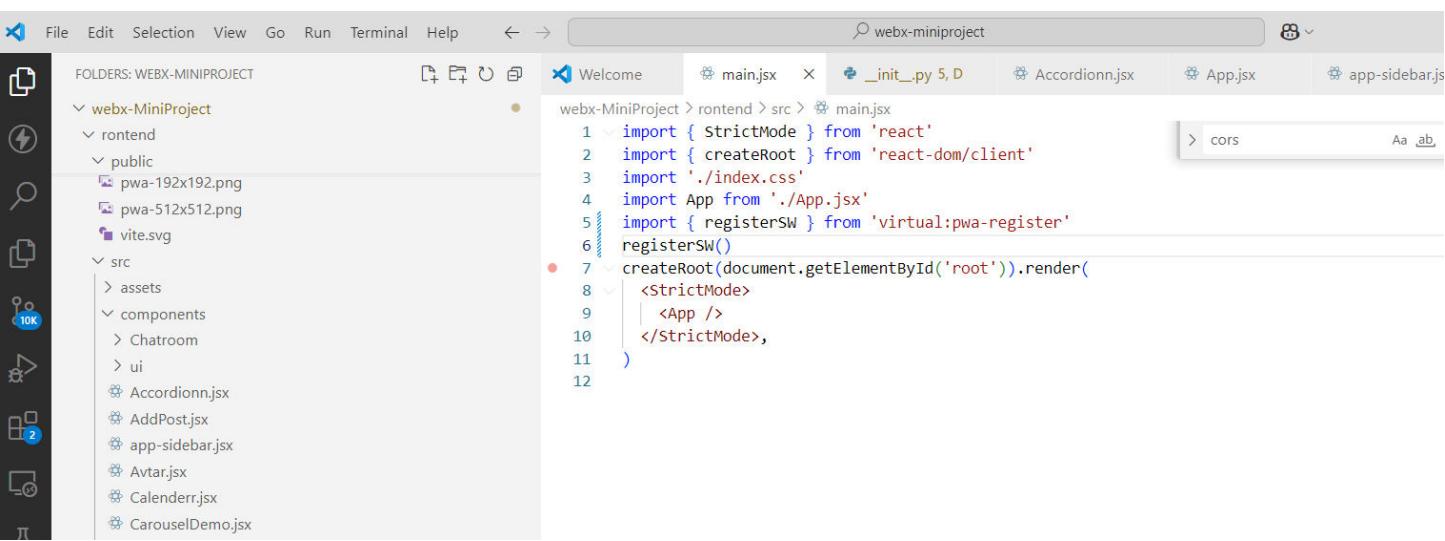
The image shows a mobile application interface. On the left, a login screen is displayed with fields for Email (dhiruv@gmail.com) and Password, followed by a large orange "Login" button. Below the login form are links for "Don't have an account? Sign up" and "Continue as Guest". On the right, a separate window titled "Authentication" is open, showing a list of users. The table has columns for Identifier, Provider, Created, Signed In, and User ID. Two entries are listed:

Identifier	Provider	Created	Signed In	User ID
dhiruv@gmail.com	Email	5 Mar 2023	5 Mar 2023	ucHDLar4HTBpMfbcJwzX...
@rougj@gmail.com	Email	6 May 2023	5 Mar 2023	sp7WdnWmpfhdgtgxkjgJ...

A message at the top of the "Authentication" screen states: "The following authentication features will stop working when Firebase Dynamic Links shuts down on 25 August 2023: email link authentication for mobile apps, as well as Cordova Auth support for web apps."

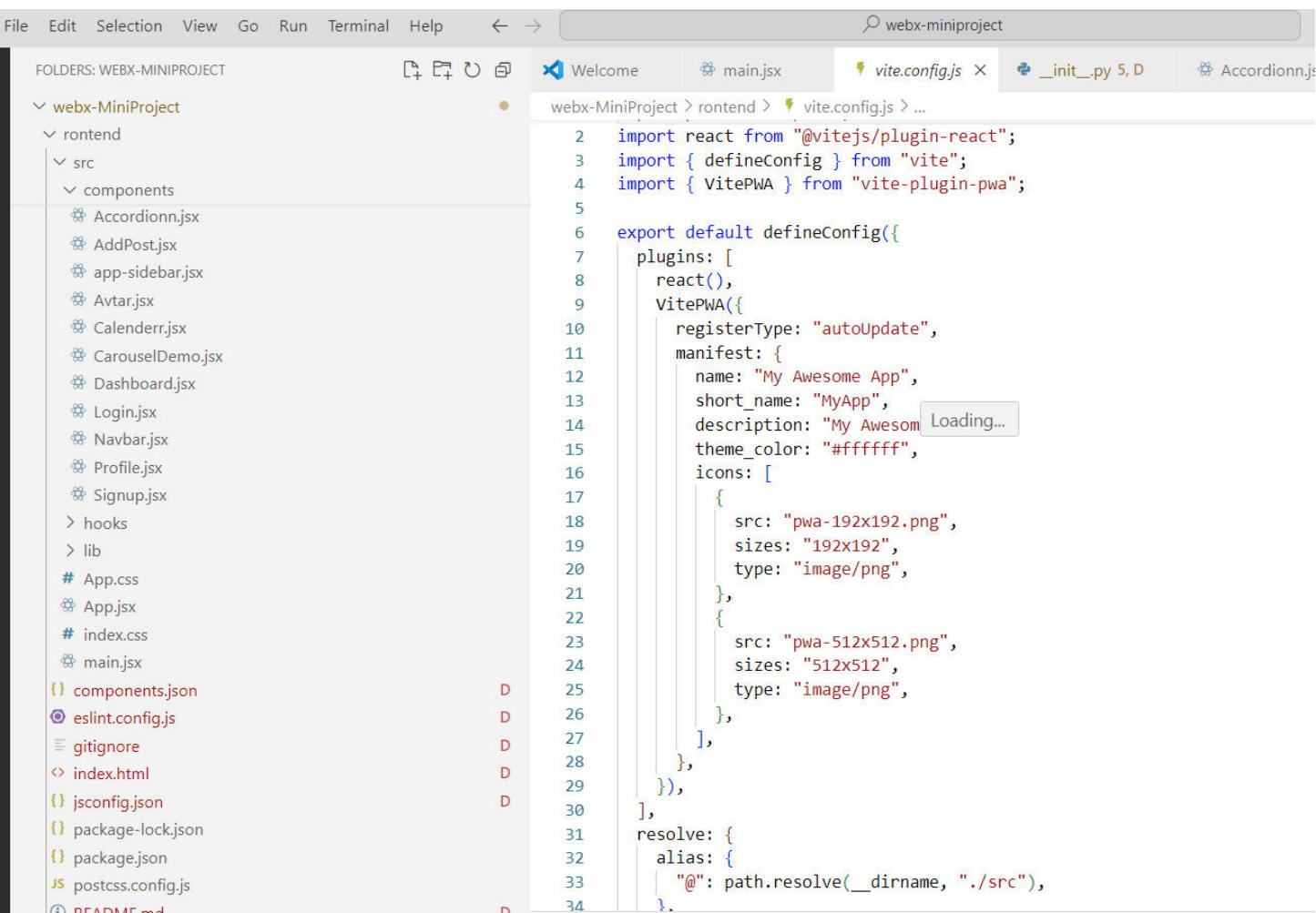
Name:Dhruv Maurya
Div:D15B
Roll No:31

PWA Practical 7



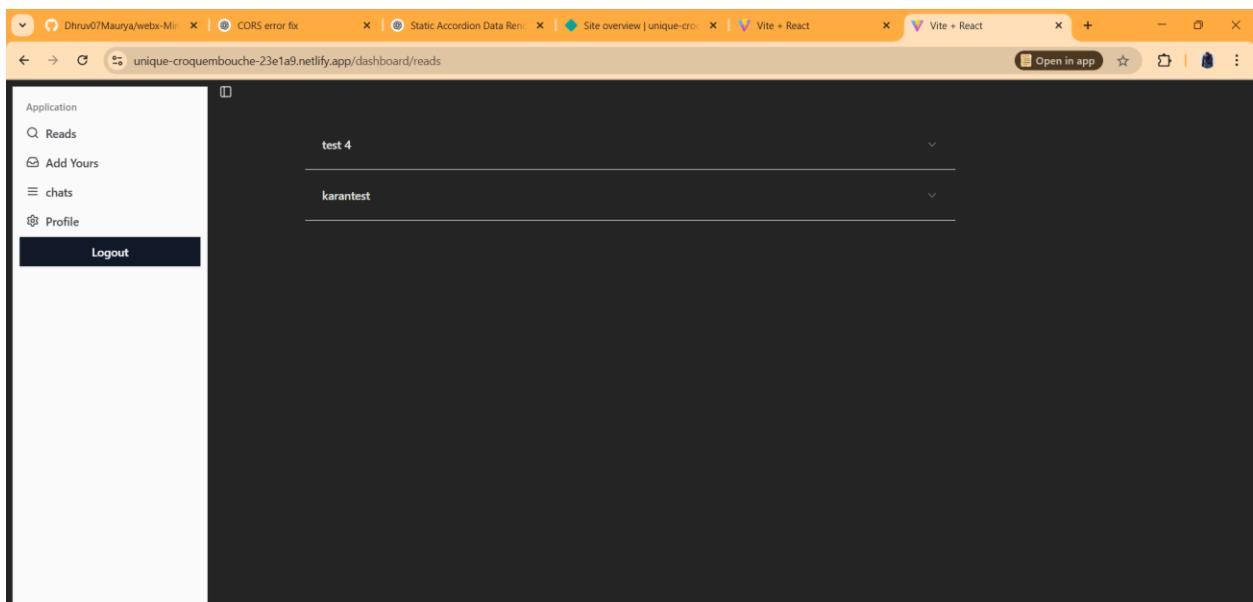
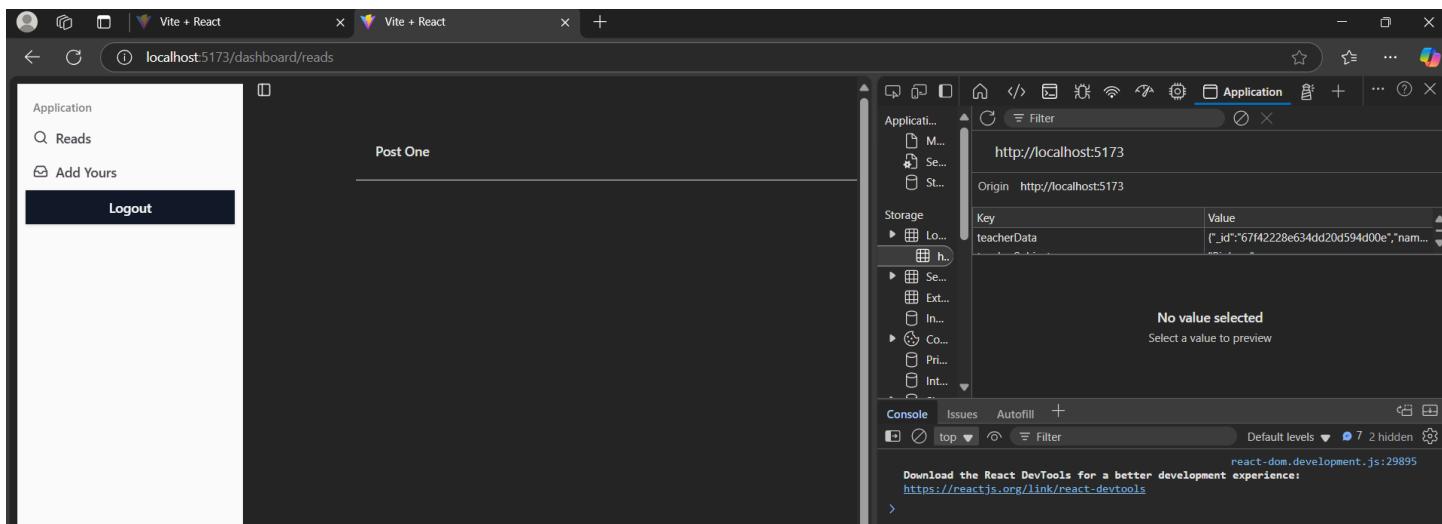
FOLDERS: WEBX-MINIPROJECT

```
1 import { StrictMode } from 'react'
2 import { createRoot } from 'react-dom/client'
3 import './index.css'
4 import App from './App.jsx'
5 import { registerSW } from 'virtual:pwa-register'
6 registerSW()
7 createRoot(document.getElementById('root')).render(
8   <StrictMode>
9     | <App />
10    </StrictMode>,
11  )
12
```



FOLDERS: WEBX-MINIPROJECT

```
2 import react from "@vitejs/plugin-react";
3 import { defineConfig } from "vite";
4 import { VitePWA } from "vite-plugin-pwa";
5
6 export default defineConfig({
7   plugins: [
8     react(),
9     VitePWA({
10       registerType: "autoUpdate",
11       manifest: {
12         name: "My Awesome App",
13         short_name: "MyApp",
14         description: "My Awesom Loading...",
15         theme_color: "#ffffff",
16         icons: [
17           {
18             src: "pwa-192x192.png",
19             sizes: "192x192",
20             type: "image/png",
21           },
22           {
23             src: "pwa-512x512.png",
24             sizes: "512x512",
25             type: "image/png",
26           },
27         ],
28       },
29     }),
30   ],
31   resolve: {
32     alias: {
33       "$": path.resolve(__dirname, "./src"),
34     },
35   },
36 });
37
```



Name:dhruv maurya

Div:D15B

Roll No.:31

MPL Experiment No.8

The screenshot shows the Chrome DevTools Application tab for a PWA. The left sidebar lists various storage and background services. The main panel displays the Service workers section for the URL <https://unique-croquembouche-23e1a9.netlify.app>. It shows a service worker named `sw.js` was received on 09/04/2025 at 10:16:20, is activated and running (#1739), and has clients at the same URL. Buttons for Test push message from DevTools, Push, Sync, and Periodic sync are present. An Update Cycle table shows three entries: Install, Wait, and Activate. The Activate entry is highlighted with a yellow bar. Below this, a section for Service workers from other origins is shown with a See all registrations link.

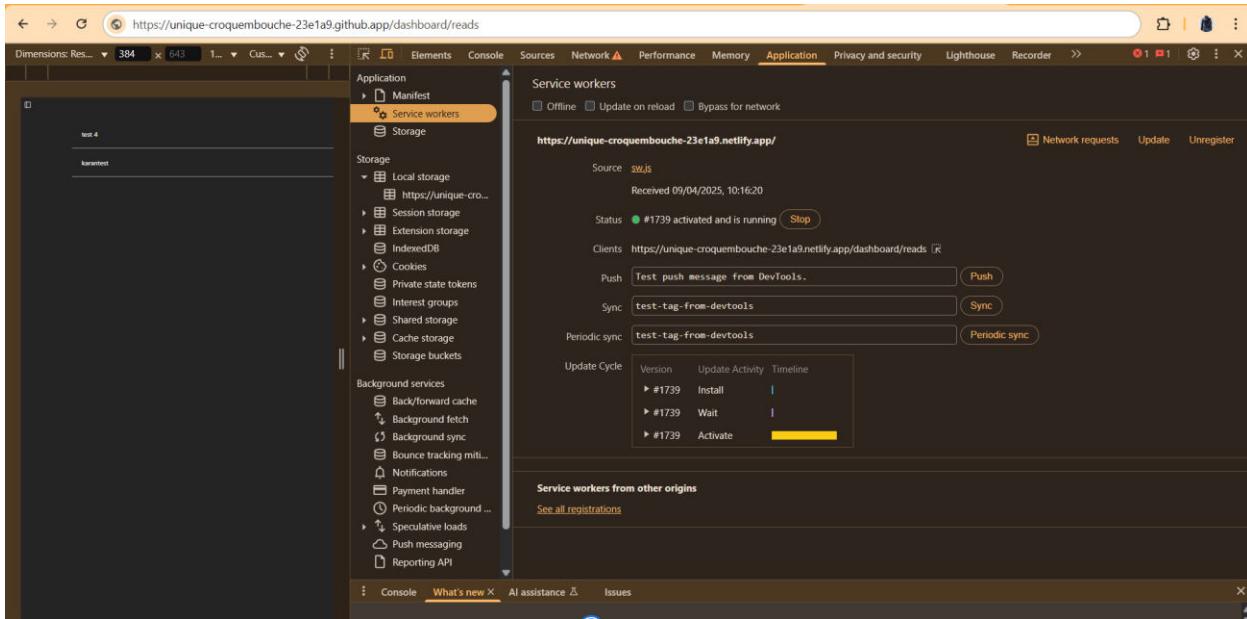
The screenshot shows the Chrome DevTools Application tab for the URL <https://unique-croquembouche-23e1a9.github.app/dashboard/reads>. The left sidebar is identical to the first screenshot. The main panel shows the App Manifest configuration. The manifest file is named `manifest.webmanifest`. It contains errors and warnings about screenshots and icon sizes. The Identity section shows the app name as "My Awesome App", short name as "MyApp", and description as "My Awesome App". The computed App ID is `https://unique-croquembouche-23e1a9.netlify.app/`. A note states that the ID is not specified in the manifest, so `start_url` is used instead. The Presentation section includes fields for Start URL, Theme color (#ffffff), Background color (#ffffff), and Orientation.

Name:Dhruv Maurya

Div:D15B

Roll No.:31

MPL Experiment No.9



Name:Dhruv Maurya

Div:D15B

Roll No.:31

MPL Experiment No.10

The image shows two screenshots. The top screenshot is a GitHub repository page for 'webx-MiniProject' owned by 'Dhruv07Maurya'. It displays a single commit from 'Dhruv07Maurya' titled 'Initial commit' at 'a1eb7e7 · 8 hours ago'. The commit message contains the text 'Initial commit'. The bottom screenshot is a browser window displaying a dark-themed dashboard. The left sidebar has links for 'Application', 'Reads', 'Add Yours', 'chats', and 'Profile'. The main area shows a list with items 'test 4' and 'karantest'.

Name:Dhruv Maurya

Div:D15B

Roll No.:31

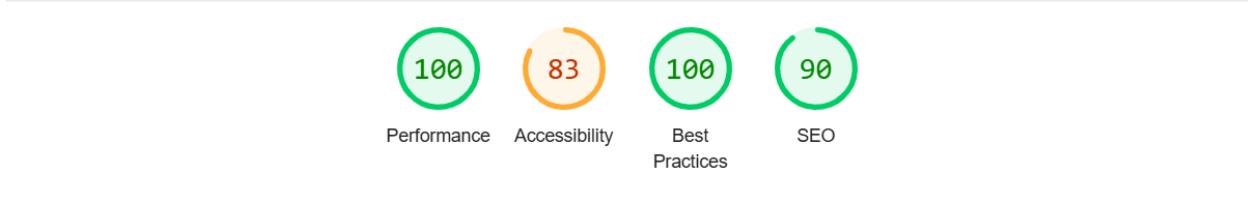
MPL Experiment No.11

The screenshot shows a code editor interface with the following details:

- Folders:** WEBX-MINIPROJECT, containing webx-MiniProject, src, and various components like Accordionn.jsx, AddPost.jsx, App-sidebar.jsx, Avatar.jsx, Calenderr.jsx, CarouselDemo.jsx, Dashboard.jsx, Login.jsx, Navbar.jsx, Profile.jsx, Signup.jsx.
- Files:** main.jsx, vite.config.js, _init_.py, Accordionn.jsx, App.jsx, app-sidebar.jsx.
- vite.config.js Content:**

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

export default defineConfig({
  plugins: [
    react(),
    VitePWA({
      registerType: "autoUpdate",
      manifest: {
        name: "My Awesome App",
        short_name: "MyApp",
        description: "My Awesome App",
        theme_color: "#ffffff",
        icons: [
          {
            src: "pwa-192x192.png",
            sizes: "192x192",
            type: "image/png",
          },
          {
            src: "pwa-512x512.png",
            sizes: "512x512",
            type: "image/png",
          },
        ],
      },
      resolve: {
        alias: {
          "@": path.resolve(__dirname, "./src"),
        },
      },
    }),
  ],
});
```
- Terminal:** Shows a log entry: 127.0.0.1 - [09/Apr/2025 10:18:21] "GET /api/posts HTTP/1.1" 200 -



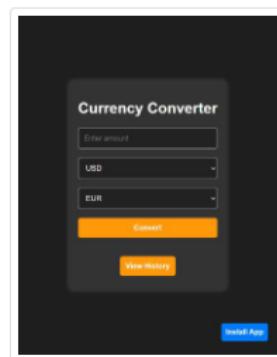
Performance

Values are estimated and may vary. The [performance score is calculated](#) directly from these metrics. See [calculator](#).

▲ 0–49

■ 50–89

● 90–100



METRICS

[Expand view](#)

- First Contentful Paint

0.3 s

- Total Blocking Time

0 ms

- Speed Index

0.3 s

- Largest Contentful Paint

0.3 s

- Cumulative Layout Shift

0

DIAGNOSTICS

- ▲ Eliminate render-blocking resources — Potential savings of 70 ms



- ▲ Page prevented back/forward cache restoration — 1 failure reason



- Enable text compression — Potential savings of 4 KiB



- Avoid chaining critical requests — 2 chains found



- Minimize third-party usage — Third-party code blocked the main thread for 0 ms



- Largest Contentful Paint element — 250 ms



More information about the performance of your application. These numbers don't [directly affect](#) the Performance score.

MAD Assignment No.1

(A)

DATE: 04/02/23

Explain key features and advantages of using flutter for mobile app development.

- ① Single code base for multiple platforms
- ② Hot Reload : Instantly see changes in the app without restarting making development faster
- ③ Fast performance : uses Dart language and a compiled approach for smooth high performance app
- ④ Open source and strong community support Backed by google and a large developer community ensuring continuous improvements & resources
- ⑤ Reduced performance issues : The app runs natively without relying on intermediate bridges like in most native reducing lag
- ⑥ Discuss how the flutter framework differs from traditional approaches & why it has gained popularity in developers' community.
Single codebase vs Separate codebase
Traditional component based code uses a single dart based code - on platform reducing development time

② Rendering Engine vs UI Components

Traditional approach: Developers rely on UI components which can lead to inconsistencies and performance issues.

Flutter: uses the Skia rendering engine to draw everything from scratch for a consistent UI across devices.

Why Flutter has gained popularity:

1. Faster development with hot reload. Can instantly see UI changes without restarting the app, making the iteration process much quicker.
2. Cross-platform inconsistency. Business scale time and resources maintaining a single codebase for multiple platforms.
3. Consistent UI across devices. Using flutter does not rely on native components; the UI looks and behaves the same across different platforms.
4. Improved performance. AOT compilation and direct access to GPU rendering ensuring smooth animations and high performance.

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

Widget tree in flutter:

In flutter, the widget tree is the fundamental structure that represents the UI of an application. It is a hierarchical arrangement of widgets where each widget defines a part of the user interface.

Flutter UI is entirely built using widgets which can be stateless or stateful.

The widget tree determines how the UI is rendered and updated when changes occur.

~~Benefits of Widgets Composition.~~

1. Reusability
2. Maintainability
3. Performance

b. Provides Example of Commonly used and their roles in creating a widget tree



1. Structural Widget:
This widget act as the foundation building the UI.
- Material App : The root widgets of an app that provides the essential Container. It provides a basic layout structure including an app bar, body floating action button etc.
- Containers : A versatile widget used for padding, margin & background customization.

Ex : Material App (

home : Scaffold (

app Bar : AppBar (Huc-Tool)

body : Container (

padding : Edge Insets (

child : Text ("Hello, flutter")

) ;



Inputs & Interaction Widgets:

Text field - Accepts text input from user

Elevation Button - A button with elevation

Gesture Detector - Detects like taps, swipes and long press

Ex: Column (

children [

Textfield (decoration : Input Deco)

Elevate Button (

onpressed: () {

print ("Button pressed") ;

child Text (Button sub) ;

) ; });

~~Display & Styling widgets~~

Text - display text on the screen

Image - Shows images from assets network or memory

Icon : Display Icons

Card - A material design card with rounded corners and elevation.

Column (

children [

Text ("Welcome");

Image.network ("url")]);

A.3)

a. Discuss the important State management in flutter applications.



In flutter State refers to data that change during the lifetime of application. This includes:

User Input

UI changes

Network changes

Animation States

These are two type of States

• Ephemeral state: Small UI specific state

does not affect the whole App.

• App wide status: Data shared across

wedgets,

Importance of state management:

1) Efficient UI updates

2) Code maintainability.

3) Data consistency

4) Synchronization



b. Compare & contrast the different state management approaches available in flutter such as state provider & riverpod. Provide scenarios where each approach is suitable.

Set state:



Scanned with OKEN Scanner

Scenarios for Each approach

- use useState when managing simple UI elements within a single widget
- use provider when changing state across multiple widget

Explain the process of integrating firebase with flutter application discuss the benefits of using firebase as a backend solution

① Create a firebase project from console

② Register the flutter app with firebase
Enter the android package name and download service

③ Install firebase dependencies

④ configure firebase for android & iOS

⑤ Initialize firebase in flutter

Benefits of using firebase:

- (1) Easy to setup & scale
- (2) Authentication
Provides Email / password, etc.
- (3) Cloud Storage : Stores Assets on cloud
- (4) Push Notification : Sends real-time Notification

b.

Highlight the firebase services used in flutter development & provide overview of how data synchronization is achieved

- =>
- (1) Firebase Authentication
 - (2) Cloud Firestore
 - (3) Realtime Database
 - (4) Firebase Cloud Messaging
 - (5) Firebase Analytics
 - (6) Firebase Hosting

Data synchronization in firebase
firebase ensures real-time data synchronization across multiple devices and platforms using firebase & real-time Database

Cloud Firestore Sync Mechanism:
 uses real-time listeners to update UI ~~instantly~~
 when data changes
 Ex: Firebase Firestore instances.selection [useid].
 snapshots().listen ((snapshot) =>
 for var doc in snapshot.docs {
 print(doc['name']); } });

Real time database Sync Mechanism
 uses Persistent websockets connections for
 live updates.
 Ex Database reference ref = FirebaseDatabase.getInstance()
 ref.ref("messages").
 onValue.listen ((event) {
 print(event.snapshot.value); })

Offline Data Sync

firestore caches data locally & syncs changes
 when the device is online

Cloud functions for automated updates

Automate backend logic on triggers
 when data changes

APP Assignment - 2

(23/24)

DATE:

- Define Progressive web App (PWA) and Explain its significance in modern web development. Discuss the key characteristics that differentiate PWA from traditional mobile apps.
 - Significance of PWA in Modern web development
 - Cross platform compatibility
 - Offline support
 - Fast performance
 - No app store required
 - Low development cost
- Key difference between PWA & Traditional mobile Apps

feature	PWA	Traditional mobile app
Installation	Direct from browser	Download from app store
Internet Required	Work offline with caching	Usually requires Internet
Performance	Fast with service workers	Faster but needs installation
Updates	Automated no app store approval	Manual update needed
Development Cost	Lower	Higher

Q) Define responsive web design and explain
in the context of progressive web app.
(Extract Responsiveness, fluid and adaptive web
design approach)

- Responsive web design (RWD) is technique that
progress adapt automatically to different screen
and device
- Best Importance of responsive design is PWA
 - Better user experience. PWA work smoothly
device
 - SEO benefit. google ranks responsive sites!

Comparision of web design approaches

Approach	How it works	Pros	Cons
Responsive	Use flexible grid and CSS media queries to adjust layout.	Work on all devices	Complex
Fluid	Use percent based width works well on less code instead of fixed pixels to different screen built charts etc & resize sizes → easy to implement	Smoothly	Complex
Adaptive	Use fixed layouts that optimized for more change at specific break points known screen required size	design for same size	



Key difference

Responsive adapts step dynamically to all client

- 3) Describe the life cycle of service workers including its registration, installation and activation phase.
- 4) Lifecycle of service workers.

A service worker is a script that runs in the background and helps a web app work offline and faster and send push notifications.

1. Registration phase.

The browser registers the service worker using JavaScript code Eg:-

```
if ('serviceWorker' in navigator) {
  navigator.serviceWorker.register('sw.js')
    .then((c) => console.log('Service worker registered'))
    .catch((err) => console.log('Registration failed', err))
```

2. Installation phase

The service worker downloads necessary (HTML, CSS, JS) and stores them in cache.

Code Ex:-

```
self.addEventListener('install', event => {
  event.waitUntil(
    caches.open('app-cache').then(cache =>
```

```
      cache.add([index.html,
        'style.css']);
    )
  );
});
```

This ensures the app loads even without the internet.

3. Activation phase.

- The old service worker is replaced with the unused code files from the previous version.
- Decided final step fetch & sync.

Q.4) Explain the use of Indexed DB in the sync workers of data storage.

Use of Indexed DB in service workers for data storage. Indexed DB is a structured database to store large amounts of structured data like returning data efficiently.

- Why use Indexed DB in service workers?
 - Offline support - Store data online, offline and sync later.

Efficient storage - stored data with structure like user setting, last time user form inputs.

Opening the database.

let db = await indexedDB.open('My Database');
Request : success = function (event) {
 let res = event.target.result;

Creating a store & adding data
Request : success = function (event) {
 let db = event.target.result;
 let store = db.createObjectStore('User', {
 keyPath: 'id' })

store.add({ id: 1, name: 'John', age: 25 })

FOR EDUCATIONAL USE



Scanned with OKEN Scanner