Name:Akshad Mhaskar

Div:D15B

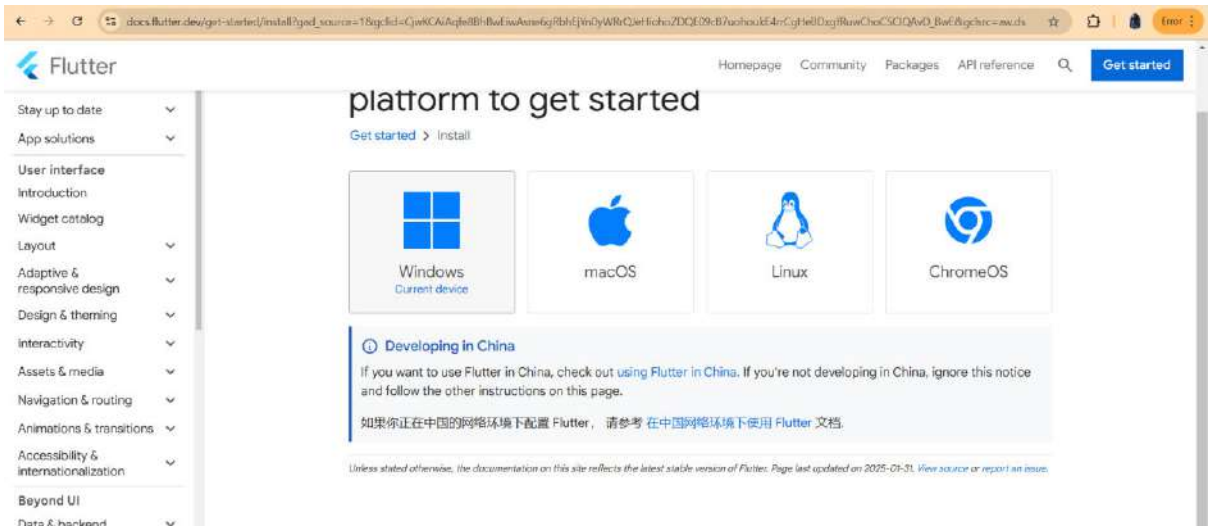
Roll No:32

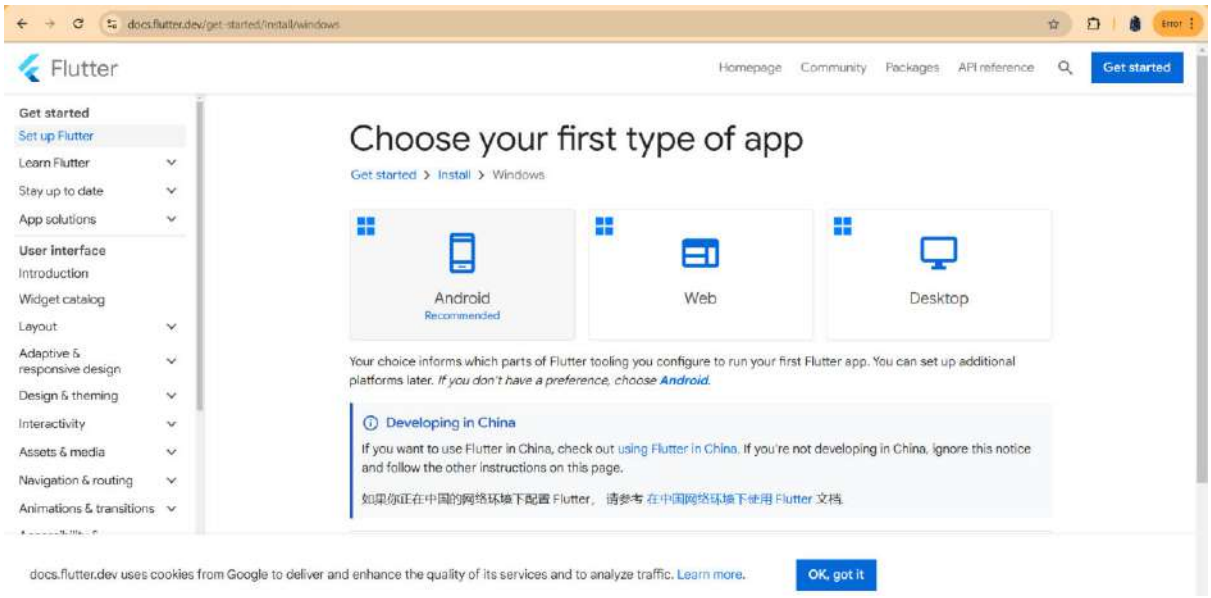
Exp No.1

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 ,

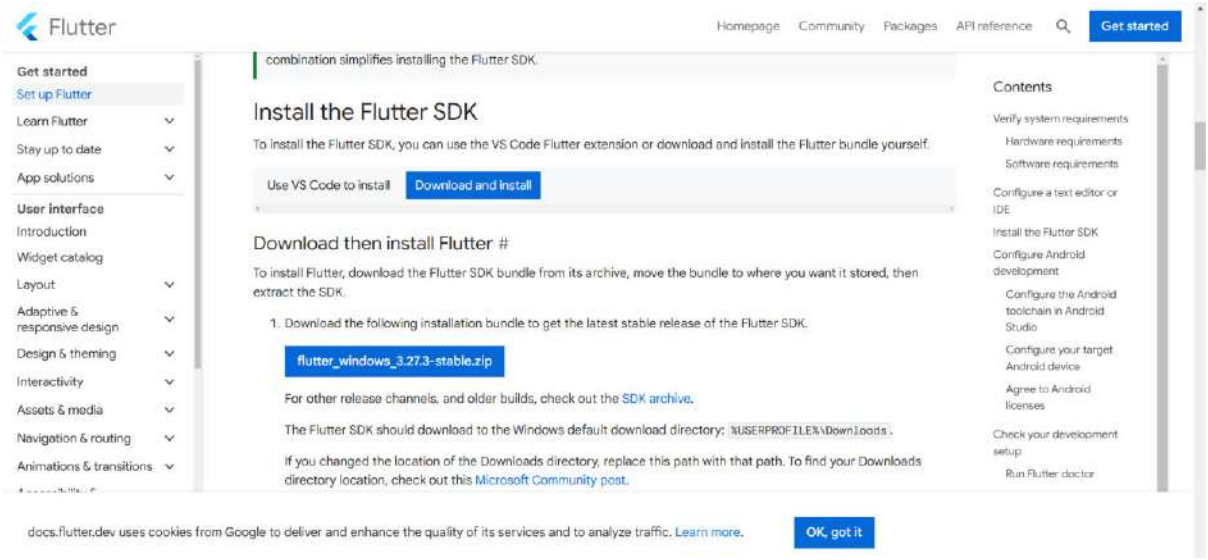
you will get the following screen.





Step 2: Next, to download the latest Flutter SDK, click on the Windows icon. Here, you will find the download link for SDK.

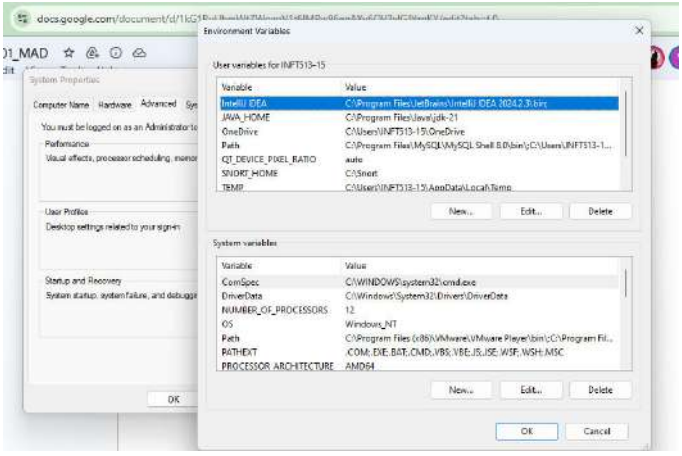
Step 3: When your download is complete, extract the zip file and place it in the desired installation folder or location, for example, C: /Flutter.



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

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

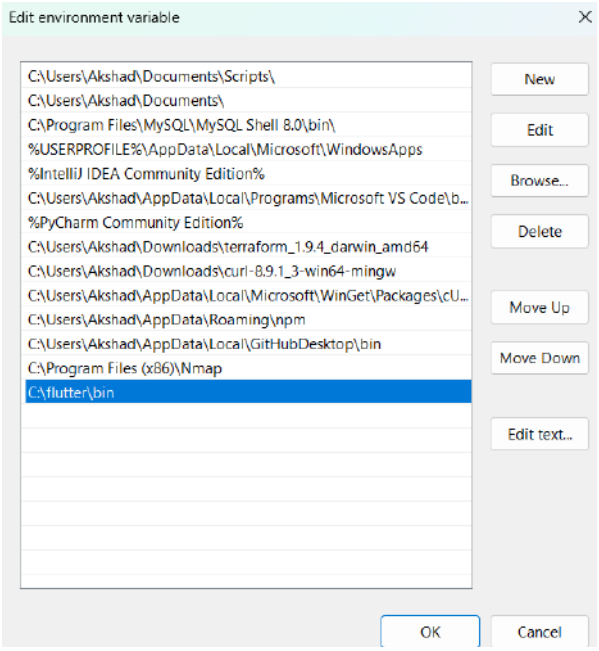
Step 4.1: Go to MyComputer properties -> advanced tab -> environment variables. You will get the following screen.



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

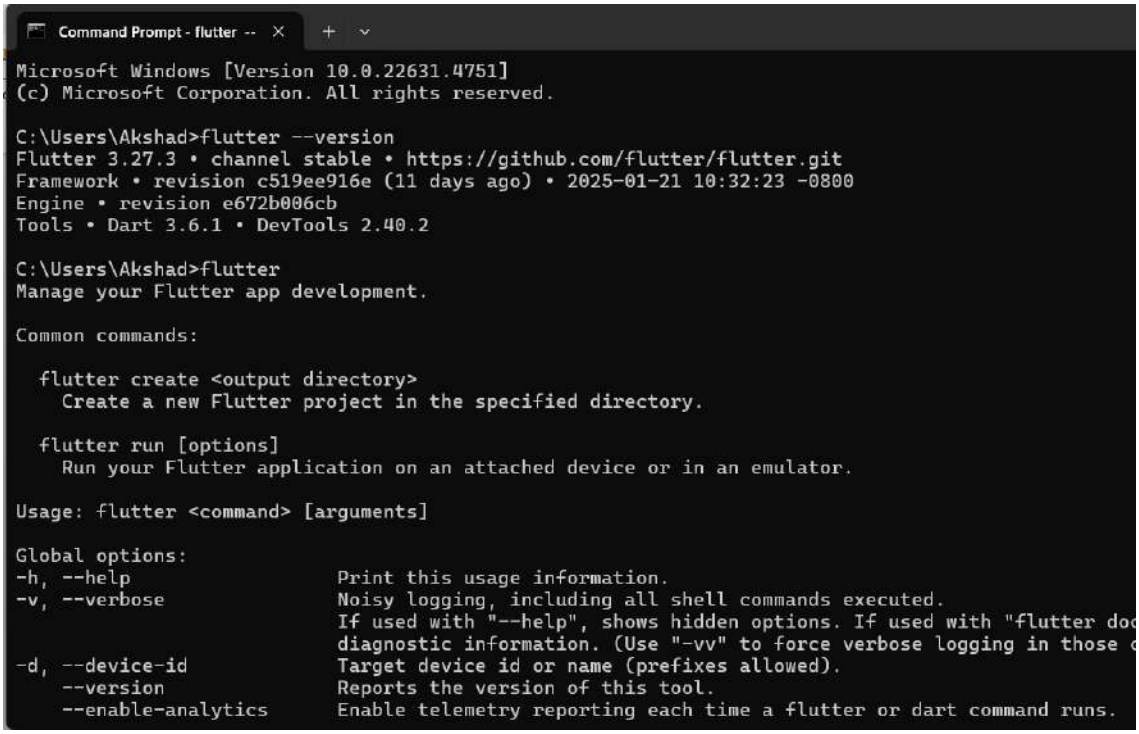
Step 4.3: In the above window, click on New->write path of Flutter bin folder in variable value -

> 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.



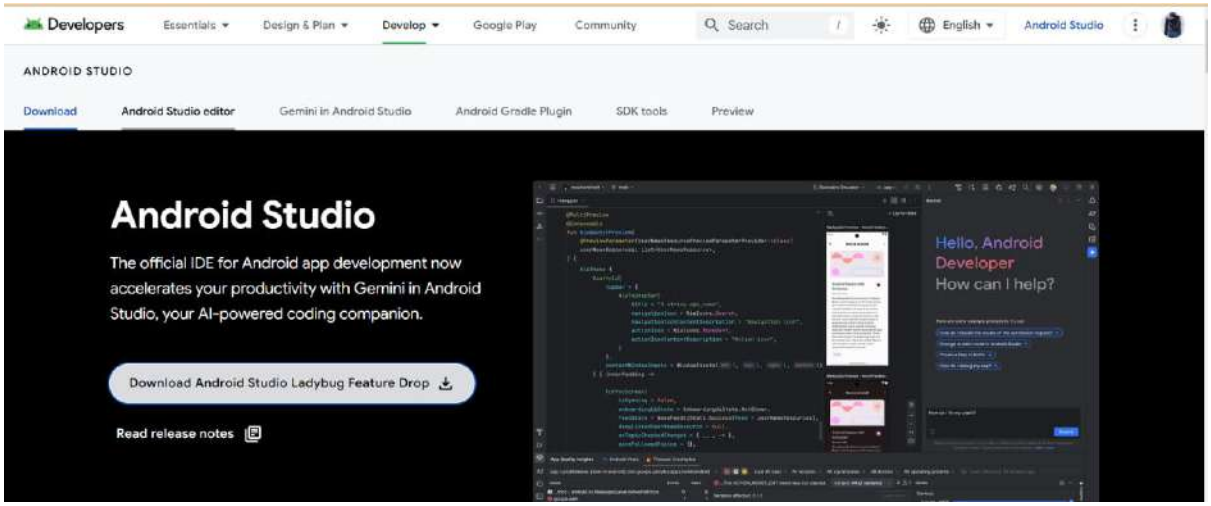
Step 6: When you run the above command, it will analyze the system and show its report, as shown in the below image. Here, you will find the details of all missing tools, which required to

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

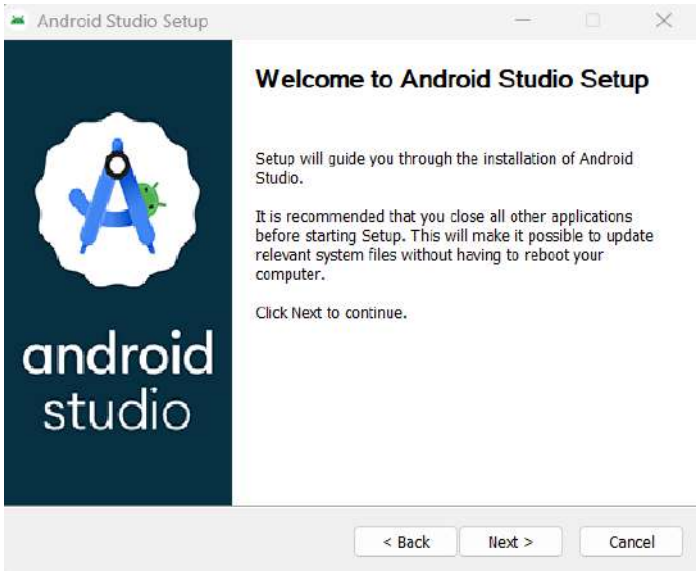
```
C:\Users\Akshad>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.27.3, on Microsoft Windows [Version 10.0.226
[X] Windows Version (the doctor check crashed)
    X Due to an error, the doctor check did not complete. If the error mess
      https://github.com/flutter/flutter/issues.
    X ProcessException: Failed to find "powershell" in the search path.
        Command: powershell
[X] Android toolchain - develop for Android devices
    X Unable to locate Android SDK.
      Install Android Studio from: https://developer.android.com/studio/ind
      On first launch it will assist you in installing the Android SDK comp
      (or visit https://flutter.dev/to/windows-android-setup for detailed i
      If the Android SDK has been installed to a custom location, please us
      `flutter config --android-sdk` to update to that location.

[√] Chrome - develop for the web
[X] Visual Studio - develop Windows apps
    X Visual Studio not installed; this is necessary to develop Windows app
      Download at https://visualstudio.microsoft.com/downloads/.
      Please install the "Desktop development with C++" workload, including
[!] 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.

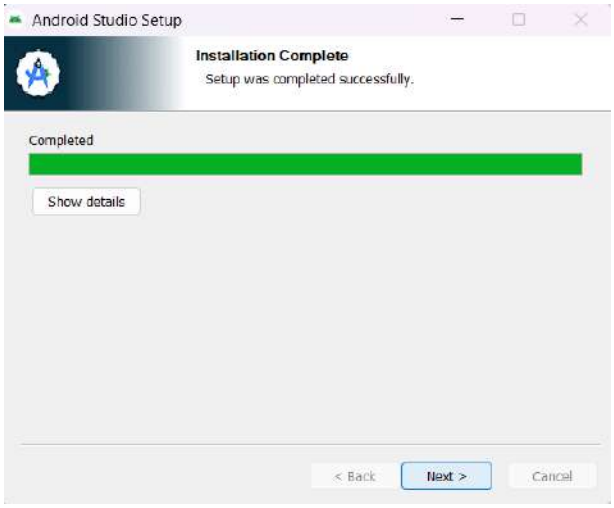
Step 7.1: Download the latest Android Studio executable or zip file from the official site.



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.



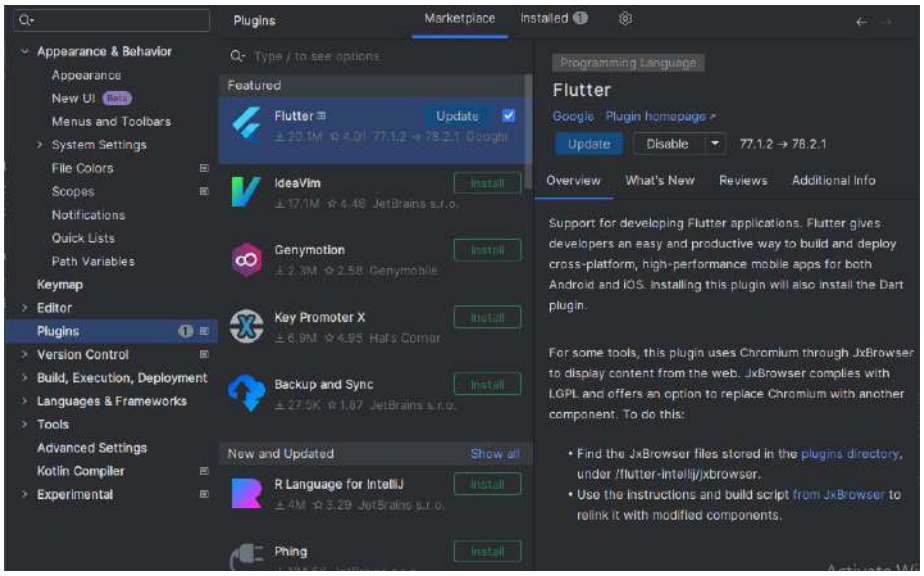
Step 7.4: In the above screen, click Next-> Finish. Once the Finish button is clicked, you need to

choose the 'Don't import Settings option' and click OK. It will start the Android Studio.

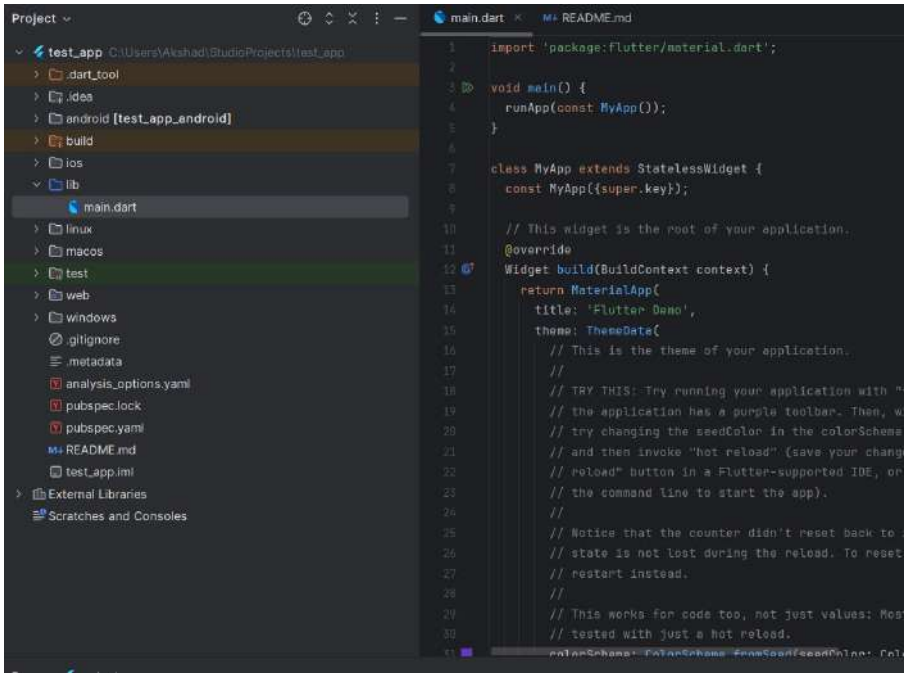
Step 8: Next, you need to set up an Android emulator. It is responsible for running and testing

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.

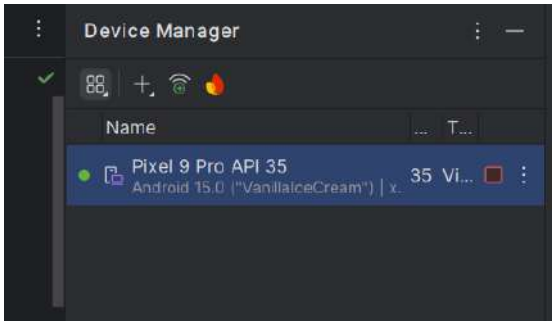


Step 8.2: Choose your device definition and click on Next.

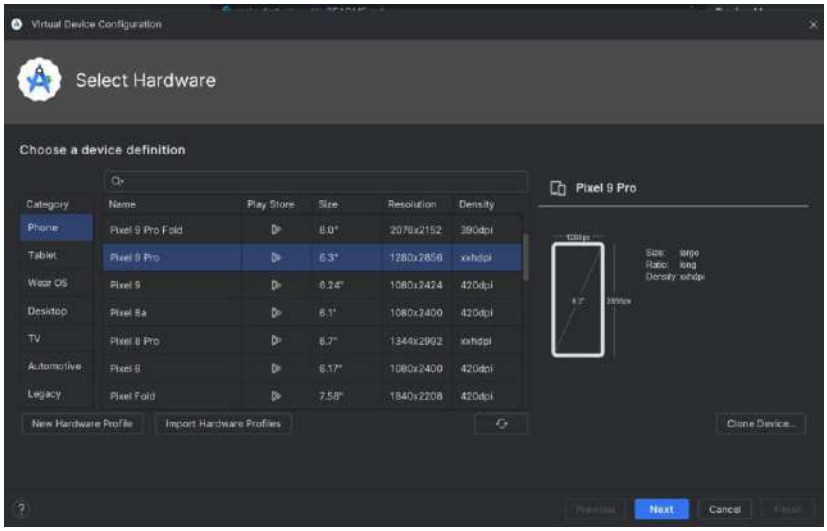


Step 8.3: Select the system image for the latest Android version and click on Next.

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

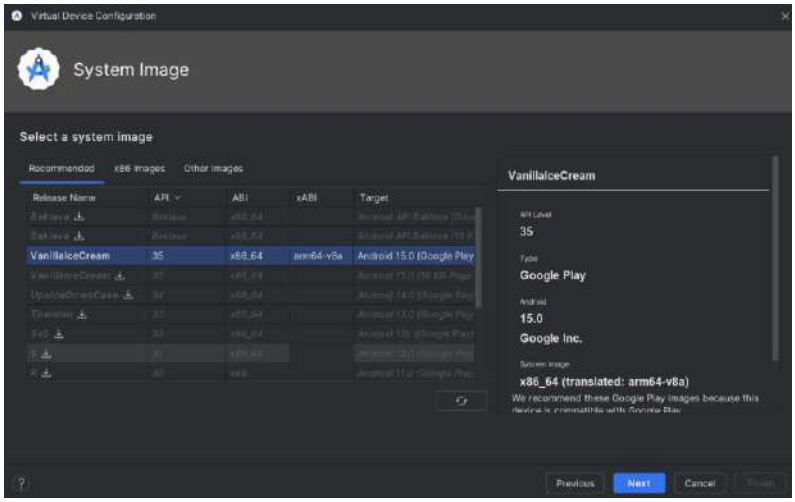


Step 9: Now, install Flutter and Dart plugin for building Flutter application in Android Studio.

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

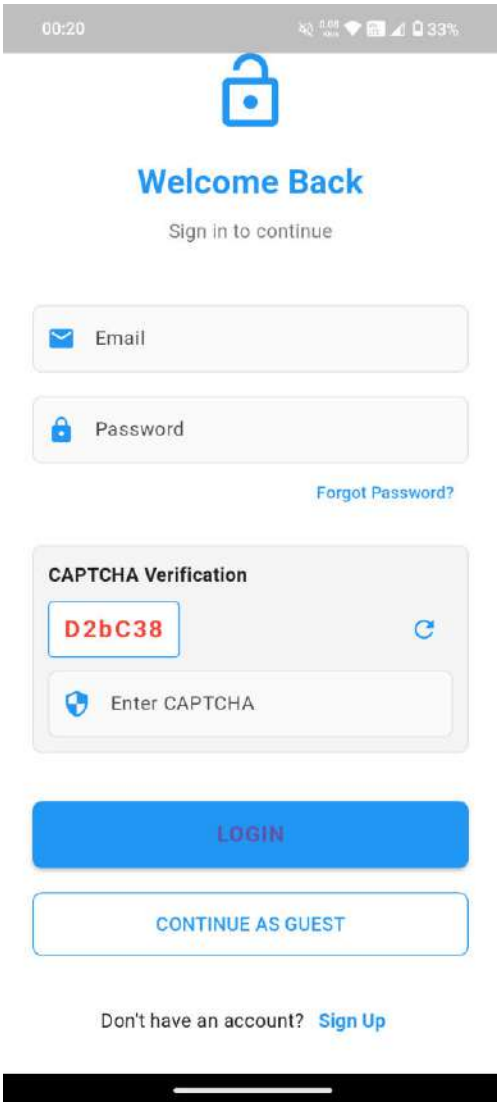
Flutter application in the Android Studio itself. Do the following steps to install these plugins.

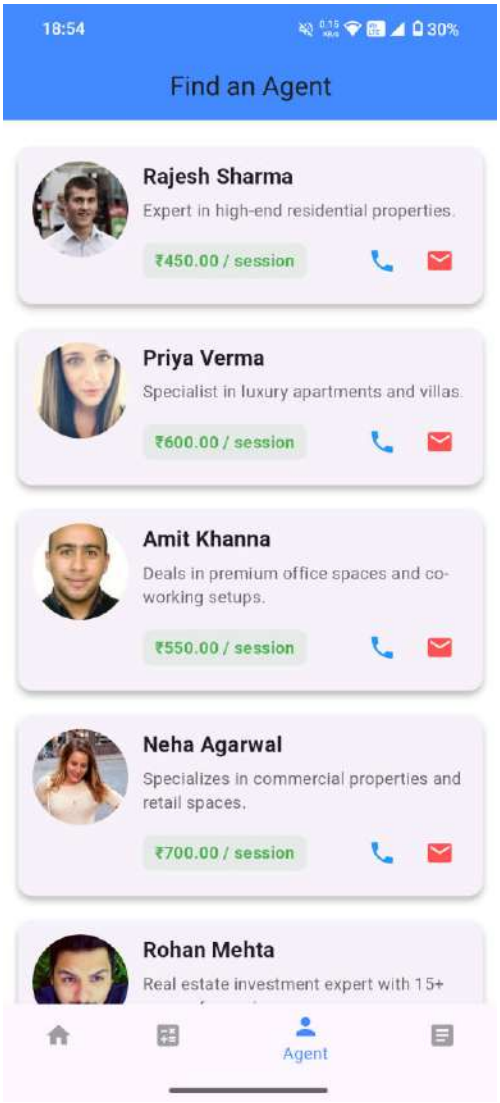
Step 9.1: Open the Android Studio and then go to File->Settings->Plugins.

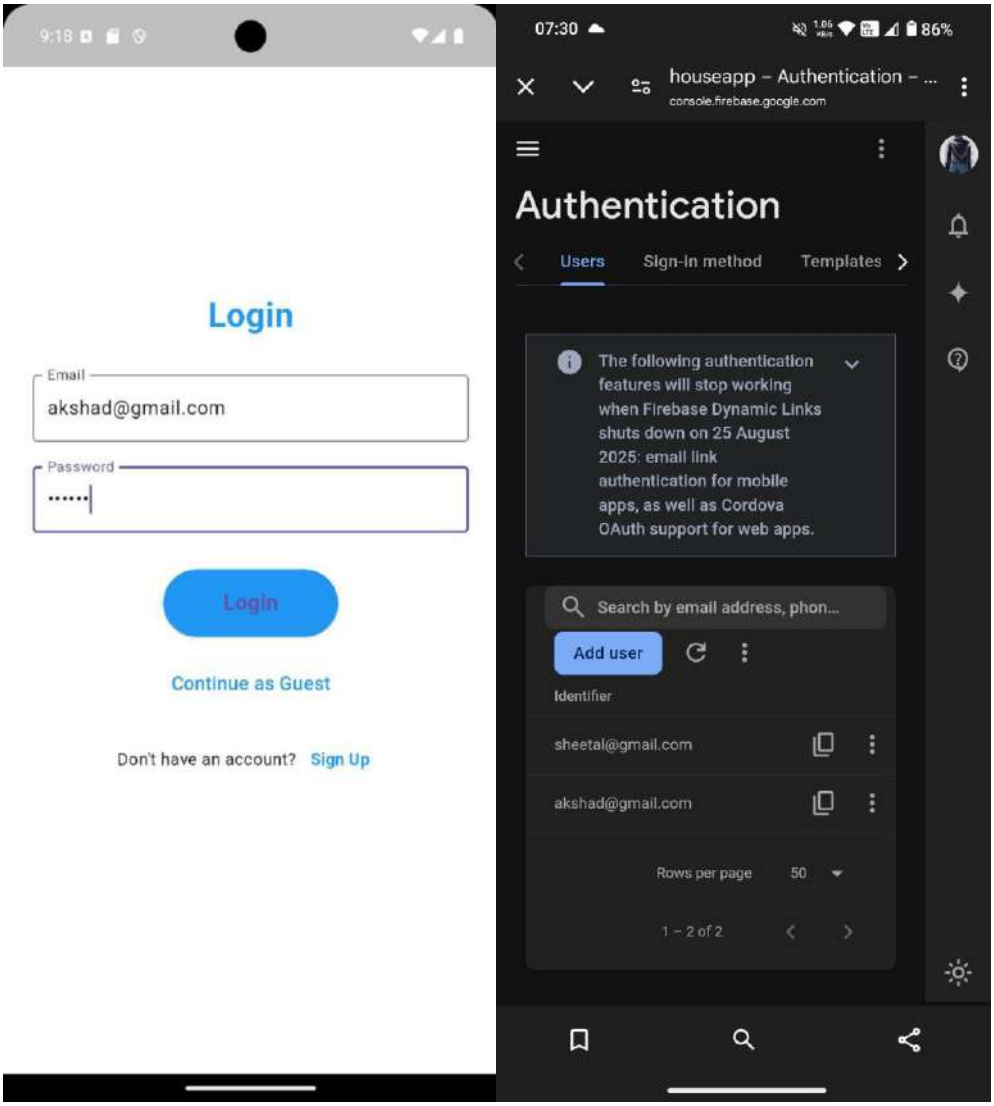


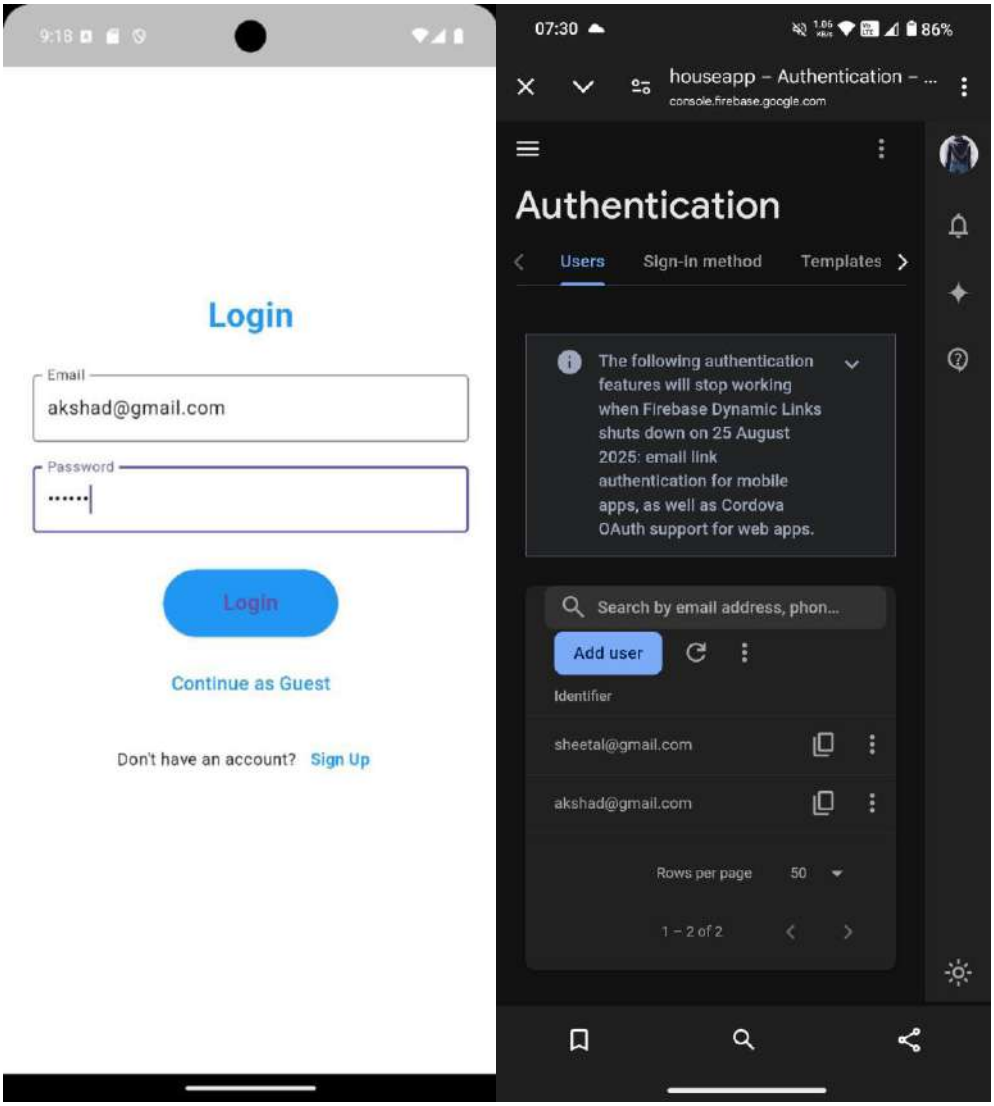
Step 9.2: Now, search the Flutter plugin. If found, select Flutter plugin and click install. When you click on install, it will ask you to install Dart plugin as below screen. Click yes to proceed.

Step 9.3: Restart the Android Studio.


00:20
33%
Welcome Back
Sign in to continue
Email
Password
Forgot Password?
CAPTCHA Verification
D2bC38
Enter CAPTCHA
LOGIN
CONTINUE AS GUEST
Don't have an account? Sign Up

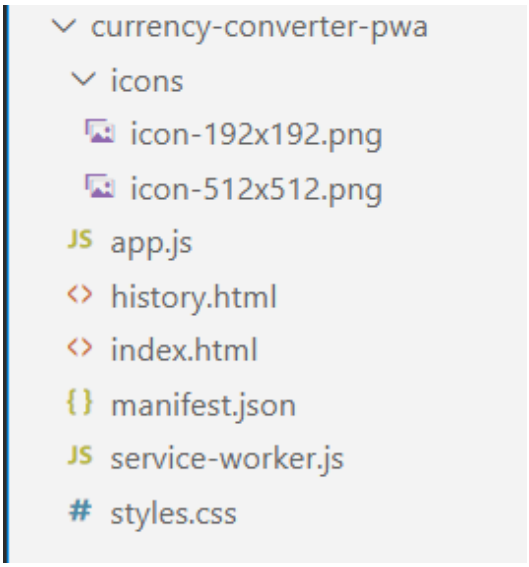

18:54
30%
Find an Agent
Rajesh Sharma
Expert in high-end residential properties.
₹450.00 / session
Priya Verma
Specialist in luxury apartments and villas.
₹600.00 / session
Amit Khanna
Deals in premium office spaces and co-working setups.
₹550.00 / session
Neha Agarwal
Specializes in commercial properties and retail spaces.
₹700.00 / session
Rohan Mehta
Real estate investment expert with 15+
Agent

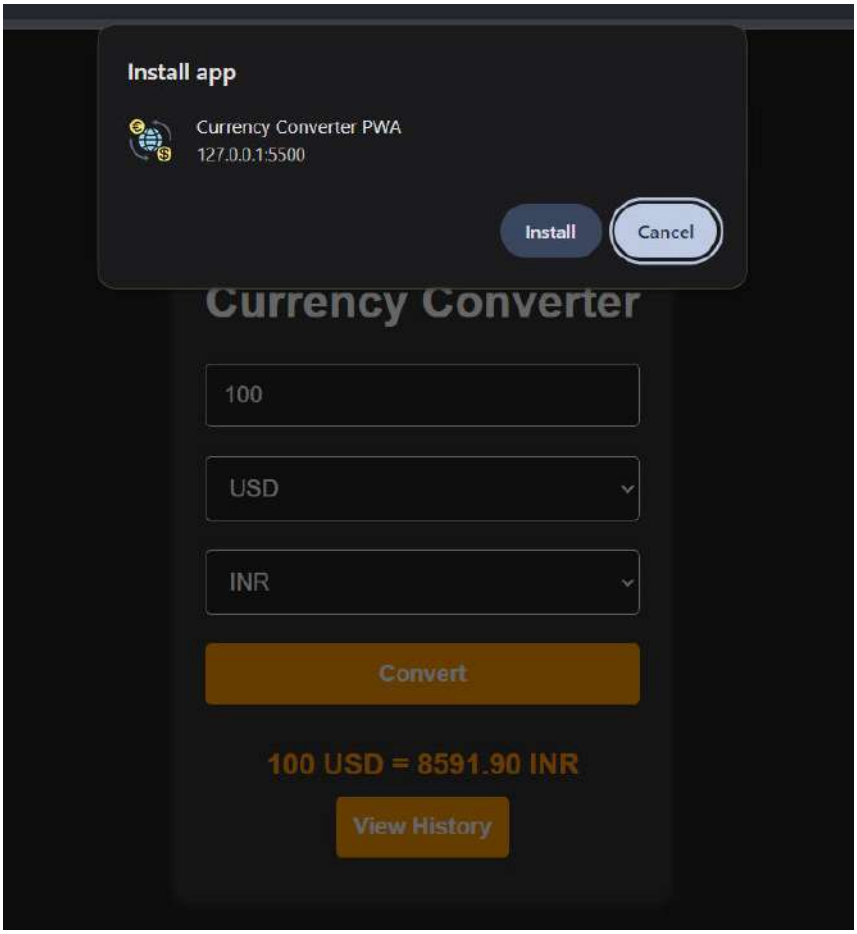

9:18
Login
Email
akshad@gmail.com
Password
Login
Continue as Guest
Don't have an account? Sign Up
07:30
86%
houseapp – Authentication – ...
console.firebase.google.com
Authentication
Users
Sign-in method
Templates
The following authentication features will stop working when Firebase Dynamic Links shuts down on 25 August 2025: email link authentication for mobile apps, as well as Cordova OAuth support for web apps.
Search by email address, phon...
Add user
Identifier
sheetal@gmail.com
akshad@gmail.com
Rows per page
50
1 – 2 of 2


9:18
Login
Email
akshad@gmail.com
Password
Login
Continue as Guest
Don't have an account? Sign Up
07:30
86%
houseapp – Authentication – ...
console.firebase.google.com
Authentication
Users
Sign-in method
Templates
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.
Search by email address, phon...
Add user
Identifier
sheetal@gmail.com
akshad@gmail.com
Rows per page
50
1 – 2 of 2

Name:Akshad Mhaskar
Div:D15B
Roll No.:32

MPL Experiment No.7

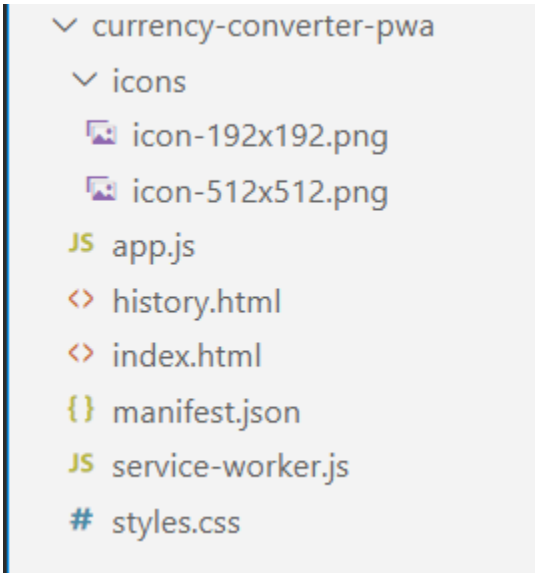


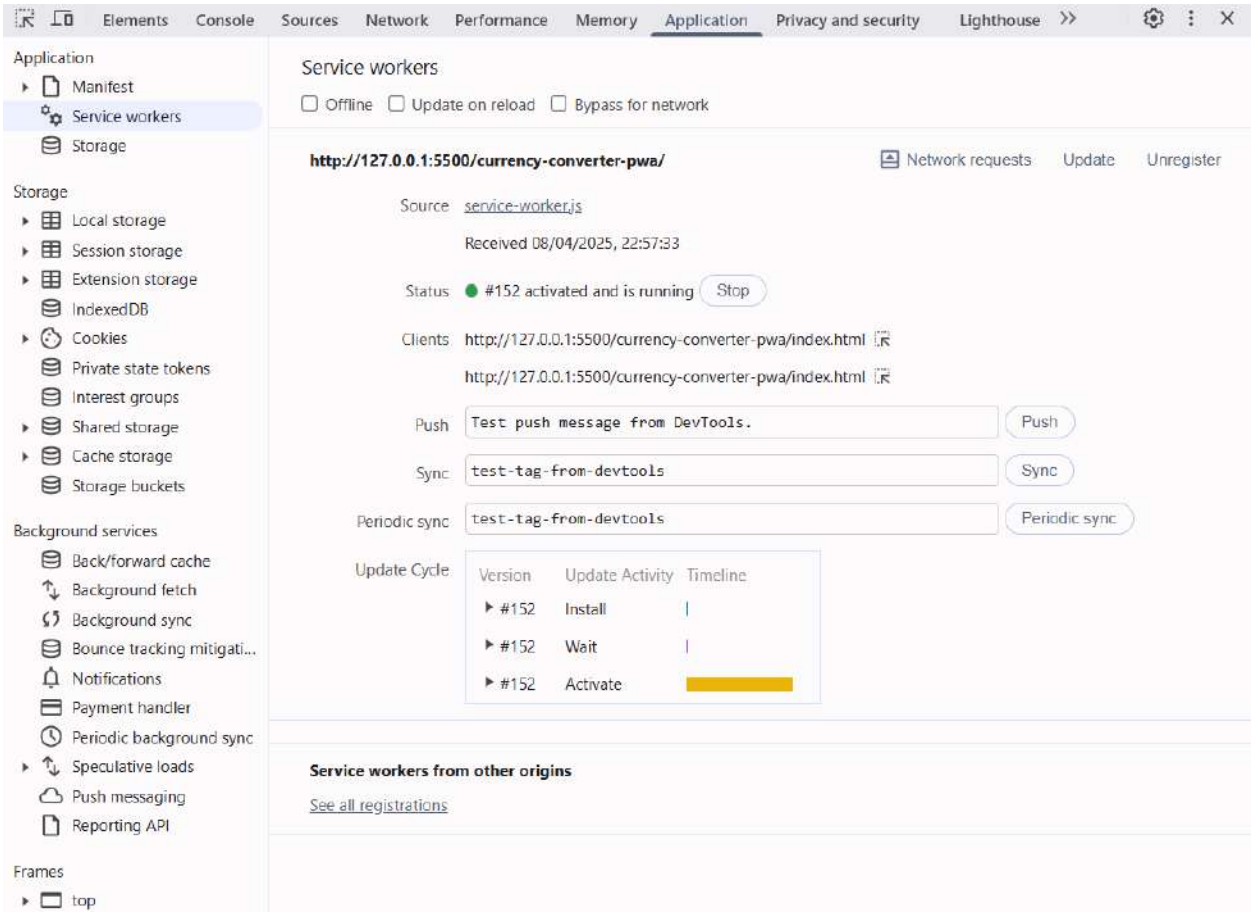

Install app
Currency Converter PWA
127.0.0.1:5500
Install
Cancel
Currency Converter
100
USD
INR
Convert
100 USD = 8591.90 INR
View History

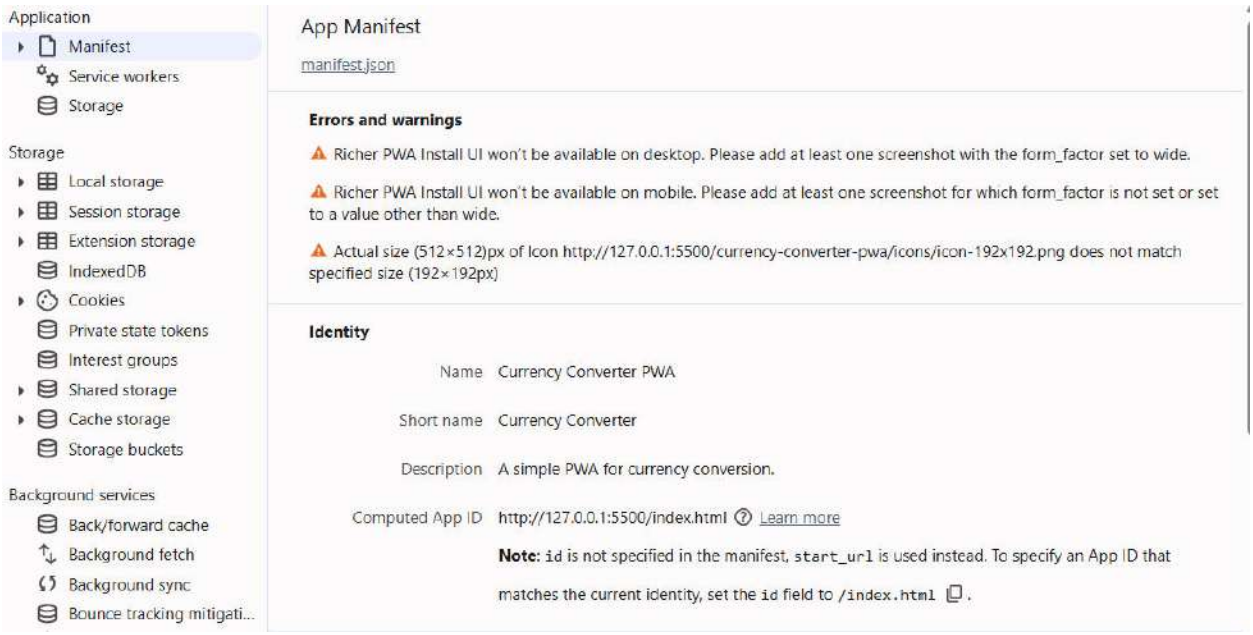

MongoDBC...
NVIDIA
Currency Convert...

Name:Akshad Mhaskar
Div:D15B
Roll No.:32

MPL Experiment No.8





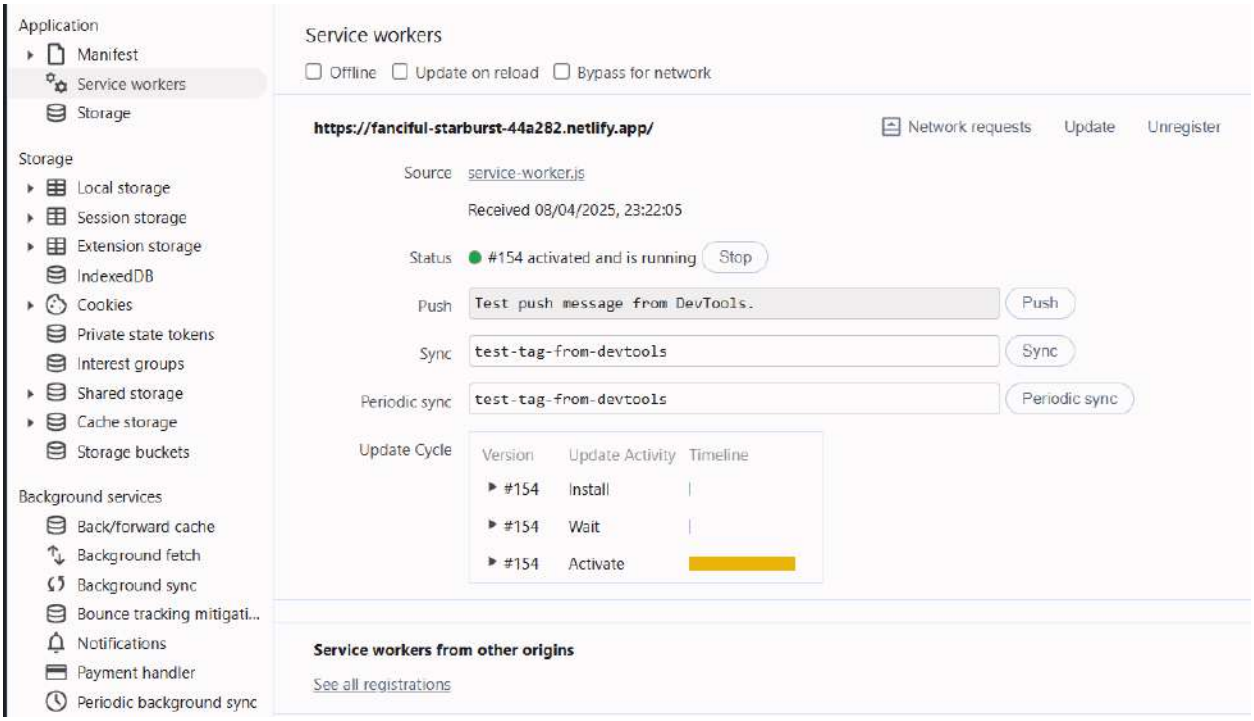

Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Storage buckets
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking mitigati...
App Manifest
manifest.json
Errors and warnings
Richer PWA Install UI won't be available on desktop. Please add at least one screenshot with the form_factor set to wide.
Richer PWA Install UI won't be available on mobile. Please add at least one screenshot for which form_factor is not set or set to a value other than wide.
Actual size (512×512)px of Icon http://127.0.0.1:5500/currency-converter-pwa/icons/icon-192x192.png does not match specified size (192×192px)
Identity
Name
Currency Converter PWA
Short name
Currency Converter
Description
A simple PWA for currency conversion.
Computed App ID
http://127.0.0.1:5500/index.html
Learn more
Note: id is not specified in the manifest, start_url is used instead. To specify an App ID that matches the current identity, set the id field to /index.html .

Name:Akshad Mhaskar

Div:D15B

Roll No.:32

MPL Experiment No.9

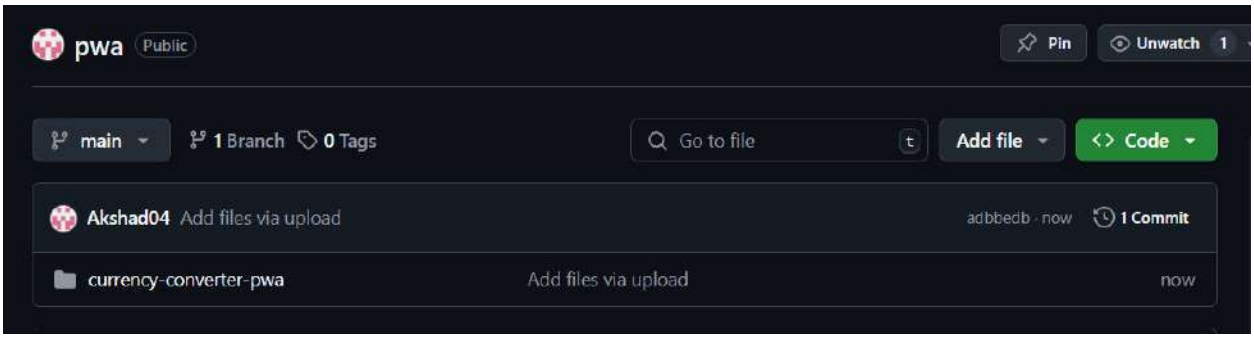


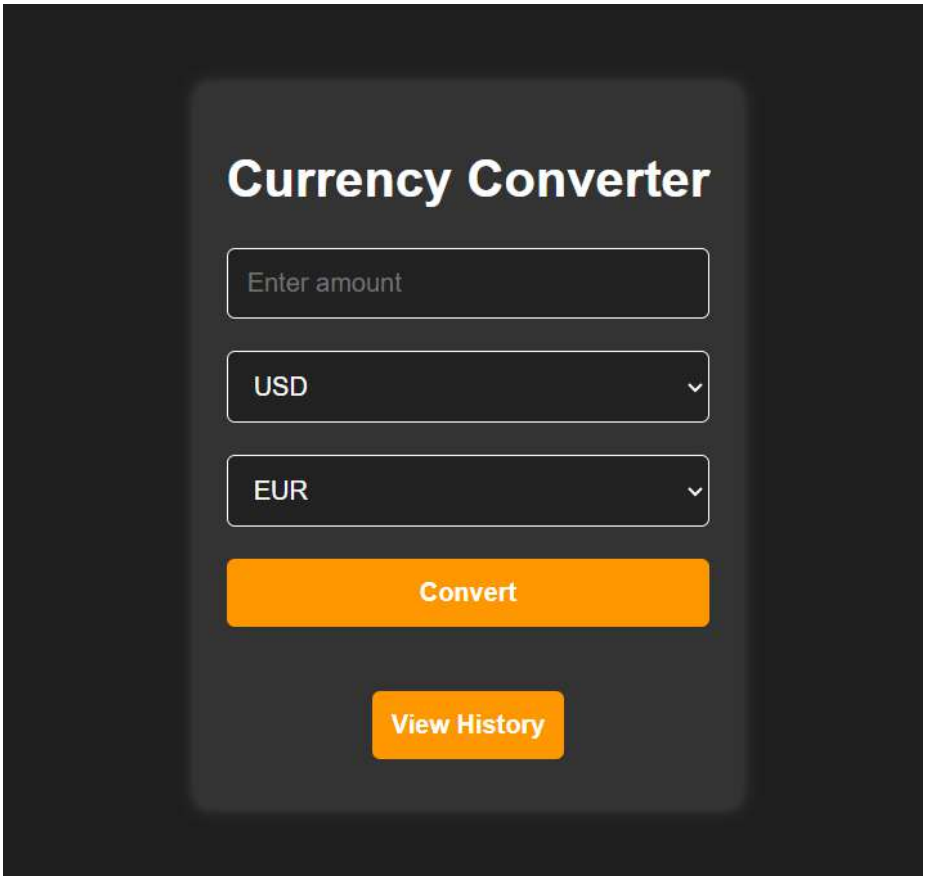
Name:Akshad Mhaskar

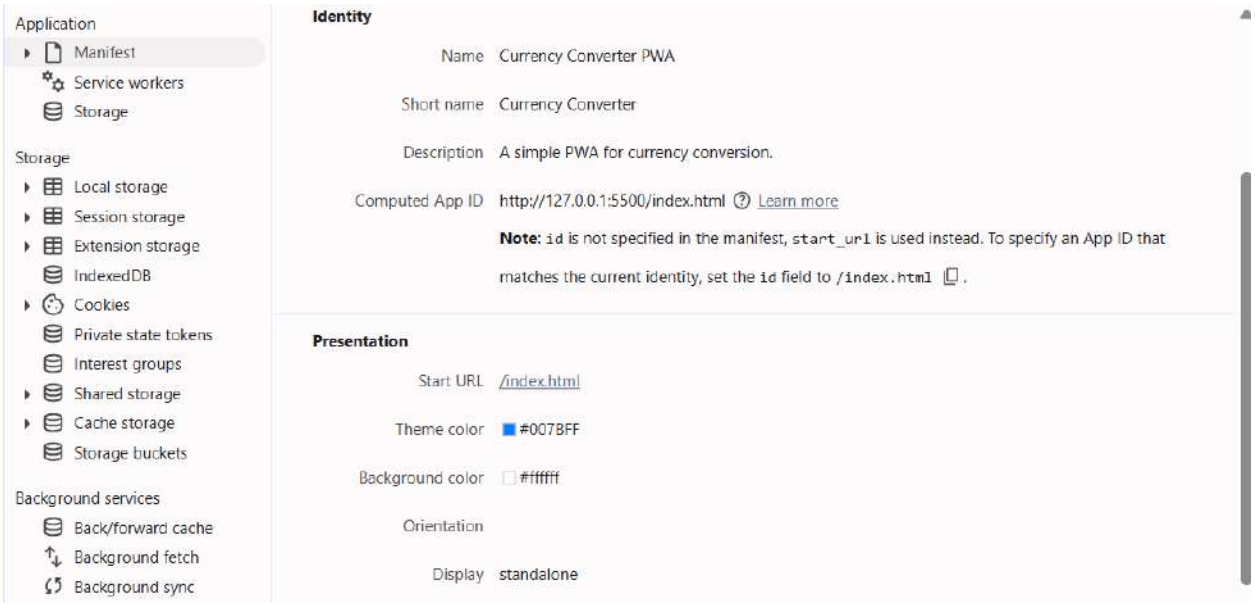
Div:D15B

Roll No.:32

MPL Experiment No.10





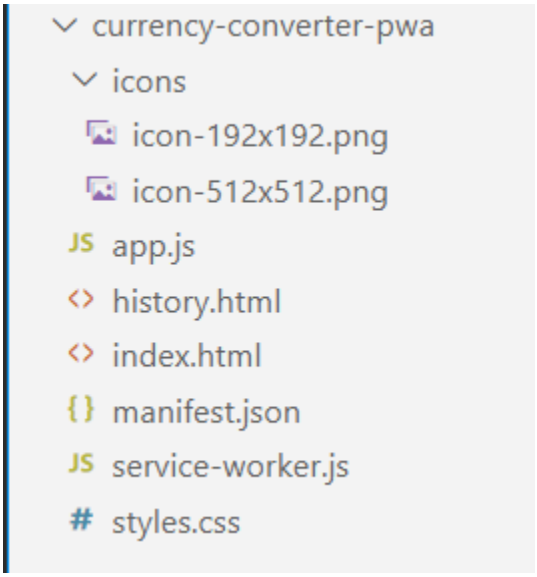

Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Storage buckets
Background services
Back/forward cache
Background fetch
Background sync
Identity
Name Currency Converter PWA
Short name Currency Converter
Description A simple PWA for currency conversion.
Computed App ID http://127.0.0.1:5500/index.html Learn more
Note: id is not specified in the manifest, start_url is used instead. To specify an App ID that matches the current identity, set the id field to /index.html .
Presentation
Start URL /index.html
Theme color #007BFF
Background color #ffffff
Orientation
Display standalone

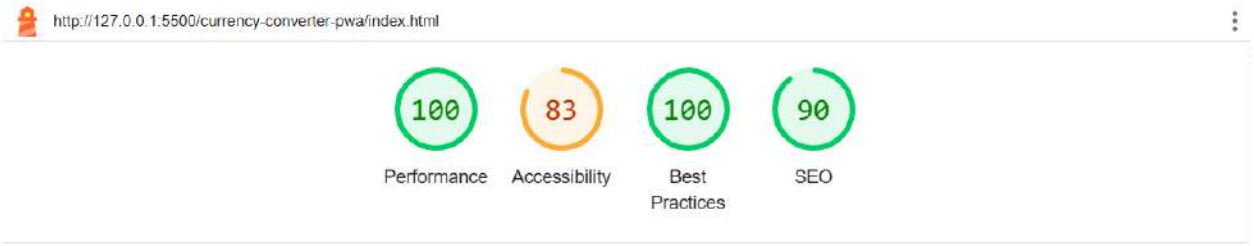
Name:Akshad Mhaskar

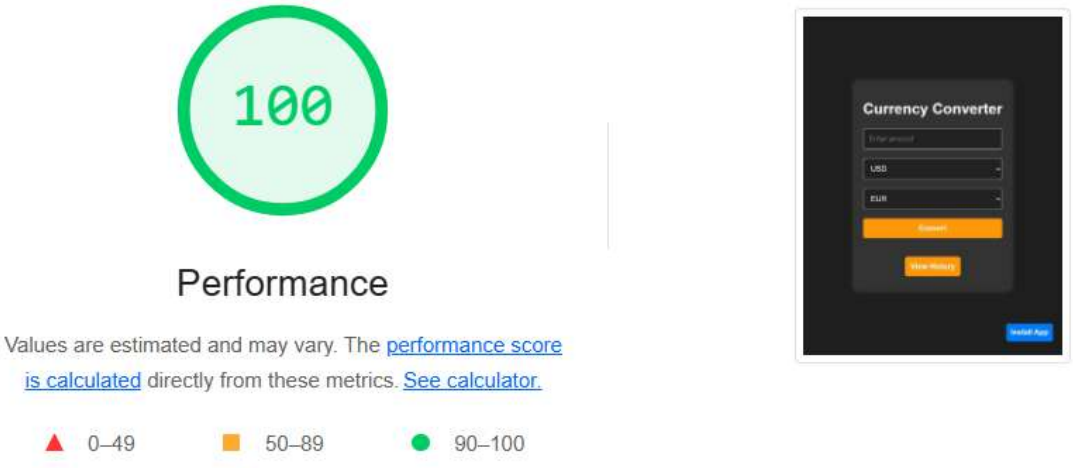
Div:D15B

Roll No.:32

MPL Experiment No.11







METRICS Expand view

- First Contentful Paint: 0.3 s
- Largest Contentful Paint: 0.3 s
- Total Blocking Time: 0 ms
- Cumulative Layout Shift: 0
- Speed Index: 0.3 s

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.

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

Akshad Mhaskar
D15B / 32

MPL Assignment 1

Q1)a) Explain the key features & advantages of using flutter for mobile app development.

→ Key features:

1) Single Codebase - Write one codebase for andriod & ios, reducing development & effort & maintainenance
2) Hot Reload - Instantly see changes in the app without restarting making development faster & more interactive
3) Fast Performance - Uses the Dart language & a compiled approach for smooth & high performance apps
4) Open Source & strong community support: Backed by Google & a large developer community, ensuring continuous improvements & resources.

Advantages.

1) Faster Development time: Hot reload & single codebase reduce development time significantly
2) Cost effective: Since the code runs on both Andriod & ios, business save on development & maintainence cost.
3) Reduce Performance issues: The app runs natively without relying on intermediate bridges like in react native reducing lag.

b) Discuss how the flutter framework differs from traditional approaches & why it has gained popularity

→ How Flutter Differs from traditional Approaches

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

1) Single codebase - Traditional methods need separate code for android & ios, but flutter uses one code for both.
2) Hot Reload - Traditional apps require full restart after changes, but flutter updates instantly.
3) UI Rendering - Traditional apps ~~require~~ use native components, while flutter has its own rendering engine (Skia) for faster performance
4) Customization - Traditional UI design depends on platform specific components, but flutter provides fully customizable widgets.

Why Flutter has gained popularity

1) Faster development with hot reload: Developers can instantly see UI changes without restarting the app making iteration process much quicker.
2) Cross Platform Efficiency: Businesses save time & resources by maintaining a single codebase for multiple platform.
3) consistent UI Across devices: Since flutter does not rely on native components, the UI looks & behaves the same across different OS versions.
4) Improved performance: AOT compilation & direct accesses to GPU rendering ensure smooth animations & high performance.

Q2) a) Describe the concept of the widget tree in flutter. Explain how widget composition is used to build complex UI.

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

→ 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 is UI is entirely built using widgets which can be stateless or stateful. The widget tree determines how the UI is rendered & updated when changes occur.

Widget Composition in Flutter

Widget composition refers to building complex UI by combining smaller, reusable UI widgets. Instead of creating large, monolithic UI components, flutter encourages breaking the UI into smaller manageable widgets that can be reused & nested with each other.

Benefits of Widget composition

1) ~~Resu~~ Reusability: Small widgets can be ~~measured~~ reused in different parts of the app.
2) Maintainability: Breaking UI into smaller widgets makes its easier to debug & update.
3) Performance: Flutter efficiently rebuilds only the necessary parts of widget tree.

b) Provide examples of commonly used widgets & their roles in creating a widget tree.

→ 1) Structural widgets

These widgets act as the foundation building the UI

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

- Material App: the root widgets of a flutter app that provid essential configuration.
- Scaffold: Provides a basic layout structure, including an app bar, body floating action button etc.
- Container - A versatile widget used for styling, padding, margin & background customization.

Ex :- MaterialApp(
home : Scaffold(
appBar : AppBar(title : Text("flutter widget tree"
body : container(
padding : EdgeInserts.all(16.0),
child : Text("Hello, Flutter !"),
),
),
);

⇒ 2) Input & Interaction Widgets
Textfold : Accepts text input from user.
Elevation Button - A button with elevation.
GestureDetector - Detects gesture like taps, swipes & long presses.

Ex : Column(
children: [
Textfield(decoration: InputDecoration(labelText:
"Enter name"
ElevatedButton(
onPressed : (){
print("Button Pressed"), },
child : Text("submit"),),
],
);

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

3) Display & Styling Widgets
- Text - displays text on the screen
- Image - shows images from assets network/memory
- Icon - Displays icon
- Card - A material design & with card with rounded corners & elevation.

Ex :-
```
column(
  children: [
  Text("Welcome to flutter!", style: TextStyle(Fontsize:20
                                  Fontweight: Fontweight.bold)),
  Image.Network("https://flutter.dev/images/flutter-logo
                                  -sharing.png"),
  ], );
```

Q3) a) Discuss the important state managment in flutter applications.

→ In flutter, state refers to data that can change during the lifetime, of an application. This includes:
- user input, UI changes, Network changes, Animation states.

There are two types of states:
1) ~~Ephemeral~~ Ephemeral state: small, UI specific state that doesn't affect the whole app.
2) App wide status - Data shared across multiple widgets

Importance of State Mgmt
- Efficient UI updates: Flutter's UI is rebuilt whenever state changes. Efficient state mgmt ensures that only necessary widgets are updated improving performance.

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

- Code Maintainability & Scalability : Managing state properly makes the code modular, readable & scalable for larger applications.
- Data Consistency & Synchronization - Proper state mgmt ensures that data remains consistent across different screens & widgets.

b) Compare & constrast the different state mgmt approaches available in flutter, such as setstate, Provider & Riverpod. Provide scenarios where each ~~components~~ approach is suitable.

→ setstate - local state
Pro - Simple built in easy to use
Cons - Not scalable causes unnecessary re-renders
Best use cases - Small UI updates (eg:- counter)
Provider - App wide state
Pros - Lightweight, recommended by flutter
Cons - Boilerplate code for nested providers.
Best use case - Medium Scale App (eg:- API data)
Riverpod : App-wide state (More scalable)
Pros : Eliminates Provider's limitation, improves performance
Cons : Requires learning new concepts
Best use cases : large apps needing global state (eg :- Shopping cart)

Scenar

Scenarios for each Approach

- Use setstate when managing simple UI elements with in a single widget like toggling dark mode in a setting screen.
- Use provider when sharing state across multiple widgets such as managing user authentication or theme changes
- Use Riverpod when building a complex, scalable app with cart mgmt.

Q3) a) Explain the process of integrating firebase with a flutter application. Discuss the benefits of using firebase as a backend solution.

→ Integrating Firebase with a Flutter Application

Step 1: Create a Firebase project

1. Go to Firebase console & create a project
2. Download the config (google-services.json for Andriod).

Step 2: Add Firebase Dependencies

In pubspec.yaml:

dependencies:
firebase_core: latest_version
firebase_auth: latest-version
cloud_firestore: latest-version

Re Run flutter pub get to install dependencies.

Step 3: Initialize firebase.

Modify main.dart.

```
import 'package:firebase_core/firebase_core.dart';
void main() asyn {
```

```
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(MyApp());
}
```

Step 4: Use Firebase service
Authentication: firebase_auth for login/signup
Database: cloud_firestore for real-time data.
Storage: firebase_storage for file uploads
Notifications: firebase_messaging for push notification.

Benefits of using firebase

1. Serverless Backend - No need to manage servers.
2. Real-Time Data Sync - Firestore enables instant updates
3. Scalable & Secure - Google-managed infrastructure
4. Easy authentication - Supports google, facebook, email login.
5. Cloud functions - Automate tasks with serverless function.
6. Push notifications - free messaging via FCM.
7. Cost-Effective - Free tier with pay-as-you-go pricing.
8. Analytics & crash Reports - Firebase Analytics & Crashlytics.

b) Highlight the firebase services commonly used in flutter development & provide a brief overview of how data synchronisation is achieved.

→ Common Firebase services in flutter

1. Firebase Authentication - Secure login via email, google, Facebook or phone.
2. Cloud firestore - NoSQL database with real-time syncing.
3. Realtime Database - low-latency database for instant updates.
4. Firebase Storage - stores & retrieves files like images & video.
5. Firebase cloud Messaging (FCM) - ~~Enab~~ Enables push notifications.
6. Firebase crashlytics - Tracks & reports app crashes.
7. Firebase Cloud functions - Runs backend tasks serverlessly.
8. Firebase Analytics - Provides insights into user behaviour

Data Synchronization in Firebase.

1. Realtime Listeners - Apps receive instant updates when data changes.

Ex:-

```
FirebaseFirestore.instance.collection('users').
snapshots().listen((snapshot){
for (var var doc in snapshot.docs){
    print(doc.data());
}});
```

2. offline persistence - caches data for offline access & syncs when online.
3. conflict Resolution - Uses timestamps to maintain data consistency.

Firebase's real-time sync & offline support make it ideal for chat apps, live updates & collaborative tools in flutter.

NAME: STD.: DIV.: DATE: 05/02/25 PAGE:

Akshad Mhaskar
D15B/32

MPL Assignment - 2

Q1) Define Progressive web App (~~DWA~~) (PWA) & Explain its significance in modern web dev. Discuss the key characteristics that differentiate PWA's from traditional mobile apps.

→ A progressive web App (PWA) is type of web application that works like a mobile app but runs in a browser. It can be installed on a device works offline & provides a fast & smooth user experience.

Significance of PWA in modern web dev.

- Cross platform compatibility
- Offline support
- Fast performance
- Lower development cost

Key diff bet^n PWA & ~~tradd~~ traditional mobile Apps.

Features	PWA	traditional Mobile App
Installation	Direct from browser	Download from App store
Internet required	works offline with caching	usually requires internet
Performance	Fast with service workers	Faster but needs installations
update	Automatic no app store approval	manual updates needed.
Dev cost	lower	Higher

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

Q2) Define responsiveness web design & explain its importance in the context of PWA compare & constrast responsiveness fluid & adaptive web design approaches.

→ Responsive Web Design (RWD) is technique that make web pages adjust automatically to different screen sizes & devices. It ensures a good user experience on mobiles, tablets & desktops without needing separate version of website.

Importance of responsiveness design in PWA.

- Better user exp - PWA's work smoothly on any device.
- Faster load time - optimized design improves speed.
- SEO Benefits - Google ranks responsive sites higher.
- Cost effective - No need to design multiple versions for different screens.

Comparision of web design approaches

Approaches	How it works	Pros	Cons
• Responsives	uses flexible grids & CSS media queries to adjus layout	works on all devices	can be complex to design
• Fluid	uses percent based widths instead of fixed pixels so elements resize smoothly	works wellon different screen sizes easy to implement	less control over layout on large screens

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

• Adaptive	Uses fixed layouts that change at specific breakpts	optimized by known screen sizes	most effort require to design by screen size.

Key differences
- Responsive adapts dynamically to all screens.
- Fluid resizes smoothly, but must be fully a proiritized.
- Adaptive loads different layouts based on device types.

Q3) Describe the lifecycle workers including to registration installation & activation phase.

→ Lifecycle of service workers

A service workers is a script that runs in the background & a helps a web app work offline load faster & send push notification, its lifecycle has 3 main phases.

1. Registration Phase

The browser registers the source worker using JS.

Ex:-

```
If ('service worker' in navigator ){
navigator service worker register (`/sw.js')
.then (()→ console.log (`service work' registered' ))
.catch (error → conside.log (Registration failed', error));
```

This tells the browser to install & activate the service worker ~~installation phase~~

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

2. Installation Phase
The service worker downloads necessary files (HTML, CSS, JS) & stores them in cache.
If successful it moves to the activation phase.
Ex:-

```
self.addEventListener('install', event => {
    event.waitUntil(
      caches.open(app.cache).then(cache => {
return cache.addAll(['/', 'index.html',
    'styles.css']);
    })
    }); });
```

This ensures the app loads even without the internet

3. Activation Phase
- the old service worker is replaced with the new one
- Unused cache files from the previous version are deleted

final step step & sync.
Once activated the service worker intercepts network requests, serves cached files & syncs data when the internet is available. This lifecycle makes PWA's faster, more ~~real~~ relatable & capable of working offline.

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

Q4) Explain the use of Indexed DB in the system service worker of data storage.

→ Use of IndexedDB in service worker for data storage IndexedDB is a browser database that stores large amounts of structured data like JSON or objects. It helps PWA's work offline by saving & retrieving data efficiently.

Why use IndexedDB in service workers ?

1. Office support - Stores data when offline & syncs it later.
2. Efficient storage - Saves structured data like user setting cost items or form inputs.
3. Faster Access - Retrieves data quickly without needing a network request.
4. Persisent Data - Data remains saved even after the browser is classed

How service workers use Indexed DB ?

opening the database

```
let db;
let request = indexedDB.open('My database');
  request.onsuccess = function (even) {
      db = event.target.result;
    };
```

NAME:_____ STD.:_____ DIV.:_____

DATE :
PAGE :

Creating a store & adding Data

```
request.onupgradeneeded = function(event){
let db = event.target.result;
let store = db.createObjectStore('users', {keyPath
    : 'id'});
store.add({id:1, name:'John Doe', age:'age 25'});
};
```

Fetching Data in service worker

```
let transaction = db.transaction('users','readonly');
let store = transaction.objectStore('users');
let getUser = store.get(1);

getusers.onsuccess = function () {
  console.log('get user detail');
};
```