Vivekanand Education Society's Institute of Technology

An Autonomous Institute Affiliated to University of Mumbai
Hashu Advani Memorial Complex, Collector Colony, Chembur East, Mumbai - 400074.

Department of Information Technology

CERTIFICATE

This is to certify that _____ of **D15A/D15B** semester **VI,** have successfully completed necessary experiments in the **MAD & PWA Lab** under my supervision in **VES Institute of Technology** during the academic year **2024-2025.**

Lab Teacher
Dr. Ravita Mishra
Signature:

Head of Department
Dr. Mrs. Shalu Chopra
Signature:

Principal

Name of the Course : MAD & PWA Lab **Course Code :** ITL604

Year/Sem/Class : D15A/D15B **A.Y.:** 24-25

Faculty Incharge : Dr. Ravita Mishra

Lab Teachers : Dr. Ravita Mishra.

Email : ravita.mishra@ves.ac.in

Programme Outcomes: The graduate will be able to:

PO1) Basic Engineering knowledge: An ability to apply the fundamental knowledge in mathematics, science and engineering to solve problems in Computer engineering.

PO2) Problem Analysis: Identify, formulate, research literature and analyze computer engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and computer engineering and sciences.

PO3) Design/ Development of Solutions: Design solutions for complex computer engineering problems and design system components or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal and environmental considerations.

PO4) Conduct investigations of complex engineering problems using research-based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of information to provide valid conclusions.

PO5) Modern Tool Usage: Create, select and apply appropriate techniques, resources and modern computer engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6) The Engineer and Society: Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to computer engineering practice.

PO7) Environment and Sustainability: Understand the impact of professional computer engineering solutions in societal and environmental contexts and demonstrate knowledge of and need for sustainable development.

PO8) Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of computer engineering practice.

PO9) Individual and Team Work: Function effectively as an individual, and as a member or leader in diverse teams and in multidisciplinary settings.

PO10) Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations and give and receive clear instructions.

PO11) Project Management and Finance: Demonstrate knowledge and understanding of computer engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12) Life-long Learning: Recognize the need for and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

Program specific Outcomes

PSO1) An ability to manage and analyze data / information effectively for making better decisions.

PSO2) Demonstrate the ability to use state of the art technologies and tools including Free and Open Source Software (FOSS) tools in developing software.

Index

Sr. No	Experiment Title	LO	DOP	DOS	Grade
1.	To install and configure the Flutter Environment	LO1			
2.	To design Flutter UI by including common widgets.	LO2			
3.	To include icons, images, fonts in Flutter app	LO2			
4.	To create an interactive Form using form widget	LO2			
5.	To apply navigation, routing and gestures in Flutter App	LO2			
6.	To Connect Flutter UI with fireBase database	LO3			
7.	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.	LO4			
8.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA	LO5			
9.	To implement Service worker events like fetch, sync and push for E-commerce PWA	LO5			
10.	To study and implement deployment of Ecommerce PWA to GitHub Pages.	LO5			
11.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.	LO6			
12.	Assignment-1	LO1,LO2 ,LO3			
13.	Assignment-2	LO4,LO5 ,LO6			

Lab Objectives:

Sr. No.	Lab Objectives
The Lab experiments aims:	
1	Learn the basics of the Flutter framework.
2	Develop the App UI by incorporating widgets, layouts, gestures and animation
3	Create a production ready Flutter App by including files and firebase backend service.
4	Learn the Essential technologies, and Concepts of PWAs to get started as quickly and efficiently as possible
5	Develop responsive web applications by combining AJAX development techniques with the jQuery JavaScript library.
6	Understand how service workers operate and also learn to Test and Deploy PWA.

Lab Outcomes:

Sr. No.	Lab Outcomes	Cognitive levels of attainment as per Bloom's Taxonomy
On Completion of the course the learner/student should be able to:		
1	Understand cross platform mobile application development using Flutter framework	L1, L2
2	Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation	L3
3	Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS	L3, L4
4	Understand various PWA frameworks and their requirements	L1, L2
5	Design and Develop a responsive User Interface by applying PWA Design techniques	L3
6	Develop and Analyse PWA Features and deploy it over app hosting solutions	L3, L4

MAD & PWA Lab
Journal

Experiment No.	01
Experiment Title.	To install and configure the Flutter Environment
Roll No.	
Name	
Class	D15B
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework
Grade:	

Name: Harsh Jadhav
Roll No: 21
Class: D15B

MPL Experiment No. 1

* **Aim:** Installation and configuration of flutter Environment.

* **Theory:**

Flutter is an open-source UI software development kit created by google. It is used for developing natively compiled applications for mobile, web and desktop from a single codebase. This experiment focuses on the installation and setup of Flutter and its dependencies, ensuring a proper development environment for flutter applications.

- **concepts:-**

1. **Flutter SDK:**
The flutter software Development Kit (SDK) component contain all necessary libraries and tools required for flutter development.

2. **System Path configuration:**
Adding the flutter binary directory to the system path allows the flutter command to be recognized globally in the command prompt.

3. **Android SDK and Emulators:** The ASDK is required for running and debugging applications.

* Steps :-

1) Download flutter SDK -

2) Extract & Set Path - Extract the SDK and add its bin folder to System path.

3) Run flutter Doctor - open CMD and run : flutter doctor to check dependencies.

4) Install Android Studio : Download and Install Android studio to set up Android SDK.

5) Set up Emulator - In Android studio go to Tools > AVD Manager create virtual device and configure it.

6) Install flutter & Dart plugins - In Android studio, go to file > Settings > plugins, search for flutter, install it and restart.

7) Run flutter doctor to ensure all dependencies are installed properly.

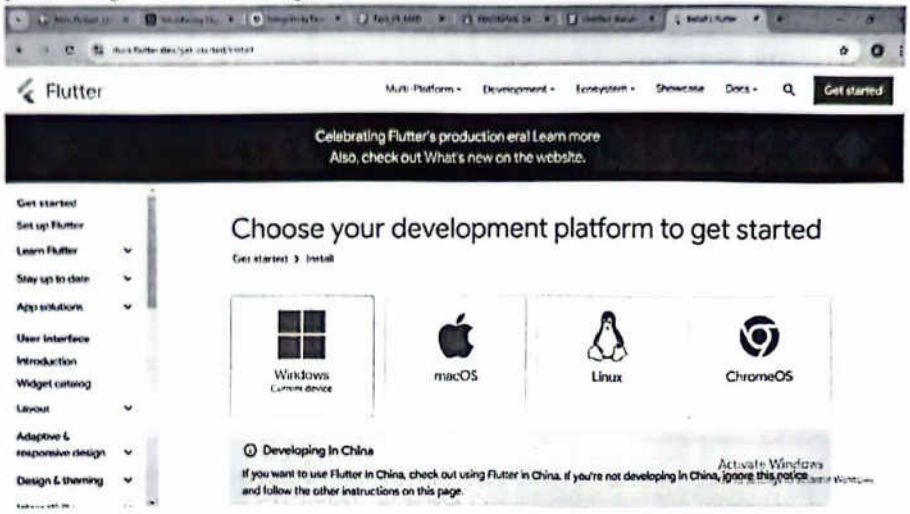
* Conclusion :-

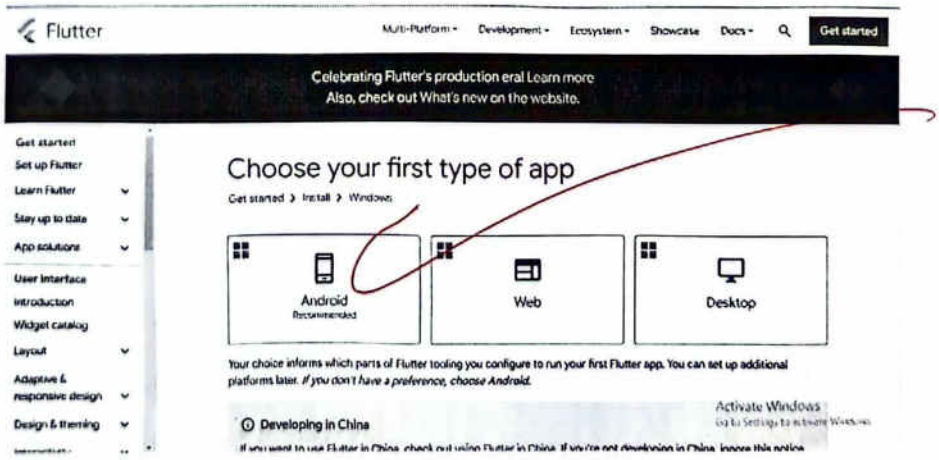
We have completed this experiment sucessfully and learned and understood about the installation and setup of flutter and Android studio as well as creating an AVD in Android studio.

Name: Harsh Jadhav
Roll No:21
Div: D15B

MPL Experiment No.1

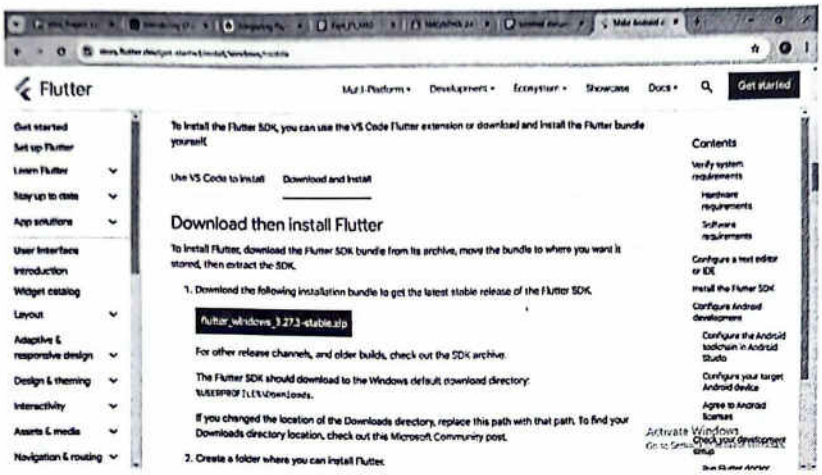
Step 1: Download the installation bundle of the Flutter Software Development Kit for windows. To download Flutter SDK, Go to its official website https://docs.flutter.dev/get-started/install , you will get the following screen.





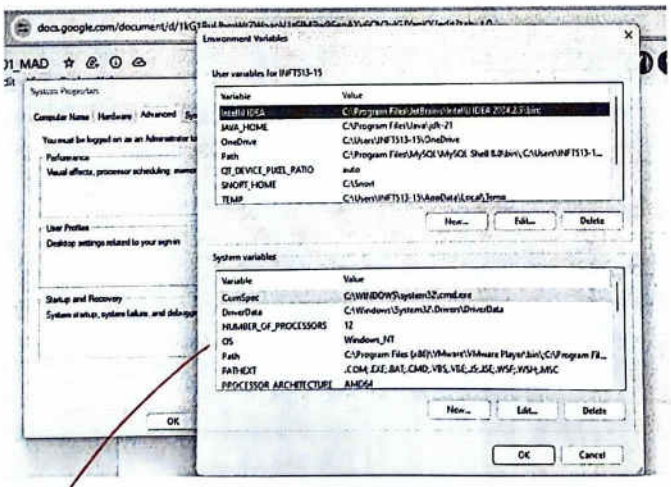
Step 2: Next, to download the latest Flutter SDK, click on the Windows icon. Here, you will find the download link for SDK.

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.



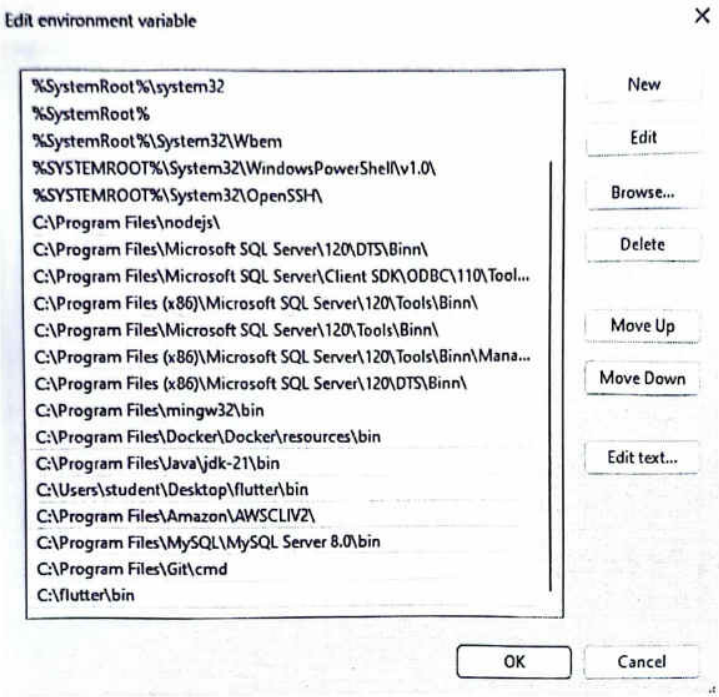
Step 4: To run the Flutter command in regular windows console, you need to update the system path to include the flutter bin directory. The following steps are required to do this:

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



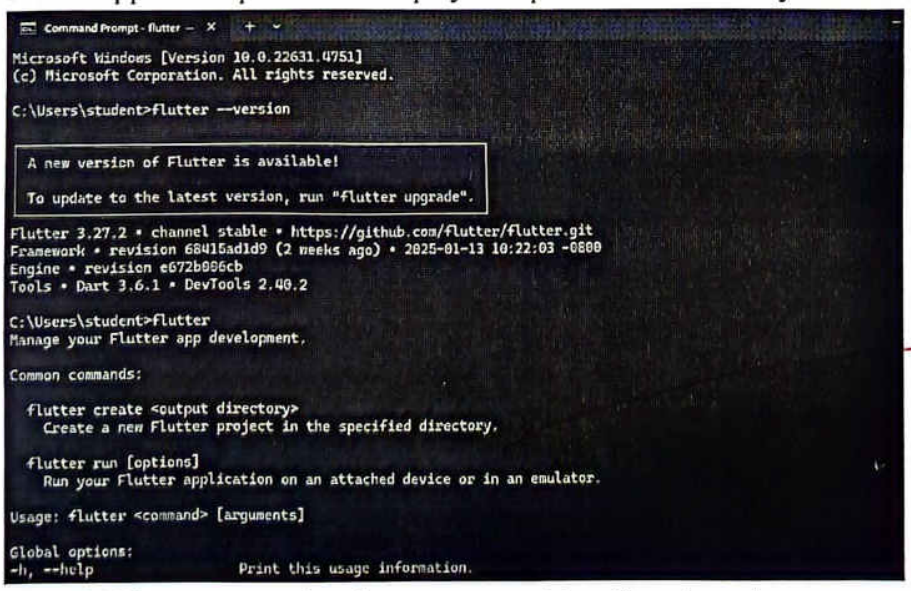
Step 4.2: Now, select path -> click on edit. The following screen appears

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.

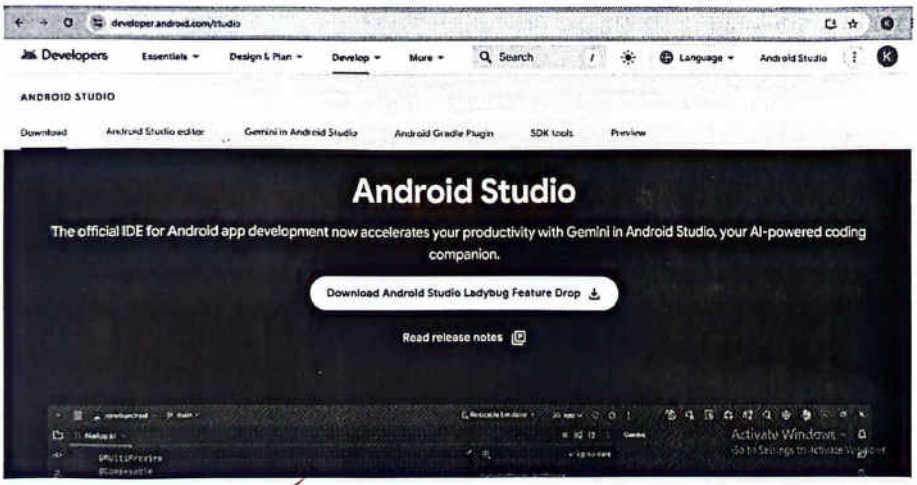


Step 6: When you run the above command, it will analyze the system and show its report, as shown in the below image. Here, you will find the details of all missing tools, which required to run Flutter as well as the development tools that are available but not connected with the device.

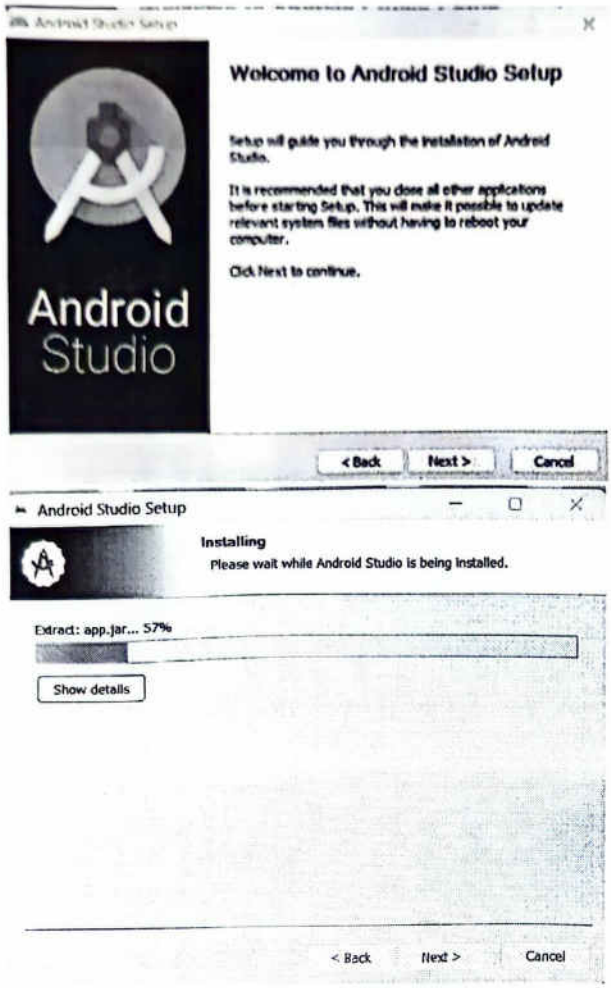
```
C:\Users\student>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.27.3, on Microsoft Windows [Version 10.0.22631.4751], locale en-IN)
[√] Windows Version (Installed version of Windows is version 10 or higher)
[!] Android toolchain - develop for Android devices (Android SDK version 34.0.0)
    X cmdline-tools component is missing
      Run 'path/to/sdkmanager --install "cmdline-tools;latest"'
      See https://developer.android.com/studio/command-line for more details.
    X Android license status unknown.
      Run 'flutter doctor --android-licenses' to accept the SDK licenses.
      See https://flutter.dev/to/windows-android-setup for more details.
[√] 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 all of its default components
[√] Android Studio (version 2023.1)
[√] VS Code (version 1.96.4)
[√] Connected device (3 available)
[√] Network resources

! Doctor found issues in 2 categories.

C:\Users\student>
```

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.

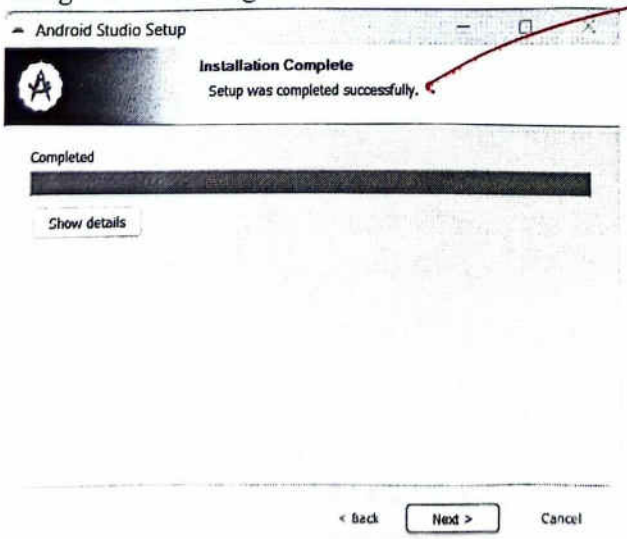
Step 7.1: Download the latest Android Studio executable or zip file from the official site.



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



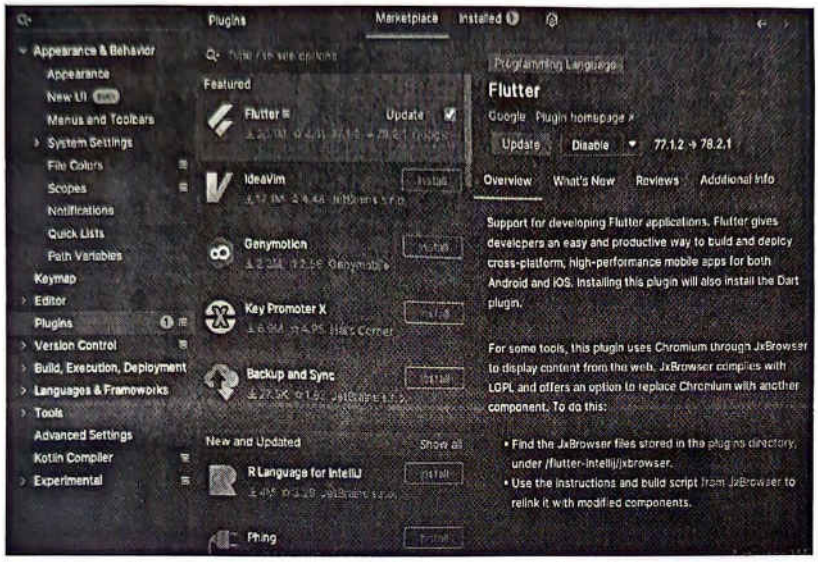
Step 7.3: Follow the steps of the installation wizard. Once the installation wizard completes, you will get the following screen.



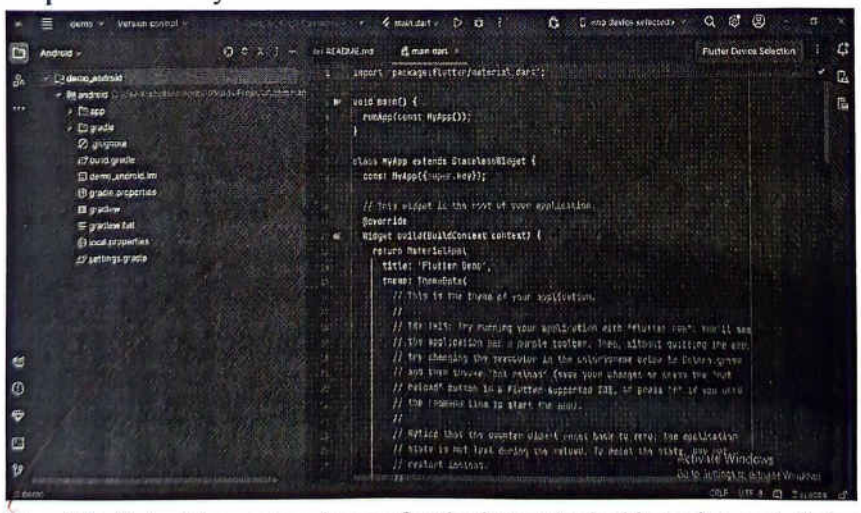
Step 7.4: In the above screen, click Next-> Finish. Once the Finish button is clicked, you need to choose the 'Don't import Settings option' and click OK. It will start the Android Studio.

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.

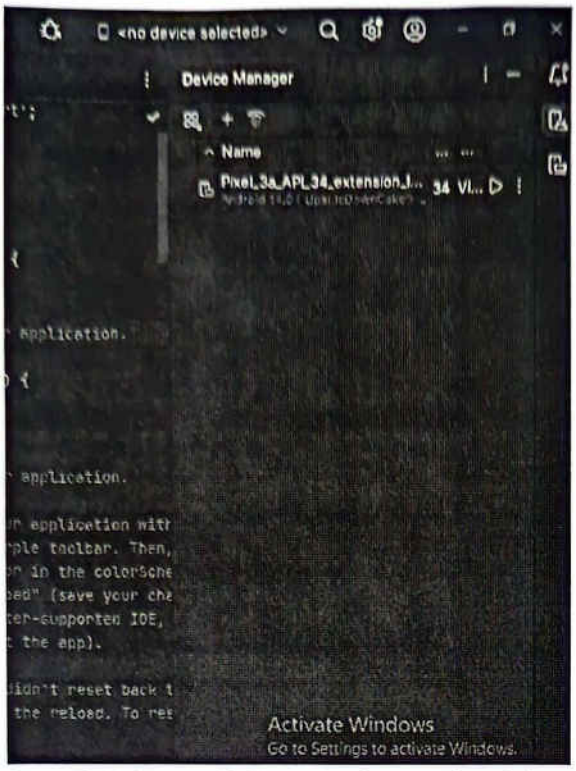


Step 8.2: Choose your device definition and click on Next.

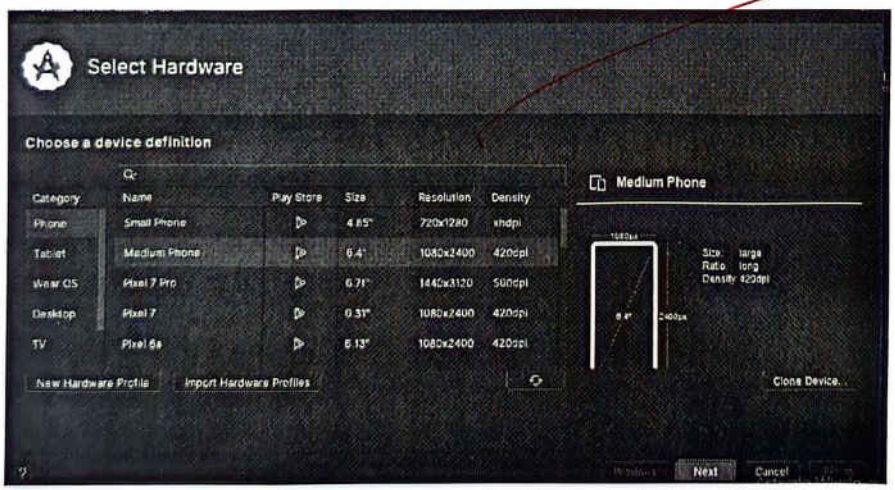


Step 8.3: Select the system image for the latest Android version and click on Next.

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

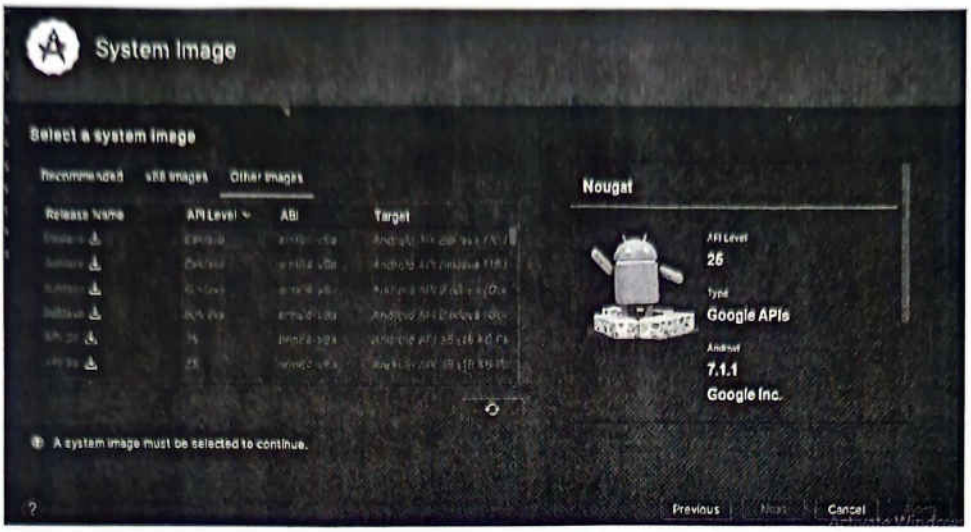


Step 8.5: Last, click on the icon pointed into the red color rectangle. The Android emulator



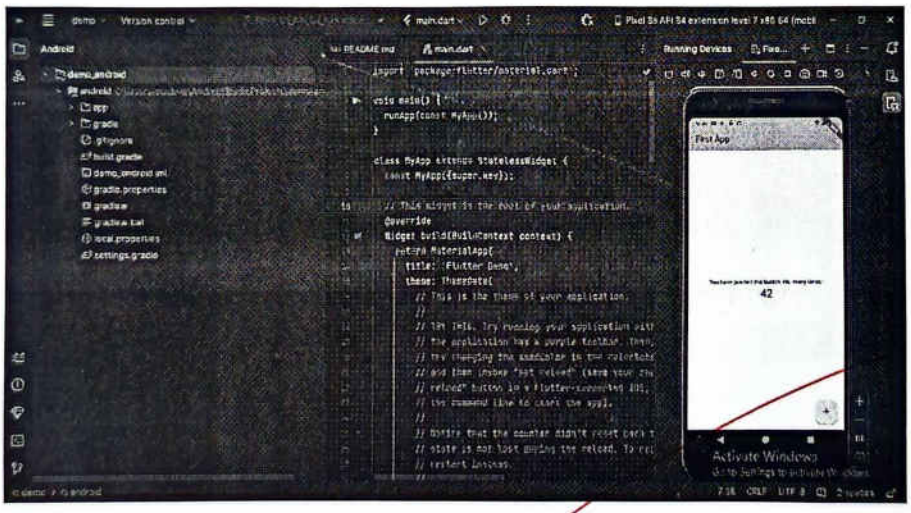
Step 9: Now, install Flutter and Dart plugin for building Flutter application in Android Studio. These plugins provide a template to create a Flutter application, give an option to run and debug Flutter application in the Android Studio itself. Do the following steps to install these plugins.

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.



MAD & PWA Lab
Journal

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	
Name	
Class	D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Name : Harsh Jadhav
Roll No : 21
Class : D15B

MPL Experiment No. 2

* **Aim:-** To design Flutter UI by including common widgets.

* **Theory:**

Flutter is an open-source UI toolkit by Google that allow developers to build natively compiled applications for mobile, web and desktop using a single codebase. It is based on the Dart programming language & focuses a widget-based architecture where everything in flutter is a widget.

- **Common Widgets in Flutter:**

1. **Scaffold:** provides a basic structure for the app, including an app bar, body, floating action button
2. **AppBar:** Displays the title and actions in the top navigation bar.
3. **Text:** used to display textual content in the app.
4. **TextField:** Accepts user input in forms or search bars.
5. **Container:** A versatile widget used for styling elements with padding, margins, borders & background colors.

⑥ ElevatedButton: A button with elevation for user interaction.

⑧ Row & Column: used to arrange widgets horizontally or vertically.

* Conclusion:-

I have completed this experiment successfully and learned to design flutter UI including common widgets such as, textfields, text, AppBar.

Forgotten password

Provide your email and we will send you a link to reset your password

Email

hjadhav866@gmail.com

Reset password

Go back

New Match

Team 1

Team 2

Toss Winner

Bat Field

Overs (Max 50)

Start Match

MAD & PWA Lab
Journal

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	
Name	
Class	D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Name : Harsh Jadhav
Roll No : 21
Class : D15B

MPL Experiment No. 3

* **Aim:** To include Icons, Images and Fonts in a Flutter App.

* **Theory :**
Flutter provides built-in support for adding Icons, images and custom fonts to enhance UI design and user experience.

1. **Adding Icons :**
Flutter has a built-in Icons class that provides various Material Design Icons. These can be used as follows.

```
Icon ( Icons.home, size:50, color: Colors.blue ).
```

2. **Adding Images :**
- Asset Images : Add images to the assets folder and declare them in pubspec.yaml.
  Image.asset ('assets /images /sample.png')
- Network Images : Directly load images from a URL:
  Image.network ('https: ').

3. Adding Fonts :

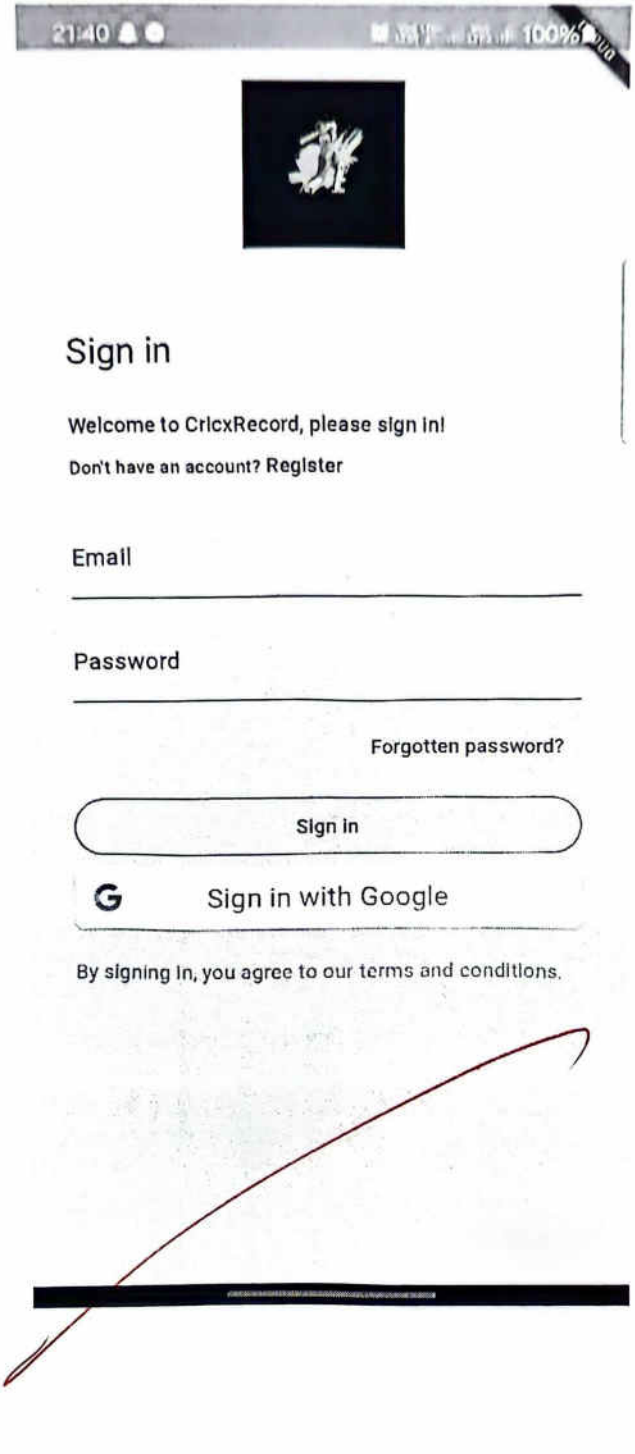
To add custom fonts.

- add font files to the assets/fonts folder.
- Declare them in pubspec.yaml.

Text ('custom font', style:
TextStyle (fontFamily : 'Roboto')).

* conclusion :-

I have implemented this experiment sucessfully and including images, icons and fonts in a flutter app named Tic x Record.


21:40
100%
Sign in
Welcome to CricxRecord, please sign in!
Don't have an account? Register
Email
Password
Forgotten password?
Sign in
Sign in with Google
By signing in, you agree to our terms and conditions.

MAD & PWA Lab
Journal

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget
Roll No.	
Name	
Class	D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Name : Harsh Jadhav
Roll No : 21
Class : D15B

MPL Experiment No. 4

* **Aim :-** To create an interactive Form using form widget.

* **Theory :-**

Forms in flutter are used to collect user input, such as login credentials, registration details or feedback. The form widget allows validation, error handling and submission of input fields.

Each input fields in a form is typically managed using a TextFormField widget.

- Key components of a form in Flutter :-

1. **Form Widget :** A container for multiple form field that supports validation.
2. **TextFormField Widgets :** used for text input fields
3. **Global Key :** Helps manage form state.
4. **Validators** - ensures valid input
5. **Submit Button :** Triggers form validation and submission.

* **Conclusion :-**

* Conclusion :-

I have completed this experiment sucessfully and learned about how to implement and create an interactive Form using Form widget.

Register

Welcome to CricxRecord, please sign up!

Already have an account? Sign in

Email

Password

Confirm password

Register

Sign in with Google

By signing in, you agree to our terms and conditions.

Sign in

Welcome to CricxRecord, please sign in!

Don't have an account? Register

Email

Password

Forgotten password?

Sign in

Sign in with Google

By signing in, you agree to our terms and conditions.

MAD & PWA Lab
Journal

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App
Roll No.	
Name	
Class	D15B
Subject	MAD & PWA Lab
Lab Outcome	LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation
Grade:	

Name : Harsh Jadhav
Roll No: 21
Class : D15B

MPL Experiment No. 5

* **Aim:-** To apply navigation, routing and gestures in Flutter App.

* **Theory:-**

Navigation and routing in flutter allows switching between different screens (pages) within an app. Gestures provide interactivity, enabling user actions like taps, swipes and long presses.

1. **Navigation and Routing in flutter:**

Flutter provides two main approaches for navigation:

1. Direct Navigation using Navigator.push() and Navigator.pop().

2. Named Routes using MaterialApp's routes property.

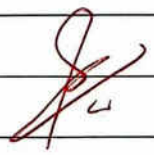
2. **Implementing Gestures in flutter:**

Flutter uses the gesture detector widget to handle touch interactions like tap, double-tap, long press and swipe.

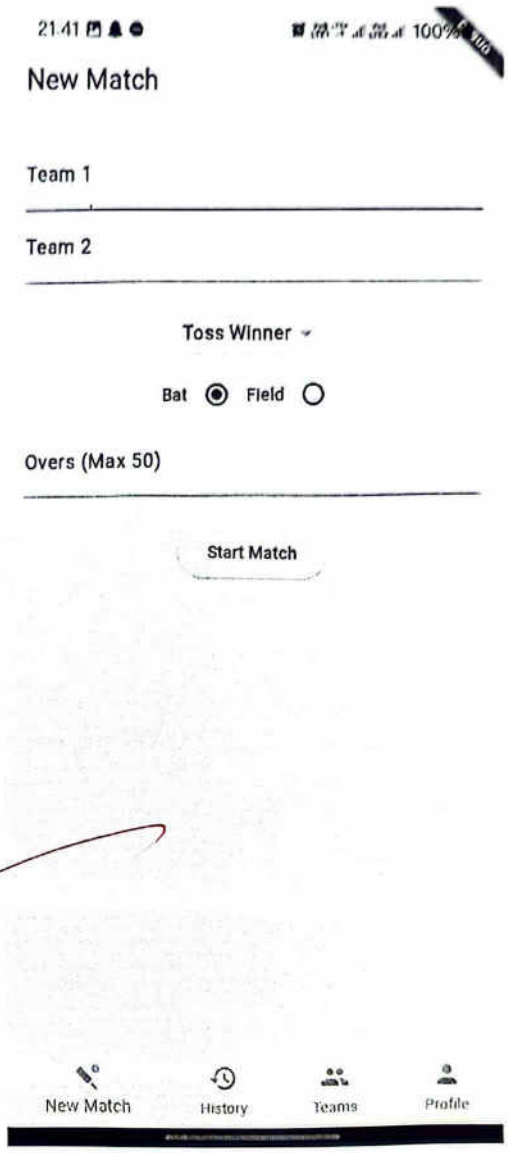
A Gesture Detector is used to detect taps and navigate to another screen.

* **Conclusion:-**

I have completed this experiment successfully and learned how to apply navigation, Routing and Gestures in a flutter App.




21:41
100%
Match History
No match history available
New Match
History
Teams
Profile


21:41
100%
New Match
Team 1
Team 2
Toss Winner
Bat
Field
Overs (Max 50)
Start Match
New Match
History
Teams
Profile

MAD & PWA Lab
Journal

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	
Name	
Class	D15B
Subject	MAD & PWA Lab
Lab Outcome	LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	

A++

Name : Harsh Jadhav
Roll No : 21
Class : D15B

MPL Experiment No. 6

* **Aim:-** How to setup firebase with flutter for iOS and Android Apps.

* **Theory:**

Firebase is a cloud-based backend service that provides authentication, real time database, cloud storage and analytics.. from mobile and web applications.

Steps to setup firebase in a flutter App:-

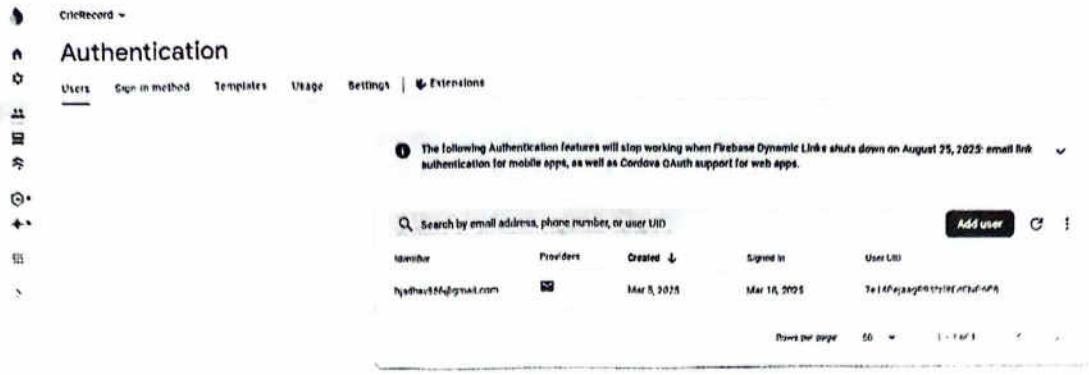
1. Create a Firebase Project:-
   - go to firebase console and create a project.
   - Enable service you want based on app's requirements.
2. Add Firebase to flutter App:-
   - Register the Android App in console.
   - Download json files for android.
3. Install Fire Base dependencies:
   - Firebase_core, firebase_auth, cloud_firebase etc.

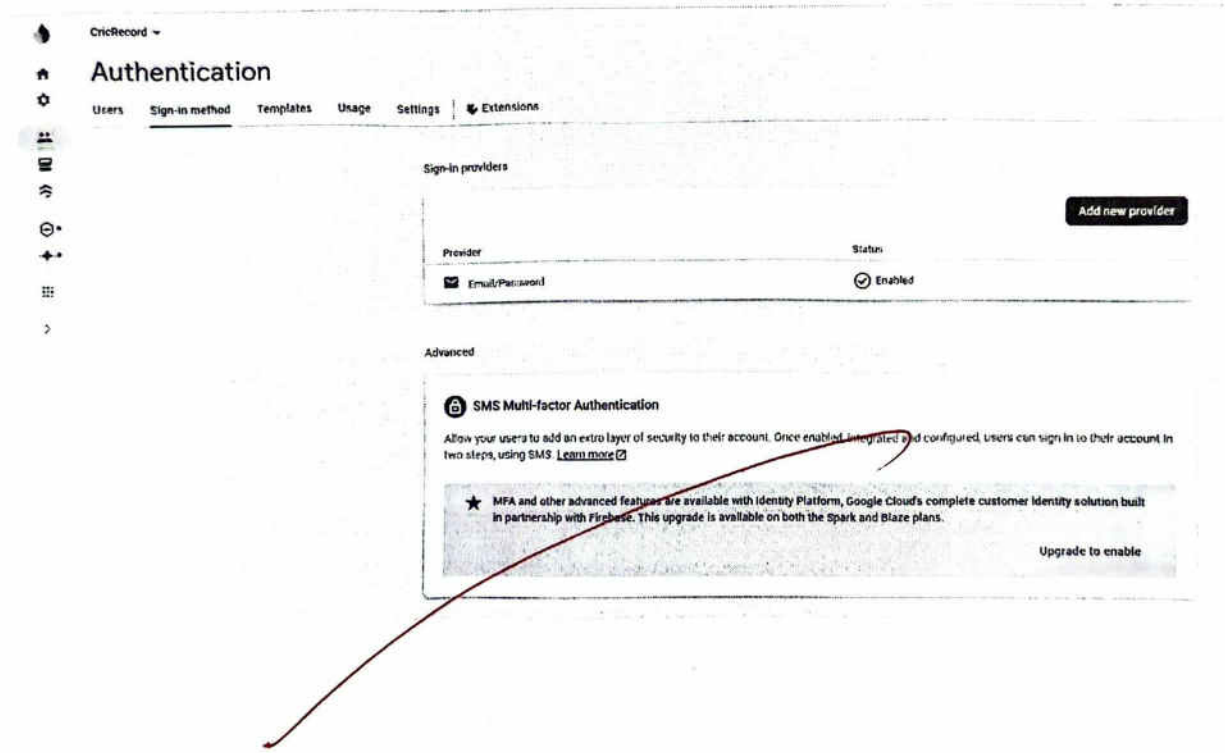
4. configure Android for Firebase:-
   - update gradle file to include firebase service.

5. ~~configure~~ initialize Firebase in Flutter:
   Modify the main.dart file to initialize Firebase in the app.

6. Run and verify the integration:-
   - Launch the Flutter app on Android using : flutter run,
   - verify service like authentication, firestore or analytics, through the firebase console.

conclusion:-
I have successfully completed this experiment and understood and learned about how to implement firebase in our flutter app and I have implemented it in my application successfully.


CricRecord
Authentication
Users
Sign-in method
Templates
Usage
Settings
Extensions
The following Authentication features will stop working when Firebase Dynamic Links shuts down on August 25, 2025: email link authentication for mobile apps, as well as Cordova OAuth support for web apps.
Search by email address, phone number, or user UID
Add user
Identifier
Providers
Created
Signed In
User UID
Mar 8, 2025
Mar 16, 2025
Rows per page
50


CricRecord
Authentication
Users
Sign-in method
Templates
Usage
Settings
Extensions
Sign-in providers
Add new provider
Provider
Status
Email/Password
Enabled
Advanced
SMS Multi-factor Authentication
Allow your users to add an extra layer of security to their account. Once enabled, integrated and configured, users can sign in to their account in two steps, using SMS. Learn more
MFA and other advanced features are available with Identity Platform, Google Cloud's complete customer identity solution built in partnership with Firebase. This upgrade is available on both the Spark and Blaze plans.
Upgrade to enable


21:41
100%
Profile
hdgh
hjadhav866@gmail.com
Logout
New Match
History
Teams
Profile

MAD & PWA Lab
Journal

Experiment No.	07
Experiment Title.	To write meta data of your Ecommerce PWA in a Web app manifest file to enable “add to homescreen feature”.
Roll No.	
Name	
Class	D15B
Subject	MAD & PWA Lab
Lab Outcome	LO4: Understand various PWA frameworks and their requirements
Grade:	

Name: Harsh Jadhav

Roll No: 21

Div : D15B

MPL Experiment 7:

* Aim: Implement add to home screen feature.

* Theory:

PWA are designed to offer users a native-like experience while still being based on web technologies one of key features of PWA's is the ability to be added to the home screen of website devices providing a seamless and engaging user experience.

o Key components of 'Add to Home screen' feature.

1) Web App Manifest: The manifest file is a crucial part of PWA that provides metadata about App, such as its name, icon, theme colors and orientation prefrences.

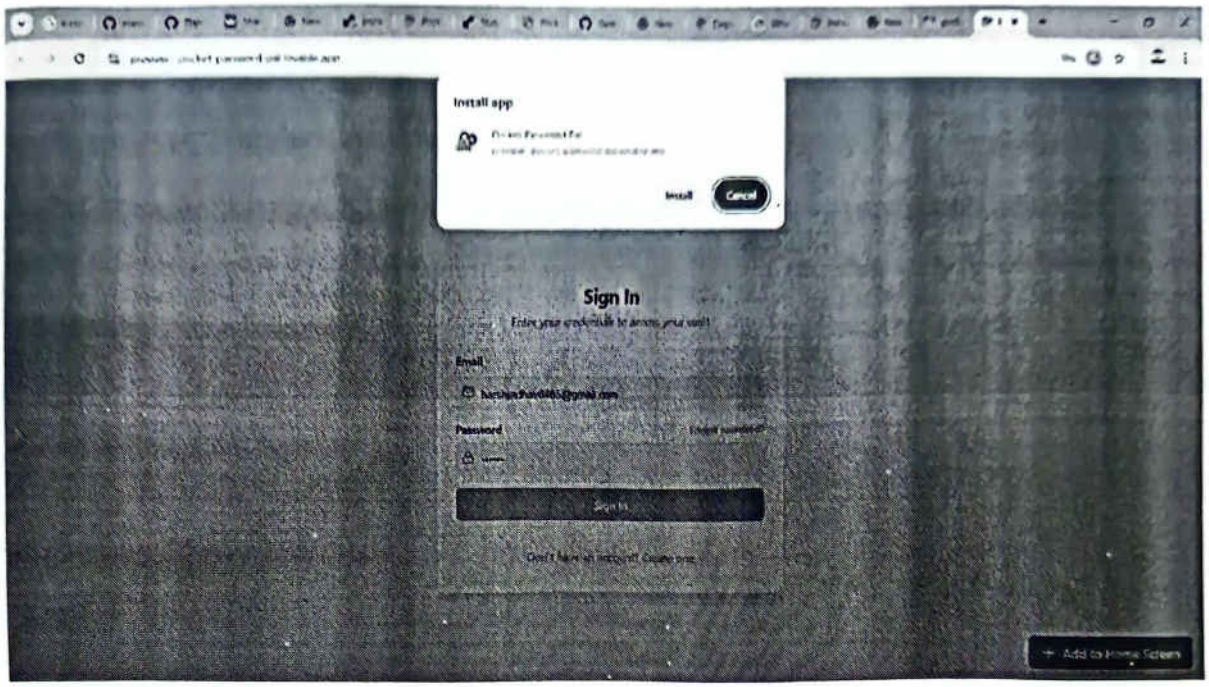
2) Service workers & Cache workers play an essential role in ensuring the app works offline, caching content for reliable use even when user is not connected to internet. This is fundamental for the PWA experience and supports the 'Add to Home screen feature'.

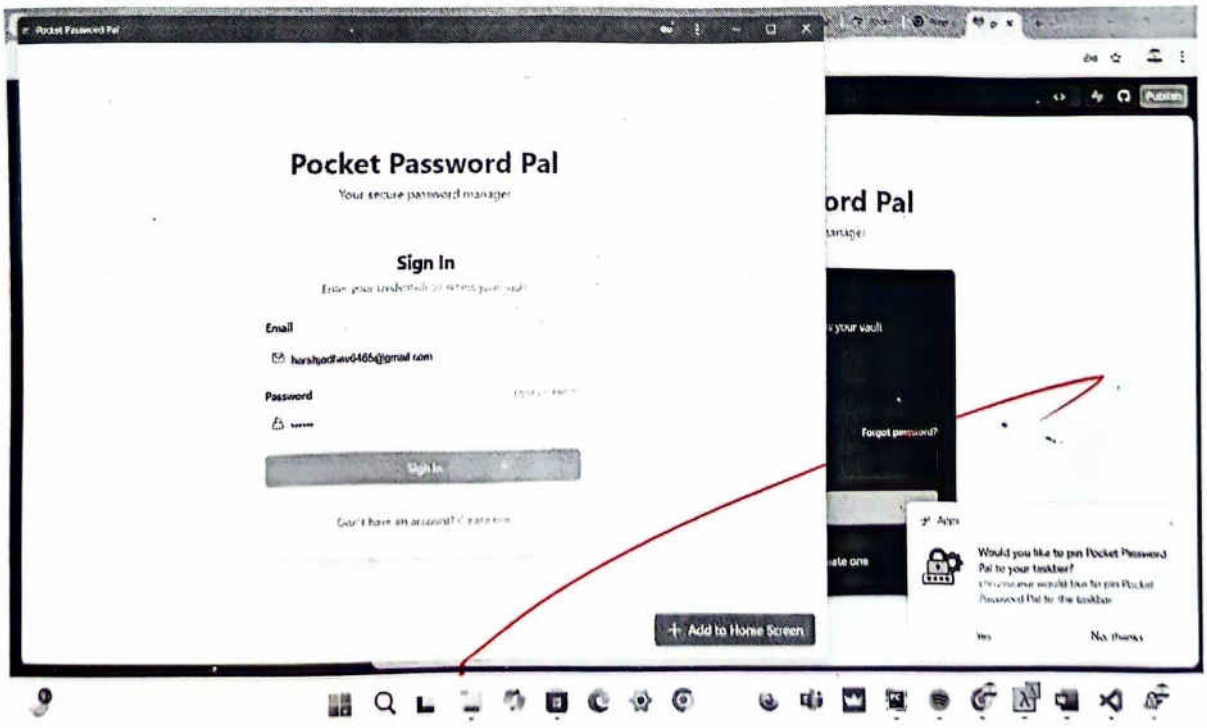
3) User engagement: For users to add a PWA to the home screen, they must be prompted with an installation option. This is often handled by a combination of browser built in prompts and a custom installation flow created by app developers.

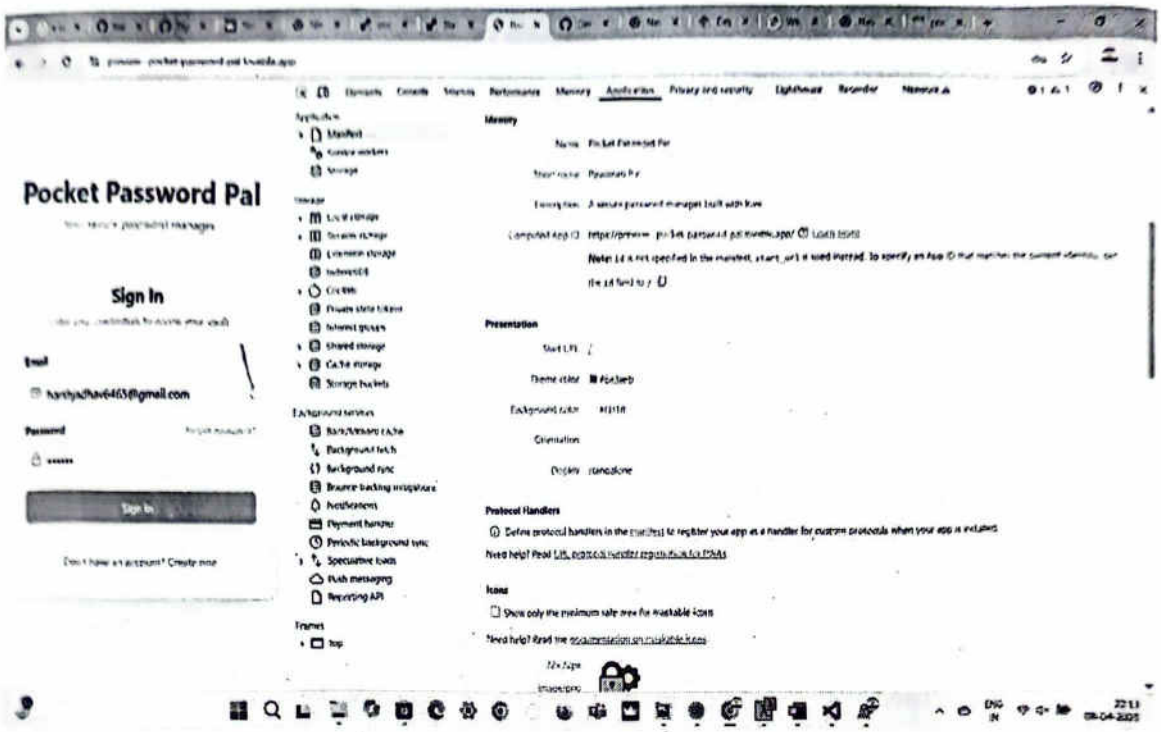
4) Browser support:- The ability to "Add to Home Screen" varies across browsers and platforms, but popular browsers like chrome, edge and safari support the features.

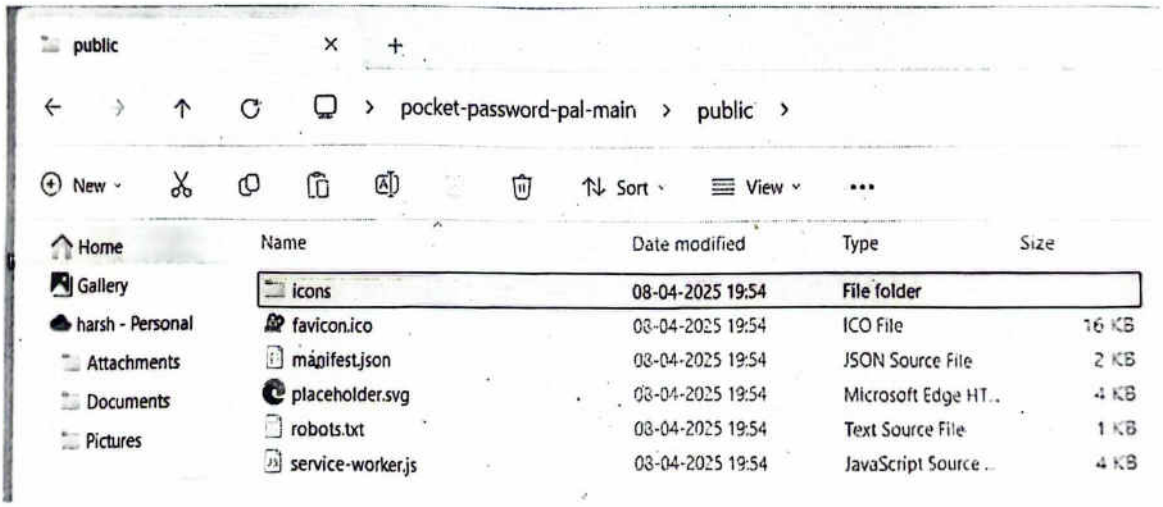
* Conclusion:

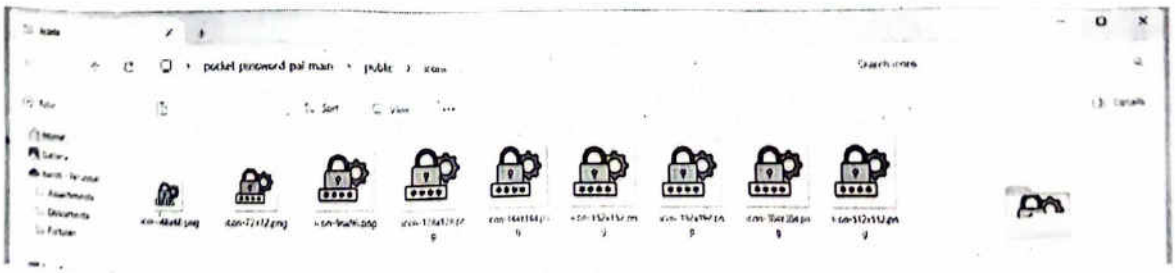
The add to home screen feature offers multiple advantages for both developers and users. I have implemented the "add to Home Screen feature".


Install app
Install
Cancel
Sign In
Email
Password
Sign In
Add to Home Screen


Pocket Password Pal
Your secure password manager
Sign In
Email
Password
Sign In
Add to Home Screen
Publish
Would you like to pin Pocket Password Pal to your taskbar?


Pocket Password Pal
Sign In
Email
Password
Sign In
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 mitigations
Notifications
Payment handler
Periodic background sync
Speculative loads
Push messaging
Reporting API
Frames
top
Memory
Presentation
Protocol Handlers
Icons


public
pocket-password-pal-main
public
New
Sort
View
Home
Gallery
harsh - Personal
Attachments
Documents
Pictures
Name
Date modified
Type
Size
icons
08-04-2025 19:54
File folder
favicon.ico
08-04-2025 19:54
ICO File
16 KB
manifest.json
08-04-2025 19:54
JSON Source File
2 KB
placeholder.svg
08-04-2025 19:54
Microsoft Edge HT..
4 KB
robots.txt
08-04-2025 19:54
Text Source File
1 KB
service-worker.js
08-04-2025 19:54
JavaScript Source ..
4 KB



DevTools - localhost:8081/

Welcome | Elements | Console | Sources | Network | Performance | Memory | Application | Lighthouse | +

Application
- Manifest
- Service w...
- Storage

Storage
- Local stor...
- Session st...
- Extension...
- IndexedDB
- Cookies
- Private st...
- Interest g...
- Shared st...
- Cache sto...
- Storage b...

Background ser...
- Back/forw...
- Backgrou...
- Backgrou...
- Bounce tr...
- Notificati...
- Payment ...
- Periodic b...
- Speculati...

Icons

☐ Show only the minimum safe area for maskable icons

Need help? Read the App Image Generator.

72×72px
image/png

96×96px
image/png

128×128px
image/png

144×144px
image/png

MAD & PWA Lab
Journal

Experiment No.	08
Experiment Title.	To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA
Roll No.	
Name	
Class	D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

Name: Harsh Jadhav

Roll No: 21

Div: D15B

MPL Experiment No. 8.

* **Aim:** To code and register a service worker and complete the instalation & activation process for a new service worker.

* **Theory:-**

A service worker is a script that is run in the background of a web page, seperate from the main thread and can manage events like fetch requests, caching, background, sync and push notifications.

* Steps for Registering a service worker and completing the instal & activation process.

STEPS:-

① **Service worker Registration:**

⇒ This is done through the javascript navigator service worker.register() method.

⇒ The service worker registration must occur often the page is loaded and only HTTPS connections

② **Install event:**

MAD & PWA Lab
Journal

Experiment No.	09
Experiment Title.	To implement Service worker events like fetch, sync and push for E-commerce PWA
Roll No.	
Name	
Class	D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

Name : Harsh Jodhav
Roll No : 21
Div : D15B

MPL Experiment No 9.

* Aim :- To implement service worker event like fetch sync for my app.

* Theory :-
Service worker is a script that works on browser background without user interaction independently. Also it reassembles a proxy that works on user guide

- Things to note about service worker :-

- A service worker is a ~~requirem~~ programmable network proxy that lets you control how network requests from your page are handled.
- Service workers only over HTTPS. Because service worker can intercept network requests and modify responses " man in the middle attack could be very bad.

* Fetch Event :-
You can trace and manage page through traffic with this event. You can check existing cache memory cache, manage "cache first" request and restore.

a response that you want.

→ Sync Event :-

Background sync is a web api that is used to delay a process until the internet connection is stable. We can adapt this definition to real life. There is an browser and we want to send an email with this tool.

→ Push Event :-

This is the event that handles push notifications are received from server, you can apply any method with received data.

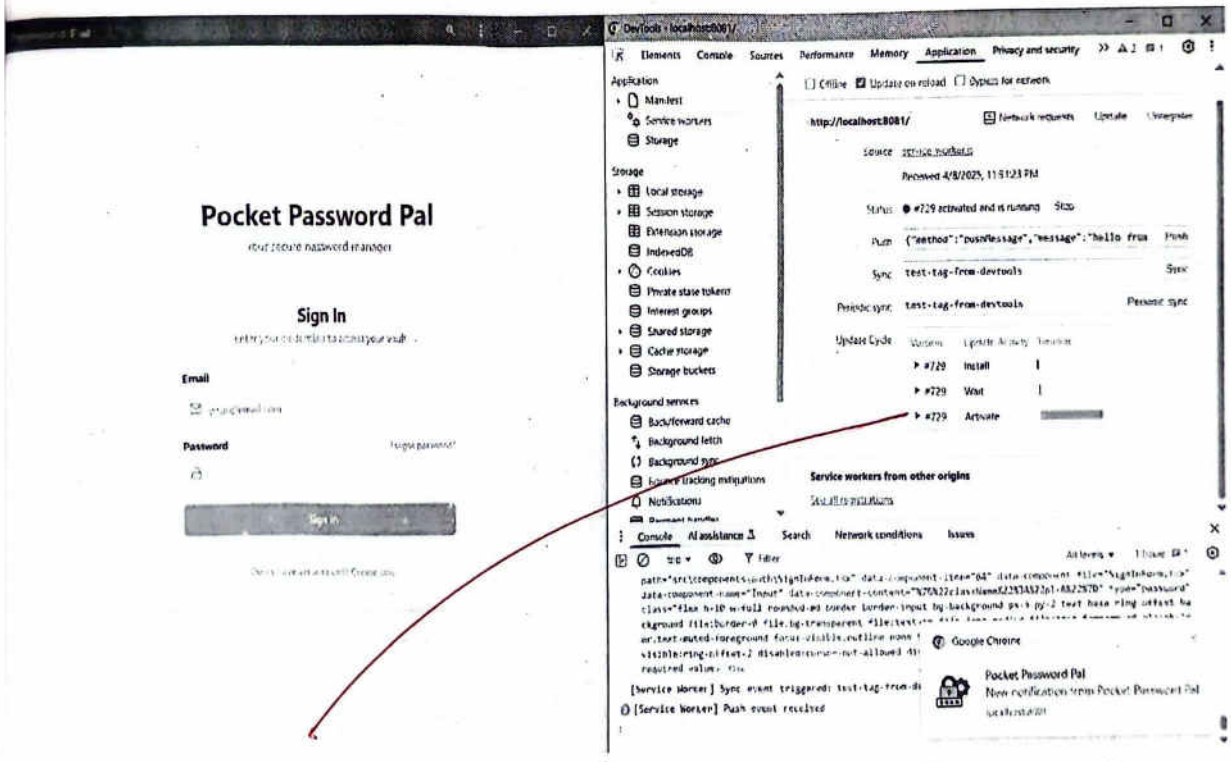
We can check it by using following methods:
Notification.requestPermission().

* Conclusion:

The implementation of fetch, sync push events in a PWA service worker significantly enhances the functionality performance and user experience of modern web application.

Push Event:

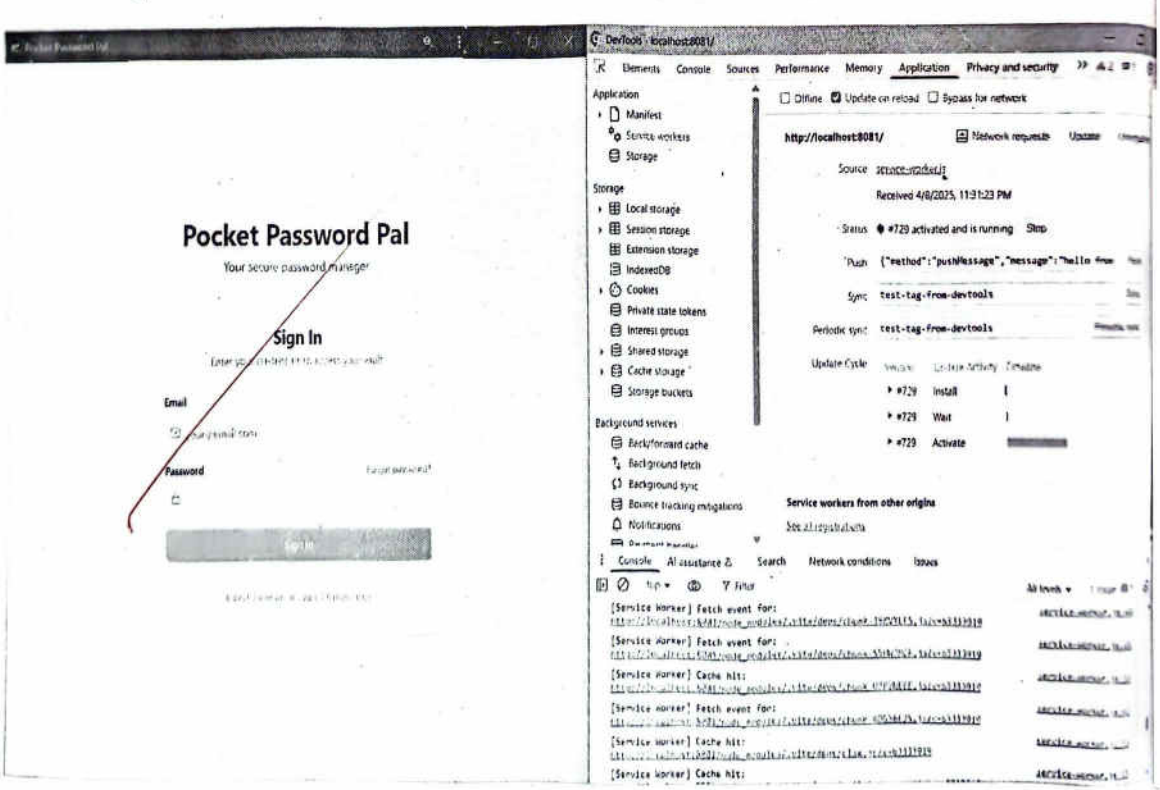
```
// Push notification event
self.addEventListener('push', event => {
  console.log('[Service Worker] Push event received');
  const data = event.data ? event.data.json() : {};

  const options = {
    body: data.body || 'New notification from Pocket Password Pal',
    icon: '/icons/icon-192x192.png',
    badge: '/icons/icon-72x72.png',
    data: {
      url: data.url || '/'
    }
  };

  event.waitUntil(
    self.registration.showNotification(data.title || 'Pocket Password Pal', options)
  );
});
```



Fetch Event:

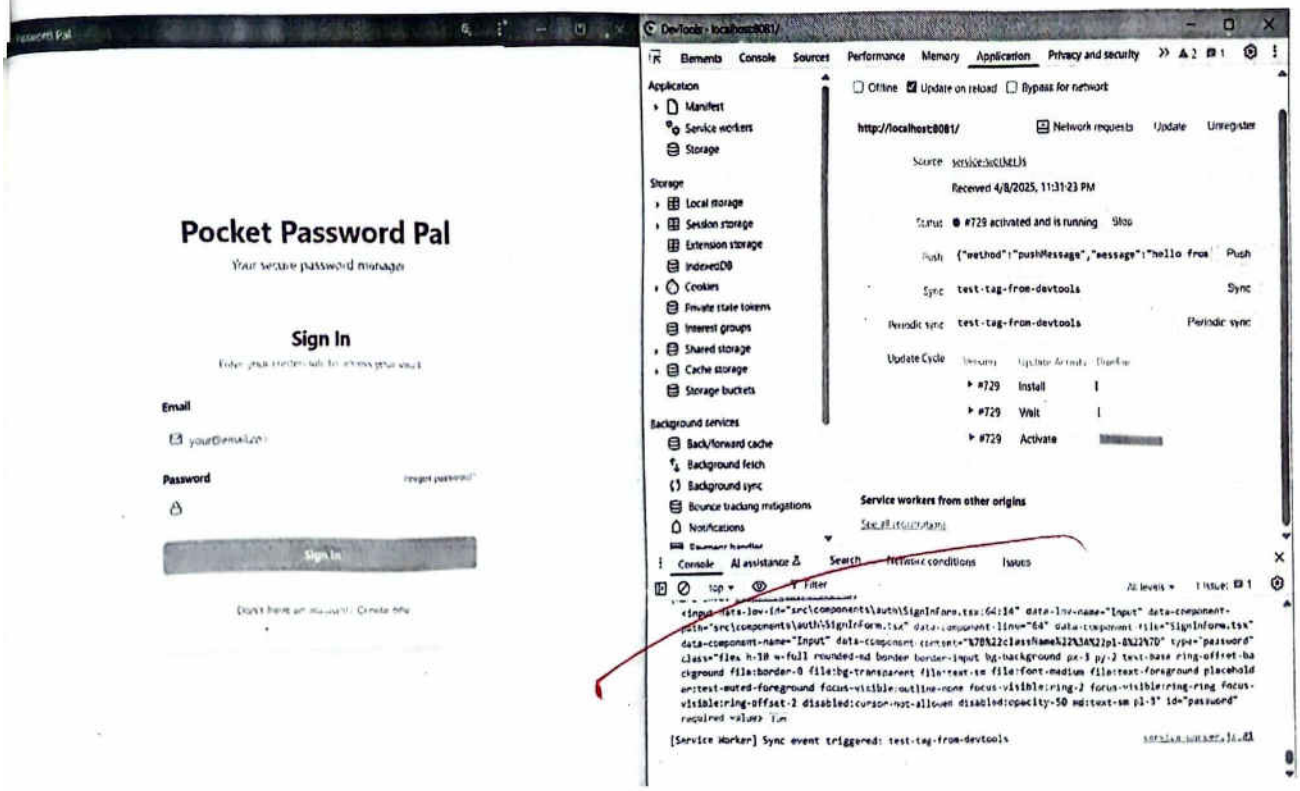
```
// Fetch event - respond with cache first, then network
self.addEventListener('fetch', event => {
  console.log(`[Service Worker] Fetch event for: ${event.request.url}`);

  if (event.request.url.includes('supabase.co')) {
    console.log('[Service Worker] Skipping cache for Supabase API call');
```

```
  return;
 }
 event.respondWith(
  caches.match(event.request)
   .then(response => {
    if (response) {
     console.log('[Service Worker] Cache hit:', event.request.url);
     return response;
    }
    console.log('[Service Worker] Cache miss, fetching:', event.request.url);
    return fetch(event.request).then(response => {
     if (!response || response.status !== 200 || response.type !== 'basic') {
      console.log('[Service Worker] Invalid response, not caching');
      return response;
     }
     const responseToCache = response.clone();
     caches.open(CACHE_NAME)
      .then(cache => {
       cache.put(event.request, responseToCache);
       console.log('[Service Worker] Cached new response:', event.request.url);
      });
     return response;
    });
   })
 );
});
```



Sync Event:

```
// Background sync event - for offline data syncing
self.addEventListener('sync', event => {
  console.log(`[Service Worker] Sync event triggered: ${event.tag}`);
  if (event.tag === 'sync-passwords') {
    event.waitUntil(syncPasswords());
  }
});

// Function to sync passwords from IndexedDB to Supabase when online
async function syncPasswords() {
  console.log('[Service Worker] Syncing passwords from offline storage');
  // Your IndexedDB + Supabase logic goes here
}
```



MAD & PWA Lab
Journal

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	
Name	
Class	D15B
Subject	MAD & PWA Lab
Lab Outcome	LO5: Design and Develop a responsive User Interface by applying PWA Design techniques
Grade:	

Name: Harsh Jodhav.
Roll No: 21
Div: D15B

MPL Experiment 10

* Aim:- To study and implement deployment of my password manager App to github pages.

* Theory:

Public Web pages are fully hosted and easily published. public webpages hosted directly from your github repository. Just edit, push your changes are fine.

Github Pages provides the following key features:-
1) Blogging with Jekyll
2) Custom URL
3) Automatic Page generator.

Reasons for favouring this over firebase:
1) Free to use
2) Right out of github.
3) Quick to set up.

firebase:
The Realtime App platform firebase is a cloud service designed to provide real time collaboration

application. Firebase cloud and write the data within milliseconds.

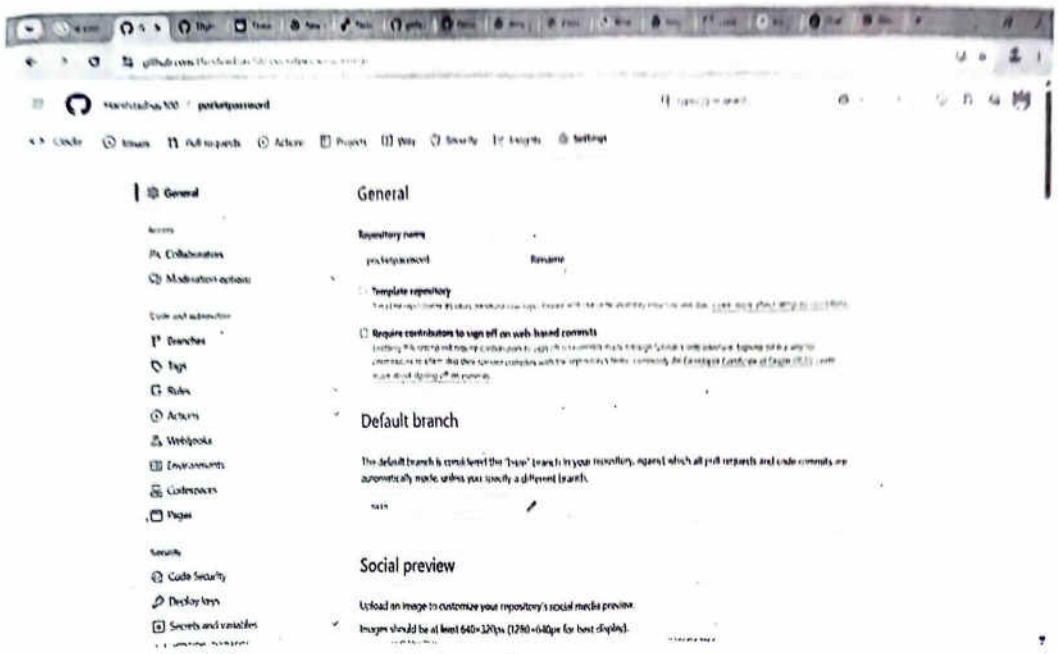
* Reasons for favouring github page.

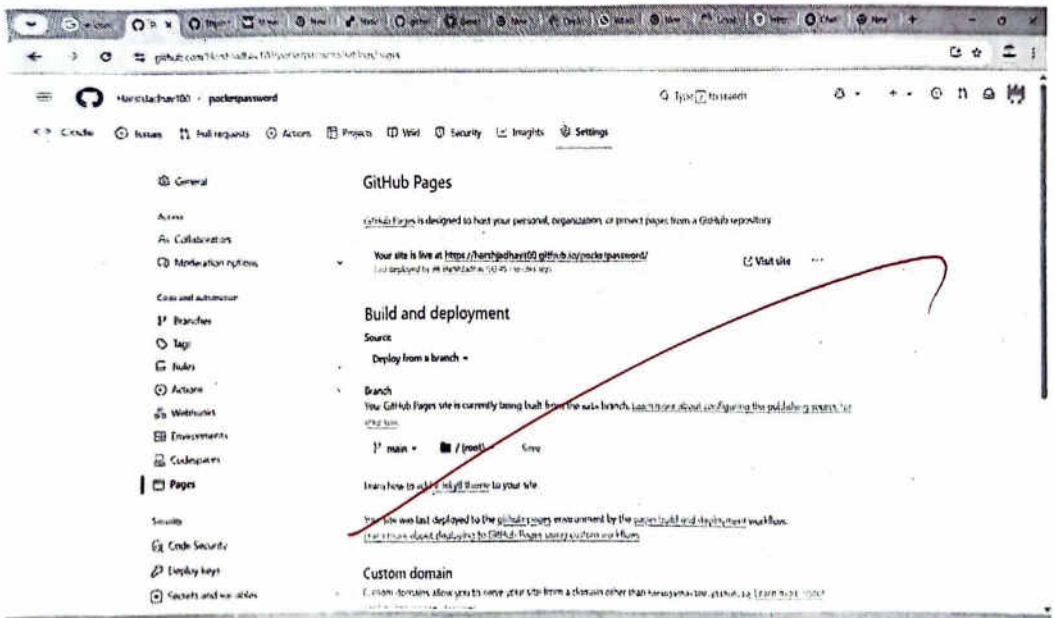
1) Realtime backend made easy.
2) Fast and responsive.
3) Quick to setup.

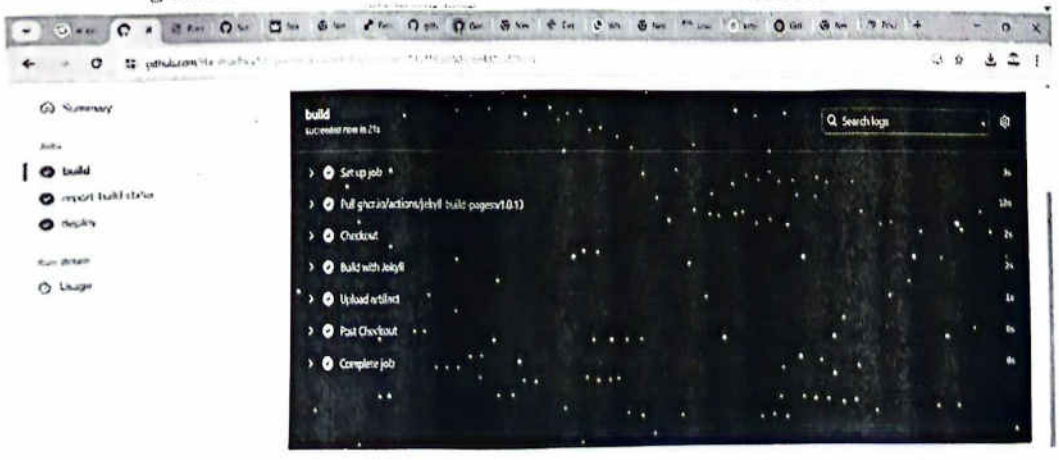
Firebase is used by many companies such as Twitch, Google and twitter are ~~too~~ popular companies that uses firebase.

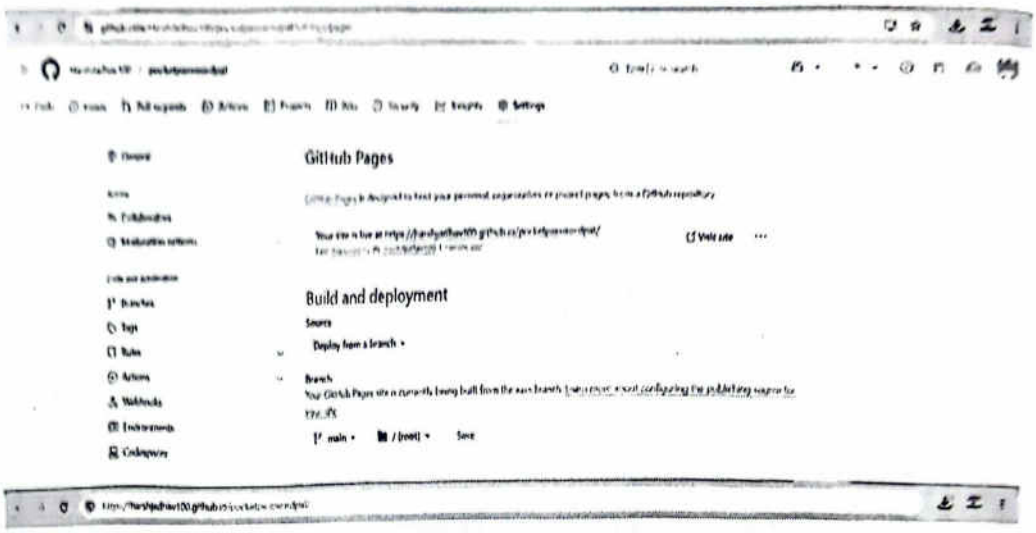
* conclusion:-

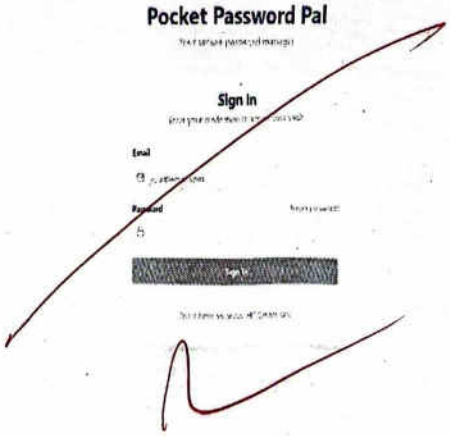
Deploying a web application using Github provides a simple, cost effective and fast way to host static websites and PWAs directly from a github repository.


General
Default branch
Social preview


GitHub Pages
Build and deployment
Custom domain


build
Search logs
Set up job
Checkout
Build with Jekyll
Upload artifact
Post Checkout
Complete job


Github Pages
Build and deployment
Source
Deploy from a branch
Branch
main
/ (root)
Save


Pocket Password Pal
Sign in
Email
Password

MAD & PWA Lab
Journal

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	
Name	
Class	D15B
Subject	MAD & PWA Lab
Lab Outcome	LO6: Develop and Analyze PWA Features and deploy it over app hosting solution
Grade:	

Name: Harsh Jadhav
Roll No: 21
Class: D15B

MPL Experiment 11

Aim: To use google lighthouse tool to test PWA functioning.

Theory:
Google lighthouse is a tool that test your web application based on number of parameters including performance, based on a number of metrics, mobile compatibility, progressive Web App implementations. All you have to do is run it on a page or pass a url to it.

Key features and Audit Metrics
Performance: This score is a aggregation of various page based in aspect of loading speed, time taken for loading basic frame, displaying meaningful content to user.

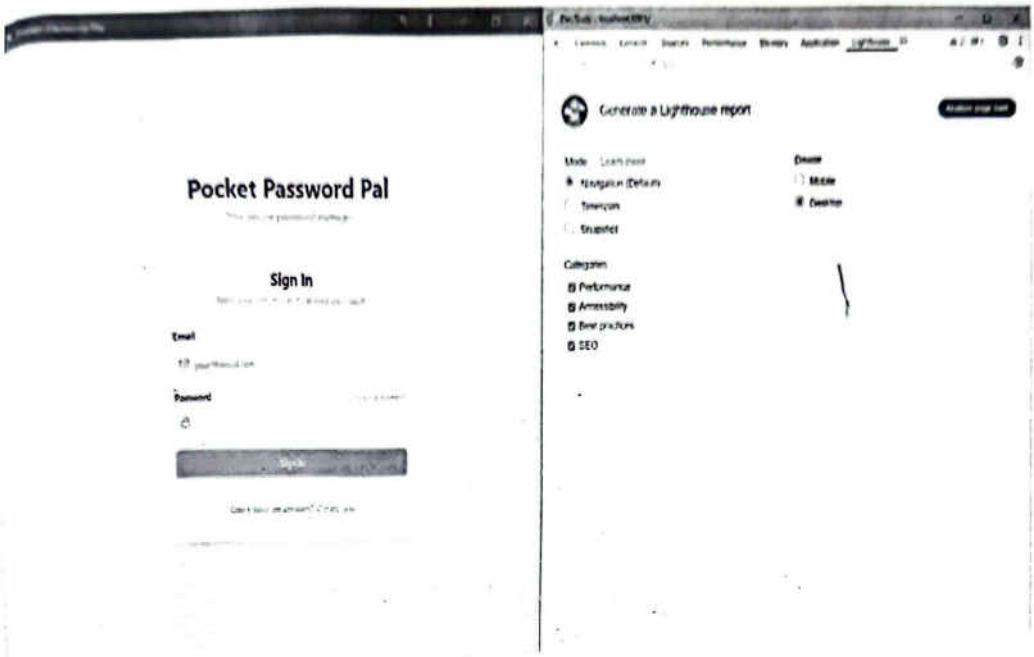
PWA score(Mobile): Thanks to the rise of service workers app manifests etc, a lot of modern Web apps moving towards PWA paradigm.

3) accessibility :- As you might have guessed, this is a measure of how accessible your website across a plethora of accessibility features implemented in your page.

4) Best practices :- As you might have guessed, this number of practices that have been deemed based on empirical data. These metrics aggregation of many such points, including not limited to use of HTTPS.

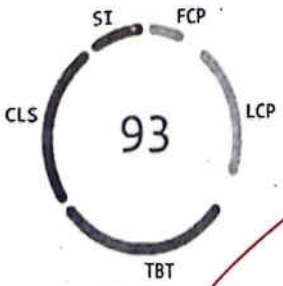
* Conclusion :-

Then we have successfully used google lighthouse PWA analysis tools for testing the PWA functioning. Lighthouse provided all the SEO, PWA, Accessibility functionality score.



https://pocket-password-pal.lovable.app/

93	87	96	91
Performance	Accessibility	Best Practices	SEO



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
2.4 s

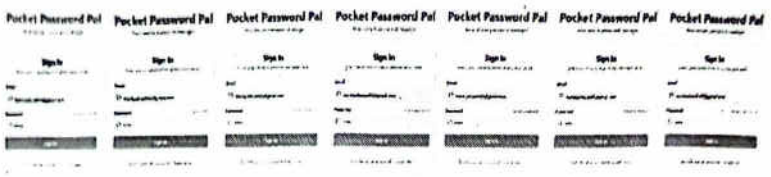
Largest Contentful Paint
2.8 s

Total Blocking Time
0 ms

Cumulative Layout Shift
0

Speed Index
2.4 s

View Treemap



Show audits relevent to: All FCP LCP TBT

DIAGNOSTICS

- ▲ Reduce unused JavaScript — Potential savings of 89 KiB
- ▲ Largest Contentful Paint element — 2,770 ms
- ▲ Page prevented back/forward cache restoration — 1 failure reason
- Serve static assets with an efficient cache policy — 4 resources found
- Initial server response time was short — Root document took 150 ms
- Avoids enormous network payloads — Total size was 212 KiB
- Avoids an excessive DOM size — 39 elements
- Avoid chaining critical requests — 3 chains found
- JavaScript execution time — 0.3 s
- Minimizes main-thread work — 0.8 s
- Minimize third-party usage — Third-party code blocked the main thread for 0 ms
- Avoid long main-thread tasks — 2 long tasks found

More information about the performance of your application. These numbers don't directly affect the Performance score

PASSED AUDITS (26) Show

87

Accessibility

These checks highlight opportunities to improve the accessibility of your web app. Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so manual testing is also encouraged.

96

Best Practices

91

SEO

These checks ensure that your page is following basic search engine optimization advice. There are many additional factors Lighthouse does not score here that may affect your search ranking, including performance on Core Web Vitals. Learn more about Google Search Essentials.

MAD & PWA Lab
Journal

Experiment No.	Assignment-1
Assignment 1 Questions	1. a) Explain the key features and advantages of using Flutter for mobile app development. b)Discuss how the Flutter framework differs from traditional approaches and why it has gained popularity in the developer community. 2. a)Describe the concept of the widget tree in Flutter. Explain how widget composition is used to build complex user interfaces. b) Provide examples of commonly used widgets and their roles in creating a widget tree. 3. a) Discuss the importance of state management in Flutter applications. b) Compare and contrast the different state management approaches available in Flutter, such as setState, Provider, and Riverpod. Provide scenarios where each approach is suitable. 4.a) Firebase Integration in Flutter: Explain the process of integrating Firebase with a Flutter application. b)Discuss the benefits of using Firebase as a backend solution. Highlight the Firebase services commonly used in Flutter development and provide a brief overview of how data synchronization is achieved.
Roll No.	
Name	
Class	D15B
Subject	MAD & PWA Lab
Lab Outcome	LO1: Understand cross platform mobile application development using Flutter framework LO2: Design and Develop interactive Flutter App by using widgets, layouts, gestures and animation LO3: Analyze and Build production ready Flutter App by incorporating backend services and deploying on Android / iOS
Grade:	

05

Name: Harsh Jadhav
Roll No: 21
Class: D15B

MPL Assignment 01

Q1) a) 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 - using dart language and a high-performance rendering engine.
3. Hot Reload - See changes instantly without restarting the app.
4. Rich UI components - comes with customizable UI.
5. Native-like experience - provides high level animations and fast execution.

• Advantages:-

1. Saves Time & Effort: Single codebase for multiple platforms.
2. High speed development: Hot reload feature speeds up coding.
3. Cost effective: Reduces development cost & time.
4. Attractive UI: Provides beautiful and customizable widgets.
5. Reduce performance issues:- The app runs natively without relying on intermediate bridges like in react native reduc logging

b. Discuss how the flutter framework differs from traditional approaches and why it has gained popularity in the dweopement community.

→ • How flutter differs from Traditional approaches:

1. Single codebase: Traditional search methods need sperate code for Android and iOS, but flutter use one code for both
2. Hot Reload:- Traditional app requires full relaun.
3. UI rendering:- Traditional apps use native engine for faster performance.
4. customization:- Traditional UI design depend on specific components but flutter fully depends on customizable widgets.

• Why flutter has gained popularity:-

1. faster dweopement with Hot Reload: Developer can instantly see UI components without restarting the app making the iteration prows much quicker.
2. Cross platform Efficiency:- Business saves time and resources by maintaining a single codebase for multiple platforms.
3. consistant UI compen across devices since flutter does not rely on native components, the UI looks and behaves the same different OS versions.
4. Improved performance:- AOT compilation and direct access to GPO rendering ensure smooth animations and high performance.

Q2) 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 built ~~with~~ ~~it~~ defines a part of the user interface. Flutters 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.

• Widget composition in flutter:-

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

* Benifits of widgets composition:-

1. Reusability: small widgets can be reused in different parts of the app.
2. Maintainability: Breaking UI into smaller widgets makes it easier to debug and update.
3. Performance: flutter efficiently rebuilds only the necessary parts of the widget tree.

b) Provide examples of commonly used widgets and their roles. In creating a widget tree.

→ 1. Structural widgets:

These widgets are the foundation for building the UI

- MaterialApp: The root widget of flutter app that provides essential configuration.
- Scaffold: Provides basic layout structure,
- Container: A versatile widget used for customization of UI

Example:-

```
MaterialApp(
    home: Scaffold(
        appBar: AppBar(title: Text("Flutter Widget Tree")),
        body: Container(
            padding: EdgeInsets.all(16.0),
            child: Text("Hello, Flutter!"),
        ),
    ),
);
```

2. Input & Interaction Widgets:-

- TextField - Accepts text input from users.
- ElevationButton - A button with elevation.
- GestureDetector - detects gestures like taps swipes and long pressed.

Example:-

```
column(children[
    TextField(decoration: InputDecoration(labelText: "Name"
    ElevatedButton(
    onPressed: (){
    print("Button Pressed");
    },
    child("Button Pressed"); },
    child: Text("Submit"); ), }, );
```

3. Display & Styling widgets (Enhance UI Apperance):

- Text - Display text content.
- Image - Shows images from assets, network, or network
- Icon - Displays icons.
- Card - provides a styled container with elevation.
- chip - Represents small pieces of information.
- Divider - Adds a horizontal separator between elements.

Example :-

```
Text("Hello, Flutter", style :
Text Style (font size : 20, color :
colors.blue));
Image.asset ("assets/logo.png");
Icon (Icons.star, size : 40, color :
Colors.amber);
```

Q3) Discuss the importance of state management in flutter application.

⟹ Importance of State Management in Flutter Applications:

State management is important because it controls how the app stores, updates and displays data when the user interact with it.

Need of State Management :-

1. Keeps UI updated - Ensures that the app reflects changes
2. Improves Performance - Updates only necessary parts of the UI instead of reloading everything.
3. Manages complex data : Helps handle user inputs, API data and navigation efficiently

4. Ensures Smooth User Experience :- Keep the app responsive and interactive.

- Types of state in flutter :-
  1. Local State - Managed within a single widget using StateFul widget.
  2. Global State : Shared across multiple screens using provider, Bloc or Redux.

Without proper state management, the app may behave unpredictably or show outdated data.

(b) Compare and contrast the different state management approaches available in flutter such as setState, Provider and Riverpod. Provide scenarios where each approach is suitable.

→	Approach	How it works	When to use.
1.	setState	updates UI by calling setState() is a stateful widget.	Best for small apps or managing state within a single widget.
2.	Provider	uses InheritedWidget to share across widget efficiently.	Suitable for medium apps where data need to be shared between multiple widgets.
3.	Riverpod	An improved version of provider with better performance and simple syntax.	Best for large apps that need complex state management with dependency injection.

choosing the right approach:
- use setState for simple UI updates.
- use provider for moderate state sharing across widgets
- use riverpod for scalable, well structured applications.

Q4)

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

→ Process of integrating Firebase with a Flutter Application:-

1. Create a Firebase project - Go to Firebase console.
2. Add Firebase to flutter app - Register the app and download the google-service-json or google service-info plist.
3. Install firebase Packages - Add dependencies like 'Firebase-core' and 'Firebase_auth' in 'pubspec.yaml'.
4. Initialize Firebase - Import Firebase in 'main.dart' and call 'Firebase.initializeApp()'.
5. Use firebase service:- Implement authentication, database, or cloud functions as needed.

Benefits of Using Firebase as a Backend Solution:-

1. Real-time database - Syncs data instantly across devices
2. Authentication - Provides ready to use sign in options.
3. cloud Firestore - Hosts web apps and stores files securely
4. Hosting and storage: Stores structured data efficiently

5. Scalability - Handles large user bases without managing servers.
6. Push Notifications: Sends alerts and updates to users.

b) 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 methods
2. Cloud Firestore - A NoSQL database that stores and syncs data in real time.
3. Firebase Realtime Database - Stores and updates data instantly across all connected devices.
4. Firebase Cloud Storage - used for storing and retrieving files like images and videos.
5. Firebase Cloud Messaging (FCM) - Sends push notifications to users.
6. Firebase Hosting - Deploys web apps fast and secure hosting.
7. Firebase Analytics - Tracks user behaviour and app performance.

Data synchronization is achieved :-

1. Real time updates :- Firestore and Realtime Database sync data across devices instantly.
2. Listeners & streams :- Widgets listens for changes and updates the UI automatically.
3. Offline support :- Firebase caches data and syncs data while online.

MAD & PWA Lab
Journal

Experiment No.	Assignment-2
Assignment 2 Questions	1. Define Progressive Web App (PWA) and explain its significance in modern web development.Discuss the key characteristics that differentiate PWAs from traditional mobile apps 2. Define responsive web design and explain its importance in the context of Progressive Web Apps.Compare and contrast responsive, fluid, and adaptive web design approaches. 3. Describe the lifecycle of Service Workers, including registration, installation, and activation phases. 4. Explain the use of IndexedDB in the Service Worker for data storage.
Roll No.	
Name	
Class	D15B
Subject	MAD & PWA Lab
Lab Outcome	LO4:Understand various PWA frameworks and their requirements LO5: Design and Develop a responsive User Interface by applying PWA Design techniques LO6:Develop and Analyze PWA Features and deploy it over app hosting solutions
Grade:	

Name: Harsh Jodhav
Roll No: 21
Class: D15B

MPL Assignment No.2

Q.1] Define progressive Web App (PWA) and explain the 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 the web. PWAs combine the best features of traditional web apps and native mobile applications, offering improved performance, offline capabilities and enhanced user engagement.

Significance in Modern Web Development:-

- Cross-platform compatibility: PWA's work seamlessly across different devices and operating systems.
- Offline functionality: Service workers allow caching, enabling PWA's to function offline or in low network conditions.
- Improved Performance: PWAs utilize caching strategies to load faster than traditional web pages.
- App like experience: They can be installed on the home screen without requiring app store downloads.
- SEO-friendly: Unlike native apps, PWAs are indexable by search engines.

• Key characteristic That differentiate PWAs from Traditional mobile applications.

	Feature	PWA	Traditional Mobile App
(1)	Installation	No app store required, add to home screen.	Installed via App store / play store.
(2)	offline Support	yes using service workers	Yes, via local storage and background sync.
(3)	Performance	Fast due to caching	optimized but may require more resources.
(4)	Development cost	Lower, single codebase for web and mobile.	Higher, requires seperate development for iOS & android.

Q.2] Define responsive web design and explain its importance in the context of progressive Web Apps. Compare and contrast responsive fluid and adaptive web design approaches.

⟹ Responsive Web Design (RWD) is a design approach that ensures a website's layout adjusts dynamically to different screen sizes and resolutions. It uses flexible grids, media queries, and CSS techniques to provide an optimal viewing experience.

*Importance in PWAs—

• PWAs needs to function seamlessly across multiple

devices (mobile, tablet, desktop).

- Ensure a consistent user experience regardless of screen size.
- Enhances accessibility and usability
- Improves SEO rankings as mobile friendly websites are favoured by search engines.
- Reduces the need for seperate designs for ~~not~~ different device making development more efficient.

comparison of Responsive, fluid and adaptive Design.

	Feature.	Responsive Design	Fluid Design	Adaptive design.
(1)	approach	use break-points and Media queries.	use percentage based layout.	use predefined layouts for specific screen sizes.
(2)	flexibility	High, adjusts dynamically.	very High, scales with screen	Medium, Fixed layouts per device.
(3)	user Experience	Smooth across all devices	consistent but may stretch too much.	optimized for predefined screens and sizes.

Q.3] Describe the lifecycle of service workers, including registration, installation and activation phases.

⟹ A service worker is a background script that helps progressive Web Apps (PWAs) work offline, cache assets and handle push notifications.

The lifecycle of a service worker includes the following phases:

1. Registration phase:-

Before a service worker can start working, it must be registered in the browser. This is done using Javascript in the main application file.

Steps in Registration:

- The browser checks if service workers are supported
- The service worker script is registered.
- If successful, the installation phase begins

2. Installation phase:-

During installation, the service worker prepares itself by caching essential assets like HTML, CSS, Javascript and images. Once installation is successfully, it moves to activation phase.

Steps in installation:-

- The install event is triggered.
- The service worker caches required files for offline access.
- If caching fails, the service worker is not installed.

3. Activation phase:-

After installation, the service worker becomes active. In this phase, old caches are cleared and the new service worker takes control of open pages.

- Step in activation:
  - The "activated" event is triggered.
  - Old caches from previous versions are deleted.
  - The new service worker takes control of all open pages.

Q.4] Explain the use of IndexedDB in the service worker for data storage.

→ IndexedDB is a client side NOSQL database that allows web applications to store large amounts of structured data.
It works asynchronously and is used in progressive Web Apps (PWAs) to enable offline data storage and synchronization when the user is online.

- Use of IndexedDB in the service worker:
  1. Offline Data Storage:- Allows data to be stored locally when the user is offline.
  2. Background Syncing:- Sync data in the background when the user goes online.
  3. Fast Data Retrieval:- IndexedDB supports indexed searches for efficient data queries.
  4. Storage of large data - unlike web storage, IndexedDB can store large JSON objects.

5. Asynchronous operations: - works asynchronously to prevent UI blocking.

A serviceworker cannot access IndexedDB directly because it runs in a seperate thread.
However it communicates with the main thread using the postMessage() API or Background Sync API to store and retrieve data.