Name- Tarun Kumar Sharma
div- D15B
Rollno - 52

SDS Page No.
Date

Subscription tracker

Aim: The aim of the subscription tracker app is to provide users with a simple and intuitive tool to manage and track their subscription effectively. The app will help users monitor their recurring expenses, avoid unnecessary charges, and stay informed about upcoming renewal dates.

Problem Statement:
In today's digital age, people subscribe to multiple services such as streaming platforms. However managing these subscription can become overwhelming.

1. Lack of Centralized tracking: Users often forget about active subscription, leading to unnecessary expenses
2. Missed Renewal Dates: Auto renewals of subscription they no longer needed
3. Budgeting challengers: Multiple subscriptions can strain monthly budgets.

Features:

1. Subscription Managment:
Allow users to easily add or remove subscriptions including the option to chose from popular services or manually enter details

2. Payment tracker
Track Payment Dates: Allow users to ... renewal dates of each subscription and ... when payment are due
3. Notification & alerts
Due date Remainders: Push notifications ... or when a trail is about to end.
4. Budgeting & Analytics
Track overall spending on subscription ... set monthly budgets.
Show analytics on how much is spent ... subscription. ~~and~~
Allow users to see how their subscriptio... impact future spending based on upcomin... payments

Conclusion - A subscription tracker can p... immense value by helping users stay on ... top of their recurring payments, mana... budget effectively and avoid unexpected ...

Vivekanand Education Society's

Institute of Technology

(An Autonomous Institute affiliated to University of Mumbai)

Department of Information Technology

A.Y. 24-25

Mobile App Development and Progressive Web App

Experiment No.	01
Experiment Title	To install and configure the Flutter Environment.
Roll No.	52
Name	Tarun Kumar Sharma
Class	D15 B
Subject	MAD & PWA
Grade:	10 (A)

Name:Tarunkumar Sharma
Div:D15B
Roll no:52

EXP 1: Installation and Configuration of Flutter Environment.

Aim: Installation and Configuration of Flutter Environment.

Theory: Flutter is an open-source framework developed by Google that is primarily used for building natively compiled applications for mobile, web, and desktop from a single codebase. It is particularly popular for creating mobile apps for iOS and Android, but it can also be used to develop apps for macOS, Windows, Linux, and the web.

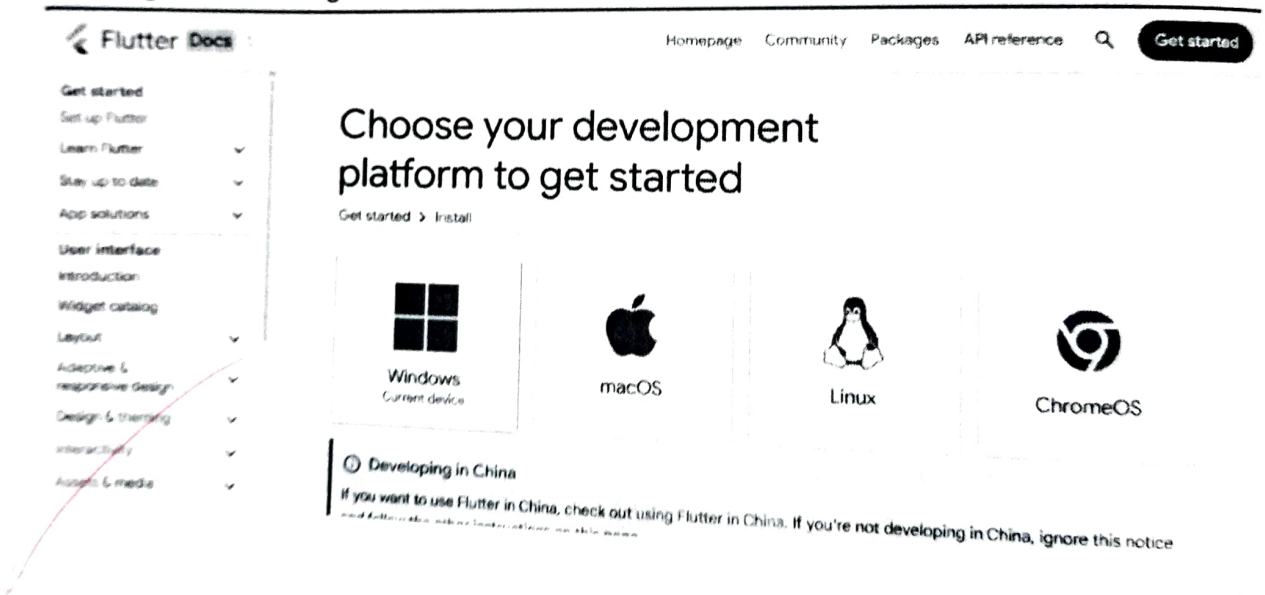
Key Features of Flutter:

1. Cross-Platform Development: With Flutter, developers can write one codebase and deploy it across multiple platforms (mobile, web, desktop). This greatly reduces the development effort and cost compared to maintaining separate codebases for different platforms.
2. Hot Reload: Flutter's "hot reload" feature allows developers to see code changes immediately reflected in the app without restarting it. This makes development faster and smoother.
3. Widgets: Flutter is built around a rich set of customizable widgets. These widgets describe how the app should look and behave, and they can be combined and customized to create complex UIs.
4. Dart Programming Language: Flutter uses Dart, a language developed by Google, which is object-oriented and compiled to native code. Dart is easy to learn and has strong support for asynchronous programming, which is crucial for building modern apps.

5. **Native Performance**: Since Flutter is compiled directly into native ARM machine code, it can provide near-native performance on both iOS and Android. This is one of the main advantages over other cross-platform frameworks like React Native.
6. **Customizable UI**: With Flutter, developers have complete control over the design and the user interface (UI). It doesn't rely on the platform's native components; instead, it provides its own set of widgets, which means you can create custom UIs that maintain consistency across platforms.

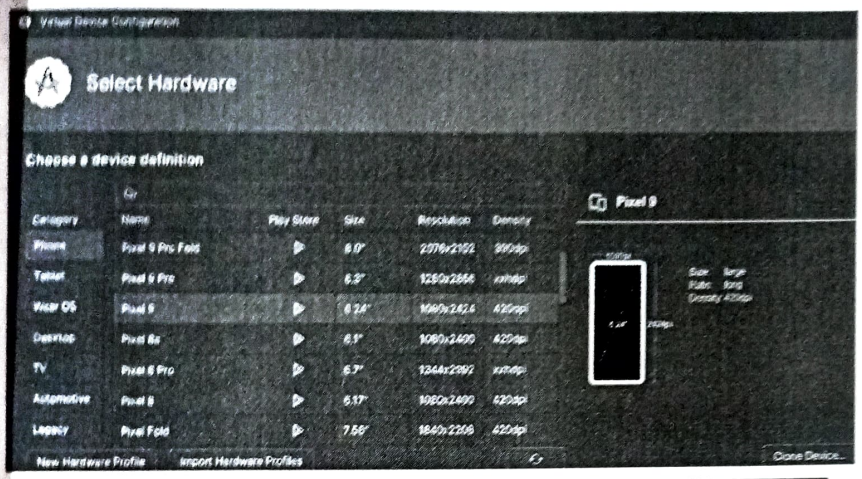
Install the Flutter SDK

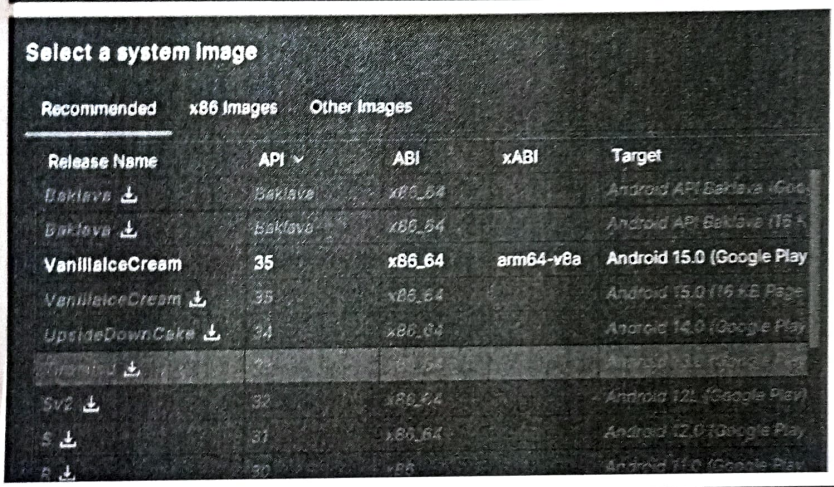
Step 1: Download the installation bundle of the Flutter Software Development Kit for windows. To download Flutter SDK, Go to its official website https://docs.flutter.dev/get-started/install , you will get the following screen.

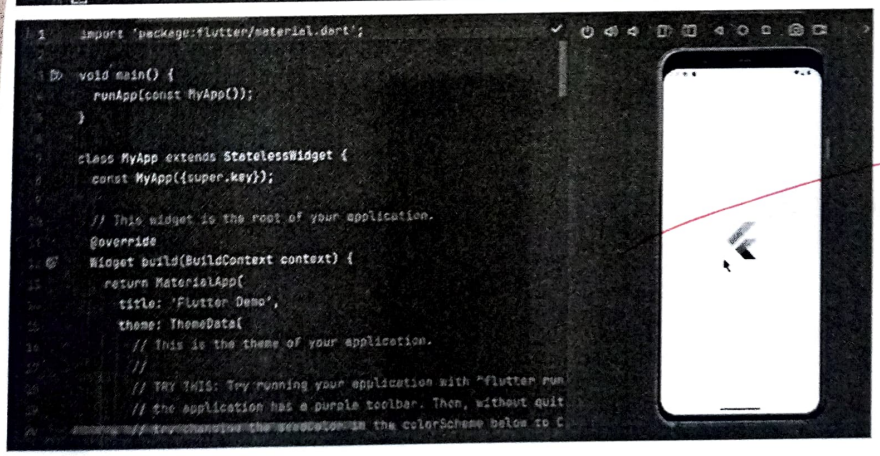


Step 2 : Download Flutter SDK: Go to the Flutter website and click on the Windows icon to download the latest SDK.

Step 3 : Extract the Zip: Once the download is complete, extract the zip file and place it in a location, like `C:\Flutter`.



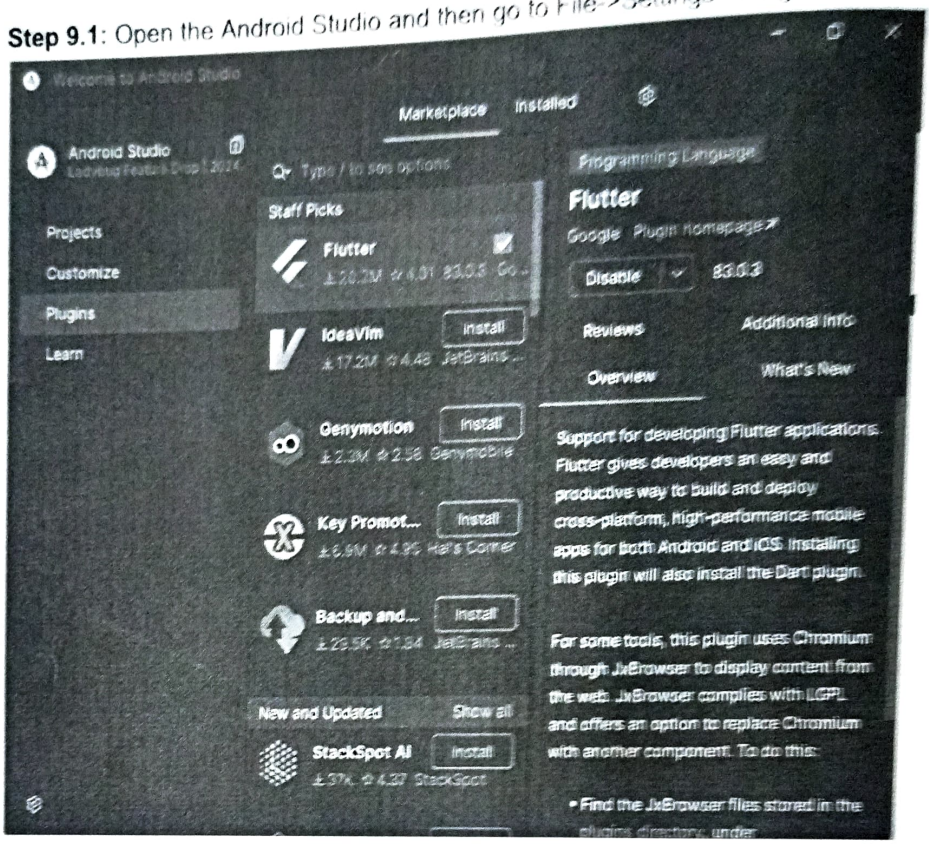




Step 9: Now, install Flutter and Dart plugin for building Flutter application in Android Studio. These plugins provide a template to create a Flutter application, give an option to run and debug

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.

Conclusion: In conclusion, you've successfully installed Flutter and Android Studio, set up the Android Emulator, and are now ready to begin developing cross-platform mobile applications. With these tools in place, you can efficiently build, test, and debug your apps. Start creating your Flutter projects, experiment with different widgets, and enjoy the development process!

Vivekanand Education Society's

Institute of Technology

(An Autonomous Institute Affiliated to University of Mumbai)

Since 1962

Department of Information Technology

A.Y. 24-25

Mobile App Development and Progressive Web App Lab

Experiment No.	02
Experiment Title.	To design Flutter UI by including common widgets.
Roll No.	52
Name	Tarun Kumar Sharma
Class	D15 B
Subject	MAD & PWA
Grade:	A

~~Advance DevOps Lab~~

(1)

MPL Exp No: 2

Aim: To design flutter UI by including common widgets

Theory:
Flutter is an open source UI toolkit by google for building natively compiled applications for mobile, web & desktop from a single codebase. It provides a wide range of pre-built widgets that helps developers design interactive & visually appealing UI.

1) Layout widgets:
Container: For styling & layout
Stack: Overlays widgets of top of each other
2) Use Navigation widgets
AppBar, Drawer, Bottom Navigation
3) List & grid widgets
ListView: Scrollable list of items
GridView: Display items in a grid format

Conclusion: Flutter widget-based UI design simplifies app dev, allowing for flexibility & efficiency. Using the right combination of Widgets ensures a responsive & User friendly experience

FOR EDUCATIONAL USE

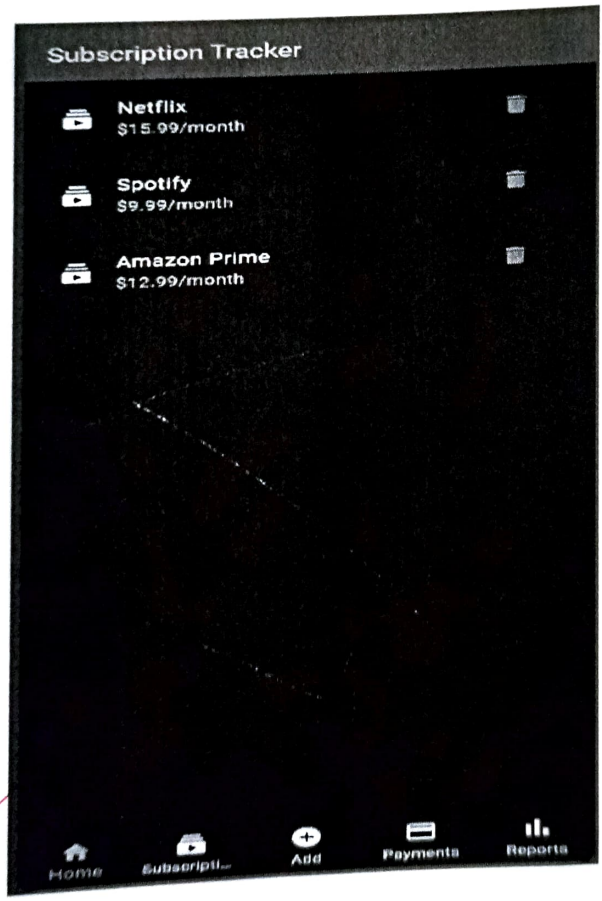
Name:Tarunkumar sharma

Div:D15B

Roll no:52

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

1) Use of scaffold Widget



Vivekanand Education Society's

Institute of Technology

(An Autonomous Institute Affiliated to University of Mumbai)

Department of Information Technology

A.Y. 24-25

Mobile App Development and Progressive Web App Lab

Experiment No.	03
Experiment Title.	To include icons, images, fonts in Flutter app
Roll No.	52
Name	Tarun Kumar Sharma
Class	D15 B
Subject	MAD & PWA
Grade:	A+

(A+)

Experiment NO-3

Aim : To include icons, images & fonts in flutter

Theory :

- Icons

Icons plays a vital role in UI design by making interfaces more intuitive and visually appealing. Flutter provides both built in Material icons

- Material icons

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

- Custom icons

Add dependency in pubspec.yaml:

```
dependencies:
  font_awesome_flutter: ^10.5.0
```

- Images

Images improve the UI experience and branching of an applications. Flutter support two main types of images

1. Assets Images
2. Network images

FOR EDUCATIONAL USE

- Fonts

Customs fonts are used to create unique branding and improve text readability. Flutter allows developers to import and use customs fonts for text styling

- Download custom fonts in .ttf or .otf font file and place it in assets/fold/.
- Use google fonts :

dependencies:

google_fonts: ^4.0.4

Conclusion:

In this experiment, we explored how to integrate icons, images and fonts in a flutter applications.

FOR EDUCATIONAL USE

Name:Tarunkumar Sharma
Div:D15B
Roll no:52

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

Icons in Flutter

Icons in Flutter help enhance UI by providing visual cues for actions, inputs, and navigation. Flutter provides **Material Icons** by default, but you can also use **custom icons** via `.ttf` font files.

1. Material Icons (Built-in)

Flutter has a built-in `Icons` class for Material Design icons. These are included in the Flutter SDK and do not require additional setup.

Usage in Code:

```
Icon(Icons.email, color: Colors.orange);

Icon(Icons.lock, color: Colors.orange);
```

2. Custom Icons (Using `.ttf` Font Files)

To use custom icons, you need a **.ttf font file** (e.g., FontAwesome or custom-designed icons).

Steps to Add Custom Icons:

Declare in `pubspec.yaml`:

```
flutter:

  Fonts

- family: CustomIcons

      fonts:
```

```
    - asset: assets/fonts/CustomIcons.ttf
```

Images in Flutter

Images are an essential part of a Flutter app's UI, used for branding (logos), illustrations, and user interaction. Flutter supports different types of images.

1. Types of Images in Flutter

- **Asset Images**: Stored in the project directory (`assets` folder).
- **Network Images**: Loaded from an online URL.
- **File Images**: Loaded from the local device storage.

Since you are using an **asset image (logo.jpg)** in your code, we will focus on that.

---

2. Using Asset Images in Flutter

Declare Image in `pubspec.yaml`

Open `pubspec.yaml` and add the image under `flutter > assets`:

```
flutter:
  assets:
    - assets/images/logo.jpg
```

Fonts in Flutter

Fonts play a crucial role in designing a visually appealing UI. By default, Flutter uses the **Roboto** font, but you can add custom fonts to give your app a unique look.

---

1. Adding Custom Fonts in Flutter

Step 1: Place the Font File in the `assets/fonts/` Folder

Save your font file (e.g., **Poppins-Regular.ttf**) inside the `assets/fonts/` folder.

Declare the Font in `pubspec.yaml`

Open `pubspec.yaml` and **register the font** under `flutter > fonts`:

```
flutter:
  fonts:
    - family: Poppins
      fonts:
        - asset: assets/fonts/Poppins-Regular.ttf
        - asset: assets/fonts/Poppins-Bold.ttf
          weight: 700
```

Track It

Create your account

Email

Password

Sign Up

Already have an account? Login

Vivekanand Education Society's
Institute of Technology
(An Autonomous Institute Affiliated to University of Mumbai)
Department of Information Technology

A.Y. 24-25

Mobile App Development and Progressive Web App Lab

Experiment No.	04
Experiment Title.	To create an interactive Form using form widget.
Roll No.	52
Name	Tarun Kumar Sharma
Class	D15 B
Subject	MAD & PWA
Grade:	8 (A+)

Experiment NO - 4

Aim: To create an interactive form using form widget

Theory:
Forms are an essential part of many applications allowing users to input and submit data. In flutter, the form widget provides a convient way to manage user input with validation and submission.
The text form field widget is commonly used inside a form for text input, offering built in validation form.

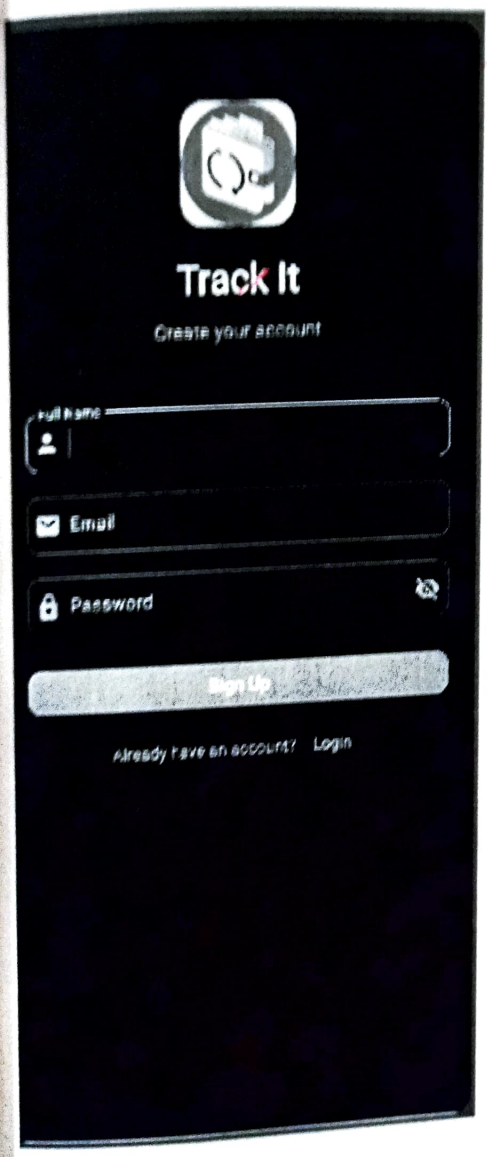
1. Form widget
A container for grouping multiple input field
Helps handle user input effectively

2) Text form field
A text input field use inside a form
Provides validation logic through the validator
Display error menages when validation fails

FOR EDUCATIONAL USE

Name: Tarunkumar Shaarma
Div: D15B
Roll no: 52

Exp 4 To create an interactive Form using form widget



Vivekanand Education Society's

Institute of Technology

(An Autonomous Institute Affiliated to University of Mumbai)

Department of Information Technology

A.Y. 24-25

Mobile App Development and Progressive Web App Lab

Experiment No.	05
Experiment Title.	To apply navigation, routing and gestures in Flutter App.
Roll No.	52
Name	Tarunkumar sharma
Class	D15 B
Subject	MAD & PWA
Grade:	A+

Experiment no - 5

Aim: Applying navigation, routing and gestures in the flutter app.

Theory:
Navigation, routing and gestures are fundamental aspects of any flutter application.

- Navigation

Navigation refers to moving between different screens in a flutter app. In flutter, navigation is handled using the Navigator class, which manages a stack of routes.

Push navigation

```
Navigator.pushC
    context,
    MaterialPageRoute : (context) => SecondScreen()),
    );
```

- Routing

Routing is the process of defining the available screens in the app and how to navigate between them.

```
Navigator.pushC
    context,
```

FOR EDUCATIONAL USE

```
MaterialPageRoute (
    builder : (context) => SecondScreen (data : 'Hello'),
    ),
);
```

- Gestures

Gestures are user interactions like tapping, swiping, pinching and dragging. Flutter provides GestureDetector widget.

```
GestureDetector (
    onLongPress : () {
        print ("Long press!");
    },
    child : Text ("Hold"),
);
```

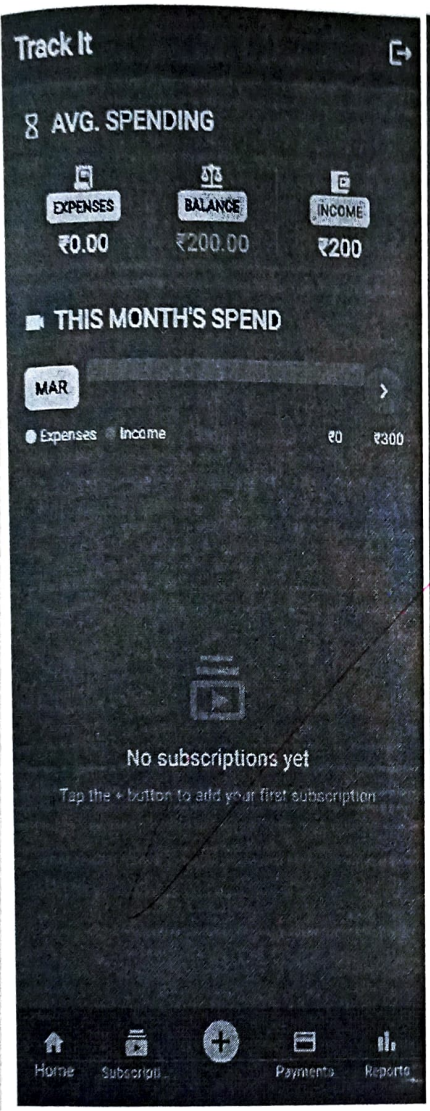
Conclusion:

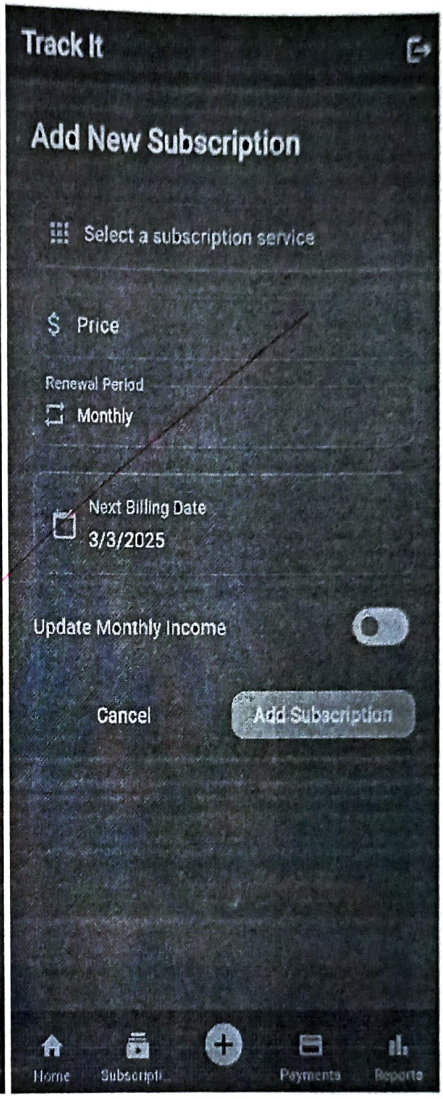
Navigation, routing and gestures are essential for creating interactive flutter application. By using the Navigator class, we can navigate screens efficiently. Mastering this concepts helps in dynamic and user friendly apps.

FOR EDUCATIONAL USE

Experiment no:5

```
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:sub/pages/signup_screen.dart';


// Import screens with correct paths
import 'pages/login_screen.dart';
import 'pages/signup_screen.dart';
import 'pages/home_screen.dart';
import 'pages/splash_screen.dart';
import 'firebase_options.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Track It',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primarySwatch: Colors.orange,
        scaffoldBackgroundColor: Colors.black,
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ),
      home: SplashScreen(),
```


Track It
AVG. SPENDING
EXPENSES
BALANCE
INCOME
₹0.00
₹200.00
₹200
THIS MONTH'S SPEND
MAR
Expenses
Income
₹0
₹300
No subscriptions yet
Tap the + button to add your first subscription
Home
Payments
Reports


Track It
Add New Subscription
Select a subscription service
Price
Renewal Period
Monthly
Next Billing Date
3/3/2025
Update Monthly Income
Cancel
Add Subscription
Home
Payments
Reports

Vivekanand Education Society's

Institute of Technology

(An Autonomous Institute Affiliated to University of Mumbai)

Since 1962

Department of Information Technology

A.Y. 24-25

Mobile App Development and Progressive Web App Lab

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	52
Name	Tarunkumar Sharma
Class	D15 B
Subject	MAD & PWA
Grade:	A

Since 1962

Vivekanand Education Society's

Institute of Technology

(An Autonomous Institute Affiliated to University of Mumbai)

Department of Information Technology

A.Y. 24-25

Mobile App Development and Progressive Web App Lab

Experiment No.	06
Experiment Title.	To Connect Flutter UI with fireBase database
Roll No.	52
Name	Tarun Kumar Sharma
Class	D15 B
Subject	MAD & PWA
Grade:	A

Exp - 6 : To setup Firebase

Theory :
Firebase is a Backend as a service platform provided by google that offers a suite of tools and services to help developers build, manage, and grow their apps. It provides features like authentication, real-time database, cloud functions, push notifications, analytics and more.

Flutter a UI toolkit by google, enable developers to build compiled applications for mobile, web and desktop from a single codebase.

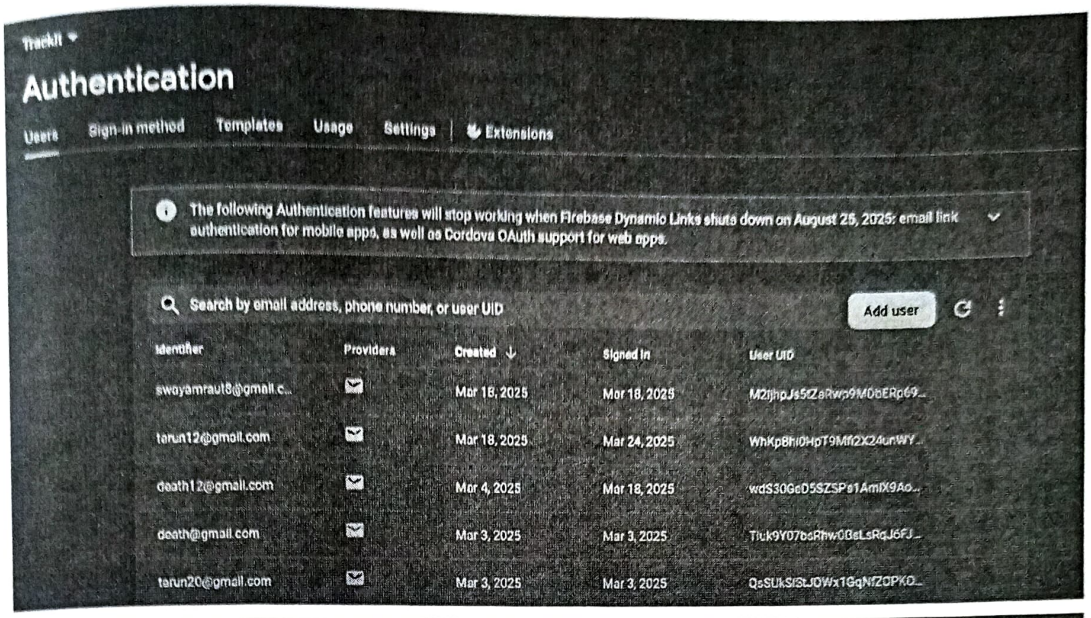
Firebase a product of google is a powerful platform that enables developers to build, manage and scale application with ease.
With features like real time cloud-storage, authentication and NoSQL database support.

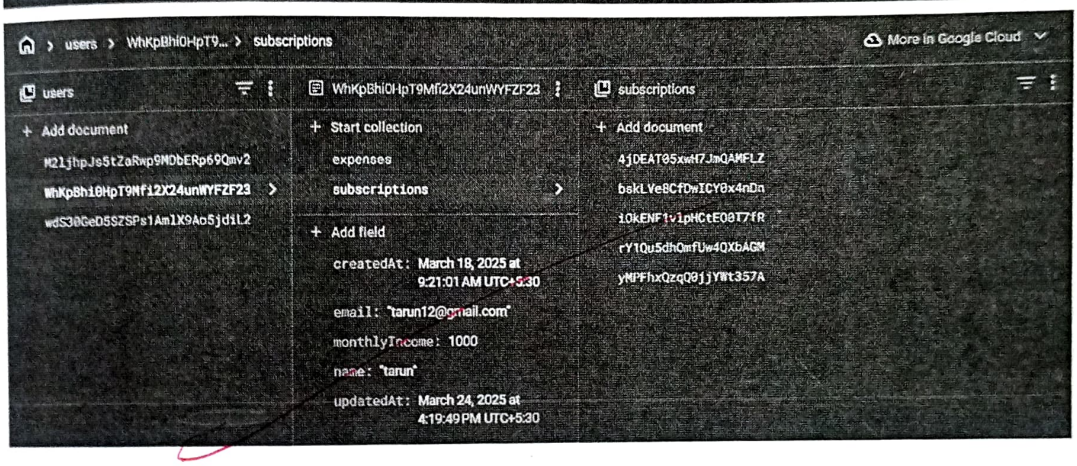
Features :-

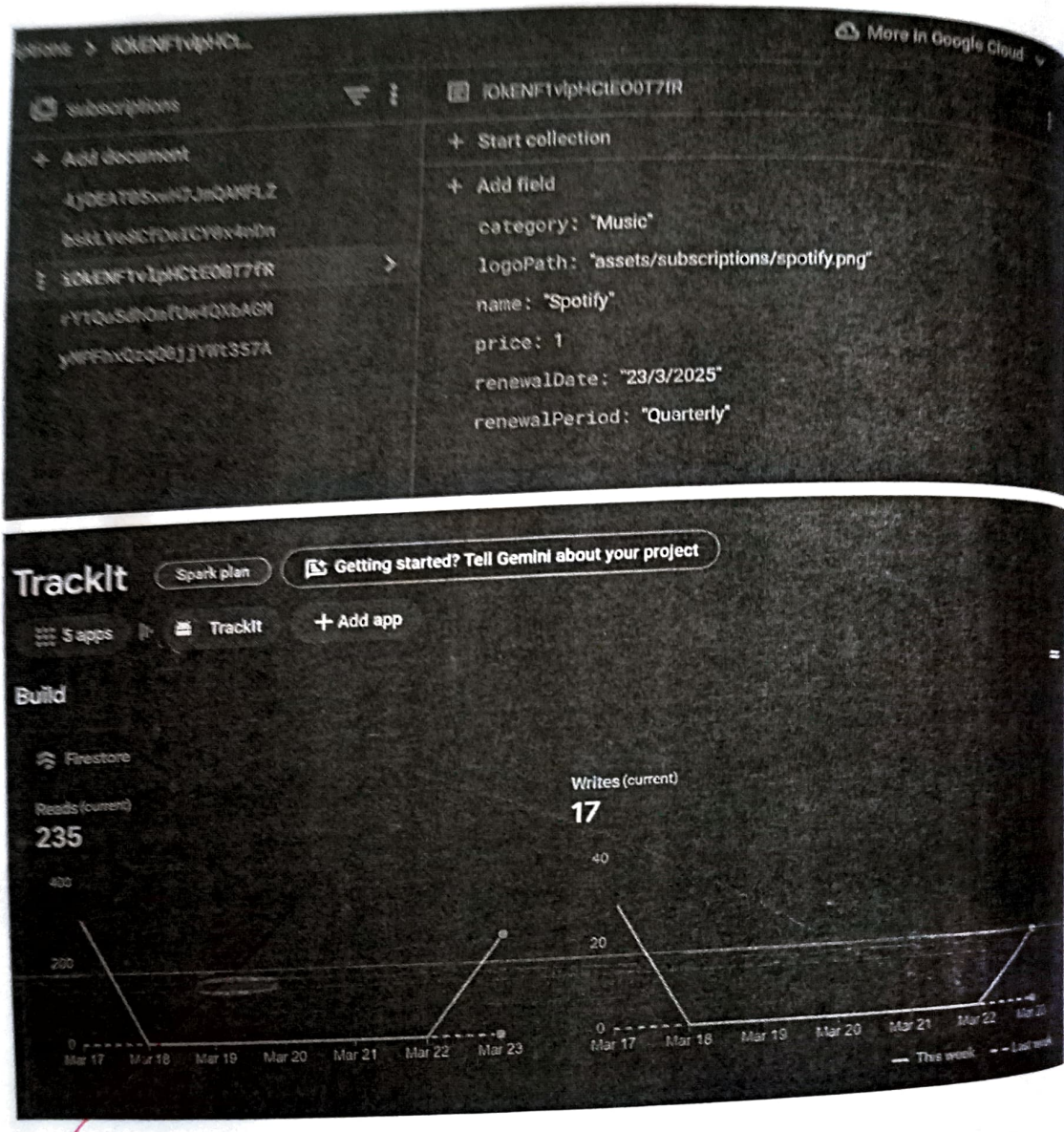
- Build Realtime database
- Cloud firestore
- Authentication
- Remote config
- Hosting

FOR EDUCATIONAL USE

```dart
// File generated by FlutterFire CLI.
// ignore_for_file: type=lint
import 'package:firebase_core/firebase_core.dart' show FirebaseOptions;
import 'package:flutter/foundation.dart'
    show defaultTargetPlatform, kIsWeb, TargetPlatform;

/// Default [FirebaseOptions] for use with your Firebase apps.
///
/// Example:
/// ```dart
/// import 'firebase_options.dart';
/// // ...
/// await Firebase.initializeApp(
///   options: DefaultFirebaseOptions.currentPlatform,
/// );
/// ```
class DefaultFirebaseOptions {
  static FirebaseOptions get currentPlatform {
    if (kIsWeb) {
      return web;
    }
    switch (defaultTargetPlatform) {
      case TargetPlatform.android:
        return android;
      case TargetPlatform.iOS:
        return ios;
      case TargetPlatform.macOS:
        return macos;
      case TargetPlatform.windows:
        return windows;
      case TargetPlatform.linux:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for linux - '
          'you can reconfigure this by running the FlutterFire CLI again.',
        );
      default:
        throw UnsupportedError(
          'DefaultFirebaseOptions are not supported for this platform.',
        );
    }
  }

  static const FirebaseOptions web = FirebaseOptions(
    apiKey: 'AIzaSyA60WlsPf1Xxmi8oV3eTG4RQDLnvTM42E0',
```


TrackIt
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
swayamraut8@gmail.c...
Mar 18, 2025
Mar 18, 2025
M2jhpJs5tZaRwp9MDbERp69...
tarun12@gmail.com
Mar 18, 2025
Mar 24, 2025
WhKpBhi0HpT9Mfi2X24unWY...
death12@gmail.com
Mar 4, 2025
Mar 18, 2025
wdS30GeD5SZSPs1AmlX9Ao...
death@gmail.com
Mar 3, 2025
Mar 3, 2025
Tiuk9Y07bsRhw0BsLsRqJ6FJ...
tarun20@gmail.com
Mar 3, 2025
Mar 3, 2025
QsSUkSfSLJOWx1GqNfZOPKD...


users
WhKpBhi0HpT9...
subscriptions
More in Google Cloud
users
Add document
M2ljhpJs5tZaRwp9MDbERp69Qmv2
WhKpBhi0HpT9Mfi2X24unWYFZF23
wdS30GeD5SZSPs1AmlX9Ao5jdiL2
WhKpBhi0HpT9Mfi2X24unWYFZF23
Start collection
expenses
subscriptions
Add field
createdAt: March 18, 2025 at 9:21:01 AM UTC+5:30
email: "tarun12@gmail.com"
monthlyIncome: 1000
name: "tarun"
updatedAt: March 24, 2025 at 4:19:49 PM UTC+5:30
subscriptions
Add document
4jDEAT05xwH7JmQAMFLZ
5skLVe8CfDwICY0x4nDn
iOkENFtv1pHCtE08T7fR
rY1Qu5dhOmfUw4QXbAGM
yMPFhxQzqQ0jjYWt357A


More in Google Cloud
subscriptions
iOkENF1vlpHCtEO0T7fR
Add document
Start collection
Add field
category: "Music"
logoPath: "assets/subscriptions/spotify.png"
name: "Spotify"
price: 1
renewalDate: "23/3/2025"
renewalPeriod: "Quarterly"
TrackIt
Spark plan
Getting started? Tell Gemini about your project
5 apps
TrackIt
Add app
Build
Firestore
Reads (current)
235
Writes (current)
17
Mar 17
Mar 18
Mar 19
Mar 20
Mar 21
Mar 22
Mar 23
This week
Last week

Vivekanand Education Society's

Institute of Technology

(An Autonomous Institute Affiliated to University of Mumbai)

Department of Information Technology

A.Y. 24-25

Mobile App Development and Progressive Web App Lab

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.	52
Name	Tarunkumar sharma
Class	D15 B
Subject	MAD & PWA
Grade:	

Experiment NO - 7

Objective : To understand and implement the basic structure and functioning of a Progressive Web App using HTML, a manifest file, and a Service worker.

What is a PWA?

A progressive web app is a type of web Application that uses modern web app capabilities to deliver an app-like experience to users.

PWA are :

Reliable : They load instantly.

Fast : They respond quickly.

Engaging : They feel like a natural app.

Conclusion :

Through this experiement, a basic Progressive web App was successfully created and tested. The app can work offline, can be installed on the device, and mimics native app behavior. This demonstrates the fundamental principles of PWA including offline capability, service worker registration, and the use of a manifest file.

FOR EDUCATIONAL USE

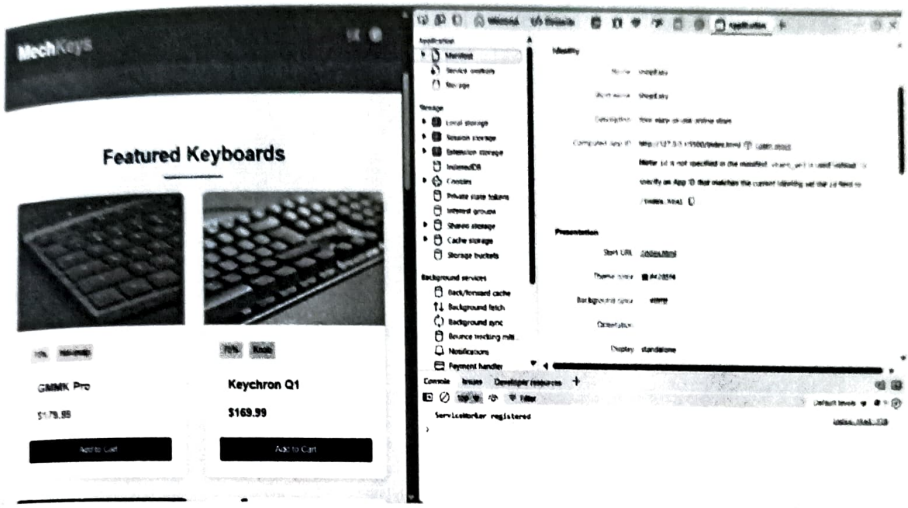
With Service Workers :

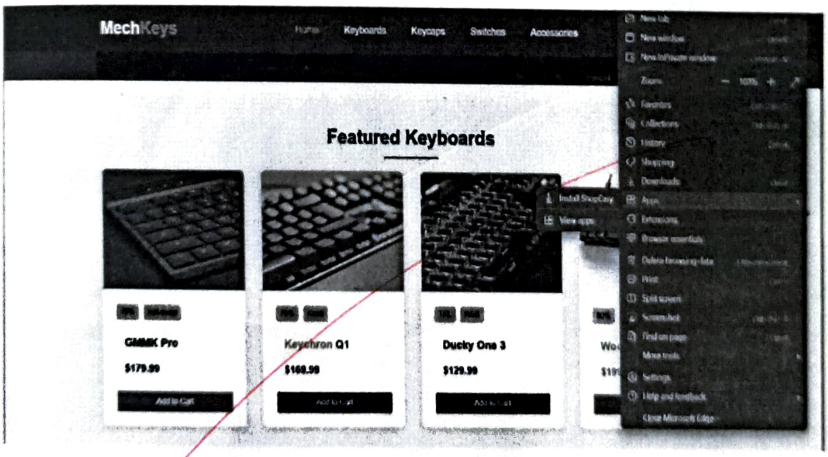
Network control : Intercept and manipulate request
Caching : offline access and faster load-times using Cache API
Push Notification : Enable real-time updates to the user
Background Sync : Handle request even when offline

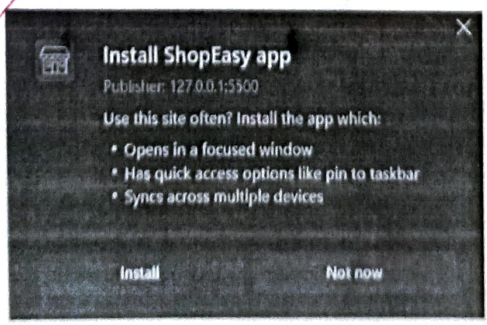
Conclusion :
This experiment demonstrates how a service worker can be effectively registered, installed and activated in a E-commerce PWA. The implementation of caching and offline support ensures reliability, faster load times, and improved user experience, even in poor network

FOR EDUCATIONAL USE

PWA EXPERIMENT - 7







Vivekanand Education Society's

Institute of Technology

(An Autonomous Institute Affiliated to University of Mumbai)

Department of Information Technology

A.Y. 24-25

Mobile App Development and Progressive Web App Lab

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.	52
Name	Tarunkumar sharma
Class	D15 B
Subject	MAD & PWA
Grade:	A1

Experiment No: 8

Aim: To code and register a service worker, and a complete the install and activation process for a new service worker in a E-commerce Progressive Web App

Theory:
A Service Worker is a script that runs in the background of the browser, separate from the web page, enabling features like:

- Background sync
- Offline support
- Push notification
- Caching strategies

It acts as a proxy between the browser and the network, intercepting request and handling responses

- Characteristics of Service Workers:

- They require HTTPS
- Cannot directly access the DOM
- Operate on a lifecycle model consisting of -
  - Registration
  - Installation
  - Activation

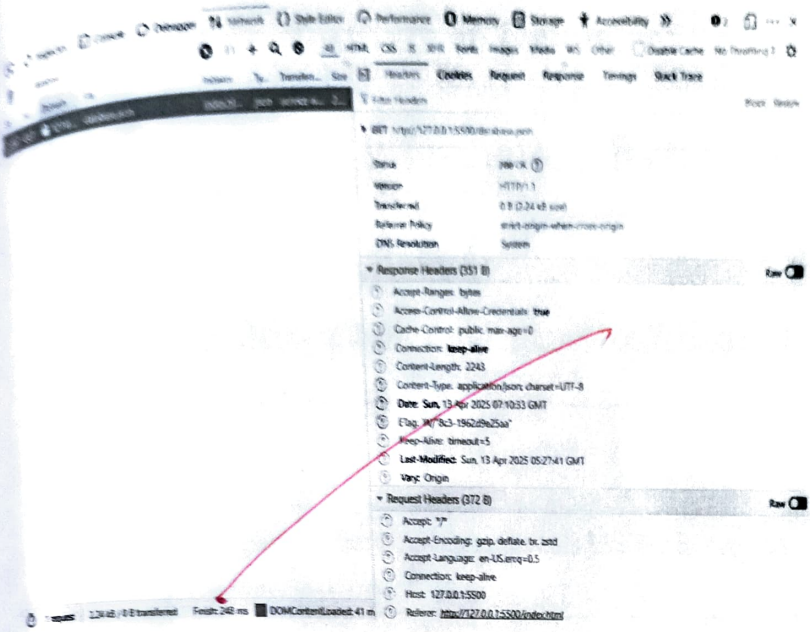
FOR EDUCATIONAL USE

Experiment N0-8

Code:

```
// Change this to a new version to trigger an update
const CACHE_NAME = "pwa-cache-v2";

const ASSETS_TO_CACHE = [
  "./",
  "./index.html",
  "./styles.css",
  "./app.js",
  "./manifest.json",
  "./images/shop_icon.png",
];

// Install event - Cache important assets
self.addEventListener("install", (event) => {
  console.log("✅ Service Worker Installed");

  // This line forces the waiting service worker to become active
  self.skipWaiting();

  event.waitUntil(
      caches.open(CACHE_NAME).then((cache) => {
      console.log("✅ Caching assets");
      return cache.addAll(ASSETS_TO_CACHE);
      })
  );
});

  // Activate event - Cleanup old caches
  self.addEventListener("activate", (event) => {
    console.log("✅ Service Worker Activated");

    // This ensures the service worker takes control immediately
    event.waitUntil(clients.claim());

    event.waitUntil(
        caches.keys().then((cacheNames) =>
```

```
      return fetchResponse;
      })
      );
      })
      .catch(() => {
      // Fallback for offline
      if (event.request.url.includes(".html")) {
      return caches.match("./index.html");
      }
      })
      );
}
});

// Rest of your code remains the same
self.addEventListener("sync", (event) => {
 if (event.tag === "sync-cart") {
      event.waitUntil(syncCart());
 }
});

self.addEventListener("push", (event) => {
 const options = {
      body: "This is a push notification!",
      icon: "/images/shop_icon.png",
      badge: "/images/badge.png",
 };
 event.waitUntil(self.registration.showNotification("New Message!", options));
});

async function syncCart() {
 // Logic to sync cart data with the server
 console.log("🔄 Syncing cart data with the server");
 // Example: Fetch unsynced cart data from IndexedDB and send to server
}

// Push event - Handle push notifications
self.addEventListener("push", (event) => {
 const data = event.data
      ? event.data.json()
      : { title: "New Notification", body: "Check out our latest products!" };
```


Response Headers (351 B)
Accept-Ranges: bytes
Access-Control-Allow-Credentials: true
Cache-Control: public, max-age=0
Connection: keep-alive
Content-Length: 2243
Content-Type: application/json; charset=UTF-8
Date: Sun, 13 Apr 2025 07:10:33 GMT
Keep-Alive: timeout=5
Last-Modified: Sun, 13 Apr 2025 05:27:41 GMT
Vary: Origin
Request Headers (372 B)
Accept: */*
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Host: 127.0.0.1:5500
Referer: http://127.0.0.1:5500/index.html
Finish: 248 ms
DOMContentLoaded: 41 m

Vivekanand Education Society's

Institute of Technology

(An Autonomous Institute Affiliated to University of Mumbai)

Department of Information Technology

A Y 24-25

Mobile App Development and Progressive Web App Lab

Experiment No.	09
Experiment Title.	To implement Service worker events like fetch, sync and push for E-commerce PWA.
Roll No.	52
Name	Tarunkumar sharma
Class	D15 B
Subject	MAD & PWA
Grade:	A+

Experiment No - 9

Aim: To implement Service Worker Events such as fetch, sync and push for an E-commerce Progressive Web App.

Theory:
A Service Worker is a Background Script that run separately from the main browser thread. It enables offline functionality, background sync and push notification by acting as a client side proxy between your app and the Network

Service Worker Events:

- Fetch Events:
  Used to intercept requests and serve response from cache or network depending on strategy
- Sync Events:
  Used to retry failed actions
- Push Events

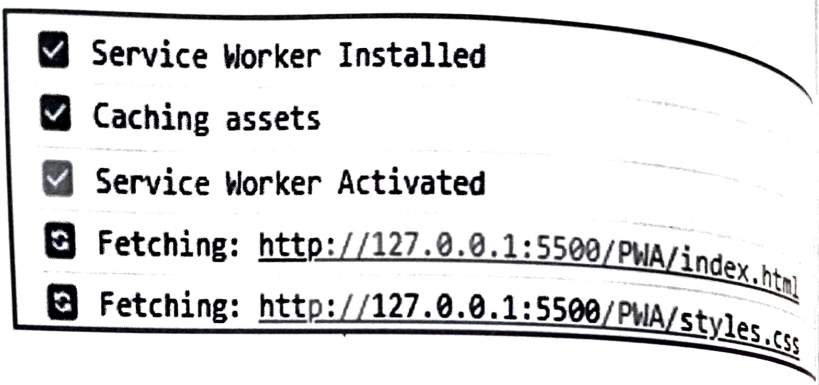
Conclusion:
This experiment successfully demonstrates the power of service workers in building reliable and engaging E-commerce PWAs

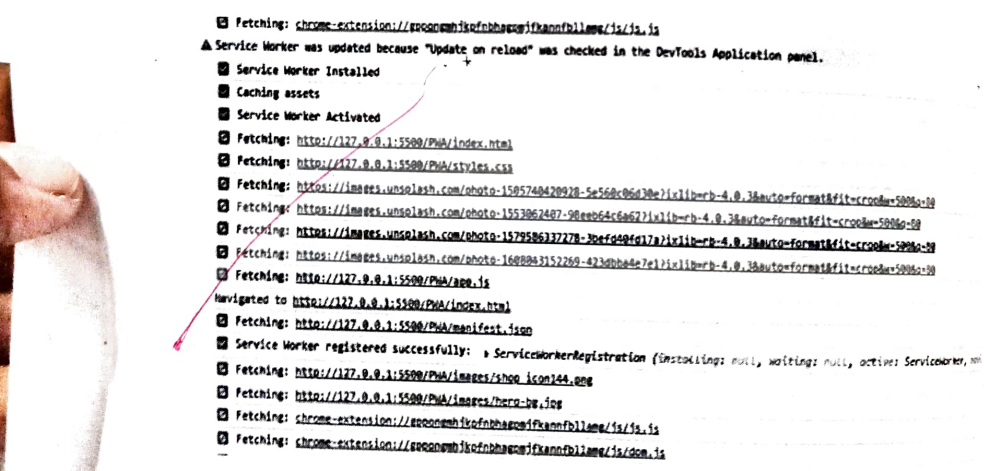
FOR EDUCATIONAL USE

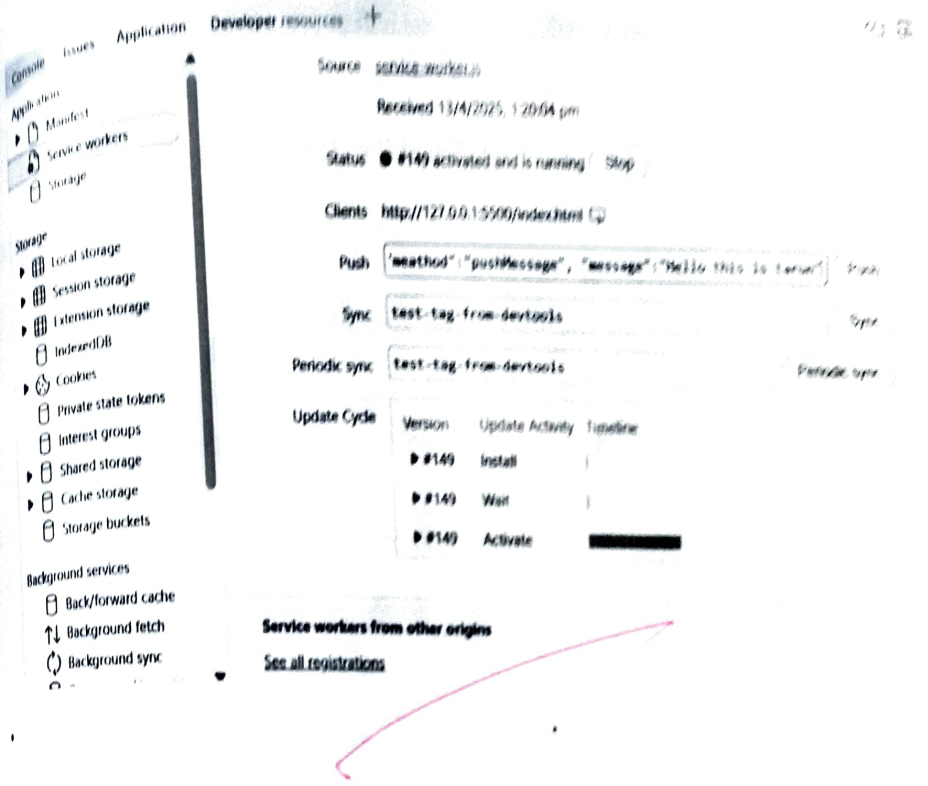
EXPERIMENT NO. 7

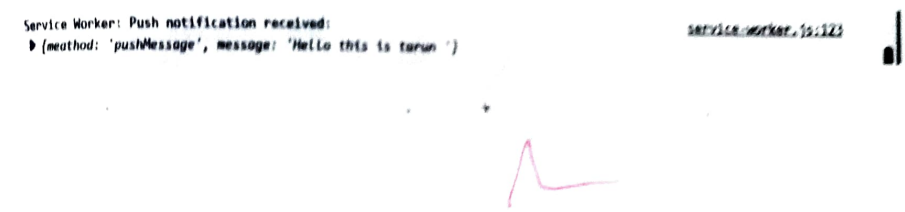
```
// Change this to a new version to trigger an update
const CACHE_NAME = "pwa-cache-v2";

const ASSETS_TO_CACHE = [
  "/",
  "/index.html",
  "/style.css",
  "/app.js",
  "/manifest.json",
  "/images/shop_icon.png",
];

// Install event - Cache important assets
self.addEventListener("install", (event) => {
  console.log("✅ Service Worker Installed");

  // This line forces the waiting service worker to become active
  self.skipWaiting();

  event.waitUntil(
    caches.open(CACHE_NAME).then((cache) => {
      console.log("✅ Caching assets");
      return cache.addAll(ASSETS_TO_CACHE);
    })
  );
});

// Activate event - Cleanup old caches
self.addEventListener("activate", (event) => {
  console.log("✅ Service Worker Activated");

  // This ensures the service worker takes control immediately
  event.waitUntil(clients.claim());

  event.waitUntil(
    caches.keys().then((cacheNames) =>
```

```
console.log("🔔 Push notification received:", data);
const options = {
        body: data.body,
        icon: "/PWA/icons/icon-192x192.png",
        badge: "/PWA/icons/icon-192x192.png",
};
event.waitUntil(self.registration.showNotification(data.title, options));
});
```






Console
Issues
Application
Developer resources
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
Source service worker.js
Received 13/4/2025, 1:20:04 pm
Status #149 activated and is running Stop
Clients http://127.0.0.1:5500/index.html
Push {"method":"pushMessage", "message":"Hello this is tarun"} Push
Sync test-tag-from-devtools Sync
Periodic sync test-tag-from-devtools Periodic sync
Update Cycle
Version
Update Activity
Timeline
#149 Install
#149 Wait
#149 Activate
Service workers from other origins
See all registrations


Service Worker: Push notification received:
{method: 'pushMessage', message: 'Hello this is tarun '}
service-worker.js:123

Vivekanand Education Society's

Institute of Technology

(An Autonomous Institute Affiliated to University of Mumbai)

Department of Information Technology

A.Y. 24-25

Mobile App Development and Progressive Web App Lab

Experiment No.	10
Experiment Title.	To study and implement deployment of Ecommerce PWA to GitHub Pages.
Roll No.	52
Name	Tarun Kumar sharma
Class	D15 B
Subject	MAD & PWA
Grade:	8 (AA)

Experiment No- 10

Aim:
To study and implement deployment of an E-commerce Progressive Web App to Github pages

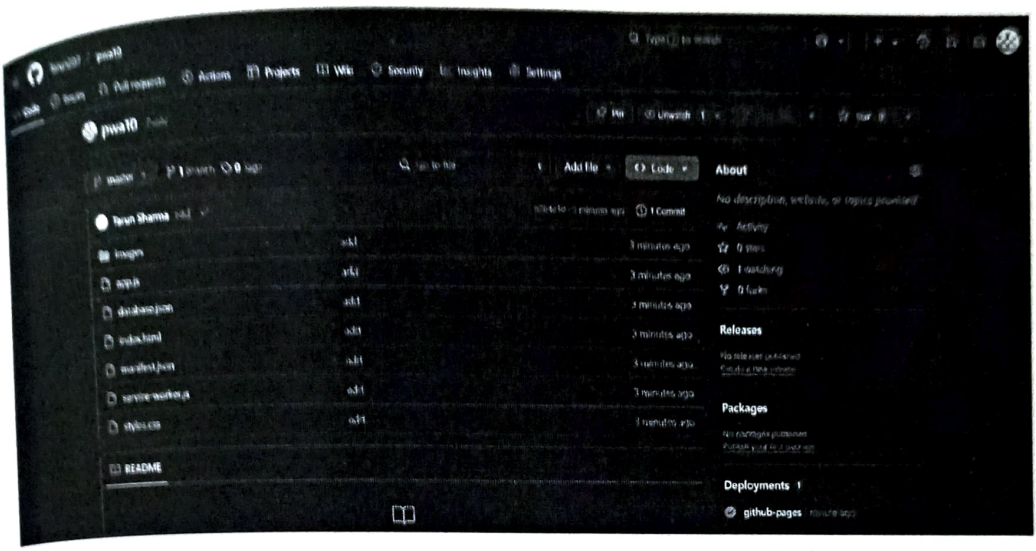
Github Pages:
Github pages is a free static hosting service that takes HTML, CSS and JavaScript straight from a repository on Github, optionally runs the files through a build process and publishes a website

Key features:
- Public webpages hosted directly from github repo.
- Simple workflow
- Supports Jekyll blogging

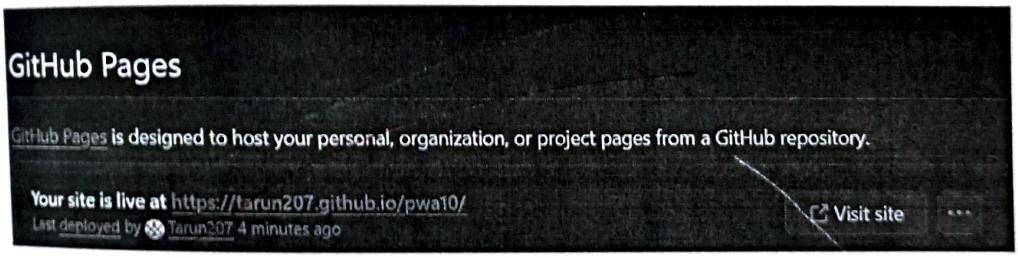
Conclusion:
Deployment using Github pages is efficient, cost-effective and suitable for static PWA. For apps requiring real-time capabilities, firebase hosting is more appropriate

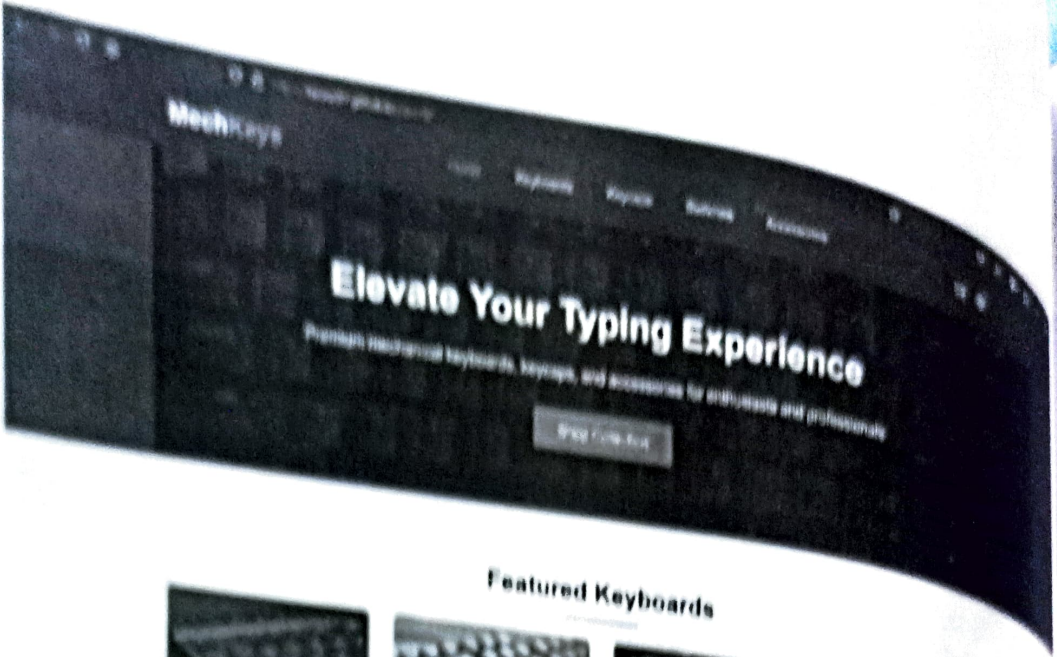
FOR EDUCATIONAL USE

Experiment No-10



https://tarun207.github.io/pwa10/




MechKeys
Elevate Your Typing Experience


Featured Keyboards

MechKeys


Featured Keyboards

Vivekanand Education Society's

Institute of Technology

(An Autonomous Institute Affiliated to University of Mumbai)

Department of Information Technology

A.Y. 24-25

Mobile App Development and Progressive Web App Lab

Experiment No.	11
Experiment Title.	To use google Lighthouse PWA Analysis Tool to test the PWA functioning.
Roll No.	52
Name	Tarun Kumar. sharma
Class	D15 B
Subject	MAD & PWA
Grade:	A+

Experiment NO-11

Aim: To use lighthouse tool to test the PWA functioning

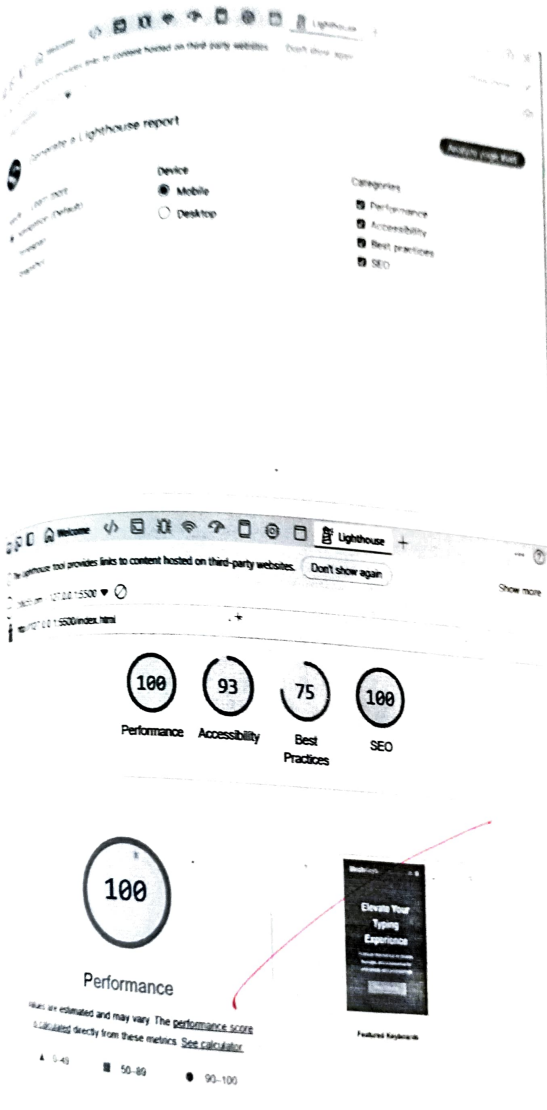
What's google lighthouse?
It's a tool that lets you audit your web-application based on a number of parameters including performance, based on number of metrics & each mobile compatability. Progressive web app implementation etc. All you have to do is run it on a page or pass it to URL site back a couple of minutes...

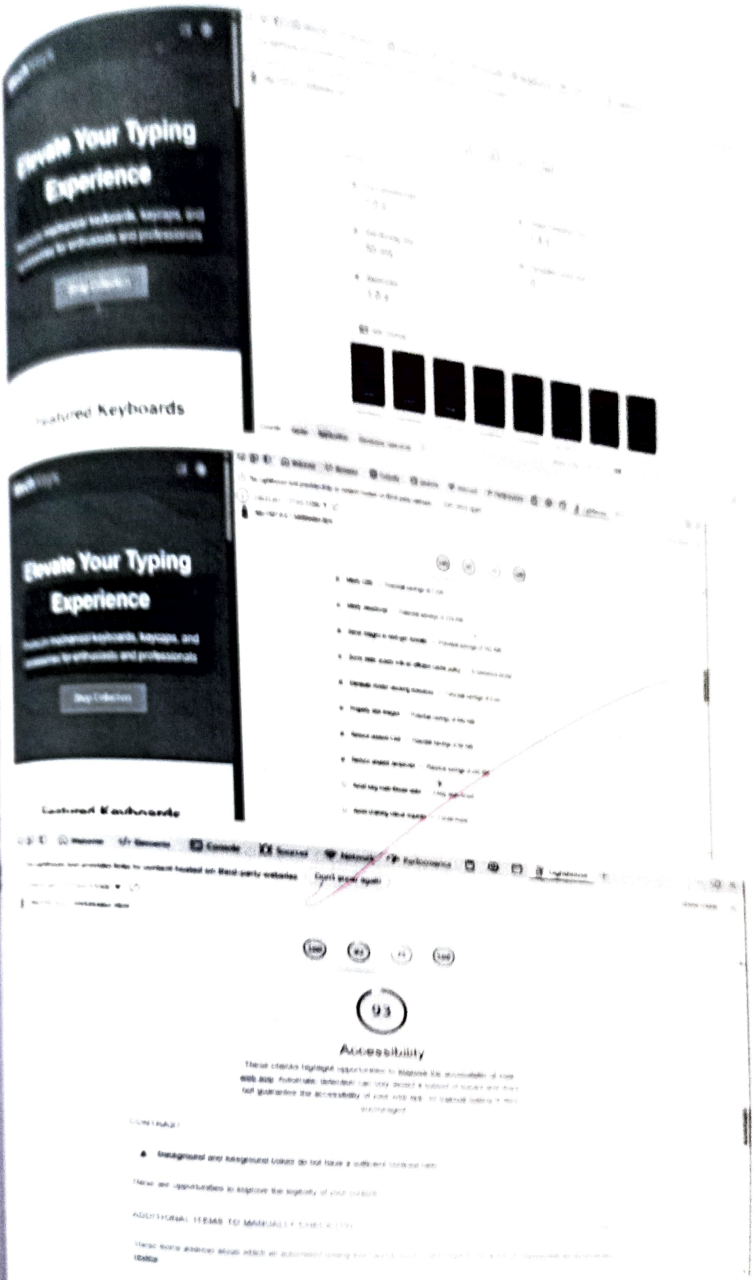
Key features:
Performance: Its an aggregation
Accessibility
Best practices
SEO
PWA

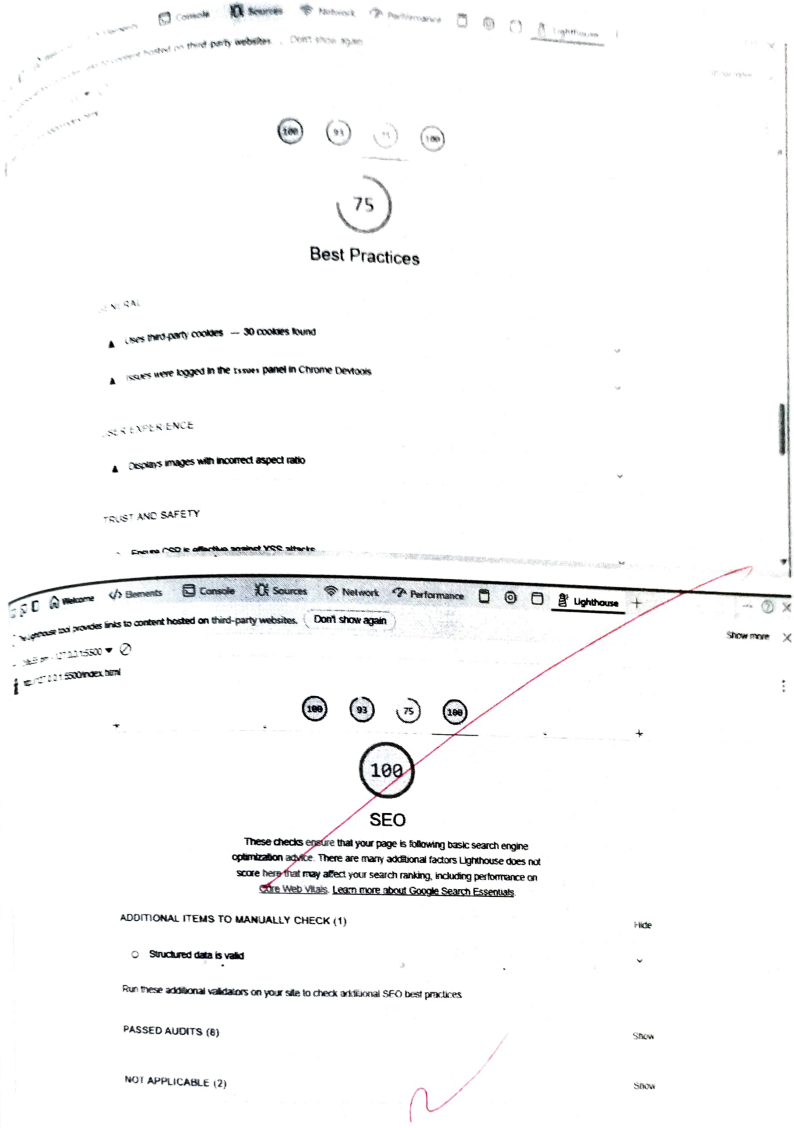
Conclusion:
Lighthouse makes your website faster, more user friendly and easier to find online

FOR EDUCATIONAL USE

Experiment No-11




Elevate Your Typing Experience
Featured Keyboards
Elevate Your Typing Experience
Shop Collection
93
Accessibility


Console
Sources
Network
Performance
Lighthouse
100
91
75
100
Best Practices
GENERAL
Uses third-party cookies — 30 cookies found
Issues were logged in the Issues panel in Chrome Devtools
USER EXPERIENCE
Displays images with incorrect aspect ratio
TRUST AND SAFETY
Welcome
Elements
Console
Sources
Network
Performance
Lighthouse
Lighthouse can provides links to content hosted on third-party websites.
Don't show again
Show more
100
93
75
100
100
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
ADDITIONAL ITEMS TO MANUALLY CHECK (1)
Hide
Structured data is valid
Run these additional validators on your site to check additional SEO best practices.
PASSED AUDITS (8)
Show
NOT APPLICABLE (2)
Show

Vivekanand Education Society's

Institute of Technology

(An Autonomous Institute Affiliated to University of Mumbai)

Department of Information Technology

A.Y. 24-25

Mobile App Development and Progressive Web App Lab

Experiment No.	Assignment-1
Title.	
Roll No.	52
Name	Tarun Kumar Sharma.
Class	D15 B
Subject	MAD & PWA
Grade:	4

Prashant Sharma
52

MAD Assignment

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

- Single database for multiple program
- Write one codebase for both Android and IOS, reducing development effort and maintenance.
- Hot Reload instantly see changes in the app without restarting making development faster and more interactive.
- Fast Performace Uses the Dart language and a compiled approach for smooth and high performance app
- Open source & strong community support. Backed by google and a large developer community ensuring continous improvement and resources

Advantages

1. Faster development time: Hot reload and single codebase reduces development time significantly
2. Cost effective: Since the same runs code run on both Android and ios, business save on development and maintaince cost

FOR EDUCATIONAL USE

Constant UI across devices since flutter does not rely on native components, the UI looks and behaves the same on different os version

Improved performance : AOT compilation and direct access to GPU rendering ensures smooth animations and high performance

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

Widget tree in flutter

In flutter, the widget tree is the fundamental structures that represent the UI of an application. It is a hierarchial arrangment of widgets where each widget defines a part of the user interface. Flutter UI are 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 refers to building complex UIs by combining smaller, reusable widgets

Example:

```
class ProfileCard extends StatelessWidget {
    final String name,
    final String imageUrl,
```

FOR EDUCATIONAL USE

```
ProfileCard ({required this.name, required ...
@override
Widget build (BuildContext context) {
return Card (
    child Column (
        children [
        Image.network (imageUrl)
        SizedBox (height: 10)
        Text (name, style: TextStyle (fon...
    )
)
}
```

Benifits

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

b) Providing examples of commonly used widg... and their roles in creating a widget ...

1. Structural Widget.
These widgets acts as the foundation for ... the UI
- Material app: The root widget of a flutter ap... that provides essential configuration
- Scaffold: Provides a basic layout structure

Sundaram®
FOR EDUCATIONAL USE

nesting an app bar using padding configuration

Example:
```
Material App (
    home: Scaffold (
        app bar. AppBar (title. Text ("Flutter
                    widget tree")
    body. container (
        padding: Edge (Inset all (16.0)
        child. text (" Hello, Flutter! ")
    ),
   ),
  ),
```

Input & interaction Widgets
Text field. accepts text input from user
Elevated Button. A button with elevation
Gesture button. Detects gestures like taps swipes and long press.

Ex.
```
Column (
    Children [
    Textfield (decoration, Input Decoration (label text, "
                        Enter name")
    Elevated Button (
        OnPressed: () {
            print ("Button Pressed");
        },
        Child. text ("Submit")
        )
    } );
```

Q3)

a Discuss the importance state managment in flutter applications

→ In flutter, state refers to data that can change during the lifetime of an application. This includes:

- User inputs
- UI changes
- Network changes
- Animation states

There are two types of States

1 Ephemeral state: small, UI specific state that doesn't affect the whole app.

2 App wide states - Data stored across multiple widgets importance of state management.

- Efficient UI updates: Flutter UI rebuild whenever State changes. Efficient state managment ensures that only necessary widgets are updated, improving performance
- Code maintainability & scaling: Managing state

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

FOR EDUCATIONAL USE

SetState : Local State
Pros: simple built in easy to use
Cons: Not scalable unnessary re-renders
Best use Cases: Small UI Updates

Provider: App wide State
Pros: Light weigh recommended by flutter, effiect
Cons: Boiler plate code for nested provider
Best use case: Large apps needing global state

Riverpod App: Wide State
Pros - Eliminates Provider limitations
Cons - Requires learing new concepts

Scenarios for each approach
UseState 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 of theme changes
Uses Riverpod when building a complex scalable app with global State managment like an ecommerce app with cart managment

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

Firebase provides a powerful backend for flutter application offering services like authentication, real time database, cloud functions, storage and more.

Steps to integrate firebase with flutter

Step - 1
Create a firebase project
Go to firebase console
Click on 'add projects' and enter a project name
Configure google analytics if needed, then

- Step 2
In the firebase project dashboard click 'Add' and select Android or IOS based on your
For android: Enter the android package name and download the google services.json file and place it on android/app/

- Step 3
Install firebase dependencies
Add firebase dependencies in pubspec.yaml
firebase_core

FOR EDUCATIONAL USE

firebase auth
cloud firestore
Run: flutter pub get

Step 4.
Configure firebase for android
for android
Open android / build.gradle and ensure the
following classpath 'com.google.gms:google-services
4.3.10'
Open android/app/build.gradle and add at
the bottom apply plugin: 'com.google.gms.google-
services'.

Step 5
Initialize Firebase in flutter
Void main() async {
Widget flutterBinding ensureInitialized();
await firebase.initializeApp();
runApp(MyApp());

Benifits of Using firebase
1. Easy to setup & scale
No need to manage backend infrastructure
Scales automatically based on usage
2. Authentication
Provides email / password, google, facebook and
phone authentication
Seamless integration with the firebase Authenti-
cation

FOR EDUCATIONAL USE

Firebase cloud Messaging
Enables push notifications and messaging between users
Eg: Firebase Messaging. instance . subscribe To topic ("news")

Firebase hosting
Deploys and serves web applications securely with automatic sol

Data Synchronization in firebase
Firebase ensures real time data synchronization across multiple devices and platform using firestore and real time Database

1. Cloud firebase Sync Mechanism
Uses real time listners to update UI instantly when data changes

Ex: Firebase Firestore. instance collection ('users').
snapshots (). listen ((snapshot) {
for (var doc in snapshot) {
print (doc [name]);
}
});

2. Realtime Database Sync Mechanism
Uses persistent Websocket Connections for live updates.

FOR EDUCATIONAL USE

3 Offline Data Sync.
Firestore caches locally and sync changes when the device is online
Ex: Firebase Firestore, instance.settings
Setting (persistenceEnabled true);

4 Cloud Functions for automated updates.
Automates backend logic to trigger updates when data changes

FOR EDUCATIONAL USE

Vivekanand Education Society's

Institute of Technology

(An Autonomous Institute Affiliated to University of Mumbai)

Department of Information Technology

A.Y. 24-25

Mobile App Development and Progressive Web App Lab

Experiment No.	Assignment-2
Title.	
Roll No.	52
Name	Tarun Kumar sharma
Class	D15 B
Subject	MAD & PWA
Grade:	

Name: Tarun kumar sharma
Div - D15B
Roll no - 52

52

04/05

MPL - 2

Define Progressive web app & explain its significance in modern web app.

A PWA is a web app that contains the both web & mobile apps to deliver a seamless PWA work offline, load quickly & provide an app like experience in web.

Platform independence
Improved performance
offline functionality
No app store dependence
Engaging User Experience

Key characharcteristics

1) Installation: Installed from browser, traditional mobile apps are download from mobile apps
2) Platform dependencies
3) offline support
4) Updates
5) Perfpernance.

PWA are faster due to caching & lightweight assets.

FOR EDUCATIONAL USE

Vivekanand Education Society's
Institute of Technology
(An Autonomous Institute Affiliated to University of Mumbai)
Department of Information Technology

Mobile App Development and Progressive Web App Lab

Experiment No	Assignment-2
Title	
Roll No.	52
Name	Tarun Kumar Sharma
Class	D15 B
Subject	MAD & PWA
Grade:	

The lifecycle of service workers.
installation & activation phases.

('service worker' in navigator) {
navigator.serviceWorker.register('/sw.js')
.then(() => console.log('Service worker registered'))
}

installation
self.addEventListener('install', event => {
event.waitUntil(
caches.open('v1').then(cache => {
return cache.addAll(
});

activation
Runs after installation & ensures old cache cleared if necessary.
Eg: self.addEventListener('activate', event => {
event.waitUntil(
caches.keys().then(keys => {
return Promise.all(keys.filter(key => key !== 'v1')
});
}

iv) Fetching & updates

→ The service workers intercepts network requests

Eg: self.addEventListener('fetch' event => {
event.respondWith(
caches.match(event.request)
);
});

Q4) Explain the use of indexedDB in the service workers for data storage?

→ IndexedDB is a low-level NoSQL database in the browser that allows web apps to store & retrieve.

Use of IndexedDB in service workers

i) Offline storage :- Saves user data when offline & sync it when

ii) Persistent Data - Unlike local storage - IndexedDB is asynchronous

3) Background Sync - Service workers can use indexedDB to store data & sync it later.

FOR EDUCATIONAL USE