Prerequisite Document

1. problem statement : Many individuals struggle with selecting outfits that suit their unique body types & personal styles. The lack of personalized fashion guidance leads to frequent wardrobe dissatisfaction.

Description of Stylume App - It is a personalized fashion recommendation mobile application developed using flutter. The app aims to guide users in choosing clothing that best suits their body type by
Taking an interactive body analysis test
Identifying user's body type
recommending outfit choices based on different occasions

3. Existing Applications & Their drawbacks

pinterest - offers fashion inspiration through photos & boards
drawbacks - lack personalization

Smart closet - virtual wardrobe & outfit planner
drawbacks - complex UI, no body type detection

4) Our solution - Stylume App
Stylume bridges the gap by providing a data driven, user centric fashion experience
- Interactive test - gathers key metrics & answers to determine body shape
- Tailored recommendations
- Simple UI/UX

FOR EDUCATIONAL USE

Advantages of StyleMe

personalization - outfit suggestions tailored to individ
body type

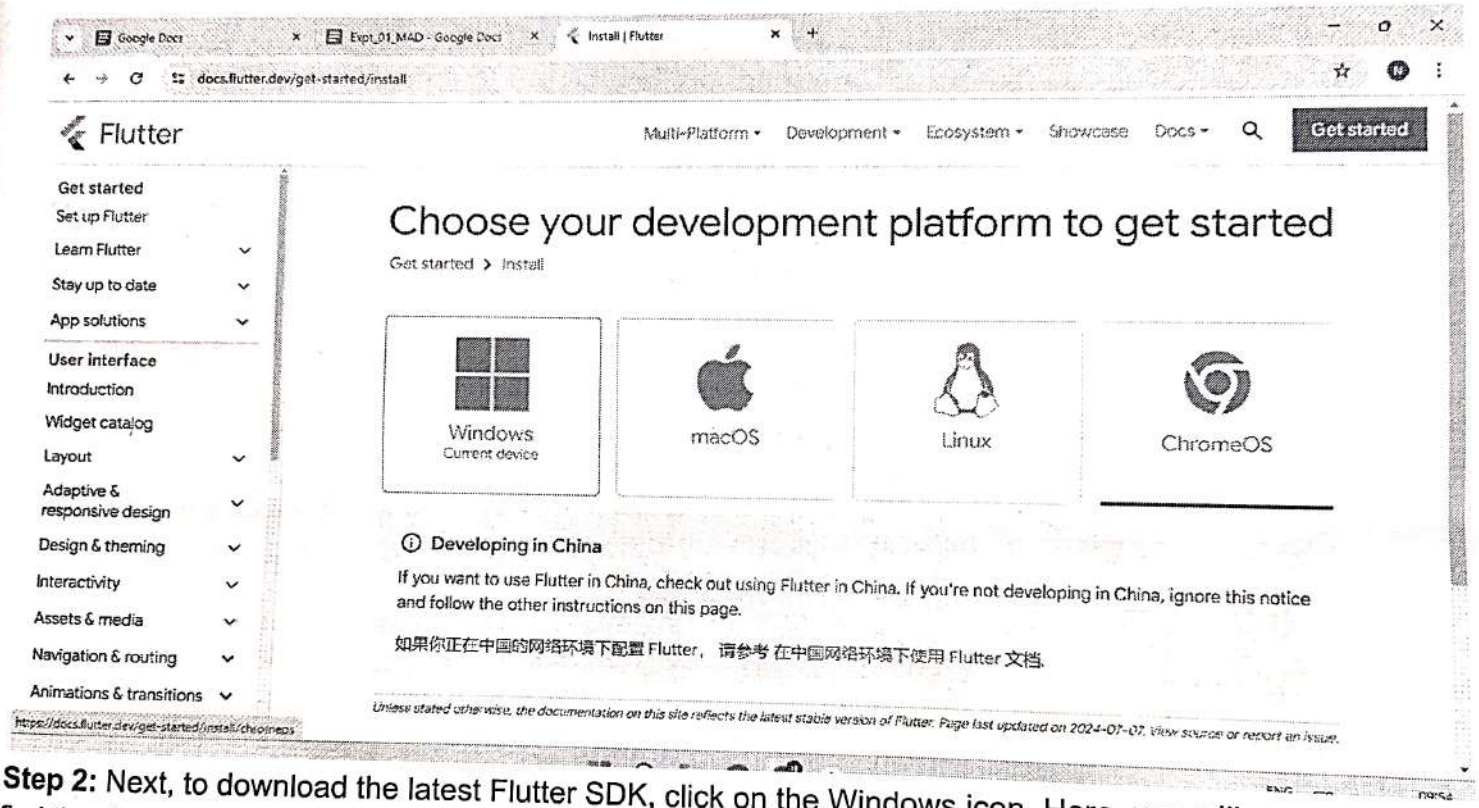
confidence boost - helps useful confident & co
in what they wear

Reduces time spent deciding what to wear or buy

EXP 1: Installation and Configuration of Flutter

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.

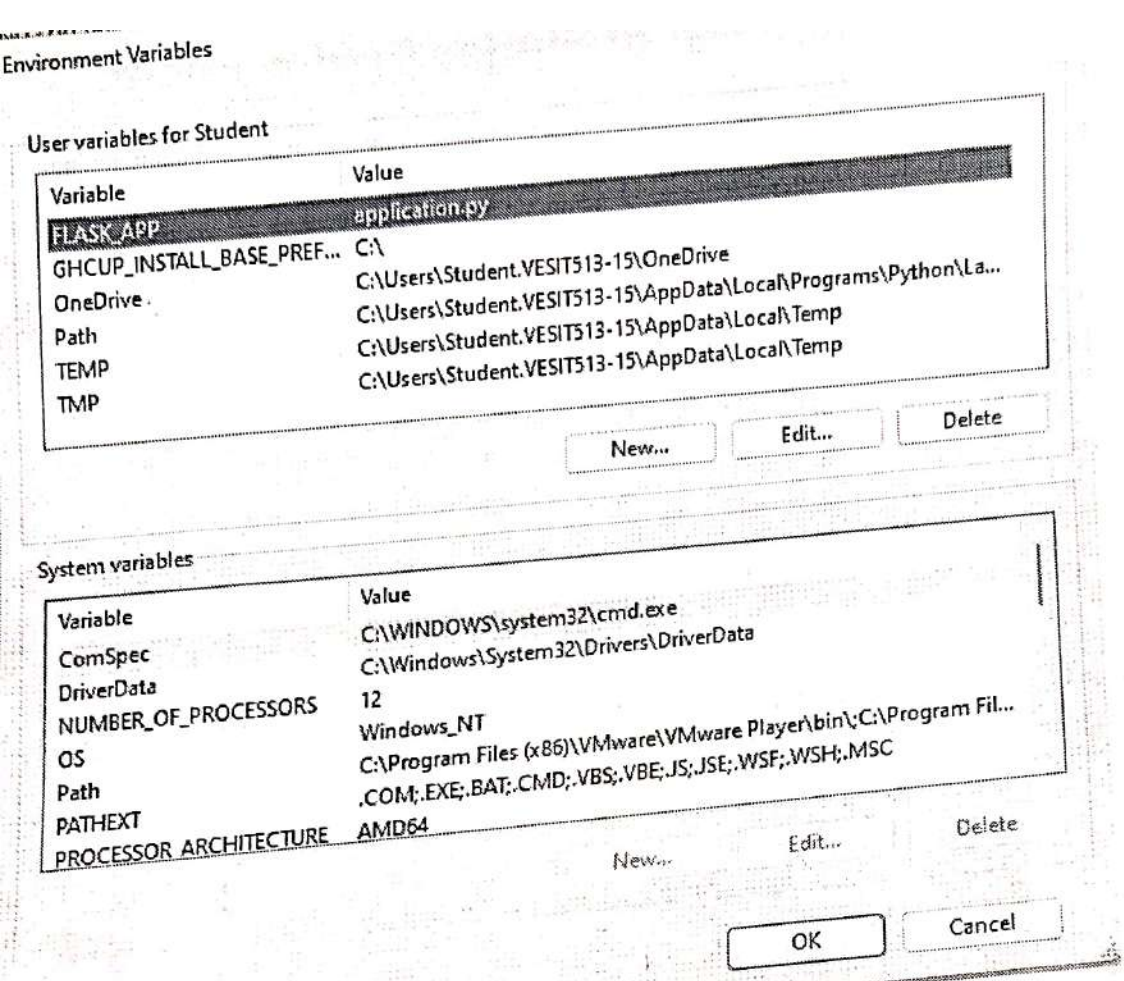


Step 2: Next, to download the latest Flutter SDK, click on the Windows icon. Here, you will find the download link for SDK.

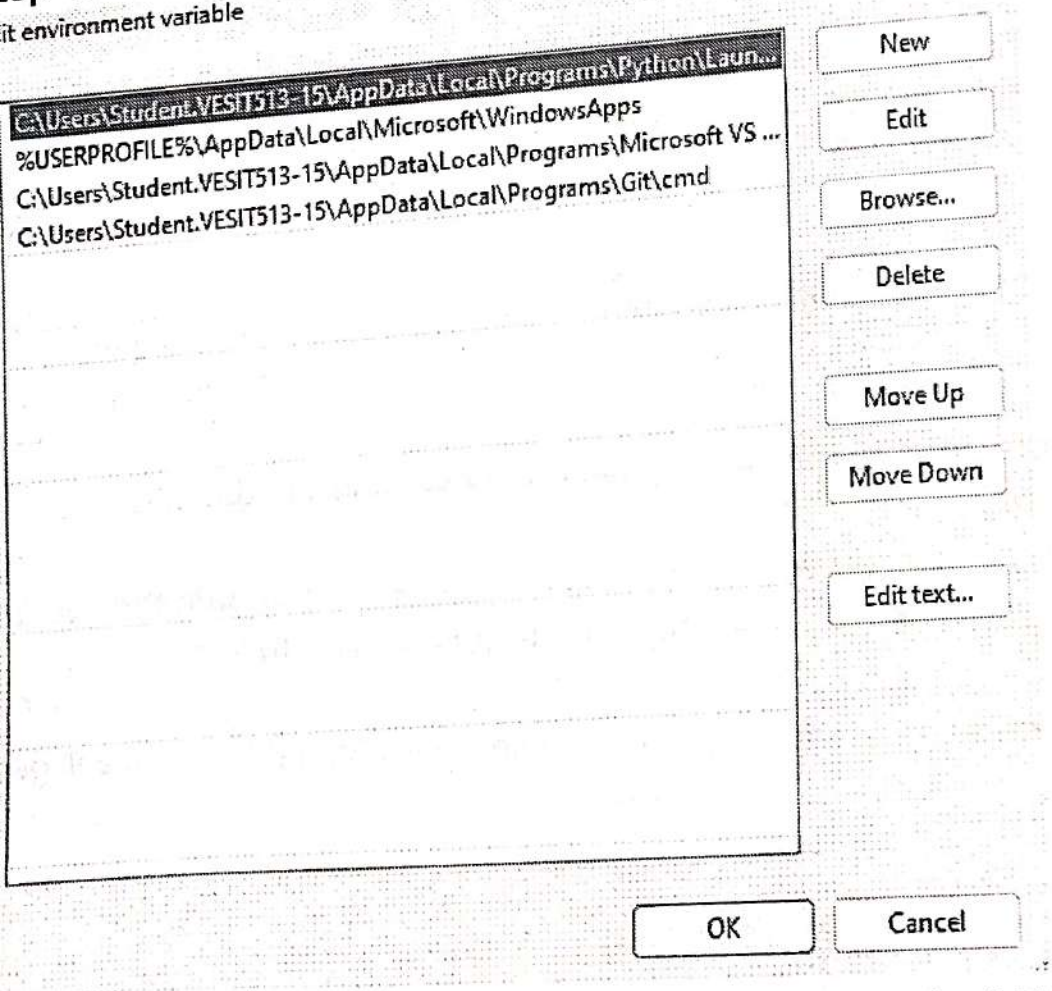
Step 3: When your download is complete, extract the zip file and place it in the desired installation folder or location, for example, C: /Flutter.

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:

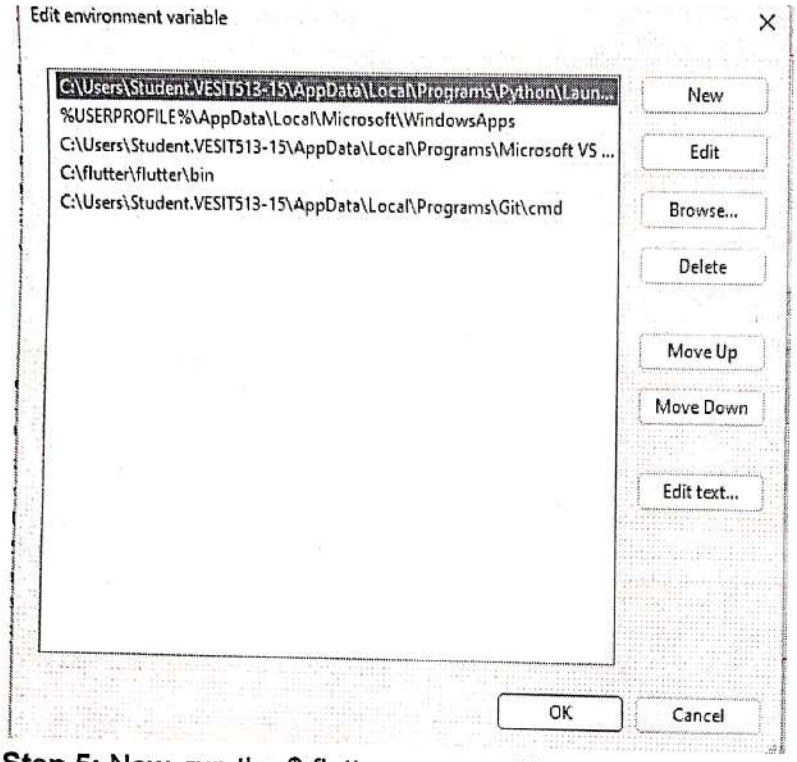
Step 4.1: Go to MyComputer properties -> advanced tab -> environment variables. You will get the following screen.



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



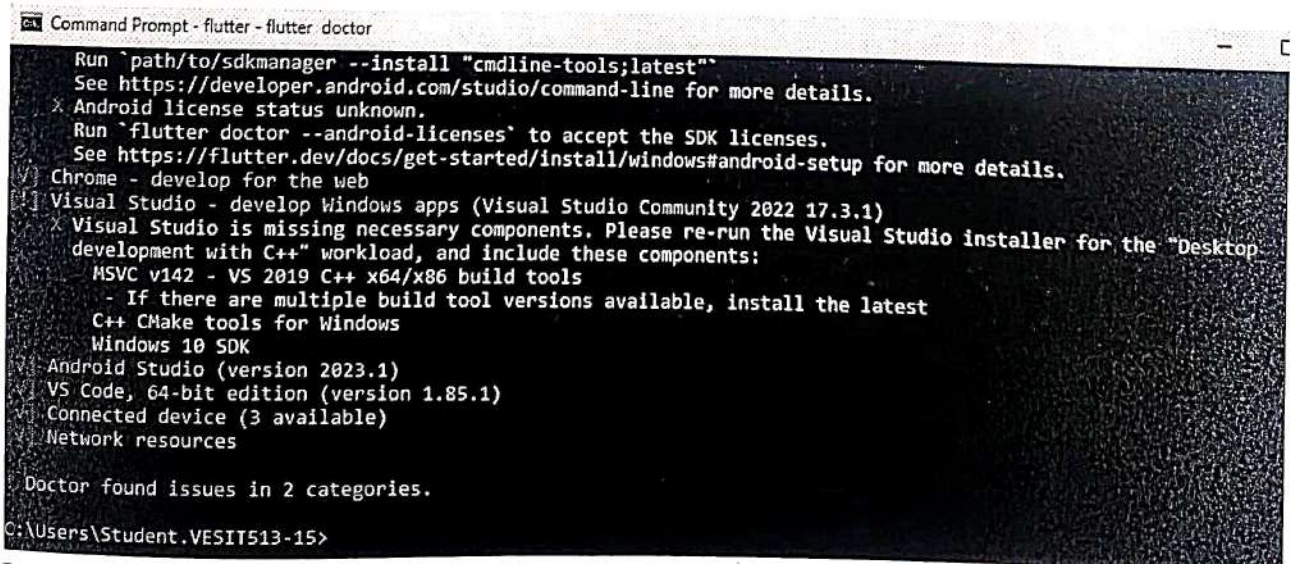
tep 4.3: In the above window, click on New->write path of Flutter bin folder in variable value



Step 5: Now, run the $ flutter command in command prompt.

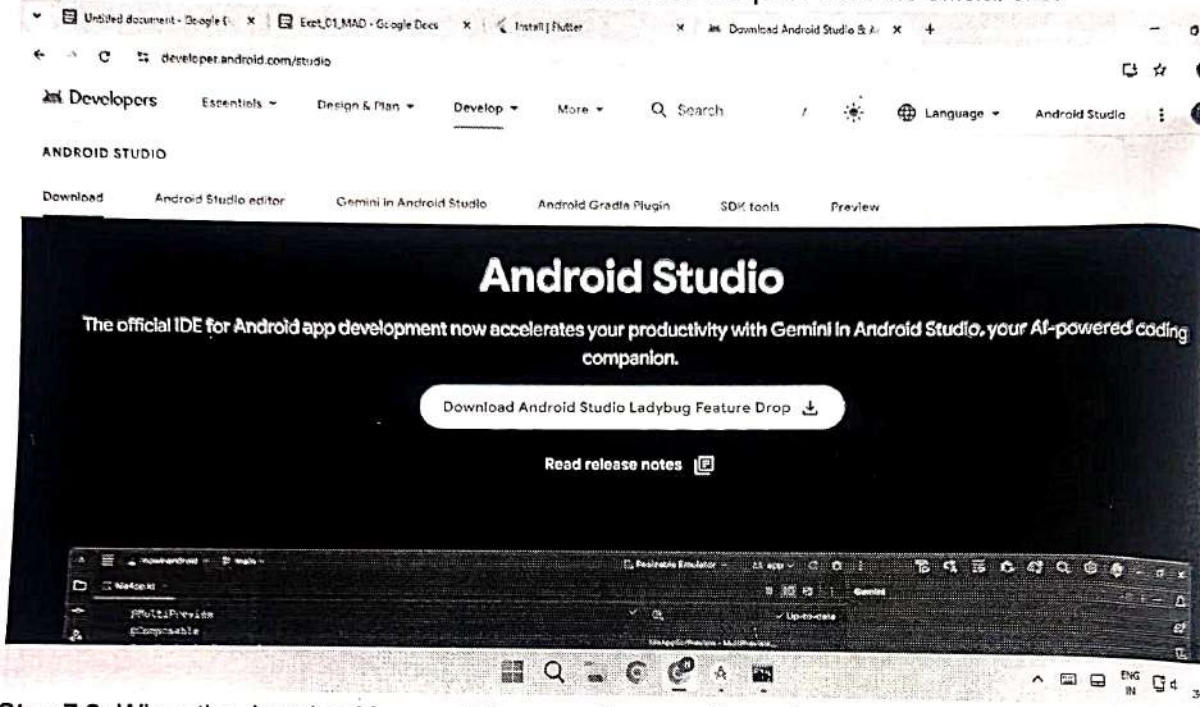
Now, run the $ flutter doctor command. This command checks for all the requirements of Flutter app development and displays a report of the status of your Flutter installation.

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.



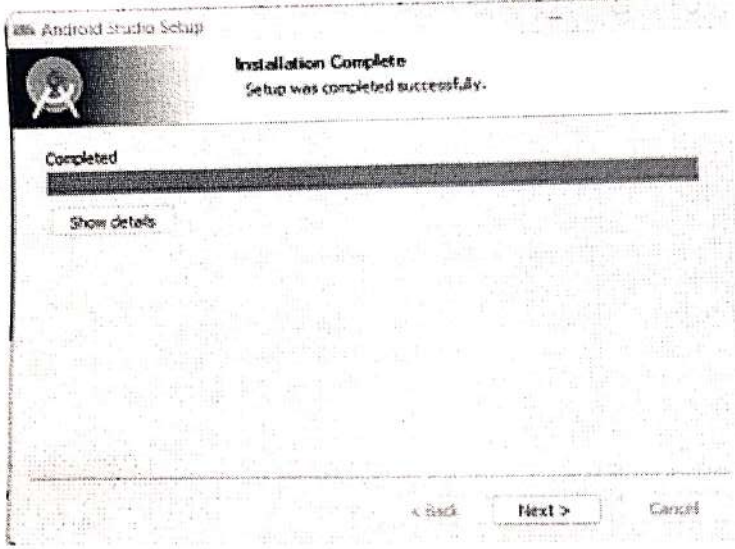
tool in your system, then you need first to install the Android Studio IDE. To install Android Studio IDE, do the following steps.

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.



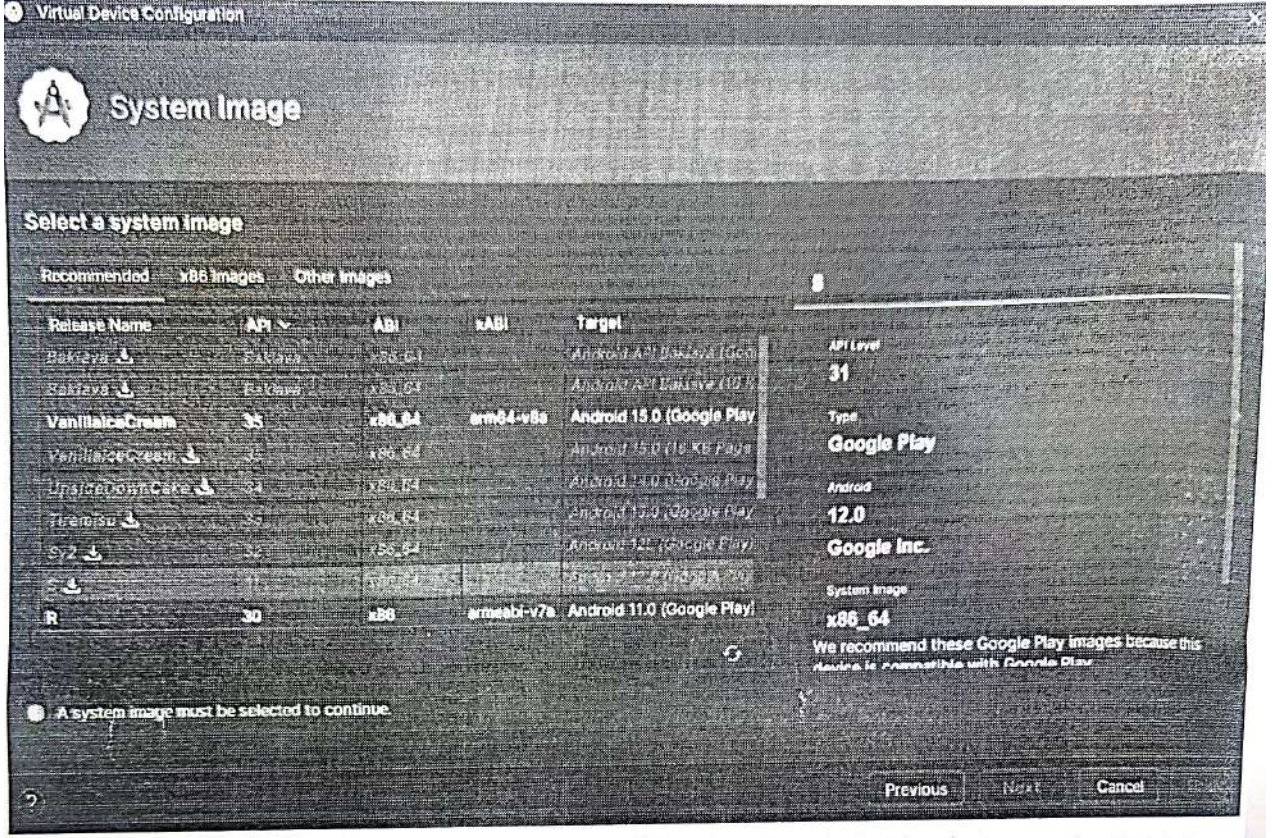
Step 7.4: In the above screen, click Next-> Finish. Once the Finish button is clicked, you need to
choose the 'Don't import Settings option' and click OK. It will start the Android Studio.

Step 7.5: run the $ flutter doctor command and Run flutter doctor --android-licenses command.

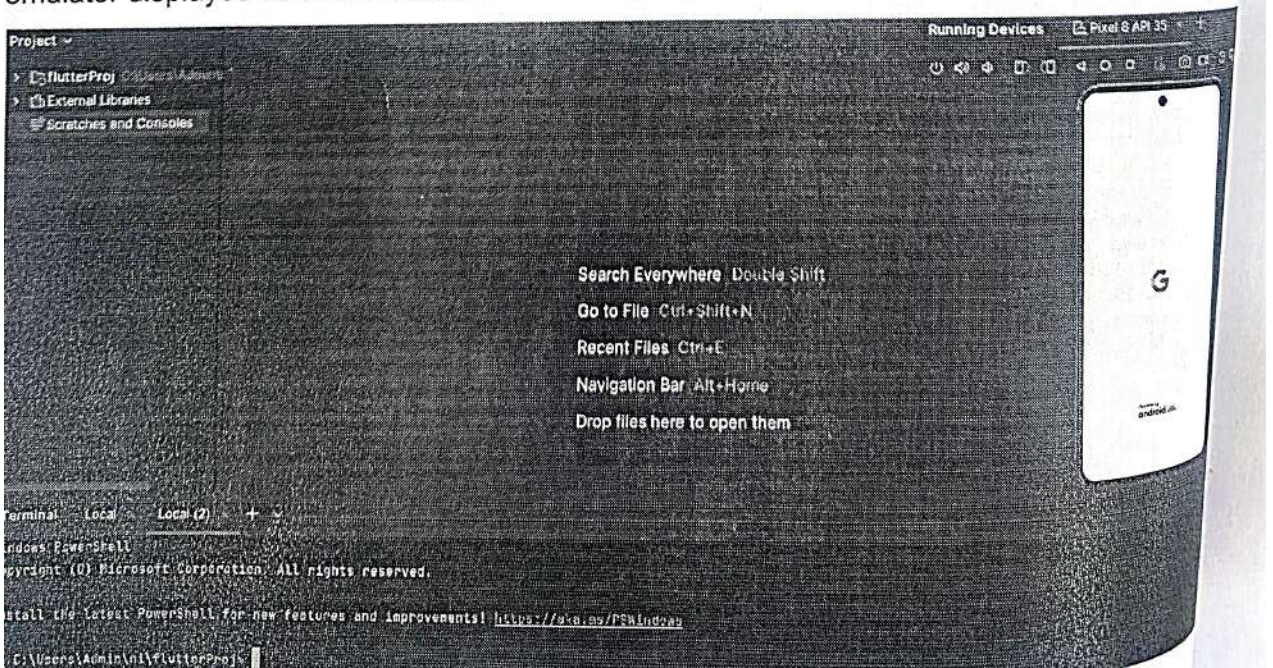
```
C:\Users\Admin>flutter doctor --android-licenses
Command exited with code 128: git -c log.showSignature=false log @{upstream} -n 1 --pretty=format:%ad --date=iso
Standard out:
Standard error: fatal: no upstream configured for branch 'stable'


Warning: Errors during XML parse:
Warning: Additionally, the fallback loader failed to parse the XML.
Warning: Errors during XML parse:       ] 49% Fetch remote repository...
Warning: Additionally, the fallback loader failed to parse the XML.ry...
[======================================] 100% Computing updates...
All SDK package licenses accepted.


C:\Users\Admin>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.27.2, on Microsoft Windows [Version 10.0.22631.4751], locale en-IN)
[√] Windows Version (Installed version of Windows is version 10 or higher)
[√] Android toolchain - develop for Android devices (Android SDK version 35.0.1)
[√] Chrome - develop for the web
[√] Visual Studio - develop Windows apps (Visual Studio Build Tools 2019 16.11.43)
[√] Android Studio (version 2024.2)
[√] VS Code (version 1.96.4)
[√] Connected device (3 available)
[√] Network resources

• No issues found!

C:\Users\Admin>
```

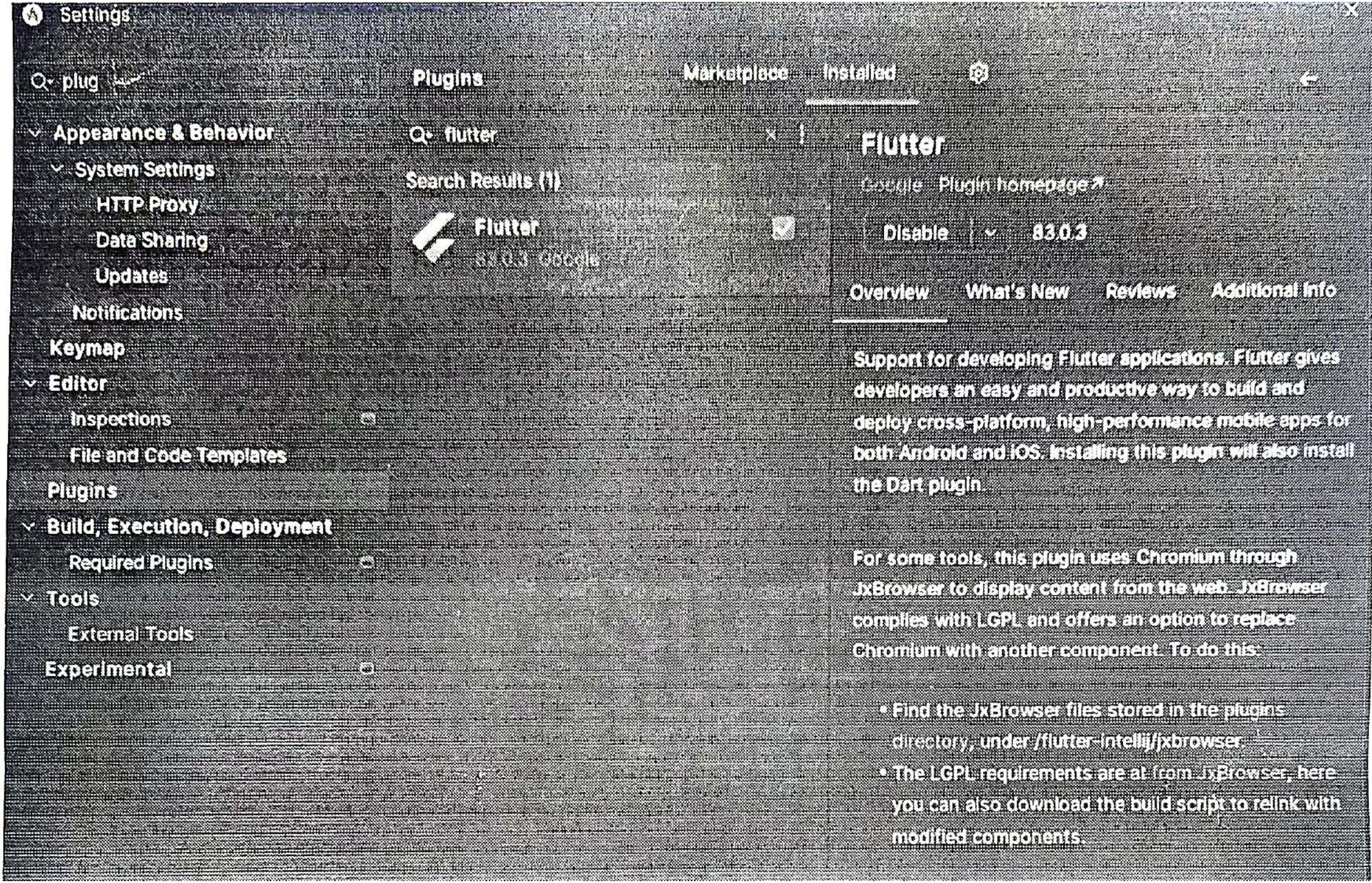
Step 8: Next, you need to set up an Android emulator. It is responsible for running and testir
the Flutter application.

Step 8.1: To set an Android emulator, go to Android Studio > Tools > Android > AVD Manag
and select Create Virtual Device. Or, go to Help->Find Action->Type Emulator in the searc



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




Settings
plug
Appearance & Behavior
System Settings
HTTP Proxy
Data Sharing
Updates
Notifications
Keymap
Editor
Inspections
File and Code Templates
Plugins
Build, Execution, Deployment
Required Plugins
Tools
External Tools
Experimental
Plugins
Marketplace
Installed
flutter
Search Results (1)
Flutter
83.0.3 Google
Flutter
Google Plugin homepage
Disable
83.0.3
Overview
What's New
Reviews
Additional Info
Support for developing Flutter applications. Flutter gives developers an easy and productive way to build and deploy cross-platform, high-performance mobile apps for both Android and iOS. Installing this plugin will also install the Dart plugin.
For some tools, this plugin uses Chromium through JxBrowser to display content from the web. JxBrowser complies with LGPL and offers an option to replace Chromium with another component. To do this:
• Find the JxBrowser files stored in the plugins directory, under /flutter-intellij/jxbrowser.
• The LGPL requirements are at from JxBrowser, here you can also download the build script to relink with modified components.

Shrusti Chintawar
D15B
17

Experiment No 2

Aim: Basic flutter UI design with common widgets. To design & implement flutter interface incorporating common widgets such as text field, buttons, switches sliders & other interactive elements to understand basic flutter

Theory:

procedure :- 1) project setup
create new project on android studio
setup main.dart as an entry point
configure basic material app structure

Implementation steps:

create a stateful widget for the form screen
add the following widget
Image widget
text field
switch widget
slider widget
button widget
check box for terms & conditions

Testing:
Tested widget rendering & layout
verified component interactions

- checked form responsiveness

Results

- properly structured widget tree
- working form elements
- responsive layout with proper spacing

Conclusion: In this lab, experiment we acquainted with the fundamental concept of programming language & learnt to implement custom UI components using

Experiment No: 02

Aim: To Design Flutter UI by Including Common Widgets

Theory:

Flutter is an open-source UI software development kit (SDK) created by Google, used to develop applications for mobile, web, and desktop from a single codebase. It provides a wide range of widgets that help in building flexible and visually appealing UIs.

Common Flutter widgets include:

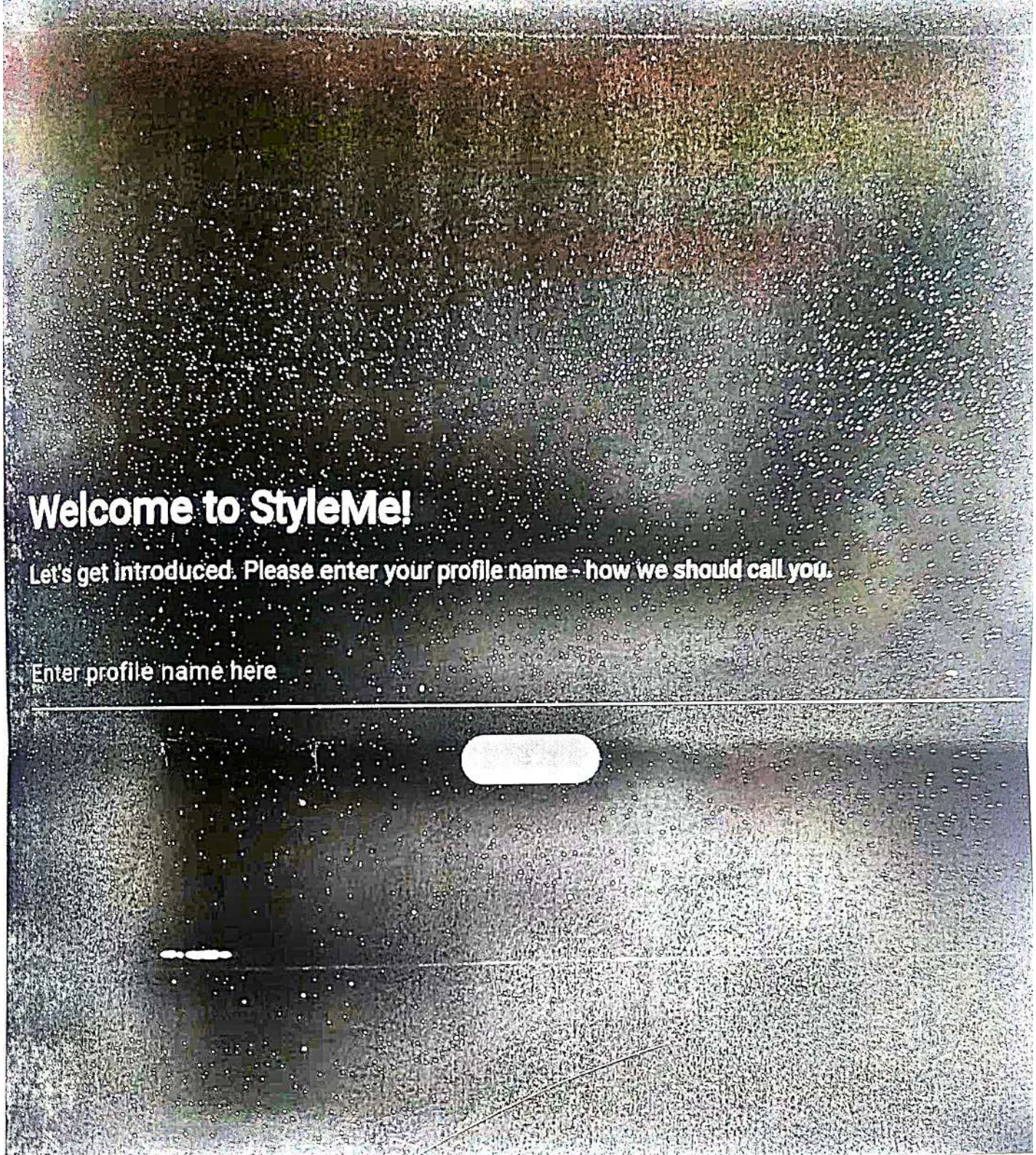
1. Scaffold – Provides the basic structure of an app, including an AppBar, FloatingActionButton, and a body.
2. Container – A versatile widget for designing UI components with padding, margin, color, and decoration properties.
3. Row and Column – Used to arrange widgets horizontally and vertically, respectively.
4. ListView – A scrolling widget used to display a list of items dynamically.
5. Text and TextStyle – Used for displaying and styling text in the application.
6. Image – Displays images from assets, network, or memory.
7. ElevatedButton – A material design button that responds to user interaction.
8. Navigator – Helps in navigating between different screens in the application.

Code:

```
import 'package:flutter/material.dart';


class WelcomePage extends StatelessWidget {

 @override

Widget build(BuildContext context) {

  return Scaffold(

   backgroundColor: Colors.black,

   body: Padding(

    padding: const EdgeInsets.symmetric(horizontal: 20.0),
```

```
ild: Column(
mainAxisAlignment: MainAxisAlignment.ce
crossAxisAlignment: CrossAxisAlignment.start,
children: [
  SizedBox(height: 40),
  Text(
    "Welcome to StyleMe!",
    style: TextStyle(
      color: Colors.white,
      fontSize: 28,
      fontWeight: FontWeight.bold,
    ),
  ),
  SizedBox(height: 10),
  Text(
```

```
hintStyle: TextStyle(color: Colors.blueAccent),
 enabledBorder: UnderlineInputBorder(
   borderSide: BorderSide(color: Colors.blueAccent),
  ),
  focusedBorder: UnderlineInputBorder(
    borderSide: BorderSide(color: Colors.blueAccent, width: 2),
   ),
 ),
),
SizedBox(height: 20),
Align(
 alignment: Alignment.center,
 child: ElevatedButton(
  onPressed: () {},
  style: ElevatedButton.styleFrom(
```


Welcome to StyleMe!
Let's get introduced. Please enter your profile name - how we should call you.
Enter profile name here

MPL lab

Page No.
Date

Shrusti Chintawar
D15B
9

Experiment No 3

Enhances flutter UI with Icons, Images, & custom fonts

AIM:- To include icons, images, fonts in flutter

procedure:

setup & config
add google fonts dependency to pubspec.yaml
configure access directory for images
setup necessary material icon imports

Implementation steps
Enhance UI components with icons
added person icon to name field
added notification icon to switch
added check icon to submit button

Testing
verified proper loading of custom fonts
checked icon rendering & re-alignment
Tested overall UI consistency

Icons- custom icons can be added using packages like flutter
-launcher - icons

Images: flutter supports asset images, network images & sum...

Page No.
Date

Exper

images
The pubspec.yaml file must be updated to inc
images

- fonts: custom fonts can be included by downl
  otf files and specifying them in pubspec.

Results:
- material icons integrated
- customs for called back
- Improved visual hierarchy
- professional looking interface element

Conclusion:
Basic flutter widgets provide a solid foundation
interactive UI
Material design icons & custom fonts significant
UI aesthetics
flutter widget system efficiently handles compon
proper planning of widget structure is crucial
-izable code.

e: Shrusti Chinta
No: 09
i: D15B

To include icons,
r allows develop
m fonts. These
cluding Icons:
er provides built
g Material Icons
(Icons.home, si
ing Custom Icor
ackage:font_awe
on(FontAwesom
. Adding Image
nages can be a
Adding Local Im
Place images in
Define them in
yaml flutter:
assets:
- asse
Use Image.a
Image.asset(
Using Netwo
Image.netw
3. Adding
Flutter allov
Steps to ac
Place font
Define the

Name: Shrusti Chintawar
Roll No: 09
Class: D15B

Experiment No: 03

Aim: To include icons, images, fonts in Flutter app **Theory:**

Flutter allows developers to enhance their app's UI by incorporating icons, images, and custom fonts. These elements help in improving the app's aesthetics and user experience.

1. Including Icons:

Flutter provides built-in Material Icons and supports custom icon sets.

Using Material Icons:
Icon(Icons.home, size: 30, color: Colors.blue);

Using Custom Icons (from packages like FontAwesome): import 'package:font_awesome_flutter/font_awesome_flutter.dart';

Icon(FontAwesomeIcons.heart, size: 30, color: Colors.red);

2. Adding Images:

Images can be added from local assets or loaded from a network.

Adding Local Images:

Place images in the assets folder.

Define them in pubspec.yaml:
yaml flutter:
assets:

- assets/images/logo.png

Use Image.asset in the app:
Image.asset('assets/images/logo.png', width: 100, height: 100);

Using Network Images:
Image.network('https://example.com/image.png', width: 100, height: 100);

3. Adding Custom Fonts:

Flutter allows integrating custom fonts for a unique design.

Steps to add a custom font:

Place font files in the assets/fonts folder.

Define them in pubspec.yaml:

localhost:63203

localhost:63203

StyleMe

Username or Email

Password

Forgot password?

Sign in

or

Sign in with Google

New to StyleMe ? Create Account

Shrusti Chintawar
D15B

Experiment No 4

Aim: Create an interactive form using form widget

Theory: Introduction to forms in flutter

forms in flutter allows user to enter & submit data. The form widget is used to group multiple form fields such as TextFormField, & manage their validation & submission

Components of flutter form

global key <formstate>: helps in managing the form's state

TextformField: Allows users to Input Text

Validation logic: It should include validation & submit the form

Submit button: used to validate & submit the form

Implementation steps:

Create a form with a global key <form state>

Add a TextField with validation logic

Create a button to validate & submit the form

Conclusion: This experiment demonstrates how to create an interactive

FOR EDUCATIONAL USE

te.dart

rt 'package:flutter/material.dart';

s Create extends StatelessWidget {
override
idget build(BuildContext context) {
return Scaffold(
appBar: AppBar(
title: Text("Create Account"),
backgroundColor: Colors.blueAccent,
),
body: Padding(
padding: const EdgeInsets.all(20.0),
child: Column(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
Text(
"Create Acccunt Form",
style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold),
),
SizedBox(height: 10),
Text("Complete your profile by filling in this account creation form."),
SizedBox(height: 20),
TextField(
decoration: InputDecoration(labelText: "First Name"),
),
TextField(
decoration: InputDecoration(labelText: "Last Name"),
),
TextField(
decoration: InputDecoration(labelText: "Email"),
),
TextField(
decoration: InputDecoration(labelText: "Birthdate"),
keyboardType: TextInputType.datetime,
),
TextField(
decoration: InputDecoration(labelText: "Preferred Username"),
),
SizedBox(height: 20),
Text("Do you want to receive updates by email?"),
Row(

Create Account Form

Complete your profile by filling in this account creation form.

First Name

Last Name

Email

Birthdate

Preferred Username

Do you want to receive updates by email?

- ○ Yes
- ○ No

Create Account

Experiment No 5

Page No.
Date

Aim: To apply navigation, routing & gestures in flutter app

Theory: Navigation in flutter:

Navigation in flutter refers to switching b/w different screen (pages) in an application. It is managed using navi-gator widget, which maintain a stack of routes.

Types of Navigation

1. Push Navigation :- (Navigator, push) add new screen on top.
2. pop navigation :- (Navigator.pop) removes the current screen
3. Named routes :- (Navigator.pushNamed) navigate using pre-defined routes name

Routing in flutter:

Routing is the mechanism that determines how a user navigates b/w diff pages.

Types of routing
1. Imperative
2. declarative

Gesture in flutter:

flutter provide the gesturedetector widget to handle user touch interactions like tapping, swaping & dragg-ing

1. Tap gesture
2. double tap gesture

Conclusion:
Navigation allows smooth transition between
Routing makes the app structured & easier to
gesture enhance user interaction by enabling
navigation

Name – Shrusti Chintawar

Roll no -09

Aim: To apply navigation,

Theory:

Navigation, routing, and gestu
applications. Navigation allows
which manages a stack of rou
push and pop methods. Name
more manageable.

Gestures enhance user intera
GestureDetector widget. Com
navigation, and pressing for a
interactions smooth and intuiti

By effectively implementing na
dynamic, resulting in seamles

Conclusion:

Implementing navigation, ro
enabling smooth screen tra
simplify navigation, while th
integrating these features e

Name – Shrusti Chintawar

Roll no -09

MAD Ex 5

Aim: To apply navigation, routing, and gestures in a Flutter app.

Theory:

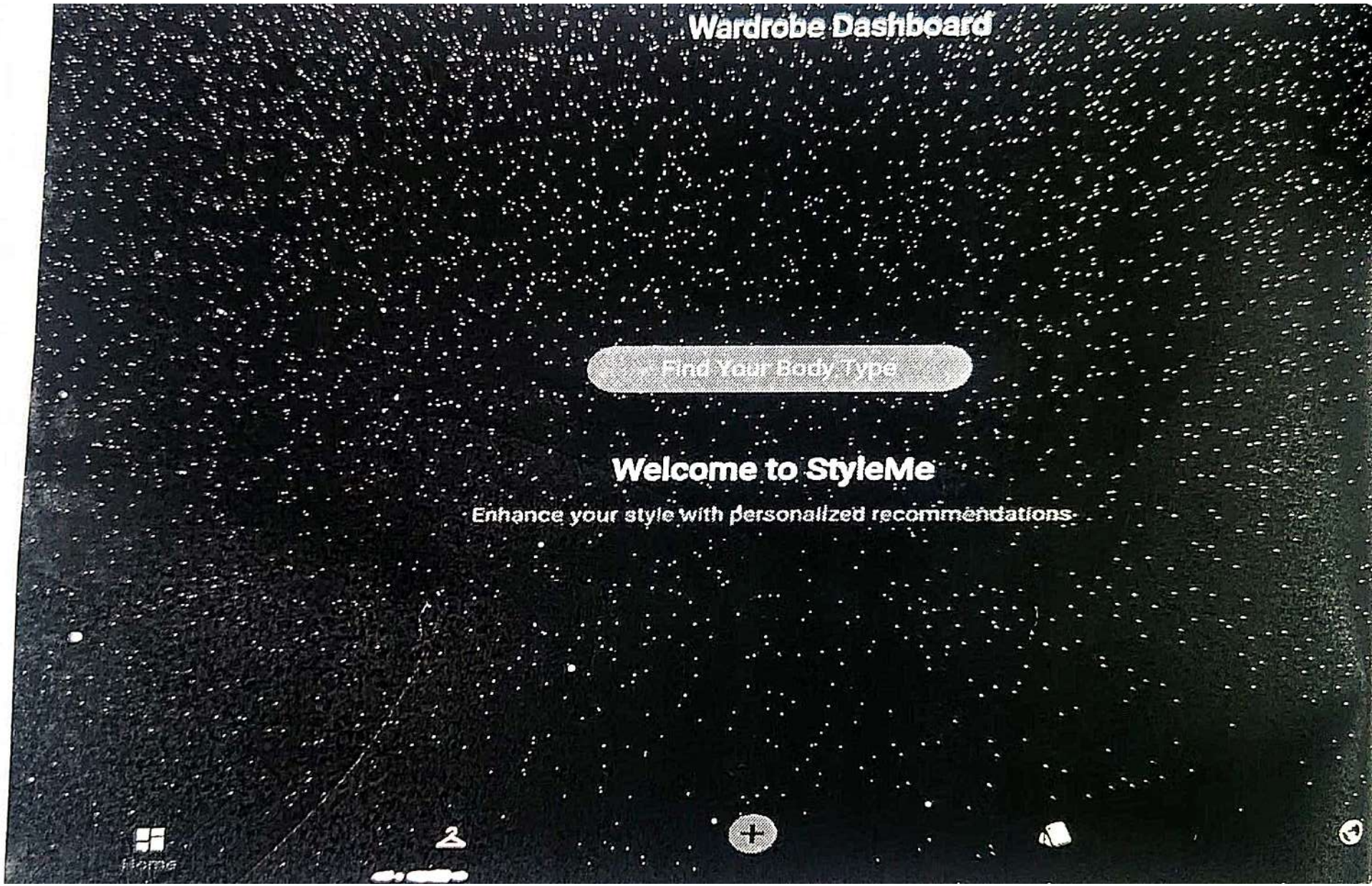
Navigation, routing, and gestures are essential for building interactive and user-friendly Flutter applications. Navigation allows users to move between screens using the Navigator widget, which manages a stack of routes. Routing can be implemented using named routes or direct push and pop methods. Named routes help in organizing navigation, making the app structure more manageable.

Gestures enhance user interaction by detecting taps, swipes, and drags using the GestureDetector widget. Common gestures include tapping for button clicks, swiping for navigation, and pressing for additional actions. Flutter's built-in gesture recognition makes UI interactions smooth and intuitive.

By effectively implementing navigation, routing, and gestures, Flutter apps become more dynamic, resulting in seamless transitions and an enhanced user experience.

Conclusion:

Implementing navigation, routing, and gestures in a Flutter app enhances user experience by enabling smooth screen transitions and interactive controls. The Navigator and named routes simplify navigation, while the GestureDetector widget allows intuitive touch interactions. Properly integrating these features ensures a seamless, responsive, and user-friendly app.


Wardrobe Dashboard
Find Your Body Type
Welcome to StyleMe
Enhance your style with personalized recommendations
Home

Shruti Chintaman
D15B IT 09

AM

Experiment No 6

Aim: setting up firebase with flutter for android & ios apps.

Theory:

What is firebase ?

firebase is a Back-end-as-a-service (Baas) provided by google that helps developers build mobile & web applications with features like authentication, real time database, cloud storage & analytics

Steps to set up firebase in a flutter app

Step 1: create a firebase project

Step 2: add firebase to your flutter app

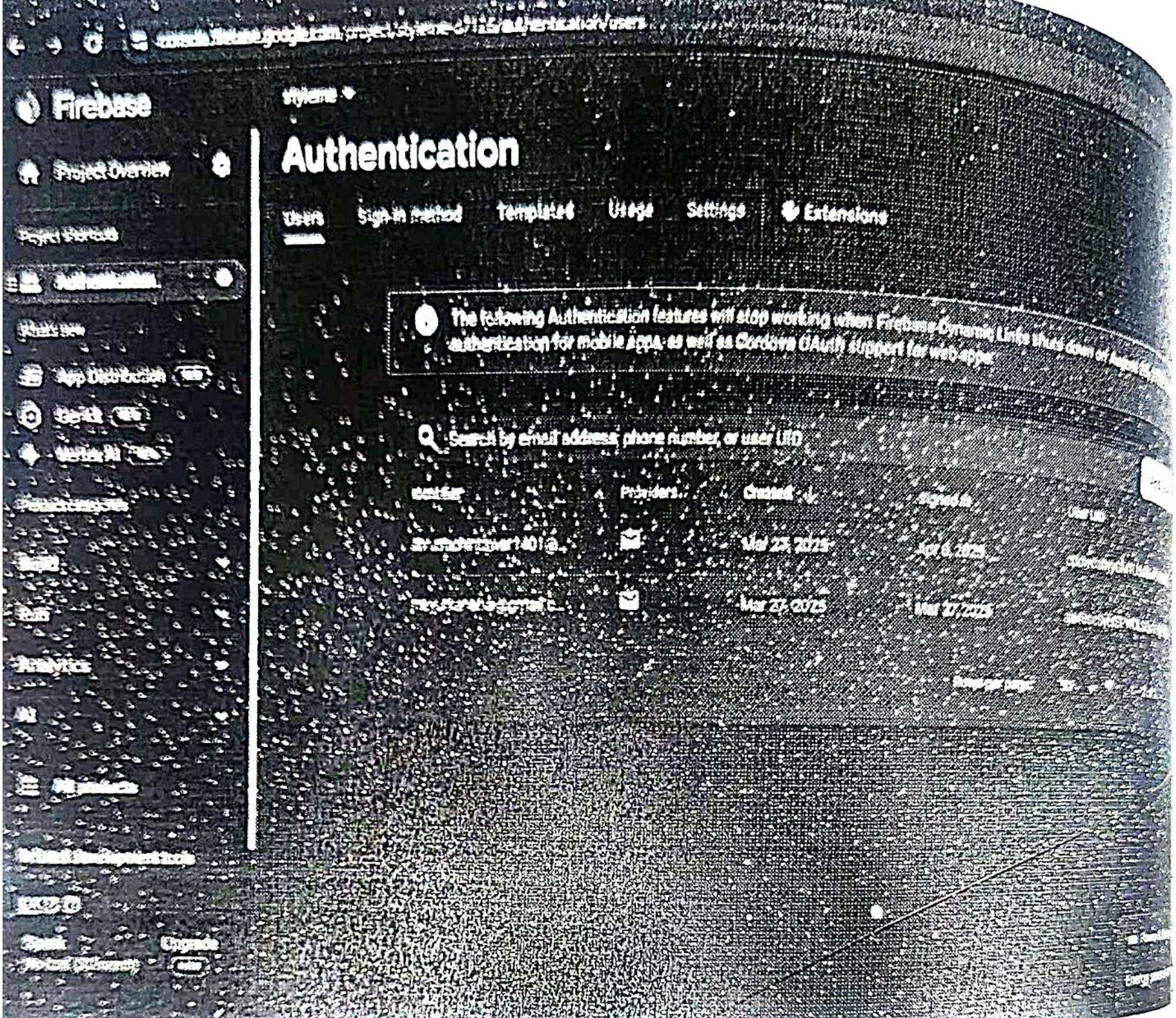
Step 3: Install firebase packages in flutter

Step 4: Initialize —11—

Step 5: verify firebase connection

Conclusion: firebase is a powerful Baas that simplifies backend development for mobile apps The integration process includes creating a firebase project, adding platform-specific configuration files & initializing firebase in

FOR EDUCATIONAL USE


Firebase
Project Overview
Authentication
Users
Sign-in method
Templates
Usage
Settings
Extensions
Search by email address, phone number, or user UID
Providers
Created
Signed In
Mar 27, 2025
Mar 27, 2025

PWA Experiment - 7 DATE:

Aim: To develop a progressive web app (PWA) that can be installed on a device like a native application & provide offline functionality using a service worker & a web manifest file

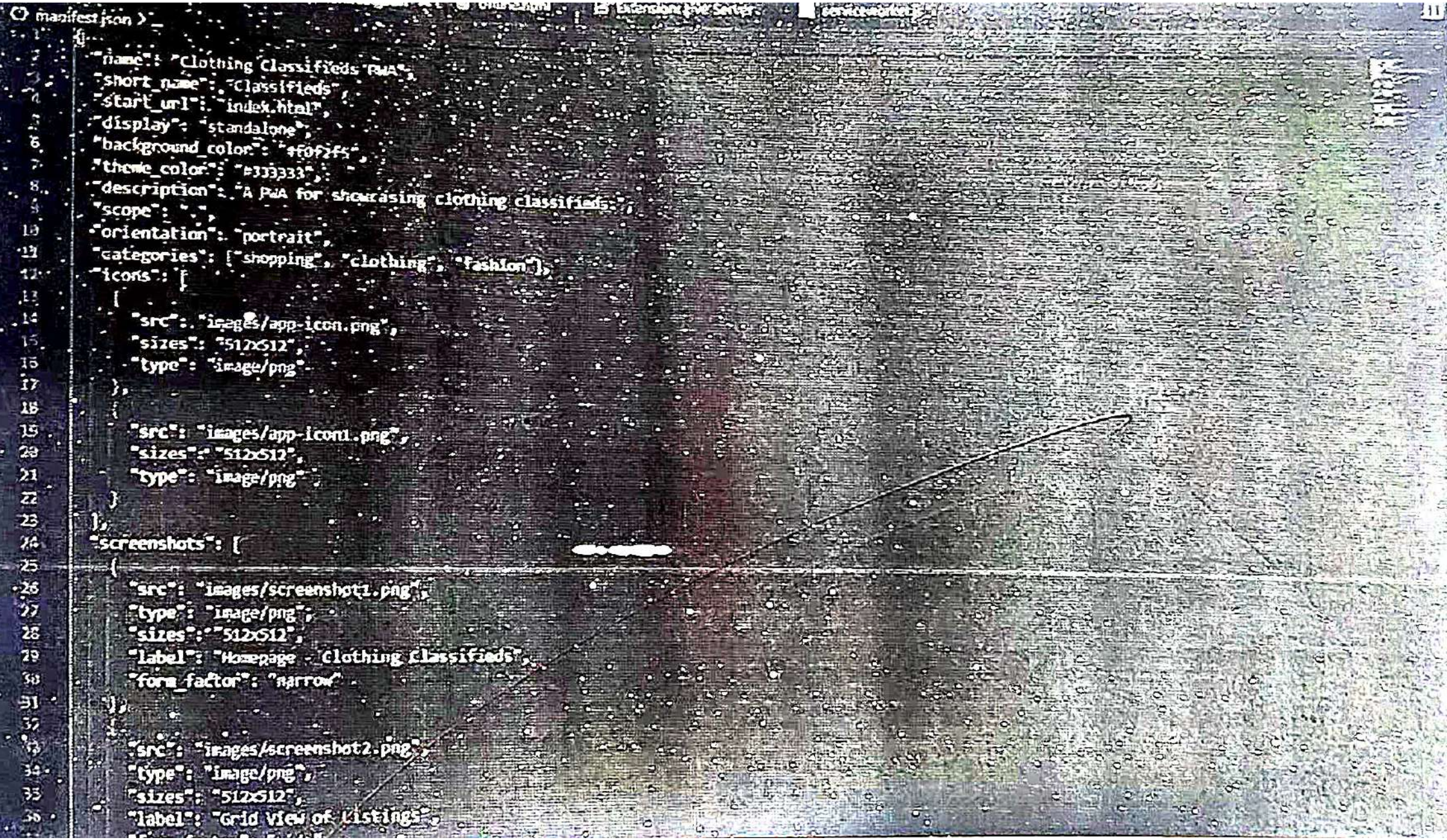
Theory: A progressive web App (PWA) is a modern web application that provides an app like experience on the web. It allows users to install the app on the app on their device, work offline, & load content faster PWA's leverage the following key components

Service worker
A background script that enables offline functionality by caching resources. It intercepts network requests & serves cached responses when offline

Web App manifest
A JSON file that provides metadata about the PWA such as the name, theme, color, start URL & icons. It makes the web app installed on device

Caching strtegy
The cache first approach is used in this PWA meaning that if resources are available in cache they are served first, otherwise they are fetched from the network

Conclusion: The implementation successfully demonstrates the PWA architecture by enabling installation offline access & caching users can install the app & even if there is not internet connected previously loaded pages will still be accessible. The project highlights the benefits of PWA's such as improved performance



manifest.json

```
{
  "name": "Clothing Classifieds PWA",
  "short_name": "Classifieds",
  "start_url": "index.html",
  "display": "standalone",
  "background_color": "#f0f2f5",
  "theme_color": "#333333",
  "description": "A PWA for showcasing clothing classifieds.",
  "scope": "./",
  "orientation": "portrait",
  "categories": ["shopping", "clothing", "fashion"],
  "icons": [
    {
      "src": "images/app-icon.png",
      "sizes": "512x512",
      "type": "image/png"
    },
    {
      "src": "images/app-icon1.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ],
  "screenshots": [
    {
      "src": "images/screenshot1.png",
      "type": "image/png",
      "sizes": "512x512",
      "label": "Homepage - Clothing Classifieds",
      "form_factor": "narrow"
    },
    {
      "src": "images/screenshot2.png",
      "type": "image/png",
      "sizes": "512x512",
      "label": "Grid view of listings",
```


Clothing Classifieds
Men's Denim Jacket
Women's Summer Dress
$35.00
Unisex Hoodie
$40.00
Comfortable cotton hoodie, various colors available. Size L
Women's Leather Jacket
ceWorker
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Service workers
Offline
Update on reload
Service workers from other origins
See all registrations
Start

DATE:

PWA ass Experiment 8.

Aim: To develop progressive Web App (PWA) that enab-
installation offline functionality using a service wo-
-ker and a web manifest file

Theory: A progressive web App (PWA) is a modern
web application that provides an app like experi-
-ence on the web. It allows users to install the
app on their device, work offline & load content
faster. PWAs leverage the following key components

1. Service worker
A background script that enables offline functionalit
by caching resources. It intercepts network requests
& serves cached response when offline

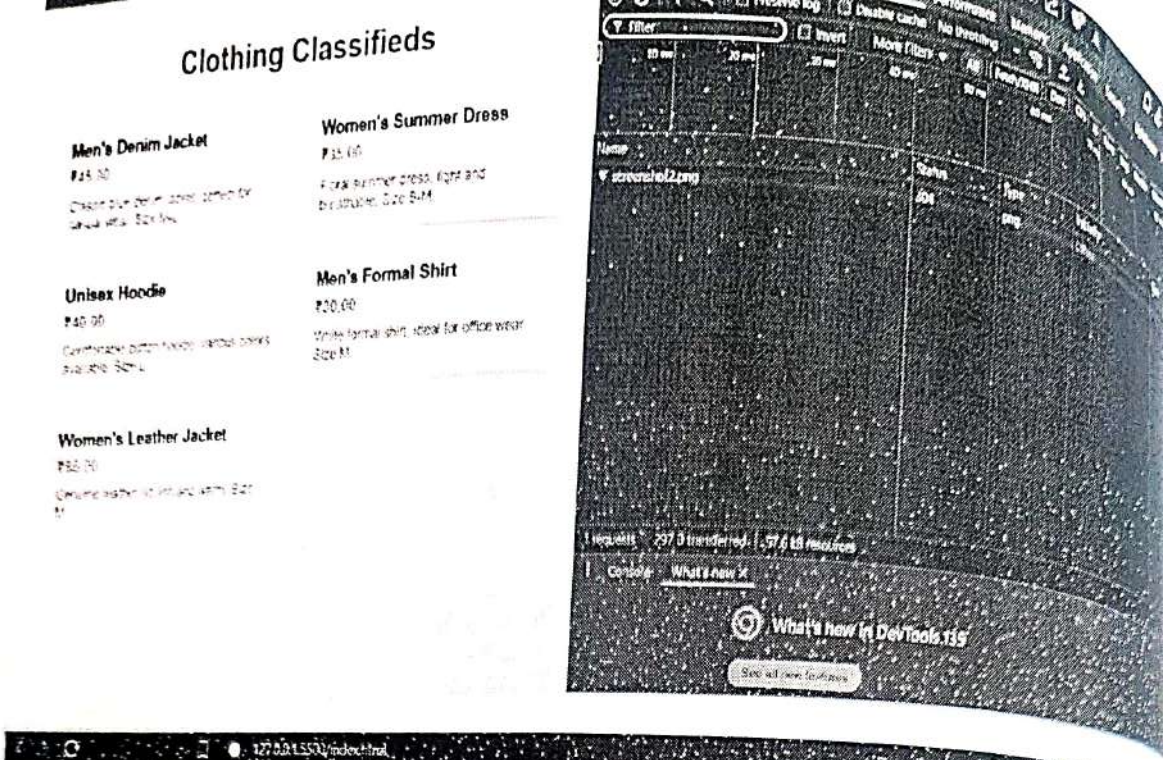
2. Web App Manifest
A JSON file that provides metadata about the PWA such
as the app's name theme start URL & icons. It
makes the web App installed on device

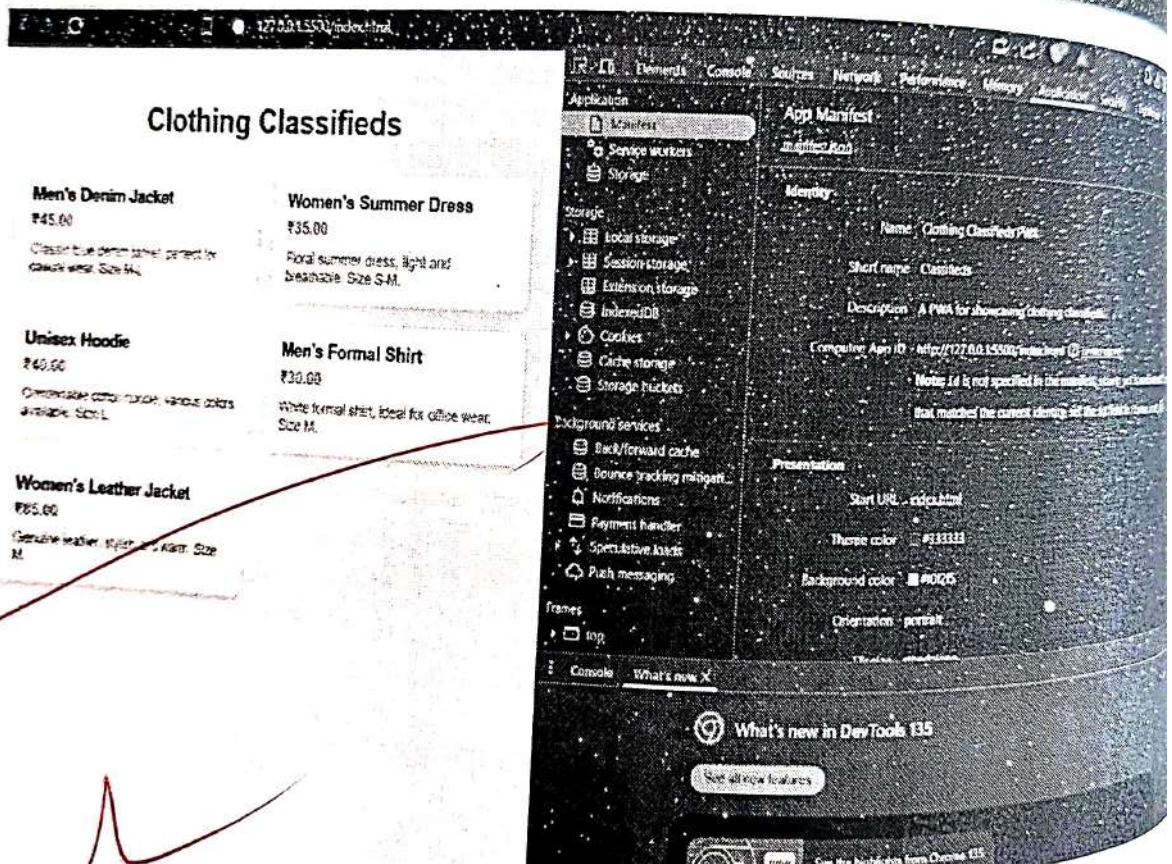
3. Caching Strategy
The cache first approach is used in this PWA meaning
that if resources are available in cache they are
served first otherwise they are fetched from the
network

Conclusion: The Implementation successfully demonstr-
-ates the PWA architecture by enabling installation
offline access & caching users can install the app
& even if there is no internet connection previously
loaded pages will be still accessible. The project high-
-lights the benefits of PWA such as improved performance

FOR EDUCATIONAL USE


Welcome
index.html
manifest.json
offline.html
Extension: Live Server
serviceworker.js
offline.html > html > body > p
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <title>Offline</title>
6 </head>
7 <body style="text-align:center; font-family:sans-serif;">
8 <h1>You're Offline</h1>
9 Please check your internet connection.
10 </body>
11 </html>
12


Clothing Classifieds
Men's Denim Jacket
Women's Summer Dress
Unisex Hoodie
Men's Formal Shirt
Women's Leather Jacket
What's new in DevTools 135


Clothing Classifieds
Men's Denim Jacket
₹45.00
Women's Summer Dress
₹35.00
Floral summer dress, light and breathable. Size S-M.
Unisex Hoodie
₹40.00
Men's Formal Shirt
₹30.00
White formal shirt, ideal for office wear. Size M.
Women's Leather Jacket
₹85.00
What's new in DevTools 135

09

MPL Exp 9

DATE:

Aim: To implement & understand service worker events like fetch sync & push for an E-commerce PWA

Theory: Service worker is a background script that runs independently in a browser & enables functionalities like offline caching background sync & push notifications. It acts as proxy between the web application & the network enhancing performance, reliability & user experience

Key features of Service Workers:

- Runs in the background without interaction
- Intercepts network requests & manages caching using the cache API
- Works only over HTTPS for security reasons
- Uses promises & indexed DB to handle data persistence

Fetch event: Manages network requests & caching

Cache first: Uses cached data first, fetching from the network only if needed

Network first: Tries fetching from network first, falls back to cache if offline

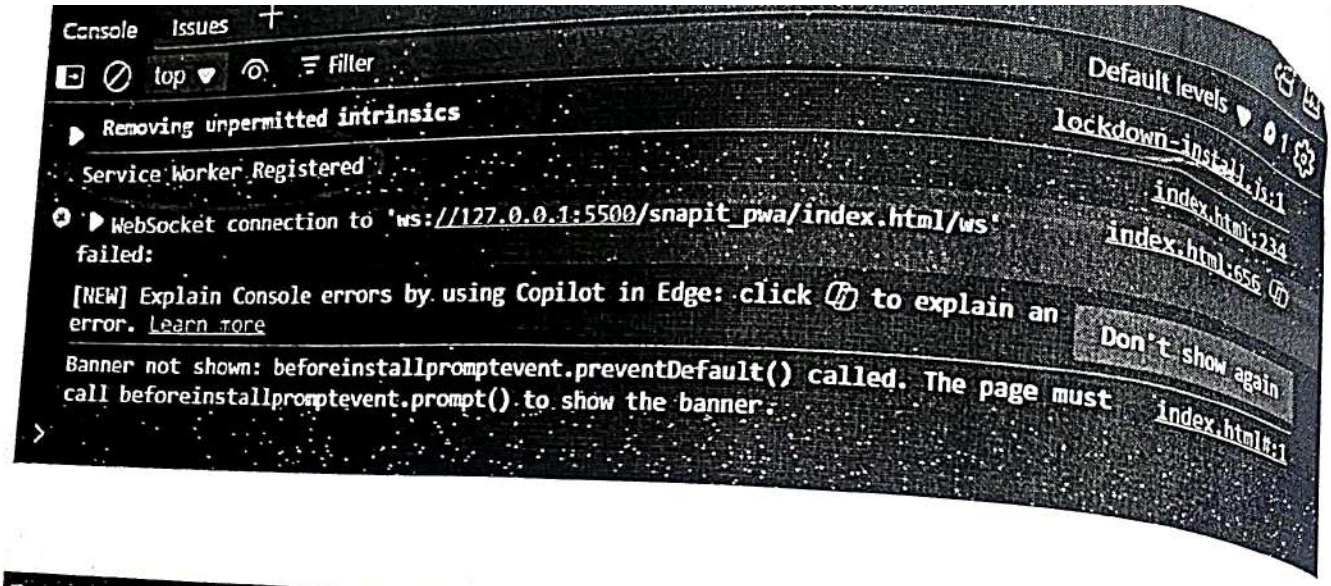
Sync event: enables background sync when internet is restored

Push event: handles push notification for updates

Conclusion: Thus we successfully implemented service worker events fetch, sync & push in PWA enabling offline functionality

FOR EDUCATIONAL USE

```
// Activate event
self.addEventListener("activate", event => {
  event.waitUntil(
    caches.keys().then(keys =>
      Promise.all(
        keys.map(key => {
          if (key !== staticCacheName) {
            return caches.delete(key);
          }
        })
      )
    )
  );
});

// Fetch event
self.addEventListener("fetch", event => {
  event.respondWith(
    caches.match(event.request).then(response => {
      return response || fetch(event.request).catch(() => caches.match('./offline.html'));
    })
  );
});
```



You're Offline

Please check your internet connection.

Conclusion: We implemented the functionality of offline web cache capture so that in the absence of a stable internet connection, the app would display a generic waiting page.

MPL Exp 10

Aim: To study & implement the development of an e-commerce PWA to github pages

Theory: github pages is a free hosting service provided by github it deploying static pages directly from repository It is used for hosting personal projects, documentations & simple web applications

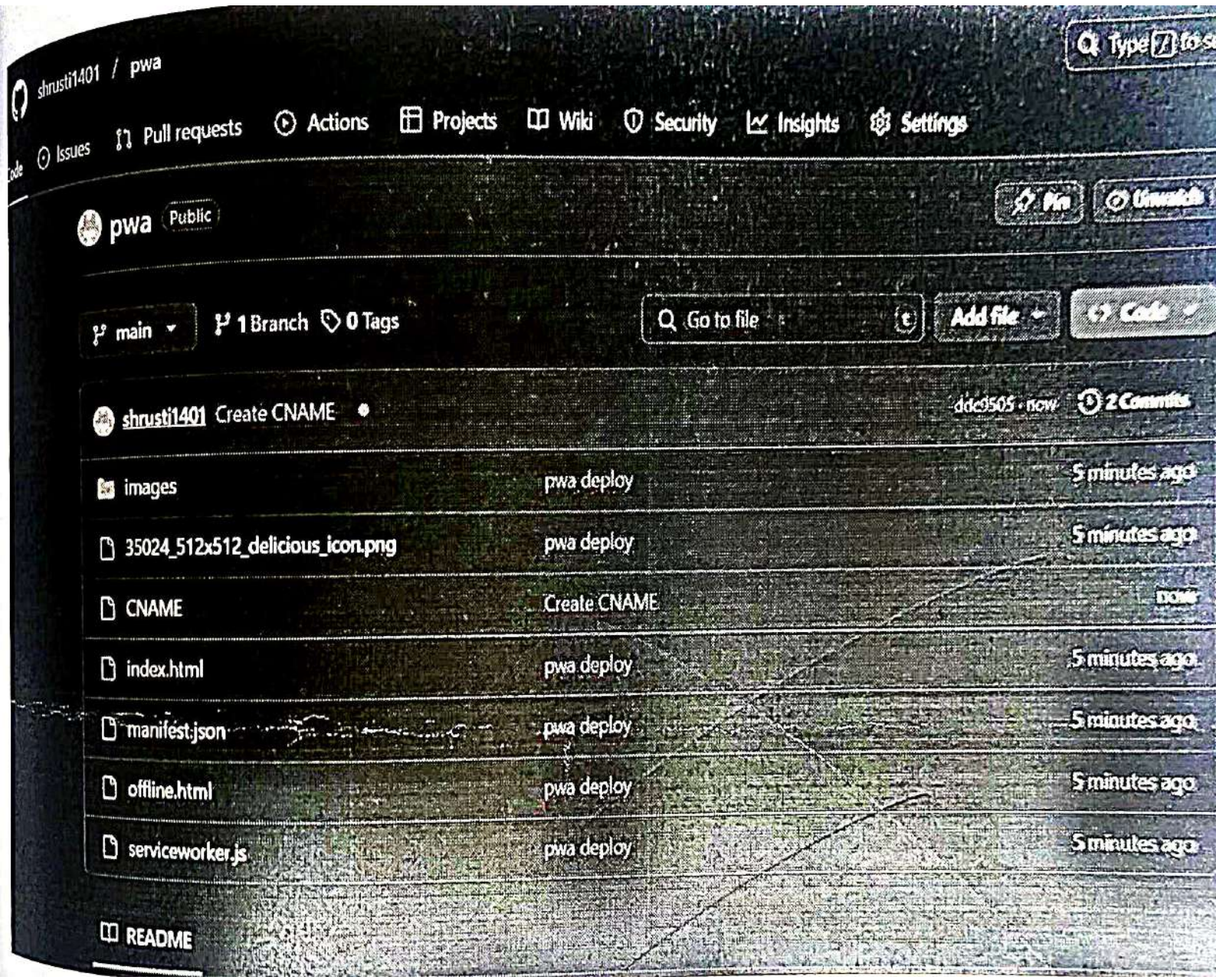
- features of github pages
supports jekyll blogging
allows custom URLs
provides an automatic page generator

- pros of github pages
free & easy to use
direct integration with github repositories
supports custom domains

cons of github pages
code is public unless using a paid pvt repository
limited plugin support for jekyll

firebase as an alternative
firebase is a realtime backend platform by google offering cloud based storage & authentication services It allows real time data synchronization & is widely used for dynamic applications

Conclusion: Thus we successfully deployed our e-commerce PWA using github pages, leveraging its free & easy to setup for static web hosting


shrusti1401 / pwa
Type / to s
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
pwa
Public
Pin
Unwatch
main
1 Branch
0 Tags
Go to file
t
Add file
Code
shrusti1401 Create CNAME
ddc9505 · now
2 Commits
images
pwa deploy
5 minutes ago
35024_512x512_delicious_icon.png
pwa deploy
5 minutes ago
CNAME
Create CNAME
now
index.html
pwa deploy
5 minutes ago
manifest.json
pwa deploy
5 minutes ago
offline.html
pwa deploy
5 minutes ago
serviceworker.js
pwa deploy
5 minutes ago
README

App Manifest

snapit_pwa/manifest.json

Errors and warnings

⚠ Richer PWA Install UI won't be available on desktop. Please add at least one screensho
form_factor set to wide.

⚠ Richer PWA Install UI won't be available on mobile. Please add at least one screenshot
form_factor is not set or set to a value other than wide.

Identity

Name	SnapShop
Short name	SnapShop
Description	A QR-based shopping app
Computed App ID	https://swayamraut8.github.io/snapit_pwa/index.html ⓘ Learn

Note: `id` is not specified in the manifest, `start_url` is used ins
specify an App ID that matches the current identity, set the `id` t
`/snapit_pwa/index.html` .

Presentation

Start URL	index.html

MPL Exp 11

Aim: To study & implement the google lighthouse PWA analysis tool to test the progressive web App (PWA) functioning

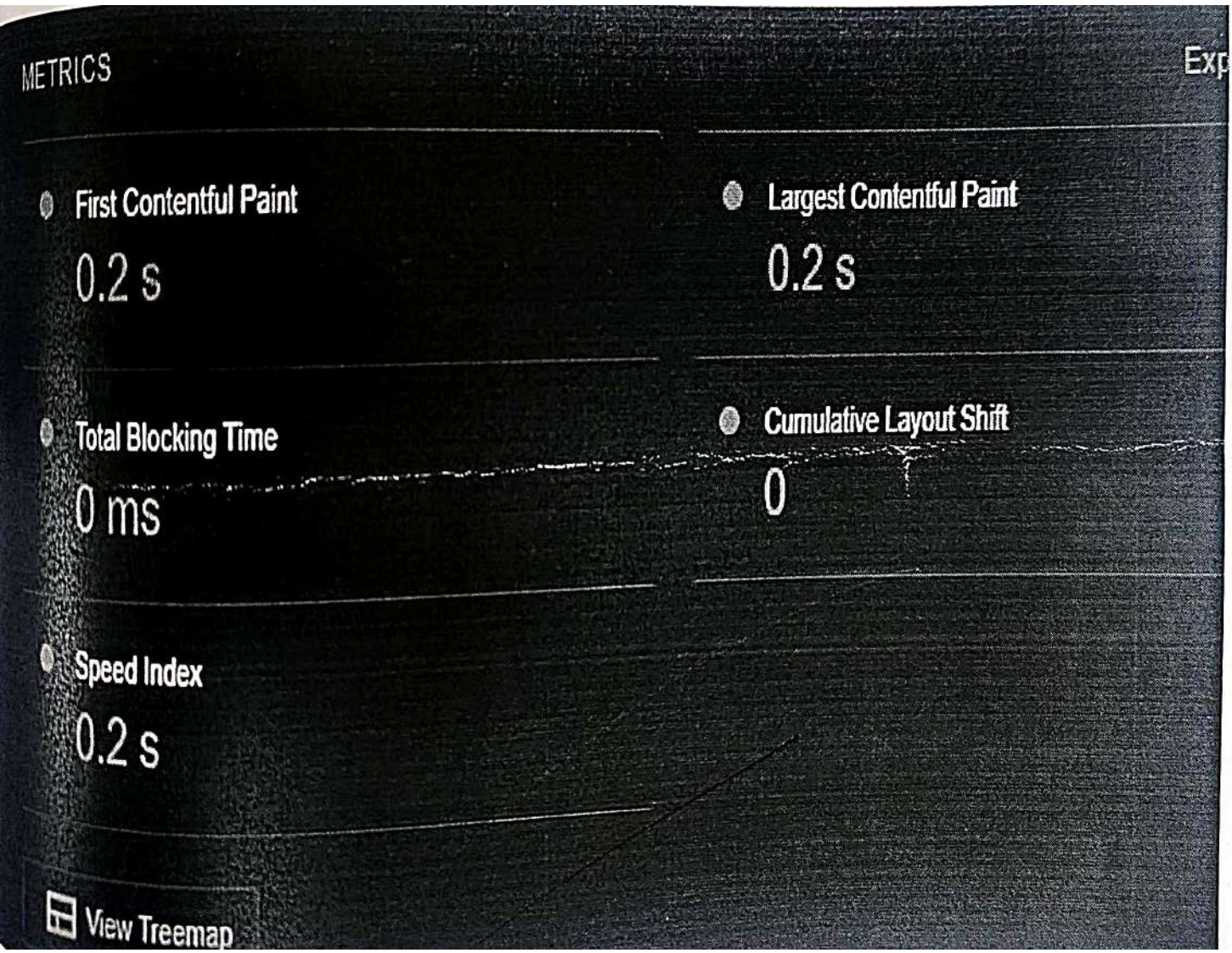
Theory: google lighthouse is an open source tool by google used to audit web applications based on various parameters including performance PWA compliance, accessibility & best practices. It helps developer analyze & optimize web apps to function like native applications

key audit metrics

1) performance - Measures loading speed, rendering time & responsiveness
2) PWA compliance - Ensures the app meets PWA standards such as service workers, offline functionality & manifests
accessibility - evaluates the apps usability, including screen - reader compatibility
Best practices - check HTTPS usage, security policies & deprecated code elements

lighthouse provides actionable insights to improve website efficiency, ensuring a better user experience & higher engagement

Conclusion :- Thus we successfully used the google lighthouse PWA analysis tool to evaluate & optimize the PWA improving its performance, accessibility & compliance with best practices


METRICS
First Contentful Paint
0.2 s
Largest Contentful Paint
0.2 s
Total Blocking Time
0 ms
Cumulative Layout Shift
0
Speed Index
0.2 s
View Treemap

How service workers use IndexedDB?

opening the database:

```
let db;
let request = indexedDB.open('MyDatabase', 1);
    request.onsuccess = function(event) {
        db = event.target.result;
    }
```

creating a store & adding Data

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

fetching data in service worker

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

    let getUser = service.get(1);
    getUser.onsuccess = function() {
        console.log(getUser.result);
    };
```

```
cache.open('app-cache')
return cache.addAll(['/', 'index.html',
});
);
});
```

iii) Activation phase
The old services worker is replaced with new one request-serves cached files & types data

Q4) Explain the use of indexed DB in the services worker for storage

Soln: use of Index DB in service worker for data storage:
DB is a browser database that stores large amount structured data like JSON objects It helps pwfs wo -line by saving & retrieving data efficiency

why use IndexedDB in service workers?

i) offline support - store data when offline & sync it lat
ii) efficient storage - saves structure data like user sett cart items or form inputs
iii) faster access - retrieves data quickly without ne a network request
iv) persistent data - data remains saved even after th browser is closed

key difference:
responsive adopts dynamically to all screens
fluid resizes smoothly but may not be fully optimized
adaptive loads different layout based on device type

describes the lifecycle of service workers, including registration, initialization & activation phases
lifecycle of service workers: A service worker is a script that runs in the background and helps a web app work offline, load faster & send push notifications its lifecycle has 3 main phase

Registration phase: The browser registers the service worker using Javascript

```
if ('serviceWorker' in navigator) {
    navigator.serviceWorker.register
    .then() => console.log("service registered")
    .catch(error => console.log("register failed:",
                                error));
```

Installation phase
The service worker downloads necessary files (HTML, CSS, JS) and
If successful, it moves to the activation phase

code eg
self.addEventListener('install' event => {

performance	fast with service workers	fast with new installation
updates	Automatic	Manual
development cost	lower	higher

Q2) define responsive web design and explain its importance in the context of progressive web. Apps compare contrast responsive, fluid, & adaptive web design approaches

Soln:- definition of responsive web design :-
Responsive web design (RWD) is a technique that web pages adjust automatically to different screen & devices it ensures good user experience on mobile tablet) & desktops without needing separate

Importance of Responsive design in PWAs:
i) better user experience
ii) faster load time
iii) SEO benefits
iv cost effective

Comparisons:

Approach	How it works	pros	cons
Responsive	CSS media queries & flexible grids	works on all devices	can b comp

separate teams for ios & android

3) Consistent UI: flutter renders everything using its own engine, ensuring a uniform look across devices

Q16. Discuss how flutter framework differs from traditional approaches & why it has gained popularity among developer community?

Ans: flutter uses a single codebase for multiple platforms. Unlike traditional native development that requires separate code for ios (swift) & android (kotlin), flutter does not rely on platform-specific UI components instead renders everything using its own skia graphics engine, ensuring consistency unlike react native which uses a Javascript bridge, flutter compiles directly to native ARM code, offering better performance. Its hot reload feature allows developer to see changes instantly, making development faster & more efficient.

flutter has gained popularity due to its fast development, cost efficiency & cross platform support. Businesses prefer it as it reduces development time & costs while delivering high performance apps. Its customizable widget system ensures a smooth, native like experience.

Q29) Describe the concept of the widget tree in flutter & explain the widget composition is used to build complex UI.

Date
Page

Describe the concept of the widget tree in flutter explain the widget composition is used to build complex UI.

In flutter everything is a widget ~~tree~~ in (button, text, layouts etc) These widgets are arranged in a hierarchial structure known as the widget tree. The widget tree determines the UI.

idget composition to build complex UI:
lutter encourages a composition-based approach
ather than inheritance
nstead of creating large, monolithic widget, developer
uild small, reusable widget that are combined to
rm complex UIs.
L. A column widget can hold multiple Text & button
idget, creating a structured widget

provide on of commonly used widgets & their role in
reating a widget tree
1) structural widget
Scaffold: provide basic structure of a screen
Container: used for layout styling
Column & row: used for vertical & horizontal layo-
ut

) Interactive widget
Textfield: for user input.
Elevated button: clickable buttons

Teacher's Sign.:

iii) Styling widget
- padding: adds spacing around widget
- align, center: adjust alignment

iv) List & scrollable widget
- ListView: scrollable widget
- GridView: provide / display items in grid

eg: Simple widget Tree

```
Scaffold (
  appBar: AppBar (title: Text (" flutter App
body: column (
  children: [
  Text (" welcome to flutter ! "),
  ElevatedButton (on pressed: () {}, child: Te
  click Me" ))
  ],
  );
  );
```

Q30) Discuss the importance of state management in flutt
-licat

Ans: Importance of State management in flutter applic
State management refers to handling dynamic da
that changes over time
In flutter, the UI rebuilds when the state chan
ensuring the app remains interactive & respons
proper state management helps in performance op
code maintainability & better UI behaviour

Page No.
Date

Compare and contrast the different state managment in flutter approaches available in flutter, such as setstate, provider & Riverpod, provide scenarios where each approach is suitable

comparison of state management approaches in flutter

approach discription suitable scenarios setstate basic state managment by calling setstate() to update UI. small apps, simple UI updates (eg., toggling a switch) provider user inherited widget to efficiently manage state across the widget tree Medium sized apps needing global state sharing (eg, user authentication) RiverpodMore scalable than provider with improved dependency injection & state handling. large, complex apps requiring modular & scalable state managment (eg. e-commerce apps)

Explain the process of integrating firebase with a flutter application

Discuss the benifits of using firebase as a backend solution

Integrating firebase with flutter & its benifits

ans: Integration process:

setup firebase console:

create a firebase project

Register the app for android & ios

download & add google-services.json (android) or google-service-info.plist (ios)

install firebase dependencies:

yaml

dependencies:
firebase_core: latest_version
firebase_auth: latest_version
cloud_firestore: latest_version

Initialize firebase in flutter

dart
void main() async {
WidgetsFlutterBinding.ensureInitialized();
await Firebase.initializeApp();
runApp(MyApp());
}

Benefits:
No need to manage servers (backend-as-a service)
-wide authentication, database & cloud fu
scalable & cost-effective

Q4b. Highlight the firebase services commonly used
flutter development & provide brief overvie
how data synchronization is achieved

Ans: commonly used firebase services in flutter &
synchronization service functionality
firebase Authentication: user-sign in (Email,
facebook) cloud firestore: NoSQL database for
time data syncing service: firebase storage
manage files (images, videos) cloud messaging
notifications, firebase analytics: app usage anal

Page No.
Date

Data synchronizatn in firebase:
firestore allows real time data syncing using snashot listner

eg: for real-time listner in firestore
dart

• Firebase.firestore.instance.collectn ('message').snapshots().
listen ((snapshot) {
for (var doc in snapshot.docs) {
print (doc.data());
}
});

MPL assignment 2

04/05

Page No.
Date

Define progressive web App (PWA) & explain its significance in modern web development. Discuss the key characteristics that differentiate PWAs from traditional mobile apps.

A progressive web app (PWA) is a type of web application that combines the best features of both traditional web pages & native mobile apps.

It leverages modern web technologies such as service workers, manifests, & responsive design, to deliver a fast reliable & engaging user experience across different devices

Significance of PWAs in modern web development.

PWAs have gained popularity in modern web development due to their ability to provide app like experiences without requiring installation from an app store.

Their significance includes

cross-platform compatibility - works seamlessly across desktops tablets & mobile devices

Improved performance: caching & efficient loading reduce page load times

offline functionality - service workers enable offline access, enhancing user engagement

difference in PWA & Traditional Apps

features	PWA	Traditional app
Installation	direct from browser	download from app store
Internet req	work offline with caching	usually requires internet

MPL - Assignment 1

Page No.
Date 4

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

Flutter is a popular open-source UI toolkit developed by google for building natively compiled applications for mobile (ios & android) web, & desktop from a single codebase

key features of flutter:

Single codebase: write once, run on multiple platform (ios, android, web, desktop)

Dart programming language: uses dart, which is optimized for fast performance and ahead-of-time (AOT) compilation

Hot reload: Instantly reflects changes in the app without restarting, making development faster & more efficient

Rich widget library: provides a vast collection of customizable widgets that support material design & cupertino styles for a native look & feel

Advantages of using flutter

faster development Time: hot reload & a single code base reduce development effort & time

cost effective: Since developers write one codebase for multiple platforms, it reduces costs associated with maintaing