Name:- Rohan Ghadge Div:-D15B Roll No.:- 63

Experiment No.:- 2

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

Theory:-

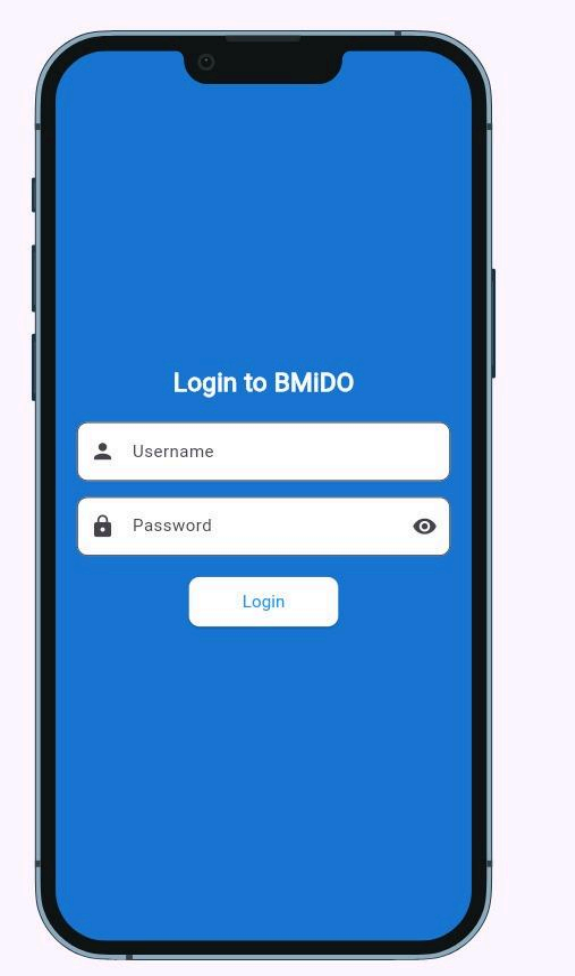
In this experiment, we designed a Flutter-based BMI Calculator application by integrating several commonly used Flutter widgets to create a smooth and functional user interface. The app features widgets like `TextField`, `ElevatedButton`, `IconButton`, `Slider`, `Text`, and layout widgets such as `Column`, `Row`, and `Container`.

Each screen of the app demonstrates practical use of these widgets — from login authentication to inputting data and displaying results. This helps in understanding how to build structured, interactive, and responsive UI components using Flutter's widget-based architecture.

Screenshots:-

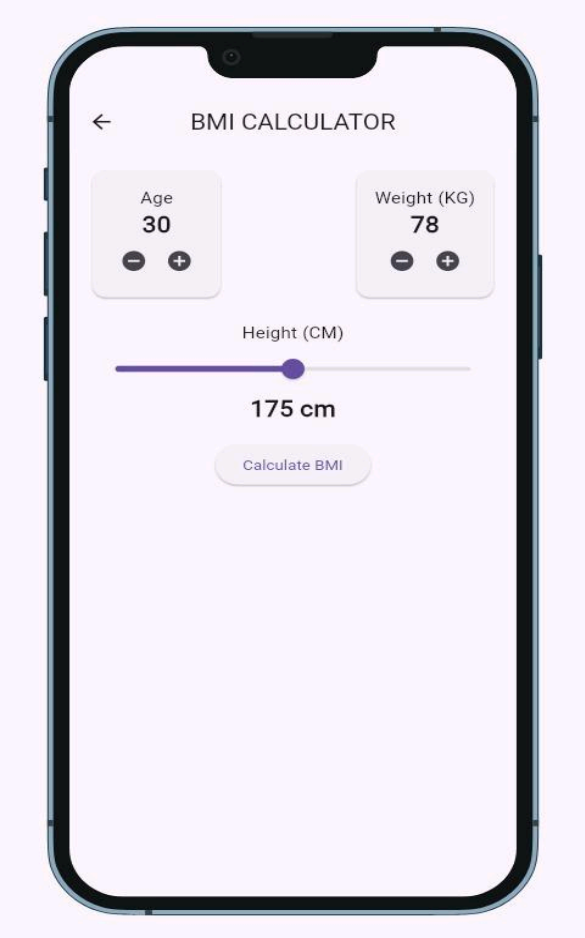
1. Login Screen

- Uses common widgets like `TextField`, `Icon`, `Button`, and layout containers.



2. BMI Input Form

- Implements interactive widgets: `Slider`, `IconButton`, `Text`, and numeric input using structured layout widgets.



Conclusion:-

This experiment helped us explore and implement commonly used Flutter widgets to design an interactive and user-friendly UI. It provided a strong foundation for building structured mobile interfaces efficiently.

Name:- Rohan Ghadge Div:-D15B Roll No.:- 63

Experiment No. 3

Aim:- To include icons, images, and fonts in a Flutter app.

Theory:-

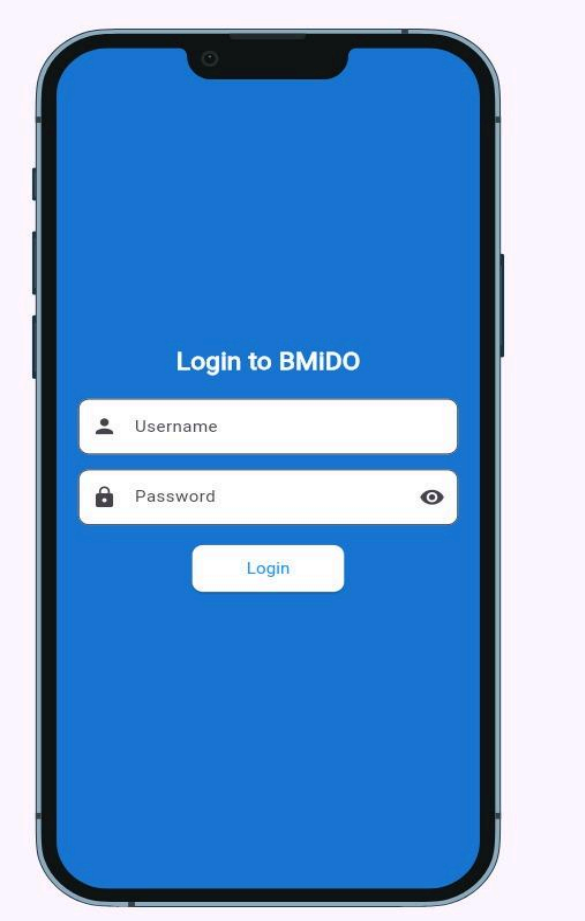
In this experiment, we developed a Flutter BMI Calculator app that demonstrates the use of icons, custom fonts, and clean UI styling to enhance user experience. We used Flutter's built-in `Icon` widget to provide visual cues in input fields and buttons, custom fonts for improved text aesthetics, and structured layout widgets for a polished look.

Icons are used for username, password, and increment/decrement controls, making the UI more intuitive. Fonts contribute to the modern appearance of headings and results. Though no external images were added in this version, the design structure supports image integration for future enhancements.

Screenshots:-

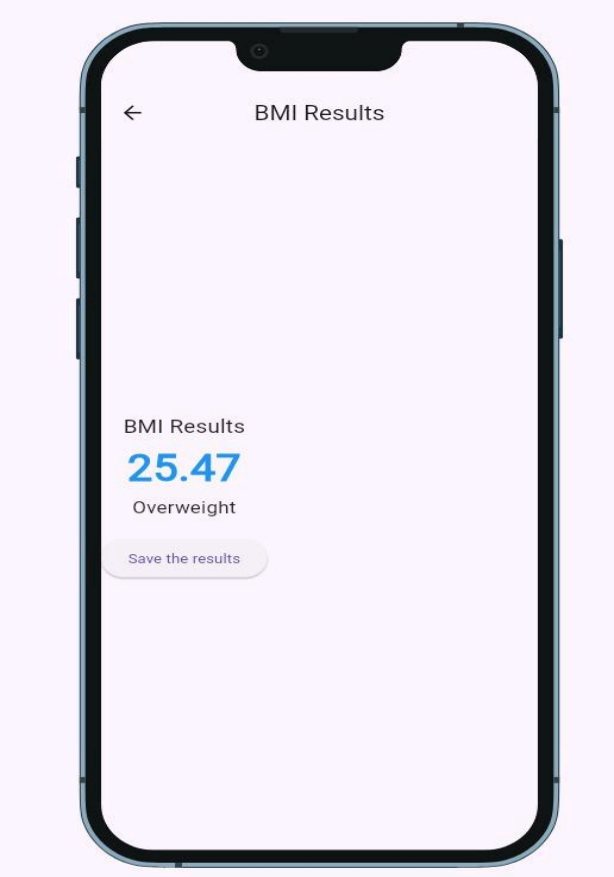
1. Login Screen

- Uses `Icon` widgets for user and password fields, with custom font styling for the title.



3. BMI Results

- Displays the result using bold, large-font text and custom styling for readability and user impact.



Conclusion:-

This experiment helped us learn how to effectively integrate icons and fonts in a Flutter app to create visually appealing and user-friendly interfaces. It emphasized the role of UI elements in improving user experience and application aesthetics.

Experiment No. 4

Name:Rohan Ghadge Roll no:63 Div:D15B

AIM:- To create an interactive Form using form widget

Theory:-

In this experiment, we have developed a BMI Calculator App using Flutter, a UI toolkit for building cross-platform applications. The app utilizes form widgets to collect input from the user, such as age, weight, and height. It then calculates the BMI using the standard formula:

$$\text{BMI} = \frac{\text{Weight (kg)}}{\left(\frac{\text{Height (cm)}}{100}\right)^2}$$

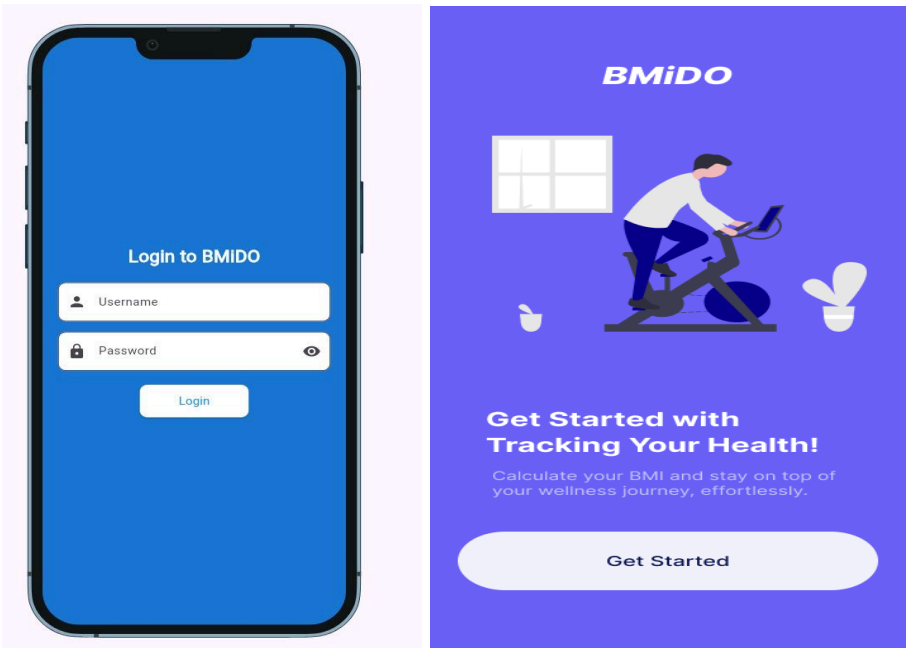
Once the user taps on "Calculate BMI", the result is displayed along with a health status (like underweight, normal, overweight). This app demonstrates the use of TextFormField, buttons, state management, and user input validation — key aspects when working with forms in Flutter.

Screenshots:-

Below are the screenshots showing different stages of the BMI Calculator app:

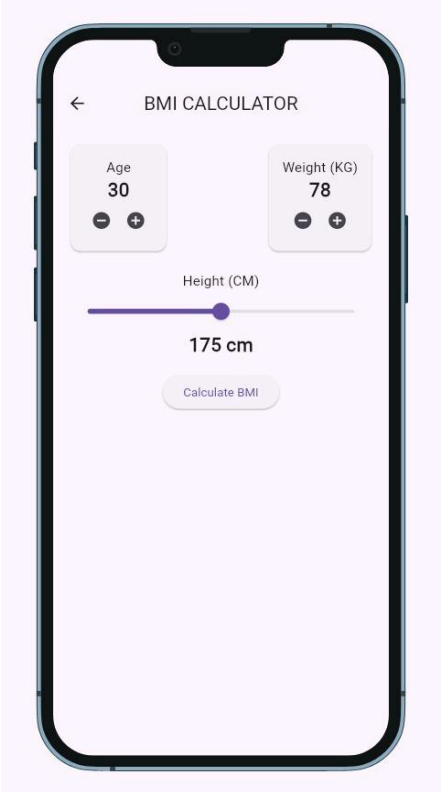
1. Login Screen

- The user is prompted to enter a username and password.
- Ensures access control to the app.



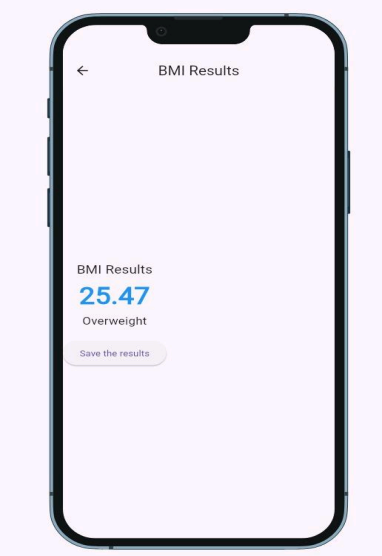
2. BMI Input Form

- Users can input their **Age**, **Weight (kg)**, and select **Height (cm)** using a slider.
- Buttons to increment/decrement values are provided for ease of use.



3. BMI Results

- The result is calculated and displayed with the BMI value and a message showing the weight category.
- Users also get the option to save the results.



Conclusion:- This experiment demonstrated the creation of an interactive BMI calculator using Flutter form widgets. It enhanced our understanding of user input handling and dynamic UI updates in mobile app development.

Name:- Rohan Ghadge Div:-D15B Roll No.:- 63

Experiment No.:- 5

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

Theory:-

This experiment demonstrates the use of navigation, routing, and gestures within a Flutter-based BMI Calculator App. Using `Navigator.push()` and `Navigator.pop()`, we move between screens like the Welcome Screen, BMI Input Screen, and Result Screen, illustrating manual and named routing.

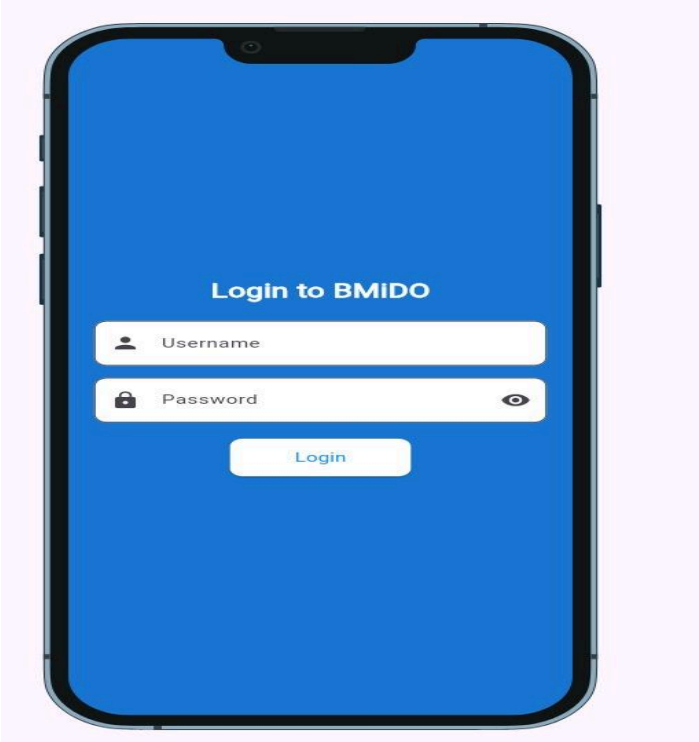
The code also implements gesture handling via buttons, sliders, icon taps, and switch toggles—showing how the app responds dynamically to user input. All these techniques come together to build an interactive and multi-screen Flutter application that enhances both user experience and app structure.

The file also shows modular organization (`login_screen.dart`, `bmi_app.dart`) which reflects proper routing practice using different screens.

Screenshots Related to the Experiment

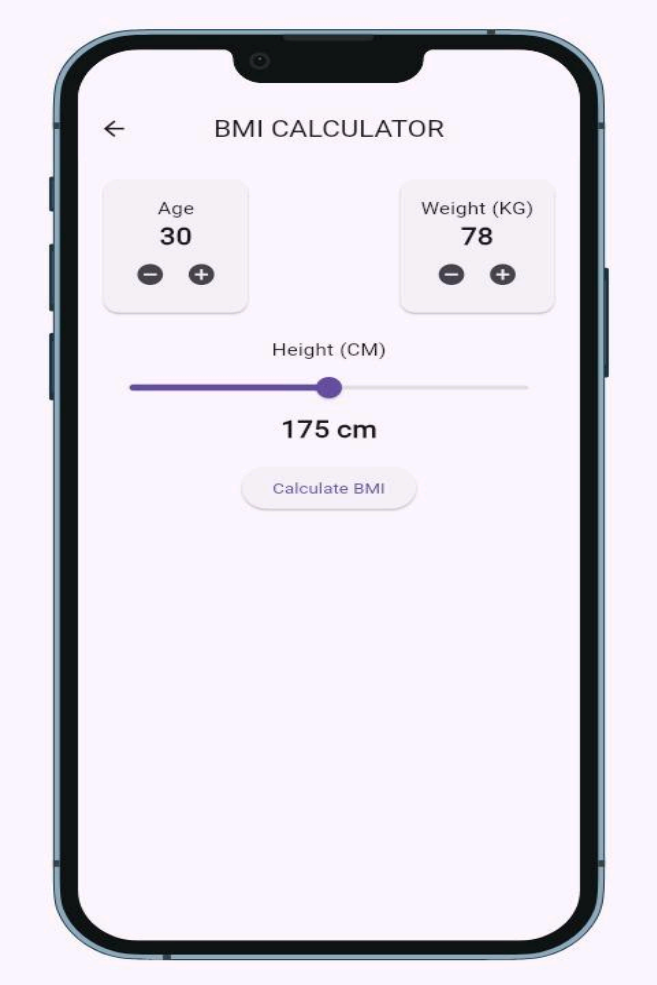
1. Welcome/Login Screen

- Initial screen that navigates to BMI calculator.
- Uses `Navigator.push` for routing.



2. BMI Input Screen

- Users interact with gesture-based widgets like sliders, icon buttons, and switches.



Conclusion:-

This experiment helped us implement navigation between multiple screens and capture user gestures using interactive widgets in Flutter. It improved our understanding of routing and dynamic UI handling.

Name: Rohan Ghadge Roll no :63 Div:D15B

Experiment no 6

Aim: To integrate Firebase with a flutter application for both Android and ios platform , cloud-based services such as authentication, real-time database, And push notifications

Theory: integrating firebase with flutter allows developer to:

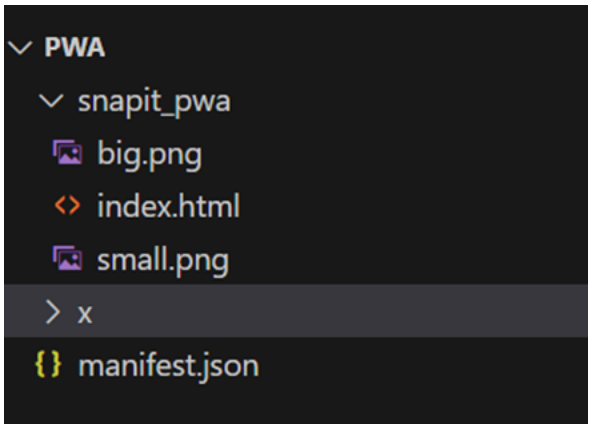
1.Use firebase Authentication to manage user securely
2. Store and sync data in cloud Firestore or Real time Database
3. TCrack user behaviour with Analysis.
4. Send targeted Push Notifications via Firebase Cloud Messaging.
5. Host backend function with cloud Function.
6. Modify native project files to include Firebase SDKs.

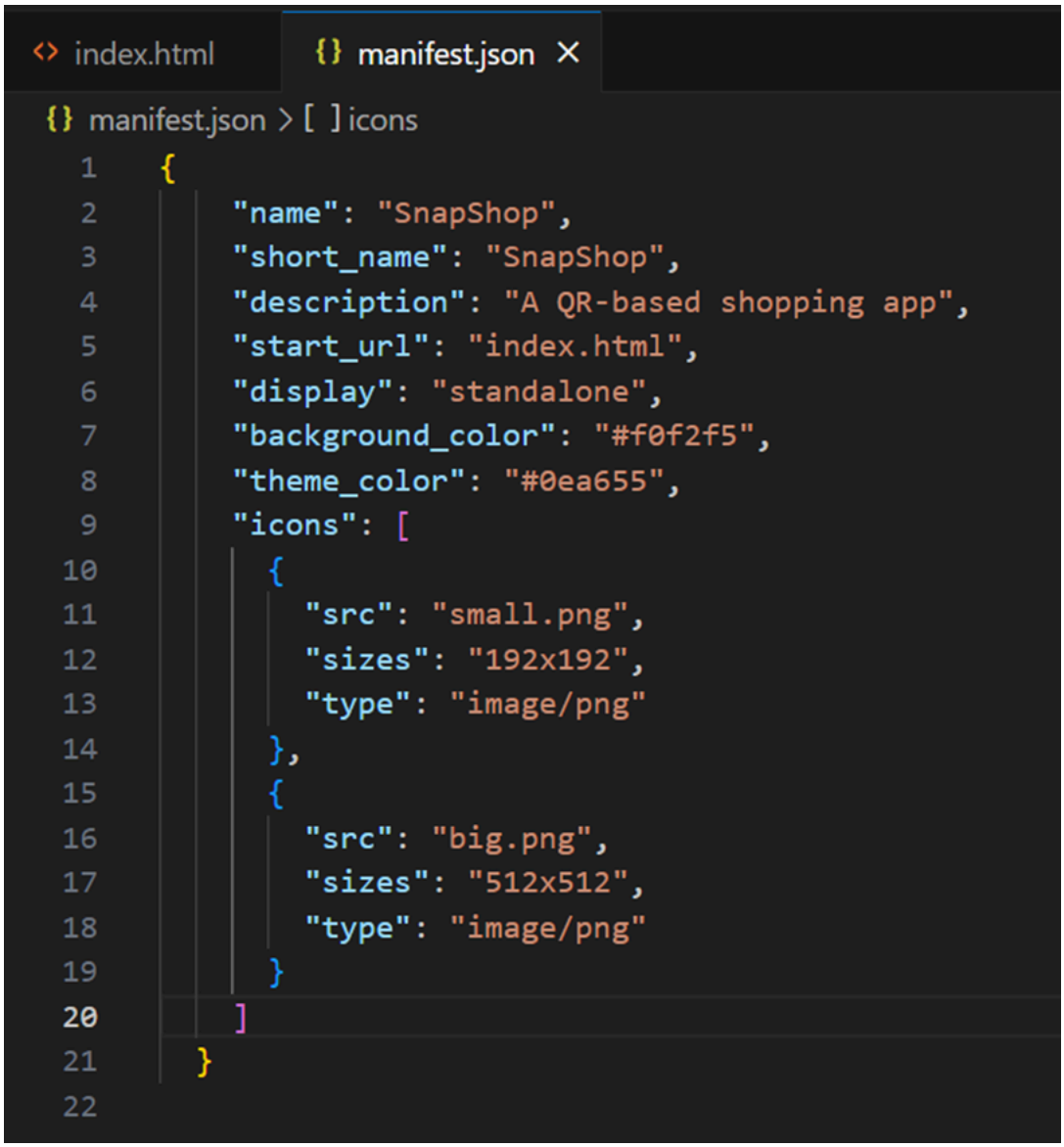
Conclusion: Firebase integration with flutter provides a robust and scalable backend Infrastructure for mobile app development. Its simplifies the implementation of essential features such as database storage authentication Reducing development time and effort. By completing the integration For both Android and ios platforms, developer can build unified platform .

Name: Rohan Ghadge Roll no:63 Div:D15B

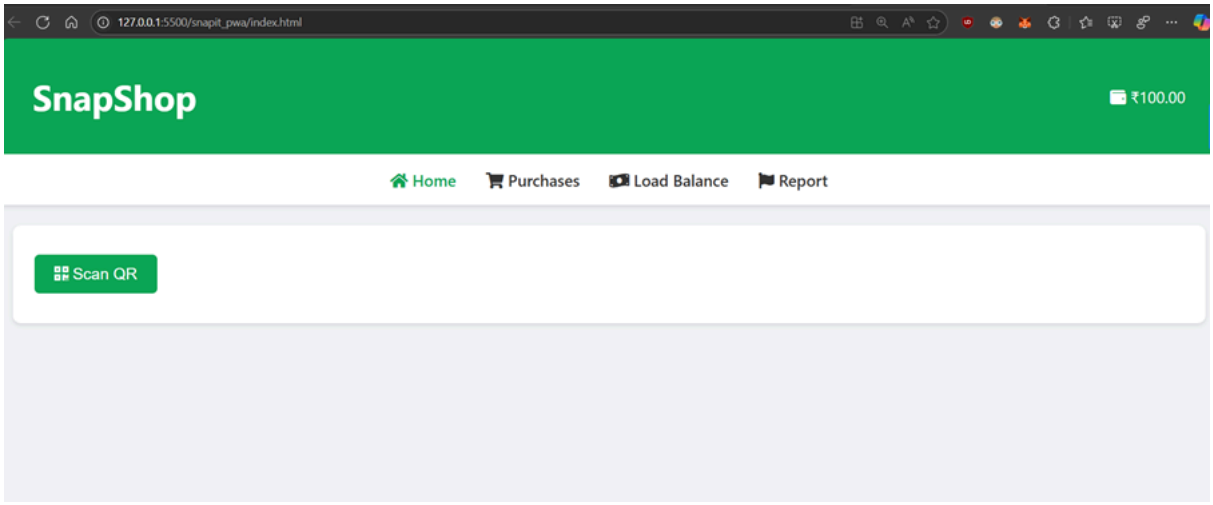
Experiment no 7

Aim: Add to your home screen feature on a web application.

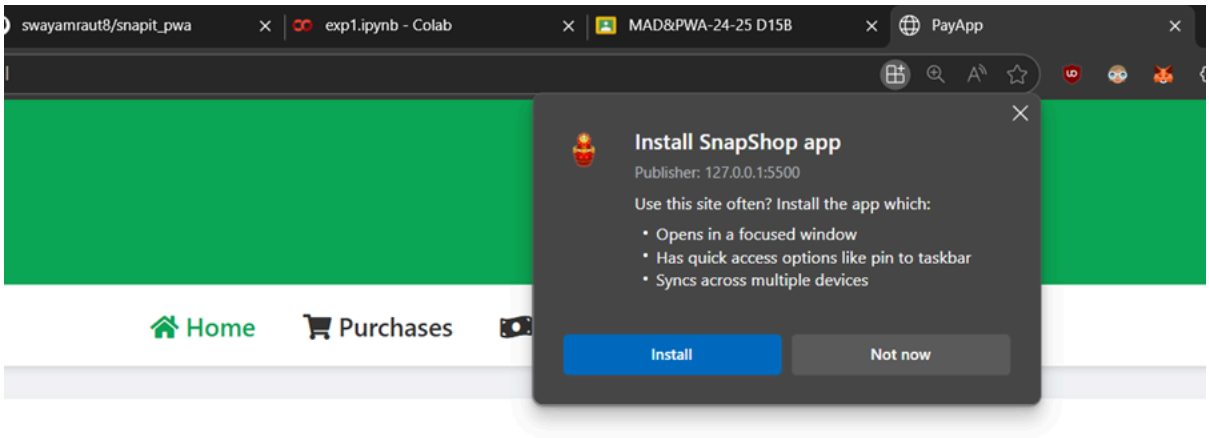


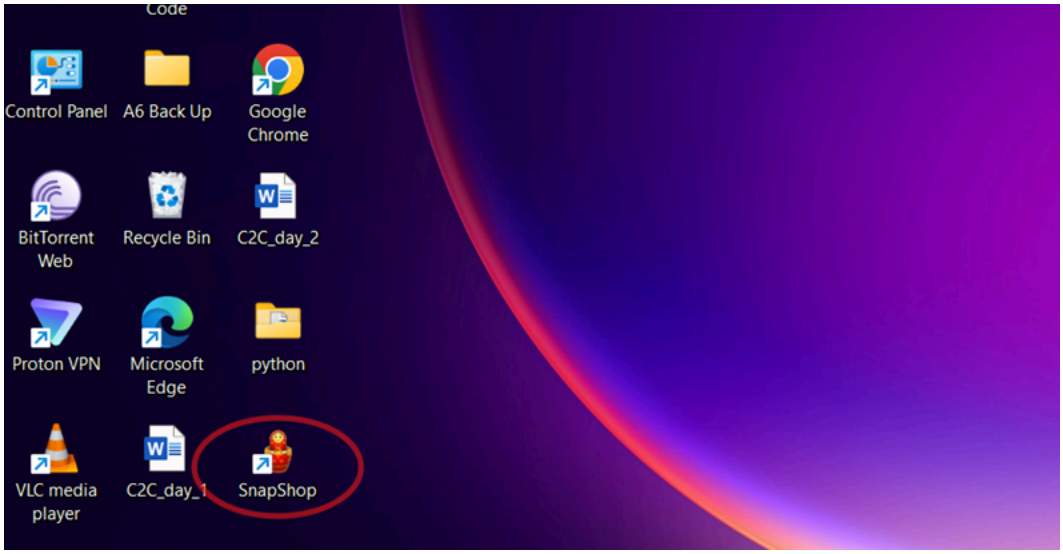


On adding Icon images and manifest.json file to the file structure, we could see the option to install the website as if it were an application.



On clicking install, the web application could be accessible from the desktop. Clicking it would create an isolated instance of that web app through which only the features of that particular web app would be accessible.





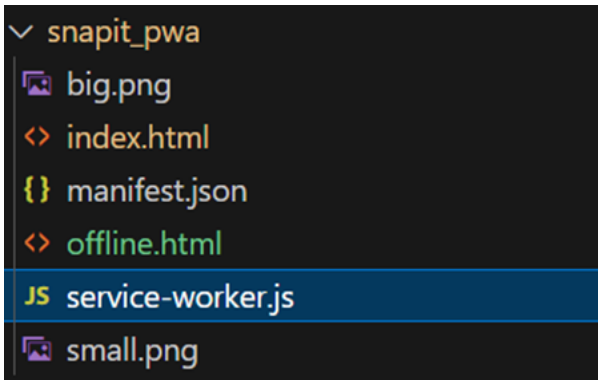
Conclusion: Through this experiment we learnt to add the ‘add to my webpage’ feature to our web application. This is the most fundamental step to be performed while building progressive web applications.

Name:Rohan Ghadge Roll no :63 Div:D15B

Experiment no 8

Aim: To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA.

Create service-worker.js

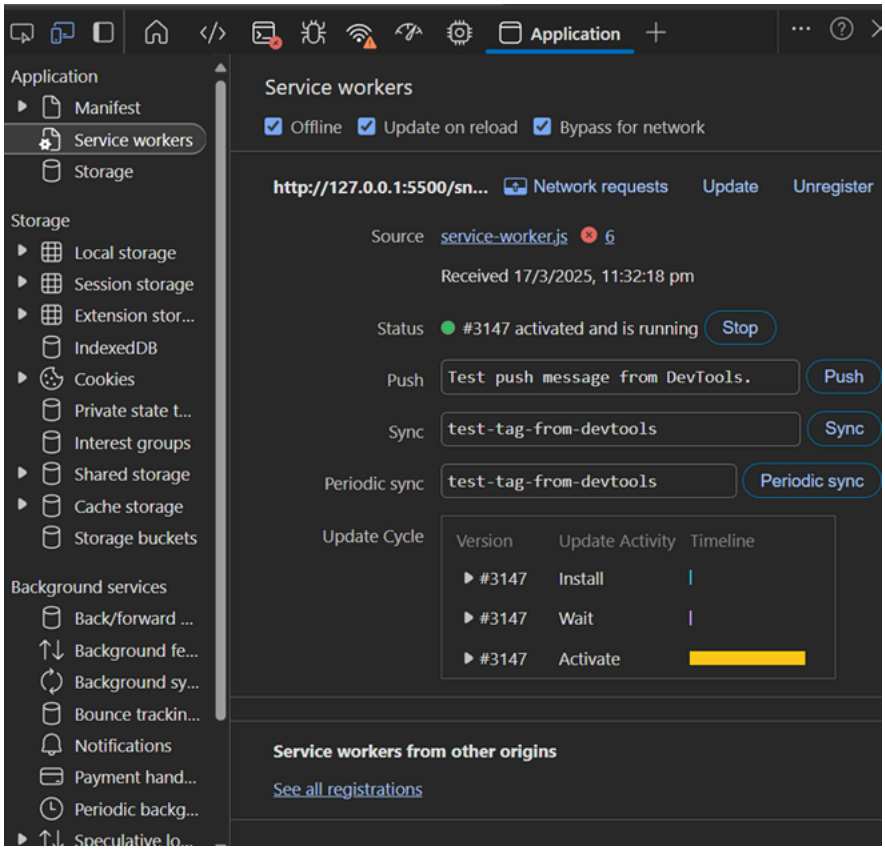


Create a cacheable file called offline.html to be displayed in the absence of an internet connection.

offline.html U

snapit_pwa > offline.html > ...

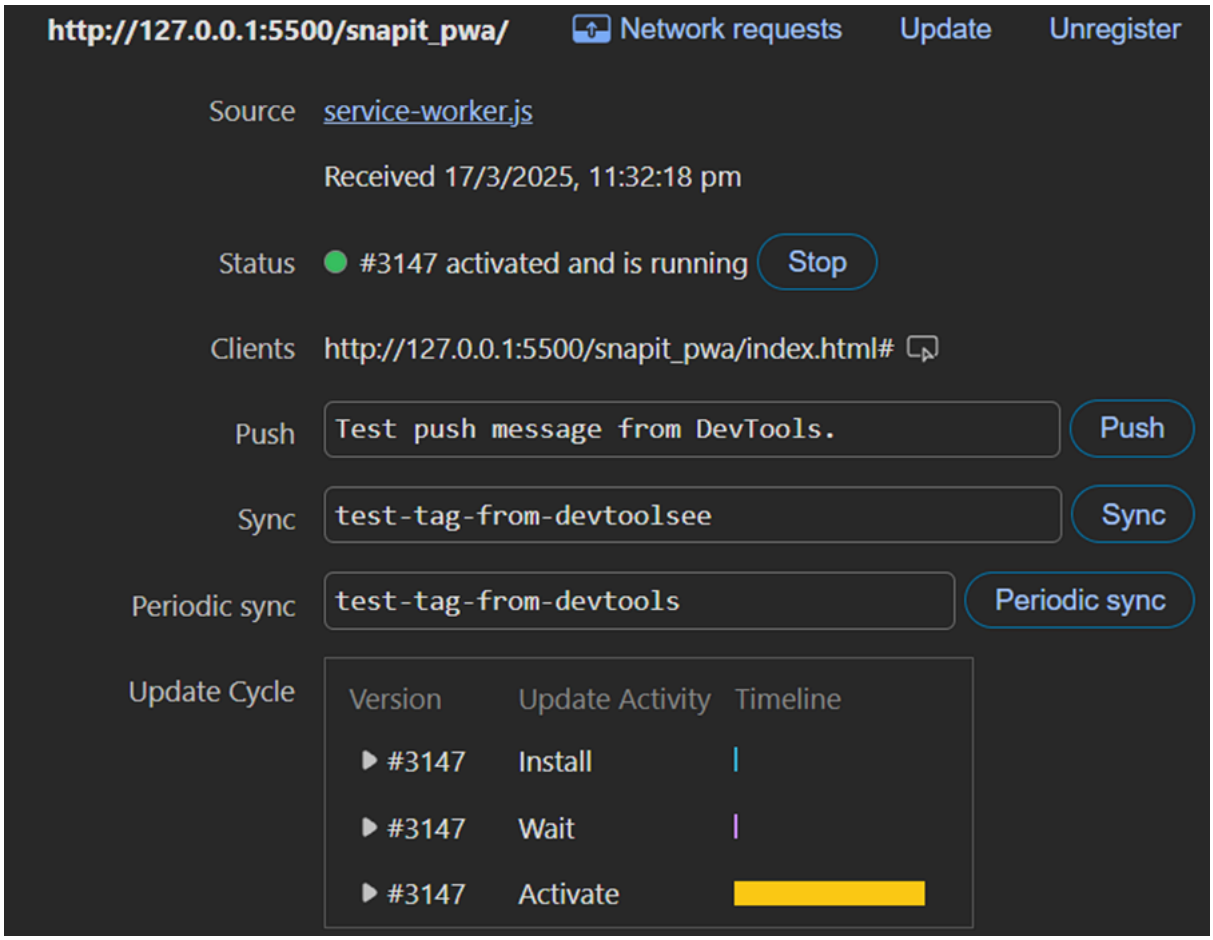
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Offline</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            padding: 50px;
        }
    </style>
</head>
<body>
    <h1>You're Offline</h1>
    <p>It looks like you have lost internet access. Some features may not work.</p>
    <img src="big.png" alt="SnapIt Logo" width="100">
</body>
</html>
```


Application
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension stor...
IndexedDB
Cookies
Private state t...
Interest groups
Shared storage
Cache storage
Storage buckets
Background services
Back/forward ...
Background fe...
Background sy...
Bounce trackin...
Notifications
Payment hand...
Periodic backg...
Service workers
Offline
Update on reload
Bypass for network
http://127.0.0.1:5500/sn...
Network requests
Update
Unregister
Source
service-worker.js
6
Received 17/3/2025, 11:32:18 pm
Status
#3147 activated and is running
Stop
Push
Test push message from DevTools.
Push
Sync
test-tag-from-devtools
Sync
Periodic sync
test-tag-from-devtools
Periodic sync
Update Cycle
Version
Update Activity
Timeline
#3147
Install
#3147
Wait
#3147
Activate
Service workers from other origins
See all registrations

Name:Rohan Ghadge Roll no:63 Div:D15B

Experiment no 9

Aim: To implement Service worker events like fetch, sync and push for E-commerce PWA.



Make the following changes to the service-worker.js

// Install Event: Cache assets

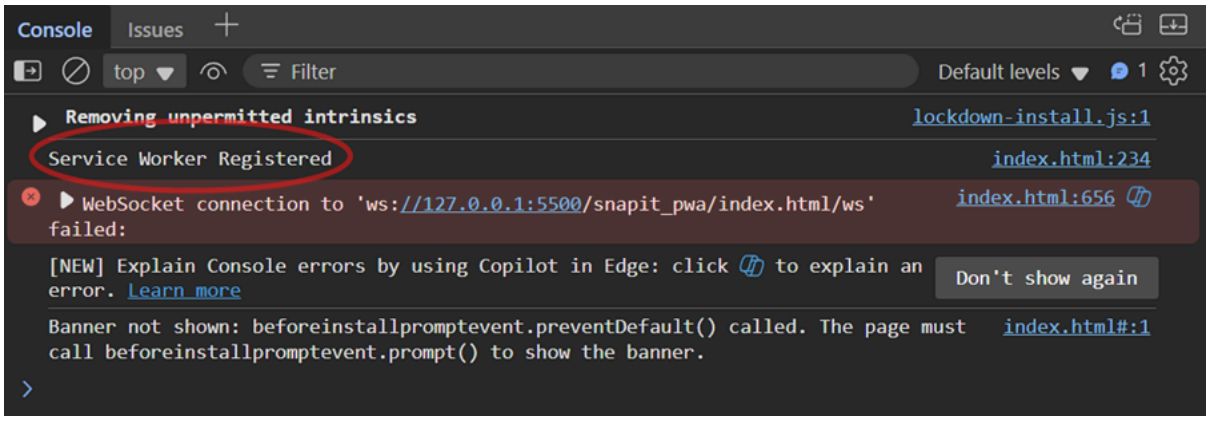
// Activate Event: Cleanup old caches

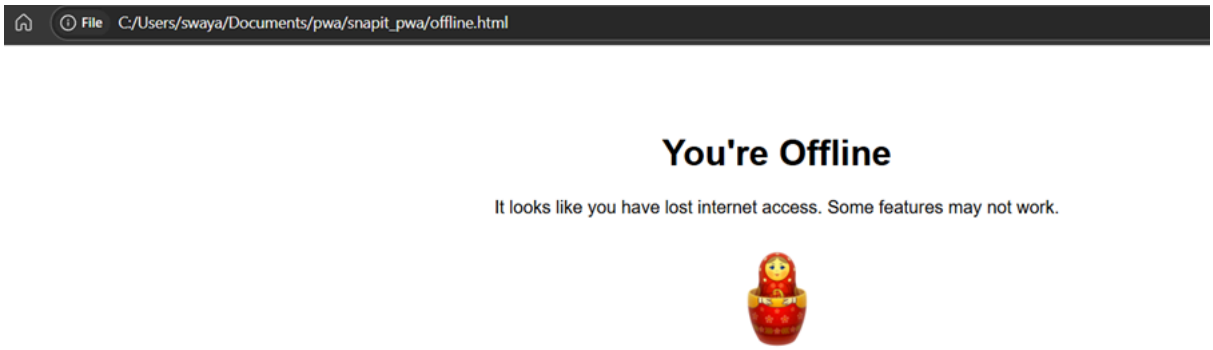
// Fetch Event: Supports both Cache-First & Network-First

// Sync Event: Retry sending data when online

// Function to send pending screenshots to the server

// Push Event: Display push notifications

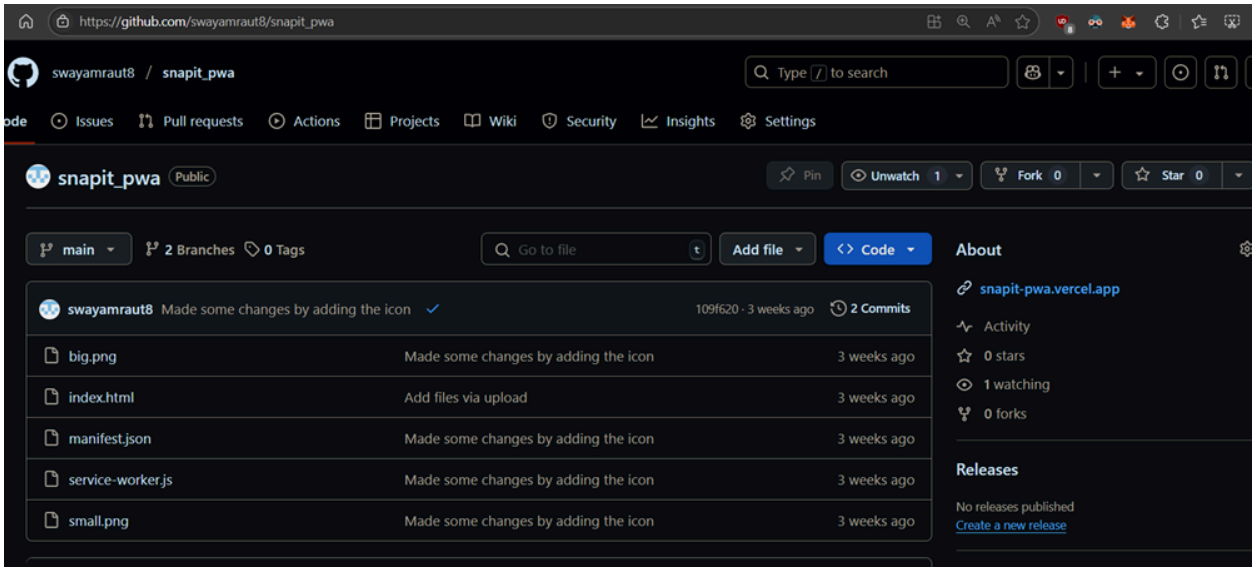


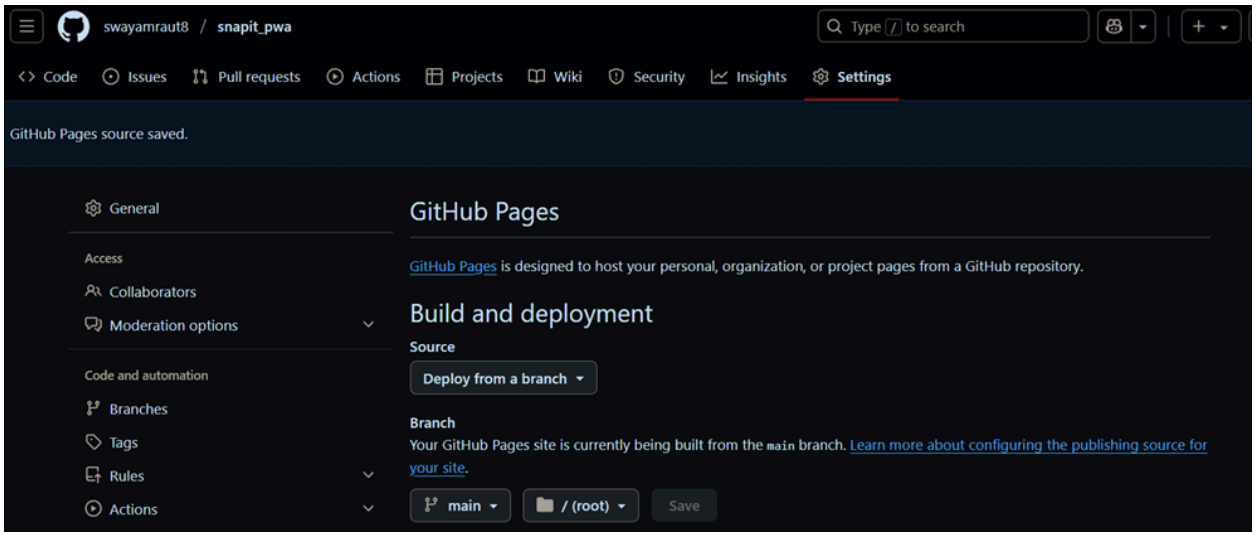


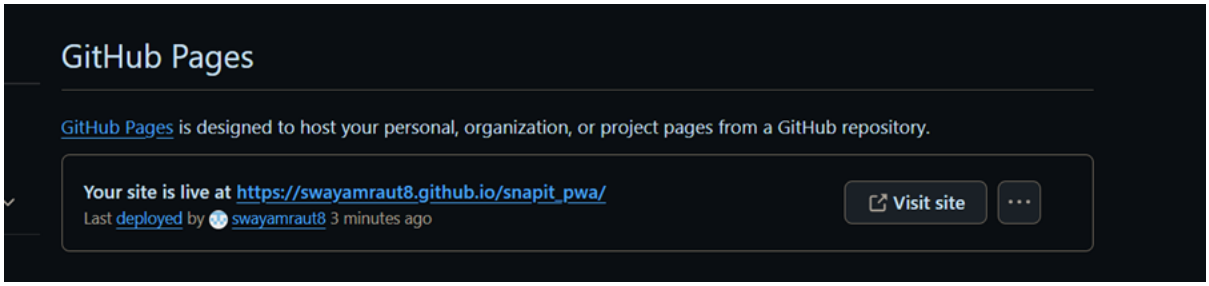
Conclusion: We implemented the functionality of offline web cache capture so that in the absence of a stable internet connection, t

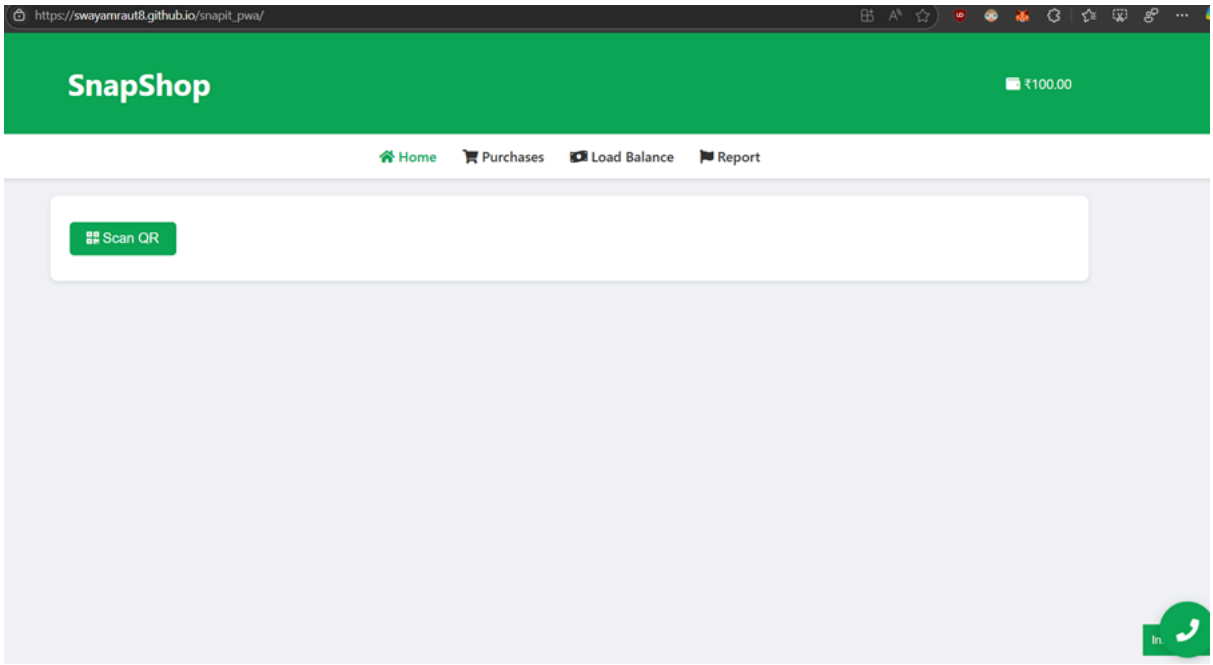
Name:Rohan Ghadge Roll no:63 Div:D15B

Aim: To study and implement deployment of Ecommerce PWA to GitHub Pages.






GitHub Pages
GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.
Your site is live at https://swayamraut8.github.io/snapit_pwa/
Last deployed by swayamraut8 3 minutes ago
Visit site


https://swayamraut8.github.io/snapit_pwa/
SnapShop
₹100.00
Home
Purchases
Load Balance
Report
Scan QR


https://swayamraut8.github.io/snapit_pwa/

Name:Rohan Ghadge

Roll no:63

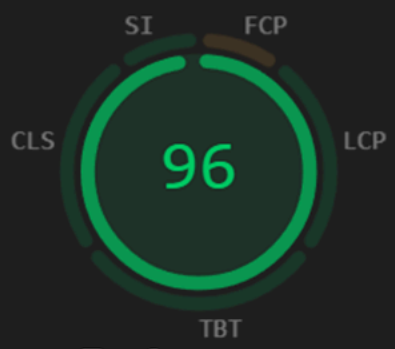
Div:D15B

Experiment no 11

Aim: Use Lighthouse chrome extension to analyse the PWA for performance.

http://127.0.0.1:5500/index.html

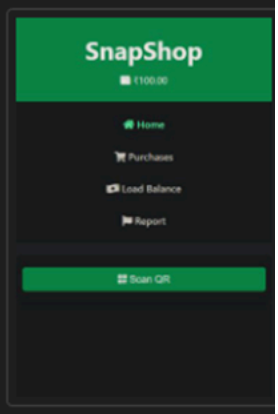
96	91	100	90
Performance	Accessibility	Best Practices	SEO



Performance

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

▲ 0–49 ■ 50–89 ● 90–100



METRICS Expand view

■ First Contentful Paint 2.2 s	● Largest Contentful Paint 2.2 s
● Total Blocking Time 0 ms	● Cumulative Layout Shift 0
● Speed Index 2.2 s	

DIAGNOSTICS

- ▲ Eliminate render-blocking resources — Potential savings of 970 ms
- ▲ Enable text compression — Potential savings of 18 KiB
- ▲ Page prevented back/forward cache restoration — 1 failure reason
- ■ Minify JavaScript — Potential savings of 135 KiB
- ■ Remove duplicate modules in JavaScript bundles — Potential savings of 16 KiB
- ■ Reduce unused CSS — Potential savings of 21 KiB
- ■ Reduce unused JavaScript — Potential savings of 121 KiB
- ■ Minimize main-thread work — 2.8 s
- ○ Avoid non-composited animations — 1 animated element found
- ○ Initial server response time was short — Root document took 10 ms

Name: Rohan Gha
Roll No: 63 Div: D15B

MAD Assignment No. 1

Explain the key feature and advantage of using Flutter for mobile app development.

Single Codebase for multiple platform. Write one codebase for both Android and ios reducing development effect and maintenance.

Hot Reload. Instantly see changes in the app without restarting making development faster and more interactive.

Each Performance uses the Dart language and a compiled approach for smooth

Open Source & Strong community support Backed by Google and a large developer community ensuring continuous improvement

Advantages :-

Faster Development Time : Hot reload and Single codebase reduce development time Significa

Cost Effective : Since the name code runs on both Android and ios, business save on development and maintenance cost.

Reduced performance issue : The app runs natively without relying on intermediate bridge like in react native.

b) Discuss how the Flutter Framework differs from traditional approach and why it has gained popularity in the development community

→ Single codebase Vs separate codebase

- Traditional Approach. Developers need to write separate code by Android and ios
- Flutter. uses a single dart based codebase for both platform reducing development time and effort.

2. Rendering Engine Vs Native UI Component

Traditional Approach Relies on platform native UI component which can lead to inconsistencies and performance issue Flutter. Uses the Skia rendering engine to draw everything from scratch ensuring a consistent UI across devices

Why Flutter Has Gained Popularity

1. Faster development with Hot. Developers can instantly see UI changes without restarting the app making the iteration process much quicker.
2. Cross platform Efficiency: Business save time and resource by maintaining a single codebase for multiple platform
3. Consistent UI Across devices Since Flutter does not rely on native component. the UI looks and behave the across different OS
4. Improved performance: AOT compilator and direction access the GPU rendering ensure smooth Animation and high performance

escribe the concept of the widget tree in Flutter
plain how widget composition is used to build
complex user interface.
idget tree in Flutter
Flutter, the widget tree is the fundamental
ructural that represent the UI of an application
t is a hierarchial arrangement of widget
here each widget define a part of the
ser interface. Flutter UI is entirely built using
idget which can be.
idget change occurs
idget composition refer to building complex UI
y combing smaller, reuseble widget. Instead
f creating large, monolithic UI component, Flutter
ncourage bracking the UI into smaller manageable
idget that can be reused and nested with
ach other.
xample: Class profile card extend StatelessWidget {
Final string nam,
Final String impe (url);
ProfileCard ({ required this.name, required this
@ overide
Widget build (Buid context context) {
return card (
child. Column (
children. [
Image. network (image url),
SizedBox (height: 10)
Text (name, style: TextStyle (FontSize: 20, Fontweigh

Benefits of widget composition

1. Reusability : Small widget can be reused in different parts of therapy
2. Maintaintability : Breaking UI into smaller widg make it easier to debug and update.
3. Performance : Flutter efficiently rebuilded only the necessary parts of the widget tree.

b Provide example of commonly used widget and their roles in creating widget tree.

→ Structural widget

These widget act as the foundation build the UI

. Material App : The root widget of Flutter ap that provide essential configuration

```
Ex. MaterialApp (
      home : Scaffold (
    appbar : Appbar ( HHc . Text ( ' Flutter widg
  body : container (
      Padding : Edge Insect all (16.0)
    Child : Text (" Hello , Flutter !"),
    ),
    ),
    ),
```

& Interaction widget

eild - Accepts text input from user.
vation Button - A button with elevation

Ex. column (
children : [
rextfield (decoration : Input Decoration (labelText :
levated Button (
Onpraced : () {
Print (" Button Prased");
},
child : text (" submit"),
)
};
];

Display & Styling widget

ext - display text on the screen
Image - Show image from asset network or memory
Icon - Display icon
card - A material design card with rounded corner and elevation

Ex (olumn (
children :)
Text (" Welcome to Flutter!", Style : Textstyle (fontsize: 24
fontweight. fontweight. bold))
Image. network (" https // Flutter. dev / image / Flutter
- logo - sharing. png);
],
);

Teacher's Sign.:

Q3) Discuss the importance of state management in a application

→ In Flutter, State refers to data that can change during the lifetime of an application
- User input
- UI changes
- Network change
- 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 State - Data, shared across multiple widget Importance of state management
3. Code maintainability & scalability : Managing state properly making the code modular, Head' readable and scalable larger application.

b) compare and contrast the different statemanagement approaches available in Flutter, such as setState Providers and Riverpad Provide scenario where each approach is suitable

→ set State : local state
Pros : simple built in easy to use
Cons : Not scalable cause unnecessary re-rende
Best use case - small UI updates
Provide : App wide State
Pros - lightweight, recommended by Fusthes efficient
Cons - Boiler plate code for nested providers
Best case medium Scale apps

Teacher's Sign.:

iverpod App-wide State
iminate Provider limitation, improves performance
cons Require learning new concept
est use cases: large, apps needing global state

ecani Scenario pr Each Approach
e set state when managing simple ui
ement within a widget like toggling dark
mode in a setting screen.
use Riverpod when building a complex, scalable
app with global state management, like an
e-commerce app with cart management.

Explain the process of integrating Firebase with Flutter application Discuss the benefit of using Firebase as a backend solution.
Firebase provide a powerful backend solution Flutter application offering services like authentication real time database, cloud function, storage and more
Steps to integrate Firebase with Flutter
ep1: Create a Firebase Project
Go to Firebase console
Click on "Add Project" and enter a project name
Configure Google analytics if needed, then click create

Step 2. Register the Flutter app with Firebase
- In the Firebase project dashboard click "Add
and select Android or ios based on your platf
- For ios
Enter the ios Bundle indentifier
Download the Google Service - info plist file and
place it in ios/ runner.

Step 3. Install Firebase dependencies
Add Firebase dependencies in
Firebase core
Firebase auth
Cloud Firestore
Run. Flutter pub get

Step 4. Configure Firebase for Android & ios
For Android
1. Open android/build.graded and ensure the
following closepath 'com.google.gms.google-servic
2. Open android/app/build.gradle and add
at the bottom apply Plugin : 'com google gms.g

Step 5 Initialize Firebase in Flutter
void main() async {
Widget Flutter Binding. ensure Initialized();
await Firebase initidlize App();
run App (MyApp());

the Firebase services commonly used
development and provide a brief
of how data synchronization is achieved
provide a suite of backend service
simplify Flutter app development

Authentication
secure authentication using email / password
numbers and third party providers like
Facebook and Apple.

Flutter
and sync data in real time across
support structured data, queries and
access

Firebase Firestore.instance.collection('users')
.add({ 'name': 'John Doe',
'email': 'John@example.com
});

Time Database
real time. JSON-based database the automatically
data across device
Database Reference ref = FirebaseDatabase
instance ref("message") ref.set({"text": "Hello, Firebase!

Firebase Cloud Messaging
enable push notification and messaging between
Ex. FirebaseMessaging.instance.subscribeToTopic

Firebase Hosting
deploy and serve web application securely
with automate set.

Teacher's Sign.: _____

Data synchronization in Firebase

Firebase ensure real-time data synchronization across multiple devices and platform using Firebase and realtime database.

1. Cloud Firebase Sync mechanism

Used realtime to update UI instantly when data changes.

Ex. Firebase instance.collection ("users").snapshot() li
((snapshot)?
Pr (var doc in snapshot.docs)?
Print (doc ['name'];
};
};

2. Realtime Database Sync Mechanism

used persistent websocket connection for live u
Ex: Database Reference ref = Firebase Database.
instance.ref ("message"); ref on value.listen ((em
Print (event.snapshot.value);
});

3. OFFline Data Sync

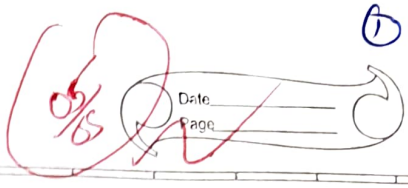
Fire store caches data locally and sync change within the device it online

Ex. Firebase Firestore.instance.setting.setting (Persistance Enabled here);

4. Cloud Function be automated update

Automate backend logic to trigger update when data changes.

Name: Rohan A. Ghadge
Roll No: 63 Div: D15B



MPL Assignment - 2

Define Progressive Web App and explain its significance in modern web development. Discuss the key characteristic that differentiate PWAs from traditional mobile apps.

A Progressive Web App (PWA) is a type of web application that uses modern web technologies to provide a user experience similar to native mobile apps. PWAs are built using standard web technologies such as HTML, CSS and Javascript, but they leverage additional feature like service worker, Web App Manifest, and push notification to offer enhanced functionality, including offline access, fast loading times and device inte

Significance of PWAs in Modern Web Development

PWAs have gained prominence due to their ability to bridge the gap between traditional web applic

1. Cross-Platform Compatability - PWAs run on an device with a modern browser, reducing the nee for separate development for iOS, Android and d
2. Improved Performance - With caching and Seor worker, PWAs load faster and perform effic even in low-network conditions.
3. Offline Functionality - Service worker enable P to function offline or with limited connectivity
4. SEO-Friendly - Unlike native apps, PWAs search engines, increasing v

Key chater characteristic that PWAs from Traditio
Mobile Apps

1. Installation : No app store required ; can be installed from the browser
2. Platform : Works across multiple platform and de
3. Offline : Uses service worker for caching support and offline access
4. Performance : Fast and lightweight ; loads instantly with caching
5. Discoverability : Indexed by search engine, improving SEO

Q) Define Responsive Web design and explain its importance in the context of Progressive Web Apps. Compare and contrast responsive, fluid and adaptive web design approaches.

→ PWAs are revolutionizing modern Web developm by offering a hybrid approach that combines the best feature of Web and mobile apps. Business looking to improve user engagement, reduce development costs, and simplify distribution can greatly benefit from adaptive PWAs over traditional native apps.

Importance of Responsive Web Design in Progressive Web Apps.

Responsive Web design is a crucial component of Progressive Web Apps because PWAs are intende to work on all devices, from mobile phones to large screen desktop. The importance of RWD in PWAs includes:

Improved User Experience - Ensure PWAs provide a consistent experience across devices.
Accessibility & Reach - PWAs should work for all users, regardless of devices or screen size
Performance optimization - Enhances usability without regarding multiple codebases.
SEO benefits - Google prioritizes mobile-friendly, responsive sites in search ranking
Reduced Development cost - A single responsive design eliminates the need for multiple version of an app.

Describe the lifecycle of service Worker, including registration, installation and activation phases.
i) Service Worker life cycle
A Service Worker is a Javascript script that runs in the background, separate from the web page, enabling feature like offline caching, background sync, and push notification in Progressive Web Apps
1. Registration Phase
Before a service worker can be used, it must be registered within the web application. This process is done in the main Javascript file of the web page.
2. Installation Phase
Once registered, the browser attempts to install the service worker
key Aspects of installation:
- This is where the service worker caches necessary asset for offline use.
- If successful, it moves to the activation phase

Teacher's Sign.:

- If an error occur, the installation fails, and the service worker does not proceed.

3. Activation Phase
After installation, the service worker moves to the activation phase, where it takes control of the pages and clean up old caches.

Q] Explain the use of IndexDB in the service Worker for data storage.

→ i) Indexed DB is a low-level, client-side database that allow developer to store, retreive and query large amount of structure data, including files and blobs. In the context of service worker, Indexed DB is particularly useful for managing data storage because service worker operates in a separate thread and cannot directly access the DOM.

Typical use cases of Indexed DB in Service Worker

1. Caching API Responses : Store data retrieved from API for OFFline access.
2. OFFline form submission : Queue form data in IndexedDB and sync it with the server once the connection is restored.
3. Background sync : Store user activity or updates and synchronize them in the background.
4. Media Storage : Save media files like image or video for offline use.