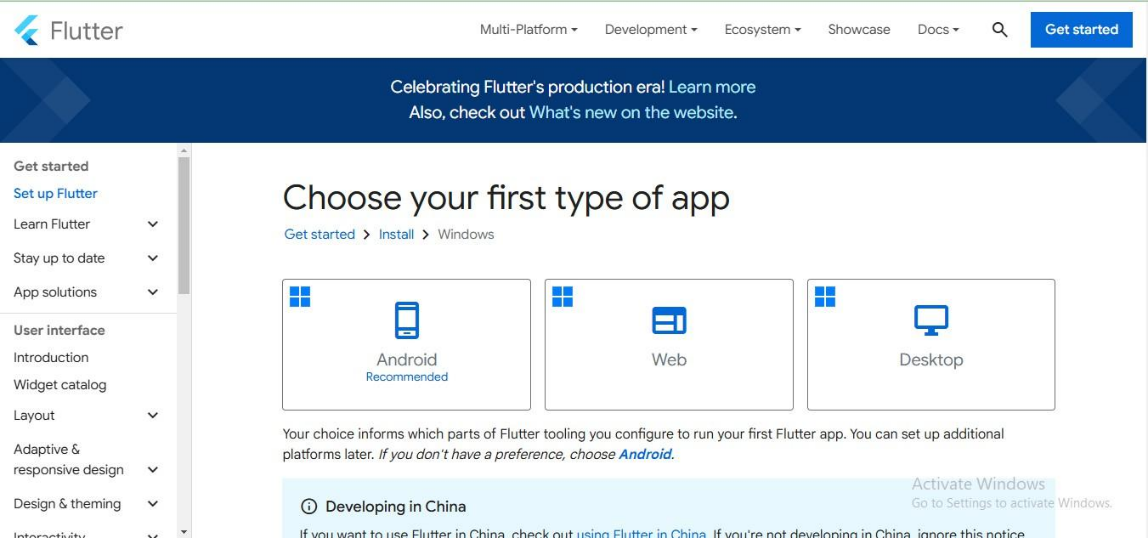
**Name: Arnav Yadav**

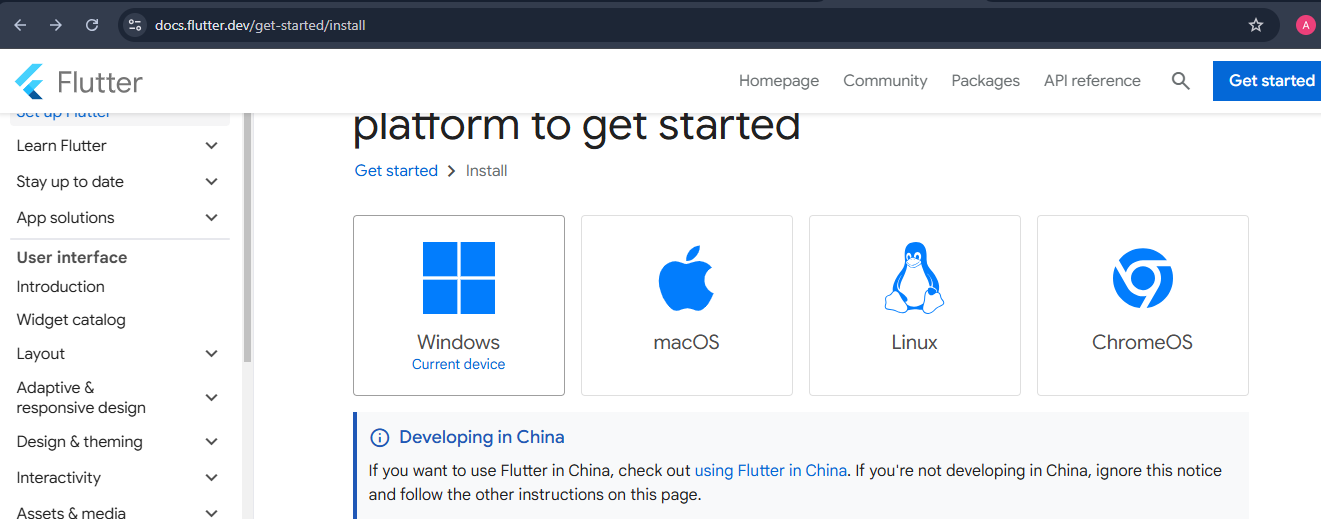
**Div: D15B**

**Roll no: 62**

**Experiment No:1**

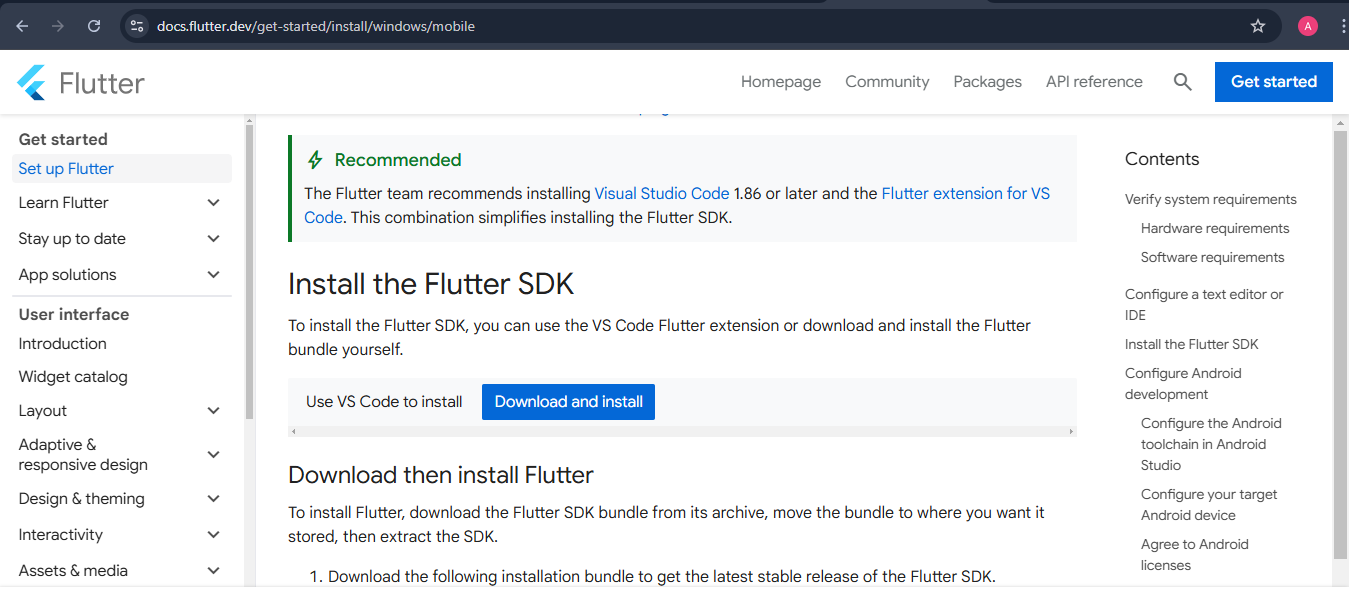
**Aim:**

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



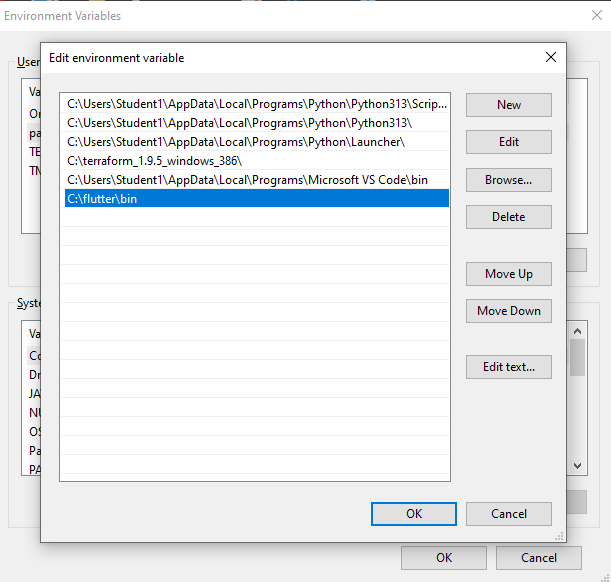
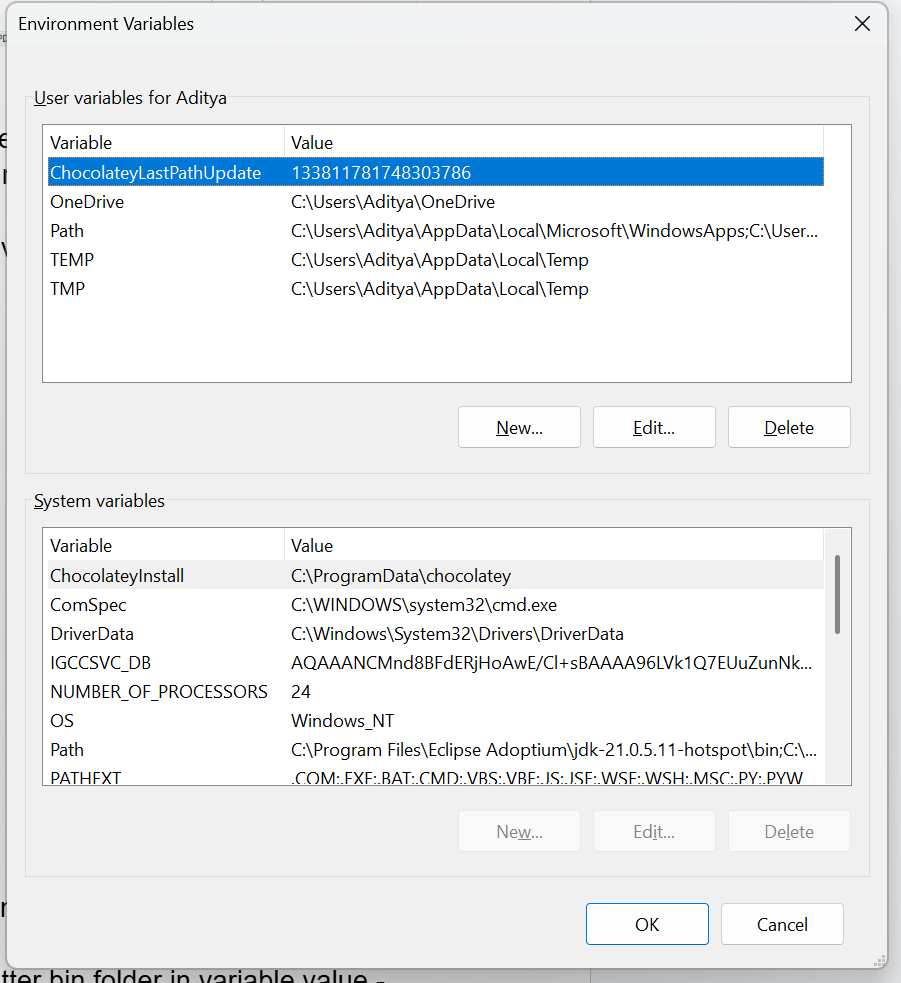
**Step 2:** Next, to download the latest Flutter SDK, click on the Windows icon. Here, you will find the download link for SDK.

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



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

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

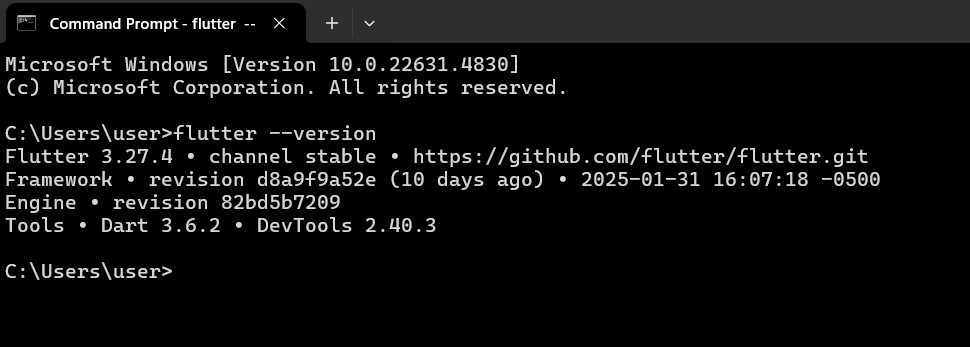


**Step 4.2:** Now, select path -> click on edit. The following screen appears

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

> ok -> ok -> ok

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



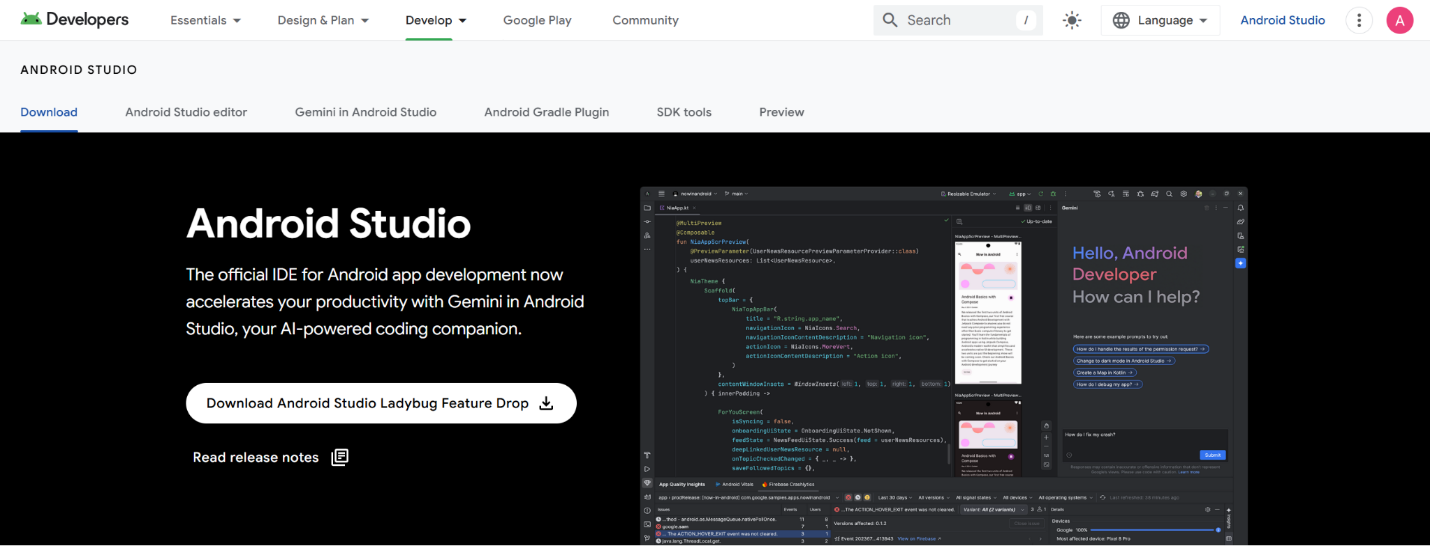
Now, run the $ flutter doctor command. This command checks for all the requirements of Flutter app development and displays a report of the status of your Flutter installation.

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

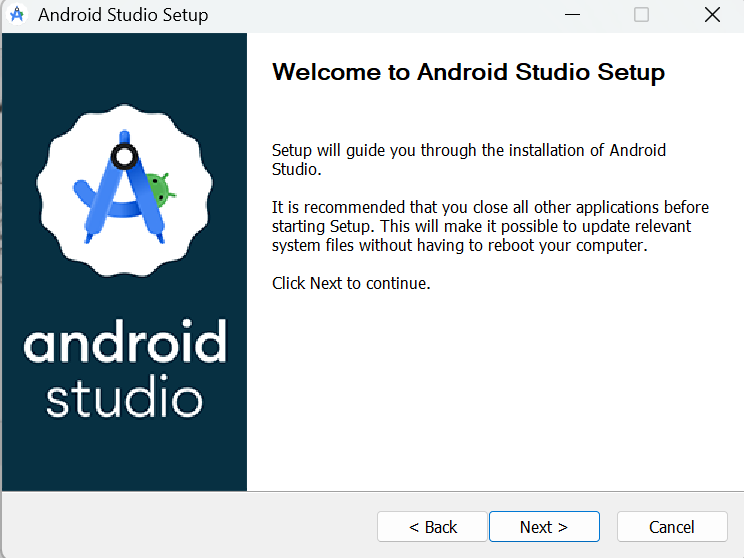


**Step 7:** Install the Android SDK. If the flutter doctor command does not find the Android SDK tool in your system, then you need first to install the Android Studio IDE. To install Android Studio IDE, do the following steps.

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



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



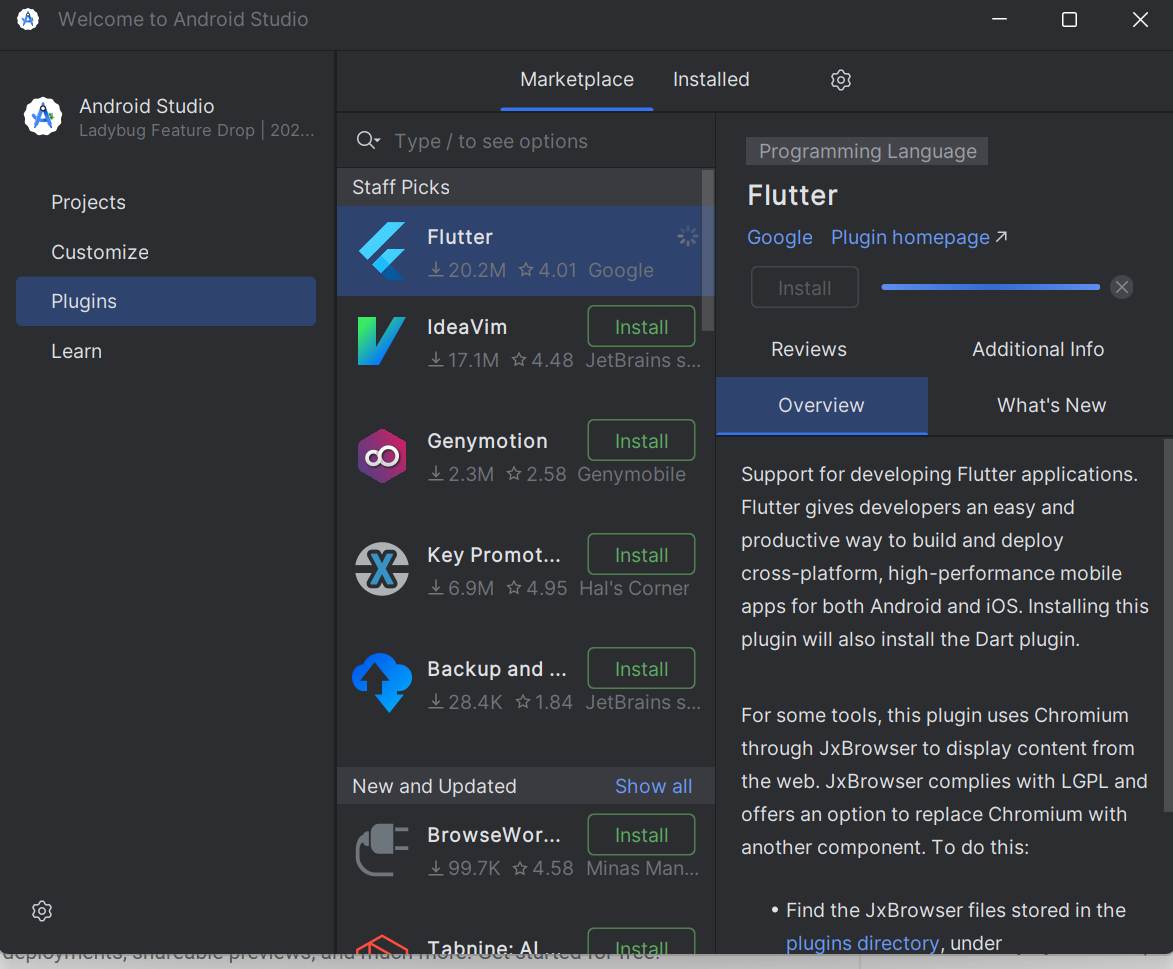
**Step 7.3**: Follow the steps of the installation wizard. Once the installation wizard completes, you will get the following screen.



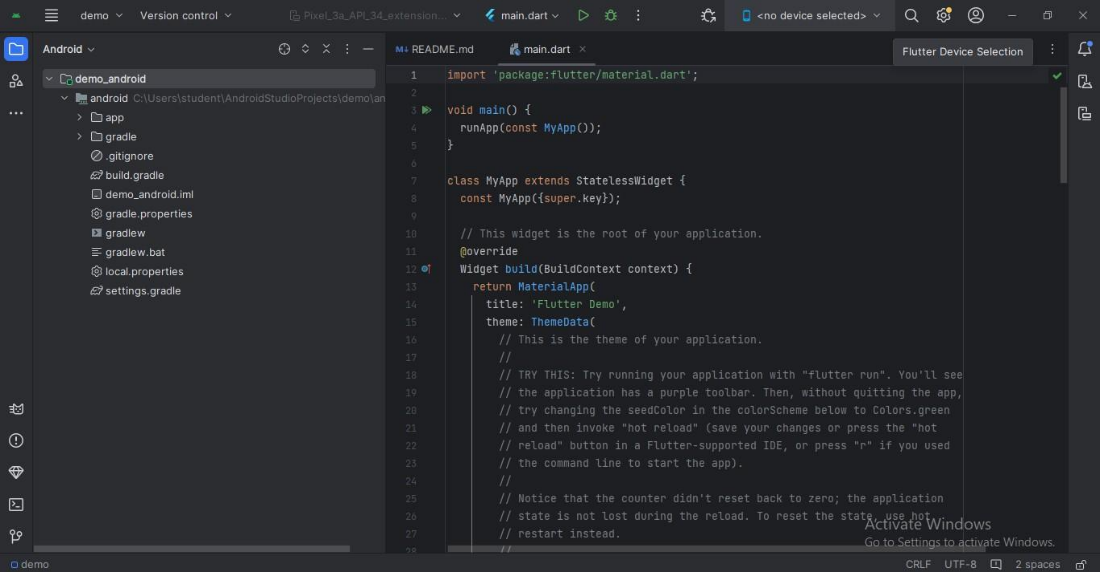
**Step 7.4:** In the above screen, click Next-> Finish. Once the Finish button is clicked, you need to choose the 'Don't import Settings option’ and click OK. It will start the Android Studio.

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

**Step 8.1:** To set an Android emulator, go to Android Studio > Tools > Android > AVD Manager and select Create Virtual Device. Or, go to Help->Find Action->Type Emulator in the search box. You will get the following screen.

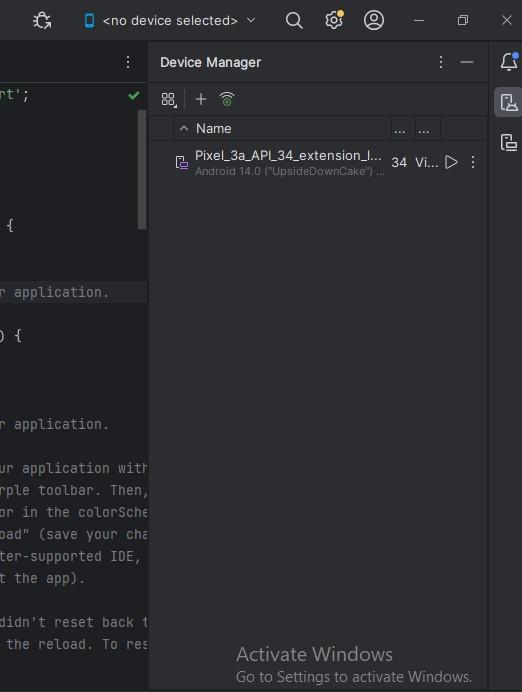


**Step 8.2:** Choose your device definition and click on Next.

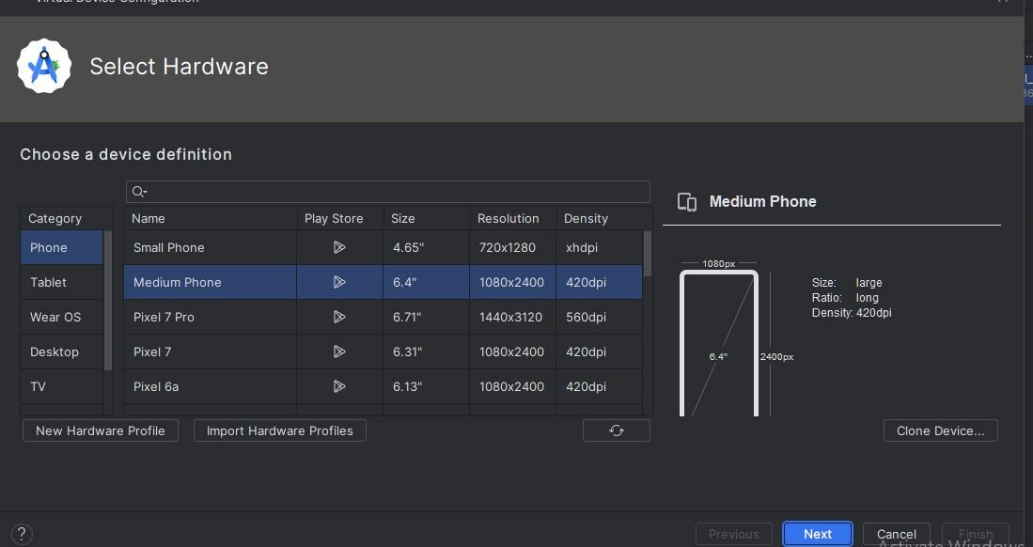


**Step 8.3:** Select the system image for the latest Android version and click on Next.

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

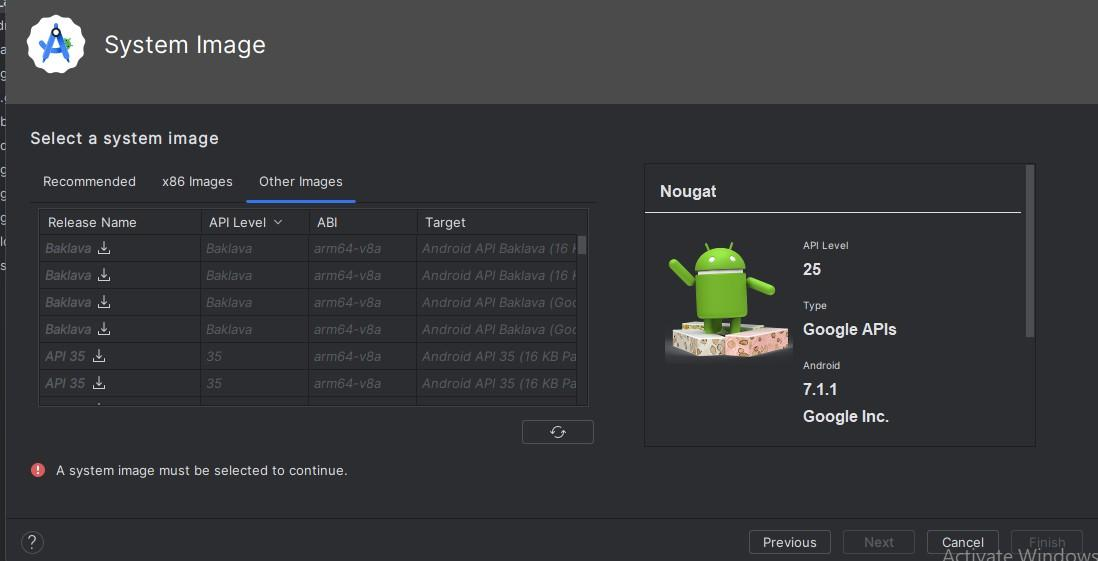


**Step 8.5:** Last, click on the icon pointed into the red color rectangle. The Android emulator



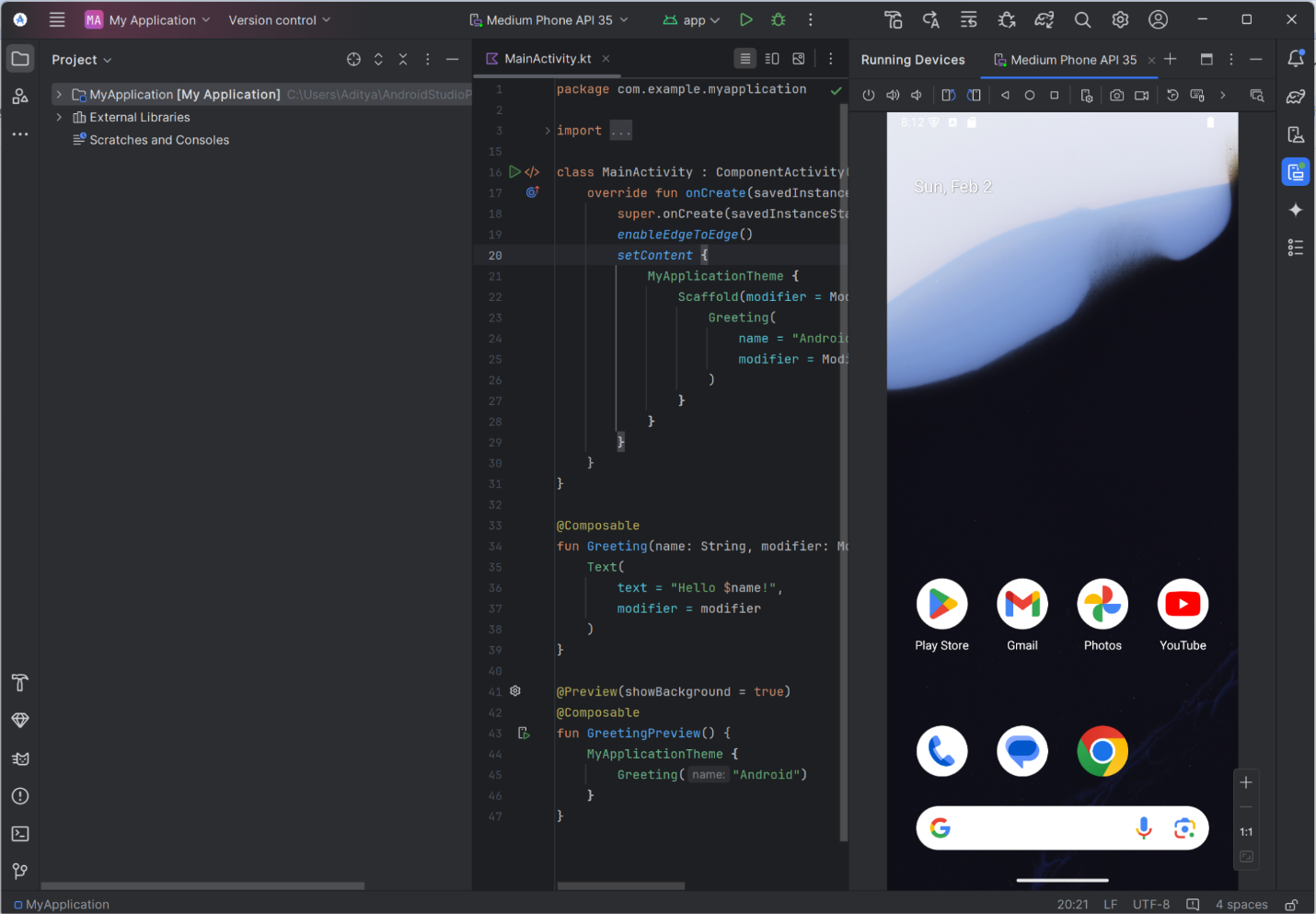
**Step 9:** Now, install Flutter and Dart plugin for building Flutter application in Android Studio. These plugins provide a template to create a Flutter application, give an option to run and debug Flutter application in the Android Studio itself. Do the following steps to install these plugins.

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



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

**Step 9.3:** Restart the Android Studio.



Conclusion: installation bundle of the Flutter Software Development Kit for windows with Android Studio.

**Experiment 02 :**

**To design Flutter UI by including common widgets.**

**Designing Flutter UI with Common Widgets**

Flutter provides a rich set of widgets to build beautiful and responsive user interfaces. Here are some commonly used widgets that help in designing a Flutter UI:

**1. Basic Structure Widgets**

* **Scaffold** → Provides a framework for material design apps (AppBar, body, FloatingActionButton).
* **AppBar** → A top navigation bar for branding, title, and actions.
* **Container** → A flexible widget for styling (padding, margin, decoration).
* **Column & Row** → For vertical and horizontal layouts.
* **Stack** → For overlaying widgets on top of each other.

**2. Common Input Widgets**

* **TextField** → For user text input.
* **TextFormField** → Same as TextField but with validation.
* **DropdownButton** → For selecting values from a dropdown.
* **Checkbox & Radio** → For selecting options.
* **Switch** → For toggling ON/OFF values.

**3. Display Widgets**

* **Text** → Displays static and dynamic text.
* **Image** → Displays images from assets or network.
* **Icon** → Displays icons from Flutter’s icon set.
* **Card** → A material design card container.
* **ListView** → For scrolling lists.

**4. Buttons and Actions**

* **ElevatedButton** → A button with elevation.
* **TextButton** → A simple text button.
* **IconButton** → A button with an icon.
* **FloatingActionButton** → A circular button for actions.

**5. Layout and Navigation Widgets**

* **Padding & SizedBox** → For spacing and alignment.
* **Drawer** → A sidebar menu for navigation.
* **BottomNavigationBar** → For bottom tab navigation.
* **PageView** → For creating swiping pages.
* **TabBar & TabBarView** → For tab-based navigation.

We can split Flutter widgets into two categories:

* **Visible (Output and Input)**
* **Invisible (Layout and Control)**

**Visible Widgets**

Visible widgets are related to user input and output data. Some of the important types of these widgets are:

Code :

import 'package:fintracker/bloc/cubit/app\_cubit.dart';  
import 'package:fintracker/screens/main.screen.dart';  
import 'package:flutter/material.dart';  
import 'package:flutter/services.dart';  
import 'package:flutter\_bloc/flutter\_bloc.dart';  
import 'package:flutter\_localizations/flutter\_localizations.dart';  
  
class App extends StatelessWidget {  
 const App({super.key});  
 @override  
 Widget build(BuildContext context) {  
 SystemChrome.*setSystemUIOverlayStyle*(SystemUiOverlayStyle(  
 statusBarColor: Colors.*transparent*,  
 statusBarIconBrightness: MediaQuery.*of*(context).platformBrightness  
 ));  
 return BlocBuilder<AppCubit, AppState>(  
 builder: (context, state){  
 return MaterialApp(  
 title: 'Fintracker',  
 theme: ThemeData(  
 useMaterial3: true,  
 brightness: MediaQuery.*of*(context).platformBrightness,  
 navigationBarTheme: NavigationBarThemeData(  
 labelTextStyle: WidgetStateProperty.*resolveWith*((Set<WidgetState> states){  
 TextStyle style = const TextStyle(fontWeight: FontWeight.*w500*, fontSize: 11);  
 if(states.contains(WidgetState.selected)){  
 style = style.merge(const TextStyle(fontWeight: FontWeight.*w600*));  
 }  
 return style;  
 }),  
 )  
 ),  
 home: const MainScreen(),  
 localizationsDelegates: const [  
 GlobalWidgetsLocalizations.*delegate*,  
 GlobalMaterialLocalizations.*delegate*,  
 ],  
 );  
 }  
 );  
 }  
}

**1. Text**

A Text widget holds some text to display on the screen. We can align the text widget by using the textAlign property, and the style property allows customization of the text, including font, weight, style, letter spacing, and color.

**Example:**

new Text(

'Hello, ALL!',

textAlign: TextAlign.center,

style: new TextStyle(fontWeight: FontWeight.bold),

)

**2. Button**

This widget allows users to perform some actions on a click. Flutter does not allow direct use of a Button widget; instead, it provides different types like FlatButton and RaisedButton.

**Example:**

// FlatButton Example

new FlatButton(

child: Text("Click here"),

onPressed: () {

// Do something here

},

),

// RaisedButton Example

new RaisedButton(

child: Text("Click here"),

elevation: 5.0,

onPressed: () {

// Do something here

},

)

**3. Image**

This widget holds an image, which can be fetched from multiple sources such as an asset folder or directly from a URL. It provides many constructors for loading images:

* Image: A generic image loader using ImageProvider.
* asset: Loads an image from the project's asset folder.
* file: Loads an image from the system folder.
* memory: Loads an image from memory.
* network: Loads an image from the internet.

**Adding an Image in Flutter**

To add an image to the project, first create an assets folder where the image is stored, and then add the following line to the pubspec.yaml file:

assets:

- assets/comp.jpg

**Example:**

import 'package:flutter/material.dart';

void main() {

runApp(const MyApp());

}

class MyApp extends StatelessWidget {

const MyApp({Key? key}) : super(key: key);

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Welcome to Flutter',

home: Scaffold(

appBar: AppBar(

title: const Text('Welcome to Flutter'),

),

body: Center(

child: Image.asset('assets/comp.jpg'),

),

),

);

}

}

**Output:**

The app will display an image from the assets folder.

**Single Child Widget Example:**

class MyButton extends StatelessWidget {

MyButton({Key? key}) : super(key: key);

@override

Widget build(BuildContext context) {

return Container(

decoration: const BoxDecoration(

border: Border(

top: BorderSide(width: 1.0, color: Color(0xFFFFFFFFFF)),

left: BorderSide(width: 1.0, color: Color(0xFFFFFFFFFF)),

right: BorderSide(width: 1.0, color: Color(0xFFFF000000)),

bottom: BorderSide(width: 1.0, color: Color(0xFFFF000000)),

),

),

child: Container(

padding: const EdgeInsets.symmetric(horizontal: 20.0, vertical: 2.0),

decoration: const BoxDecoration(

border: Border(

top: BorderSide(width: 1.0, color: Color(0xFFFFDFDFDF)),

left: BorderSide(width: 1.0, color: Color(0xFFFFDFDFDF)),

right: BorderSide(width: 1.0, color: Color(0xFFFF7F7F7F)),

bottom: BorderSide(width: 1.0, color: Color(0xFFFF7F7F7F)),

),

color: Colors.grey,

),

child: const Text(

'OK',

textAlign: TextAlign.center,

style: TextStyle(color: Colors.black),

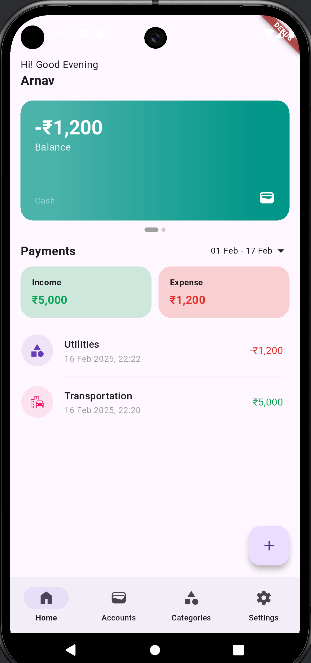
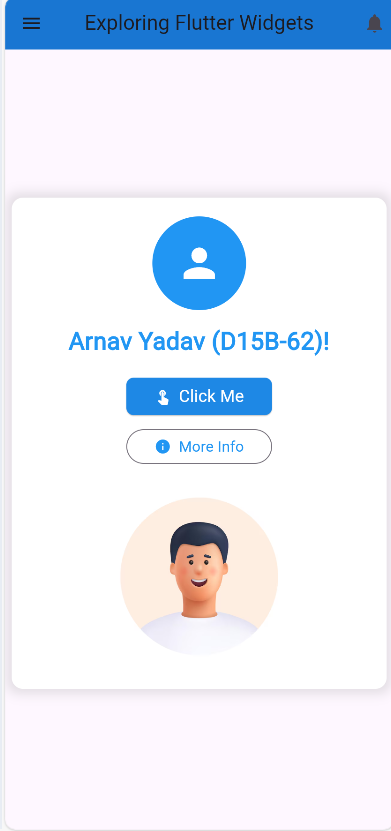
),

),

);

}

}



***Conclusion:***

In this experiment, we explored Flutter widgets, specifically focusing on Text, Button, and Image widgets. These widgets play a crucial role in building interactive UI elements in Flutter applications. We investigated their properties, customization options, and how they can be combined to create more complex layouts. Furthermore, we examined how to handle user interactions with buttons and display images from various sources.

Experiment No – 6

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 android = FirebaseOptions(

    apiKey: 'AIzaSyDHwT2KEliuAewEJZiWN9XhaPOIFOI2tQM',

    appId: '1:293282519154:android:3980abd5ef153a86904aeb',

    messagingSenderId: '293282519154',

    projectId: 'mpl-finance-buddy',

    storageBucket: 'mpl-finance-buddy.firebasestorage.app',

  );

  static const FirebaseOptions ios = FirebaseOptions(

    apiKey: 'AIzaSyAASp57dwfIm56CYNHHkO1iN\_l2z4r7ZCs',

    appId: '1:293282519154:ios:abb65e9b985bd6d3904aeb',

    messagingSenderId: '293282519154',

    projectId: 'mpl-finance-buddy',

    storageBucket: 'mpl-finance-buddy.firebasestorage.app',

    iosBundleId: 'com.example.arnavmobile',

  );

  static const FirebaseOptions macos = FirebaseOptions(

    apiKey: 'AIzaSyAASp57dwfIm56CYNHHkO1iN\_l2z4r7ZCs',

    appId: '1:293282519154:ios:abb65e9b985bd6d3904aeb',

    messagingSenderId: '293282519154',

    projectId: 'mpl-finance-buddy',

    storageBucket: 'mpl-finance-buddy.firebasestorage.app',

    iosBundleId: 'com.example.arnavmobile',

  );

  static const FirebaseOptions windows = FirebaseOptions(

    apiKey: 'AIzaSyAl52vyqXJUiNUq87rt\_dYTqXsxfandtpw',

    appId: '1:293282519154:web:0da86bd3a9d34bb2904aeb',

    messagingSenderId: '293282519154',

    projectId: 'mpl-finance-buddy',

    authDomain: 'mpl-finance-buddy.firebaseapp.com',

    storageBucket: 'mpl-finance-buddy.firebasestorage.app',

  );

  static const FirebaseOptions web = FirebaseOptions(

    apiKey: 'AIzaSyAl52vyqXJUiNUq87rt\_dYTqXsxfandtpw',

    appId: '1:293282519154:web:2a8ae265b158d4c8904aeb',

    messagingSenderId: '293282519154',

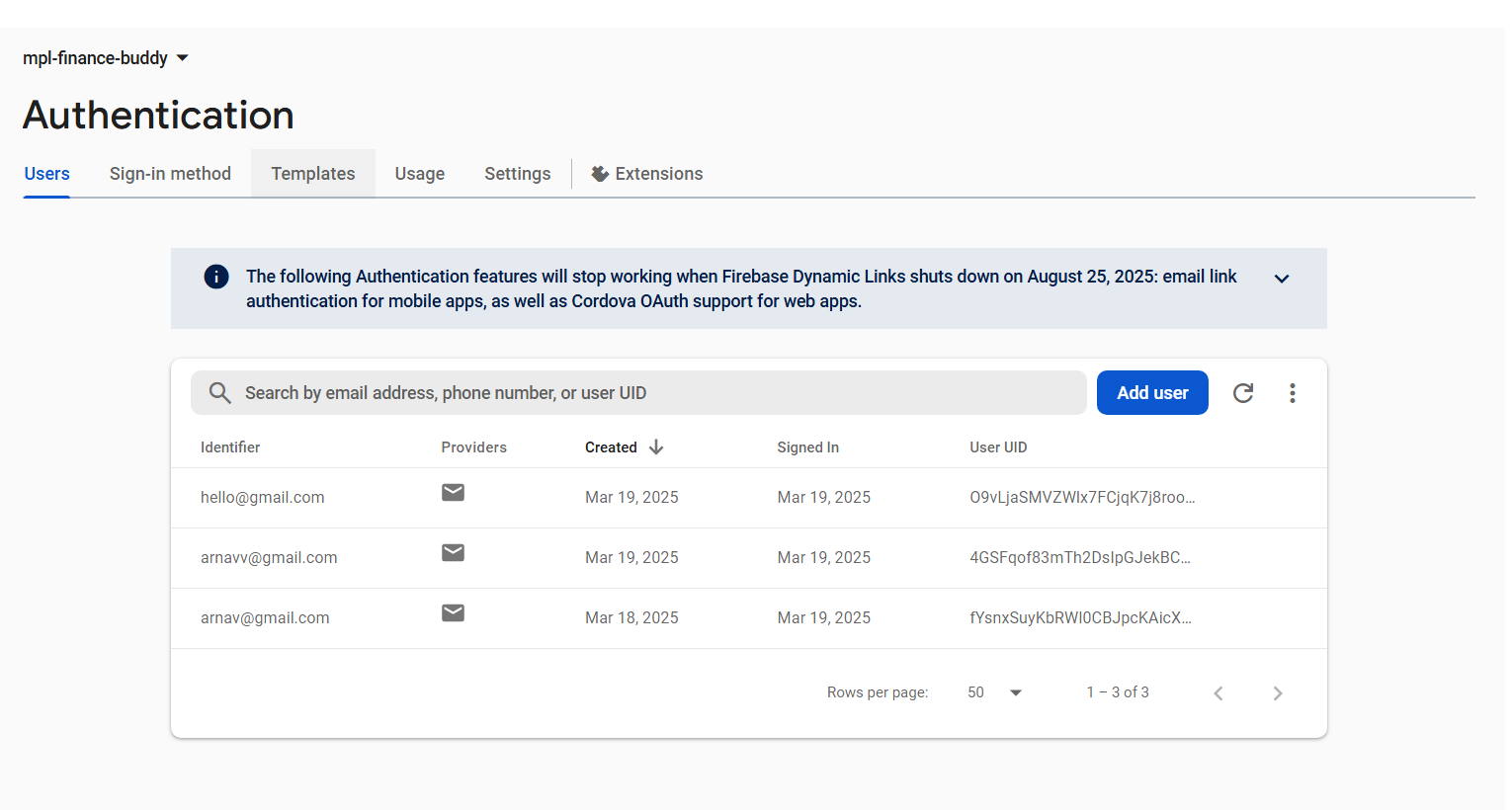
    projectId: 'mpl-finance-buddy',

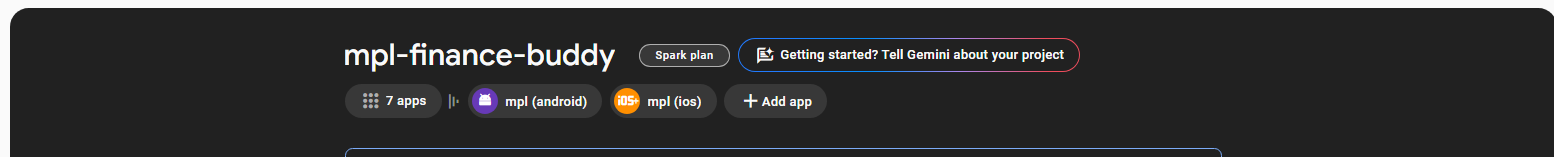
    authDomain: 'mpl-finance-buddy.firebaseapp.com',

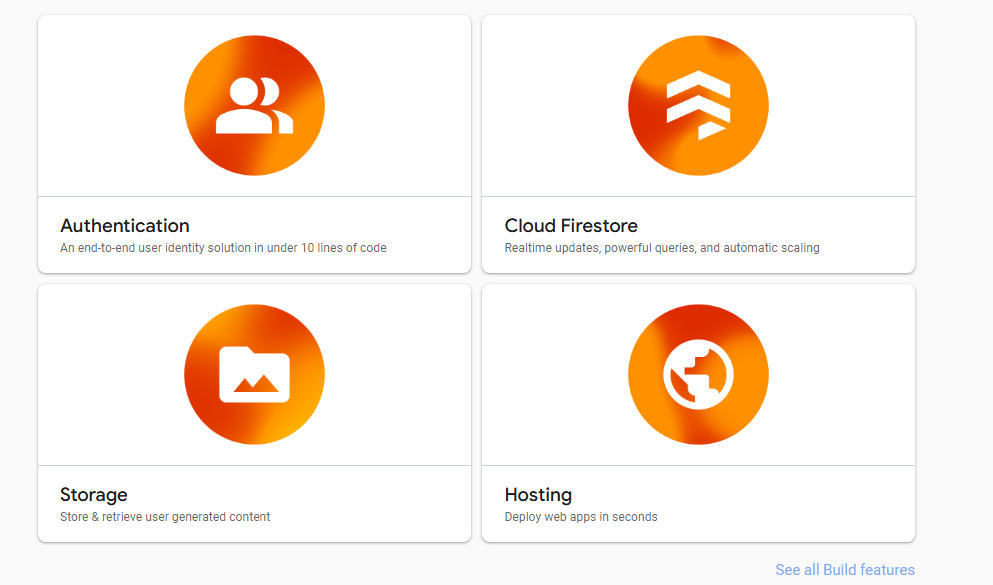
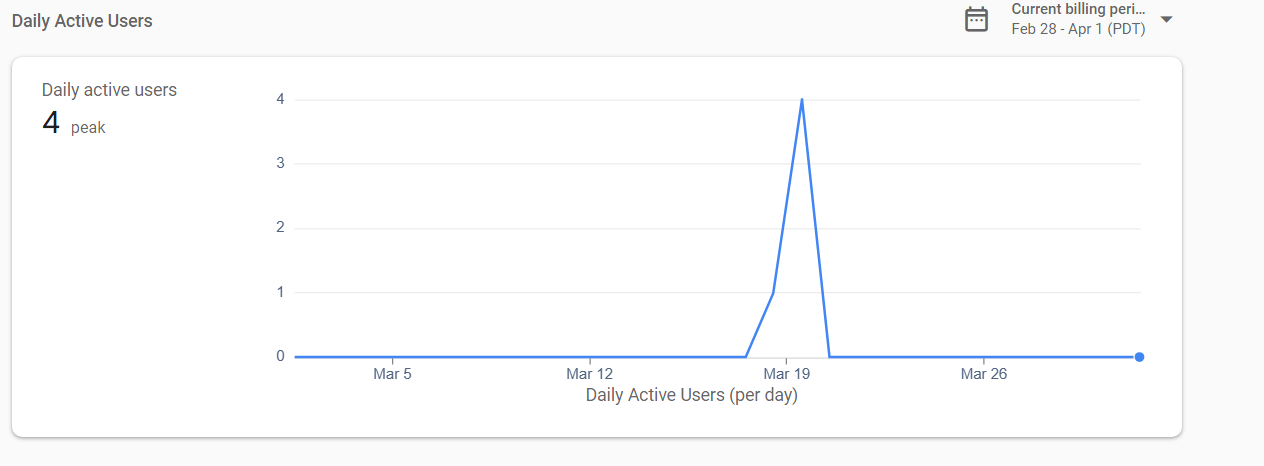
    storageBucket: 'mpl-finance-buddy.firebasestorage.app',

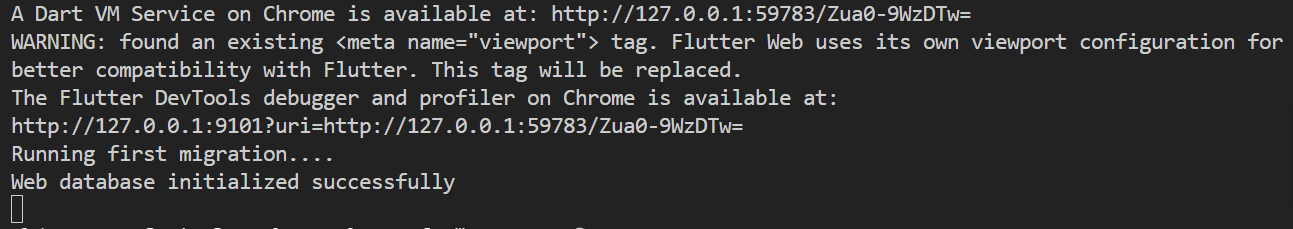
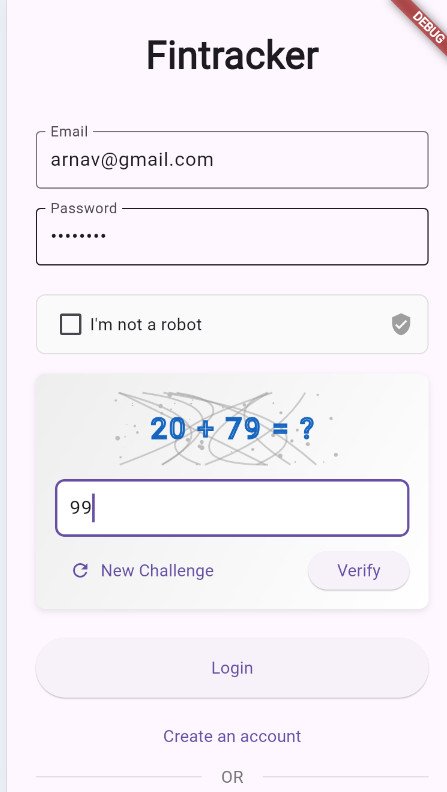
  );

}

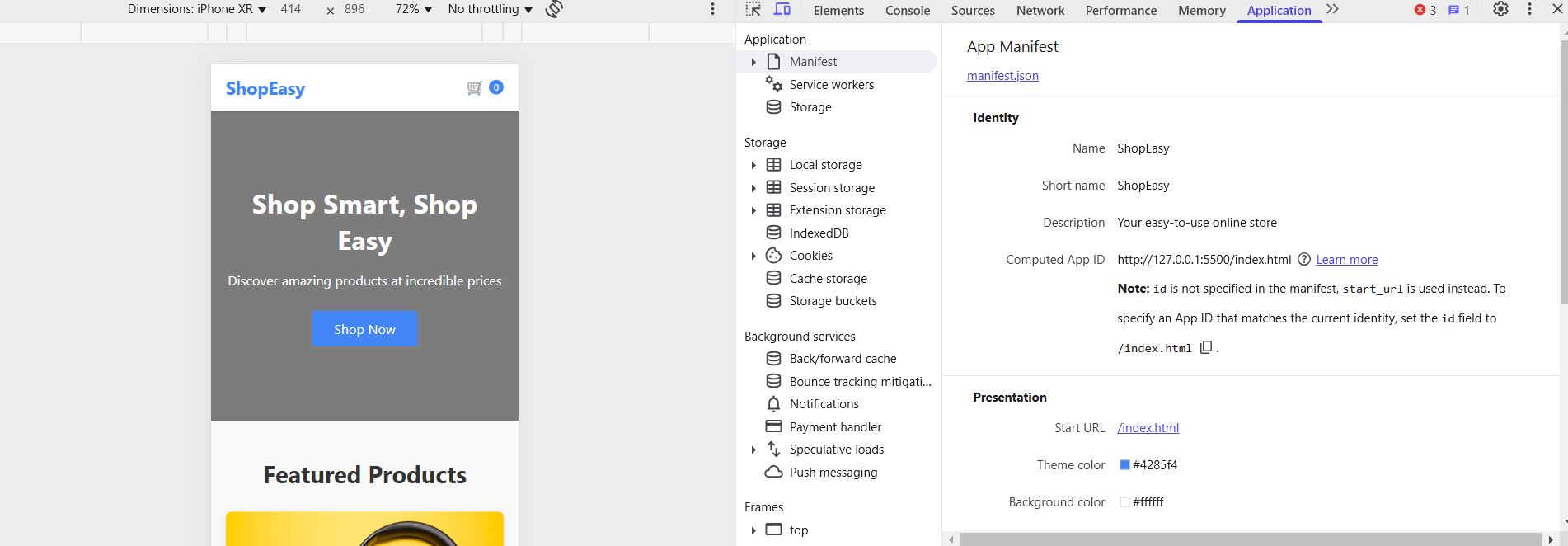


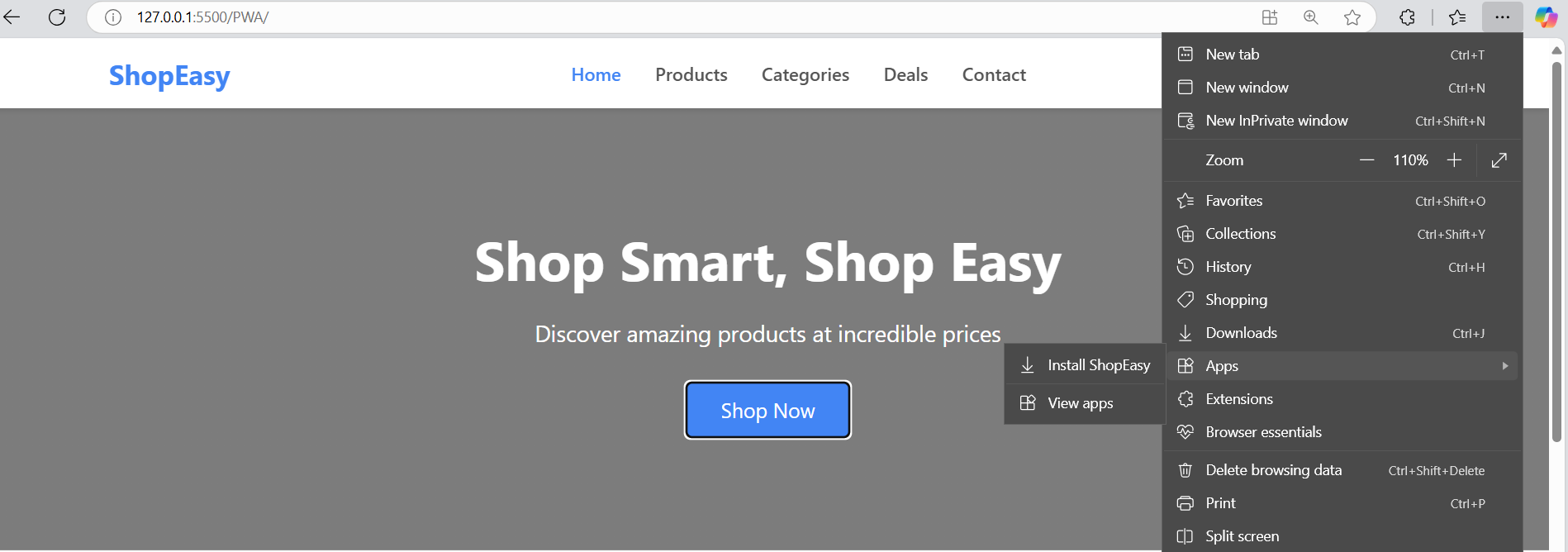


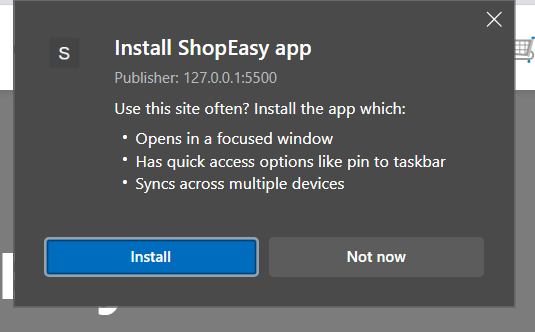


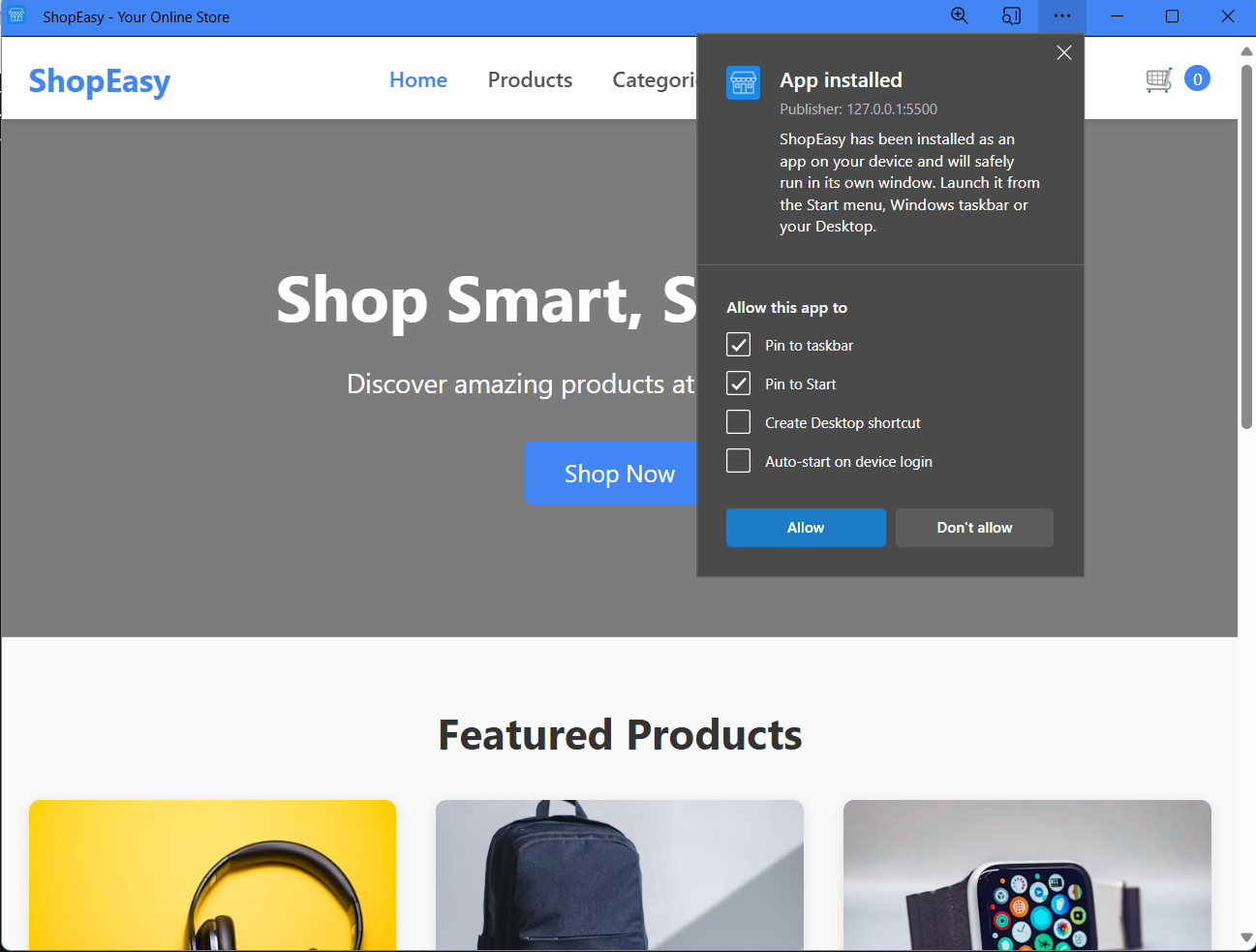


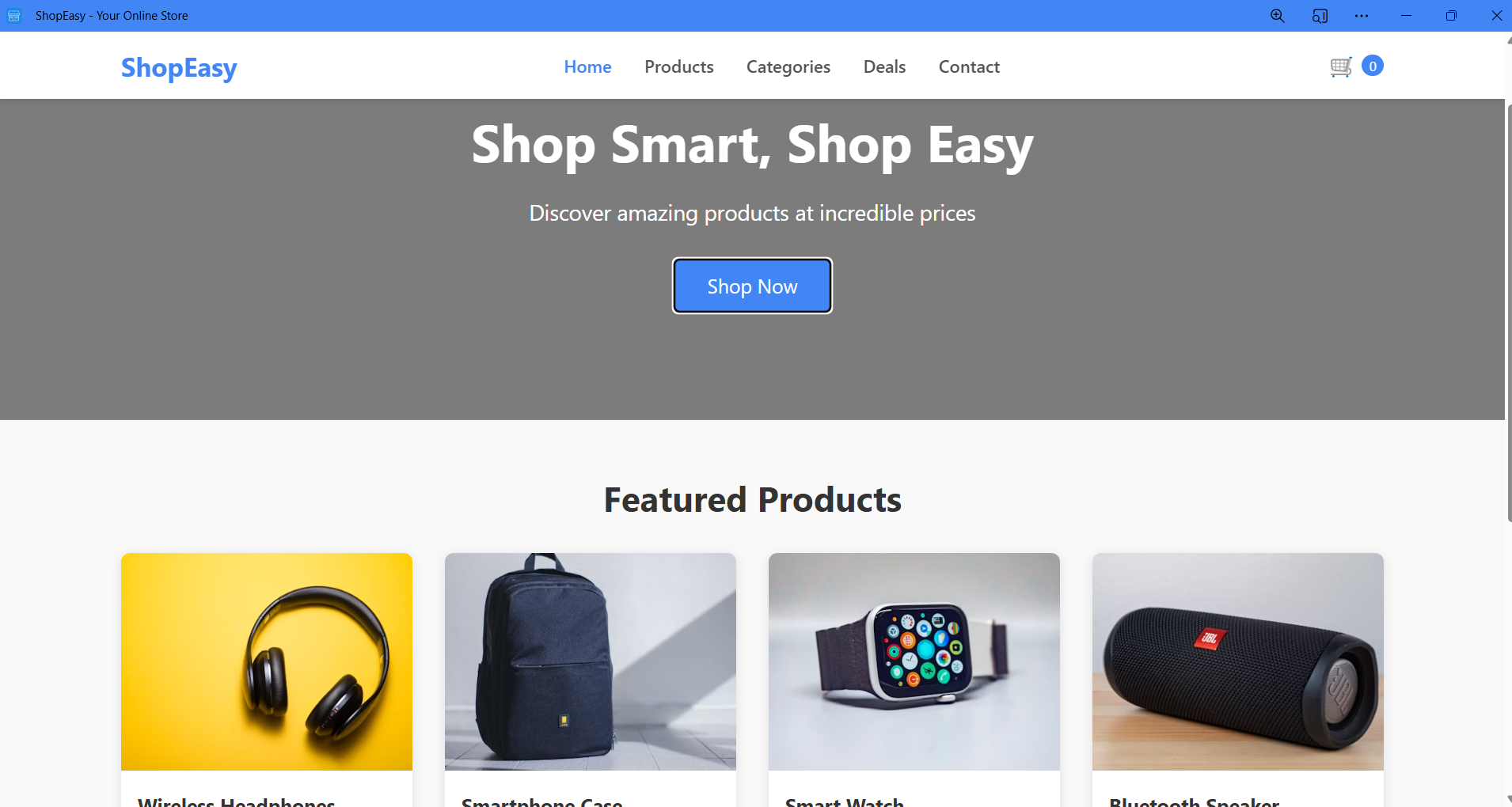
**Experiment No 7**











Experiment N0 8

Code:

// PWA functionality

let deferredPrompt;

const installBanner = document.getElementById('install-banner');

const installButton = document.getElementById('install-button');

const dismissButton = document.getElementById('dismiss-button');

// Check if service worker is supported

if ('serviceWorker' in navigator) {

    window.addEventListener('load', () => {

        navigator.serviceWorker.register('service-worker.js')

            .then(registration => {

                console.log('Service Worker registered with scope:', registration.scope);

            })

            .catch(error => {

                console.error('Service Worker registration failed:', error);

            });

    });

}

// Listen for the beforeinstallprompt event

window.addEventListener('beforeinstallprompt', (e) => {

    // Prevent Chrome 67 and earlier from automatically showing the prompt

    e.preventDefault();

    // Stash the event so it can be triggered later

    deferredPrompt = e;

    // Show the install banner

    installBanner.classList.remove('hidden');

});

// Install button click handler

installButton.addEventListener('click', () => {

    // Hide the install banner

    installBanner.classList.add('hidden');

    // Show the install prompt

    if (deferredPrompt) {

        deferredPrompt.prompt();

        // Wait for the user to respond to the prompt

        deferredPrompt.userChoice.then((choiceResult) => {

            if (choiceResult.outcome === 'accepted') {

                console.log('User accepted the install prompt');

            } else {

                console.log('User dismissed the install prompt');

            }

            // Clear the deferredPrompt variable

            deferredPrompt = null;

        });

    }

});

// Dismiss button click handler

dismissButton.addEventListener('click', () => {

    installBanner.classList.add('hidden');

});

// E-commerce functionality

const cartCount = document.querySelector('.cart-count');

const addToCartButtons = document.querySelectorAll('.add-to-cart');

let cartItems = 0;

// Add to cart functionality

addToCartButtons.forEach(button => {

    button.addEventListener('click', () => {

        cartItems++;

        cartCount.textContent = cartItems;

        // Animate button

        button.textContent = 'Added!';

        button.style.backgroundColor = '#4caf50';

        setTimeout(() => {

            button.textContent = 'Add to Cart';

            button.style.backgroundColor = '#4285f4';

        }, 1000);

    });

});

// Service Worker installation and activation

self.addEventListener('install', (event) => {

    console.log('Service Worker installing.');

    // Perform install steps

    event.waitUntil(

        caches.open('ecommerce-cache-v1')

            .then((cache) => {

                console.log('Opened cache');

                return cache.addAll([

                    '/',

                    '/index.html',

                    '/styles.css',

                    '/script.js',

                    '/images/logo.png',

                    '/images/shop\_icon.png',

                    // Add other assets to cache

                ]);

            })

    );

});

self.addEventListener('activate', (event) => {

    console.log('Service Worker activating.');

    const cacheWhitelist = ['ecommerce-cache-v1'];

    event.waitUntil(

        caches.keys().then((cacheNames) => {

            return Promise.all(

                cacheNames.map((cacheName) => {

                    if (cacheWhitelist.indexOf(cacheName) === -1) {

                        return caches.delete(cacheName);

                    }

                })

            );

        })

    );

});

self.addEventListener('fetch', (event) => {

    event.respondWith(

        caches.match(event.request)

            .then((response) => {

                // Cache hit - return response

                if (response) {

                    return response;

                }

                return fetch(event.request);

            })

    );

});

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

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

We can split Flutter widgets into two categories:

* **Visible (Output and Input)**
* **Invisible (Layout and Control)**

**Visible Widgets**

Visible widgets are related to user input and output data. Some of the important types of these widgets are:

Code :

import 'package:fintracker/bloc/cubit/app\_cubit.dart';  
import 'package:fintracker/screens/main.screen.dart';  
import 'package:flutter/material.dart';  
import 'package:flutter/services.dart';  
import 'package:flutter\_bloc/flutter\_bloc.dart';  
import 'package:flutter\_localizations/flutter\_localizations.dart';  
  
class App extends StatelessWidget {  
 const App({super.key});  
 @override  
 Widget build(BuildContext context) {  
 SystemChrome.*setSystemUIOverlayStyle*(SystemUiOverlayStyle(  
 statusBarColor: Colors.*transparent*,  
 statusBarIconBrightness: MediaQuery.*of*(context).platformBrightness  
 ));  
 return BlocBuilder<AppCubit, AppState>(  
 builder: (context, state){  
 return MaterialApp(  
 title: 'Fintracker',  
 theme: ThemeData(  
 useMaterial3: true,  
 brightness: MediaQuery.*of*(context).platformBrightness,  
 navigationBarTheme: NavigationBarThemeData(  
 labelTextStyle: WidgetStateProperty.*resolveWith*((Set<WidgetState> states){  
 TextStyle style = const TextStyle(fontWeight: FontWeight.*w500*, fontSize: 11);  
 if(states.contains(WidgetState.selected)){  
 style = style.merge(const TextStyle(fontWeight: FontWeight.*w600*));  
 }  
 return style;  
 }),  
 )  
 ),  
 home: const MainScreen(),  
 localizationsDelegates: const [  
 GlobalWidgetsLocalizations.*delegate*,  
 GlobalMaterialLocalizations.*delegate*,  
 ],  
 );  
 }  
 );  
 }  
}

**1. Text**

A Text widget holds some text to display on the screen. We can align the text widget by using the textAlign property, and the style property allows customization of the text, including font, weight, style, letter spacing, and color.

**Example:**

new Text(

'Hello, ALL!',

textAlign: TextAlign.center,

style: new TextStyle(fontWeight: FontWeight.bold),

)

**2. Button**

This widget allows users to perform some actions on a click. Flutter does not allow direct use of a Button widget; instead, it provides different types like FlatButton and RaisedButton.

**Example:**

// FlatButton Example

new FlatButton(

child: Text("Click here"),

onPressed: () {

// Do something here

},

),

// RaisedButton Example

new RaisedButton(

child: Text("Click here"),

elevation: 5.0,

onPressed: () {

// Do something here

},

)

**3. Image**

This widget holds an image, which can be fetched from multiple sources such as an asset folder or directly from a URL. It provides many constructors for loading images:

* Image: A generic image loader using ImageProvider.
* asset: Loads an image from the project's asset folder.
* file: Loads an image from the system folder.
* memory: Loads an image from memory.
* network: Loads an image from the internet.

**Adding an Image in Flutter**

To add an image to the project, first create an assets folder where the image is stored, and then add the following line to the pubspec.yaml file:

assets:

- assets/comp.jpg

**Example:**

import 'package:flutter/material.dart';

void main() {

runApp(const MyApp());

}

class MyApp extends StatelessWidget {

const MyApp({Key? key}) : super(key: key);

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Welcome to Flutter',

home: Scaffold(

appBar: AppBar(

title: const Text('Welcome to Flutter'),

),

body: Center(

child: Image.asset('assets/comp.jpg'),

),

),

);

}

}

**Output:**

The app will display an image from the assets folder.

**Single Child Widget Example:**

class MyButton extends StatelessWidget {

MyButton({Key? key}) : super(key: key);

@override

Widget build(BuildContext context) {

return Container(

decoration: const BoxDecoration(

border: Border(

top: BorderSide(width: 1.0, color: Color(0xFFFFFFFFFF)),

left: BorderSide(width: 1.0, color: Color(0xFFFFFFFFFF)),

right: BorderSide(width: 1.0, color: Color(0xFFFF000000)),

bottom: BorderSide(width: 1.0, color: Color(0xFFFF000000)),

),

),

child: Container(

padding: const EdgeInsets.symmetric(horizontal: 20.0, vertical: 2.0),

decoration: const BoxDecoration(

border: Border(

top: BorderSide(width: 1.0, color: Color(0xFFFFDFDFDF)),

left: BorderSide(width: 1.0, color: Color(0xFFFFDFDFDF)),

right: BorderSide(width: 1.0, color: Color(0xFFFF7F7F7F)),

bottom: BorderSide(width: 1.0, color: Color(0xFFFF7F7F7F)),

),

color: Colors.grey,

),

child: const Text(

'OK',

textAlign: TextAlign.center,

style: TextStyle(color: Colors.black),

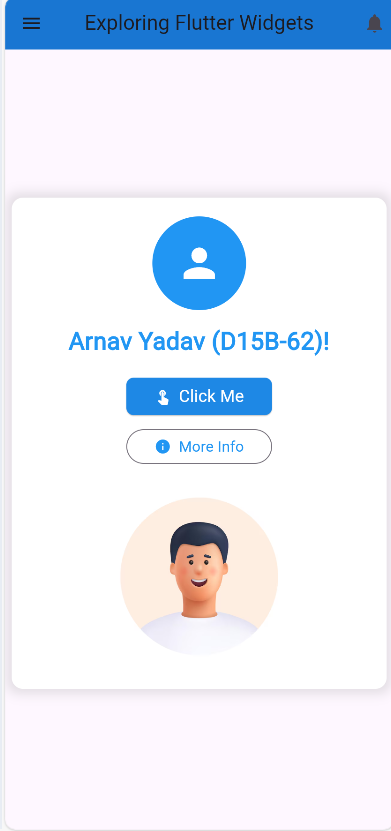
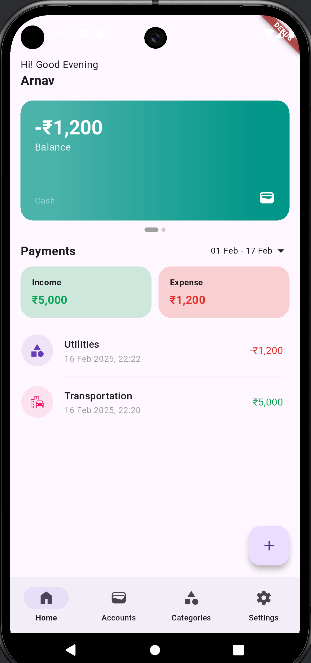
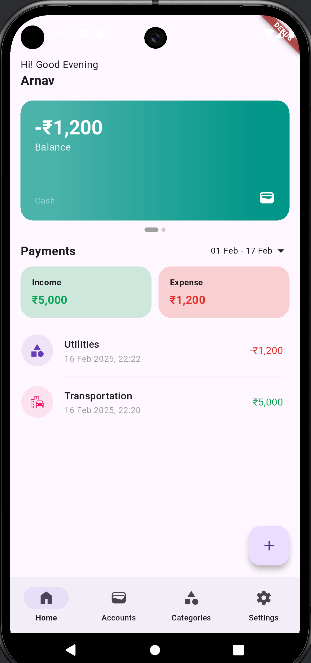
),

),

);

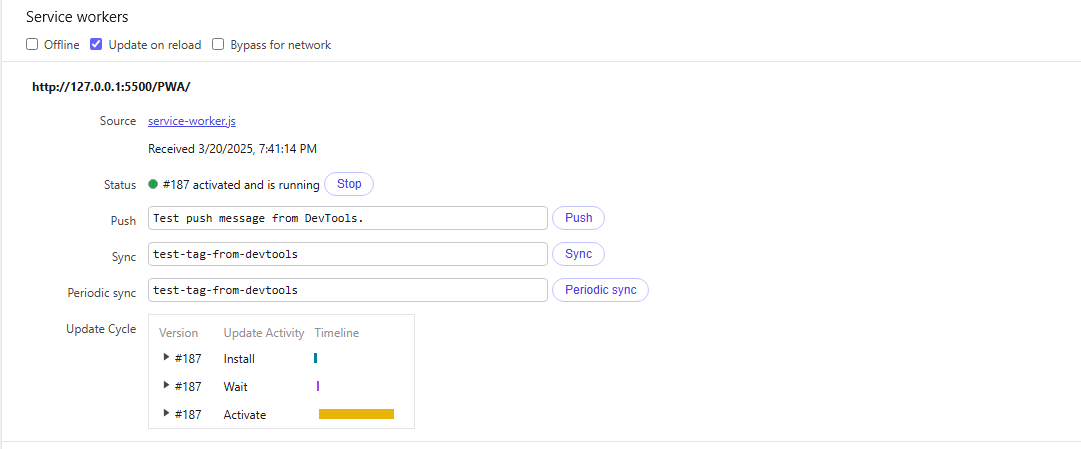
}

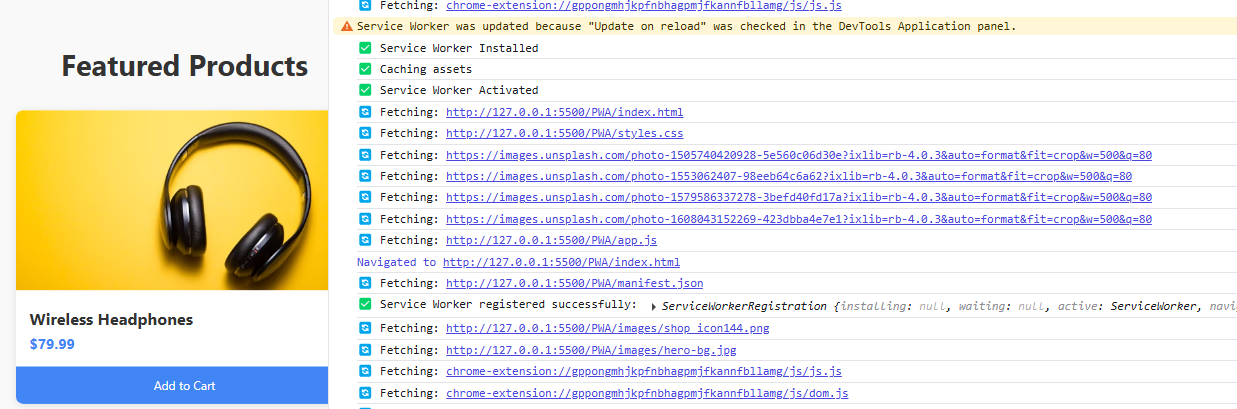
}

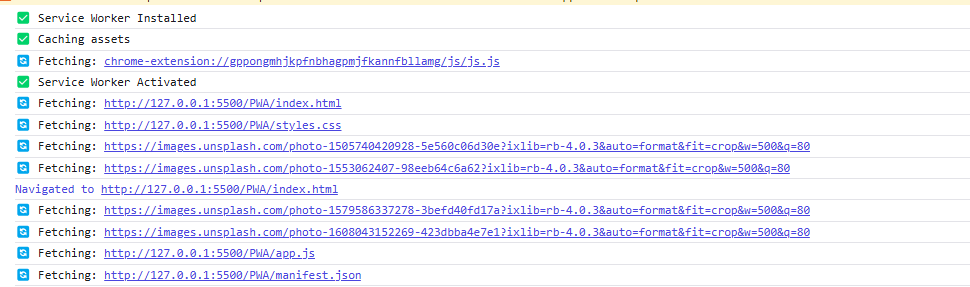


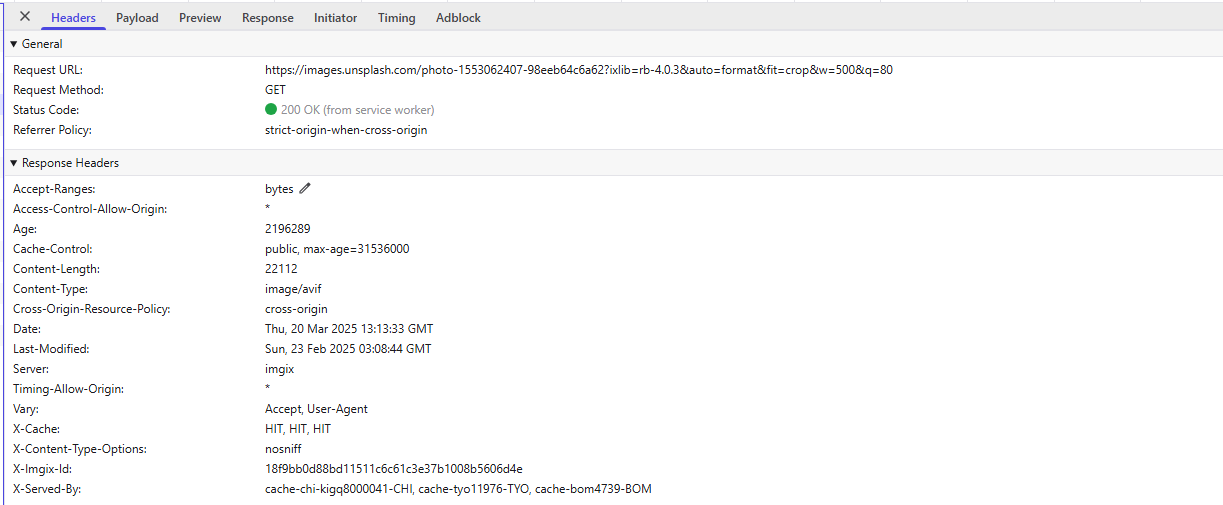
***Conclusion:***

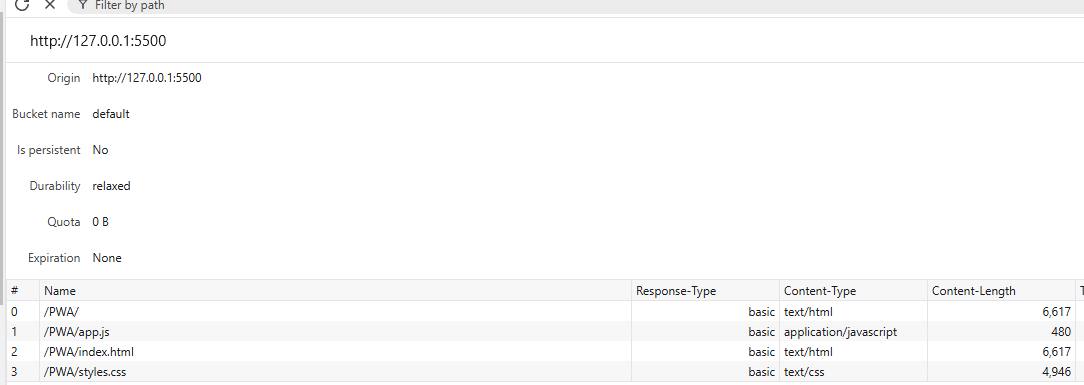
In this experiment, we explored Flutter widgets, specifically focusing on Text, Button, and Image widgets. These widgets play a crucial role in building interactive UI elements in Flutter applications. We investigated their properties, customization options, and how they can be combined to create more complex layouts. Furthermore, we examined how to handle user interactions with buttons and display images from various sources.











Experiment No 9

**Code:**

const CACHE\_NAME = "pwa-cache-v1";

const ASSETS\_TO\_CACHE = [

"/PWA/",

"/PWA/index.html",

"/PWA/styles.css",

"/PWA/app.js",

// Add other assets to cache

];

// Install event - Cache important assets

self.addEventListener("install", (event) => {

console.log("✅ Service Worker Installed");

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");

event.waitUntil(

caches.keys().then((cacheNames) =>

Promise.all(

cacheNames.map((cache) => {

if (cache !== CACHE\_NAME) {

console.log("🗑️ Deleting old cache:", cache);

return caches.delete(cache);

}

})

)

)

);

});

// Fetch event - Serve cached files when offline

self.addEventListener("fetch", (event) => {

console.log("🔄 Fetching:", event.request.url);

event.respondWith(

caches

.match(event.request)

.then((response) => response || fetch(event.request))

.catch(() => caches.match("/PWA/index.html")) // Fallback to index.html if offline

);

});

// Sync event - Background sync for failed requests

self.addEventListener("sync", (event) => {

if (event.tag === "sync-cart") {

event.waitUntil(syncCart());

}

});

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.json();

console.log("🔔 Push notification received:", data);

const options = {

body: data.body,

icon: "/PWA/icons/icon-192x192.png",

badge: "/PWA/icons/icon-192x192.png",

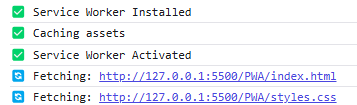
};

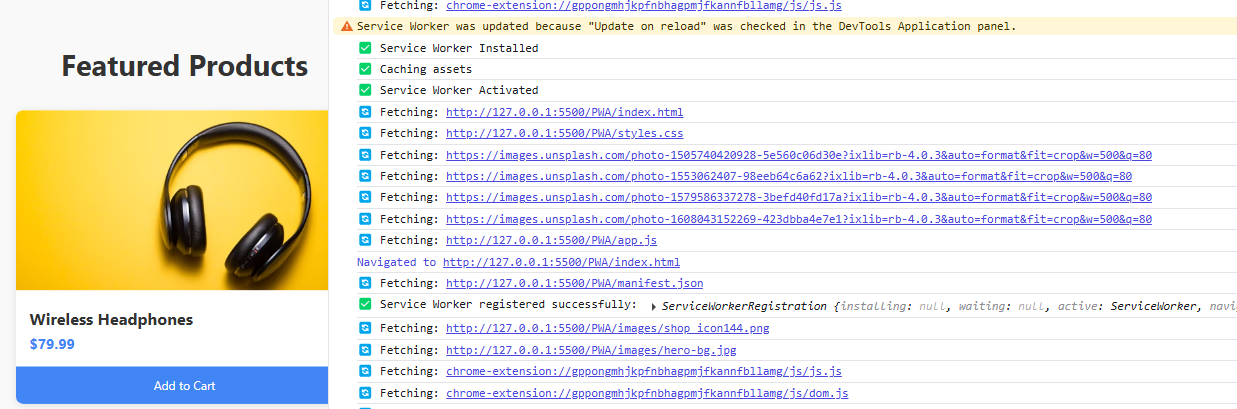
event.waitUntil(

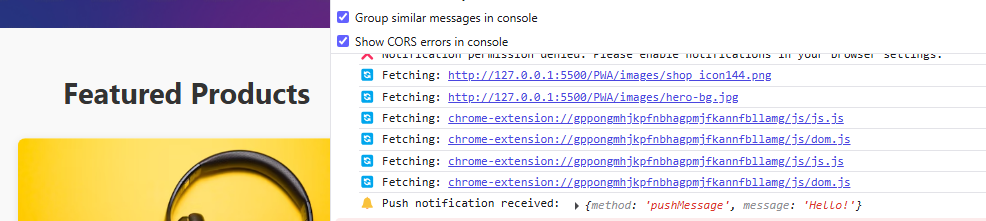
self.registration.showNotification(data.title, options)

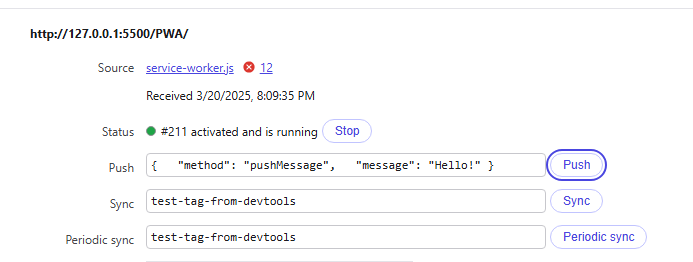
);

});





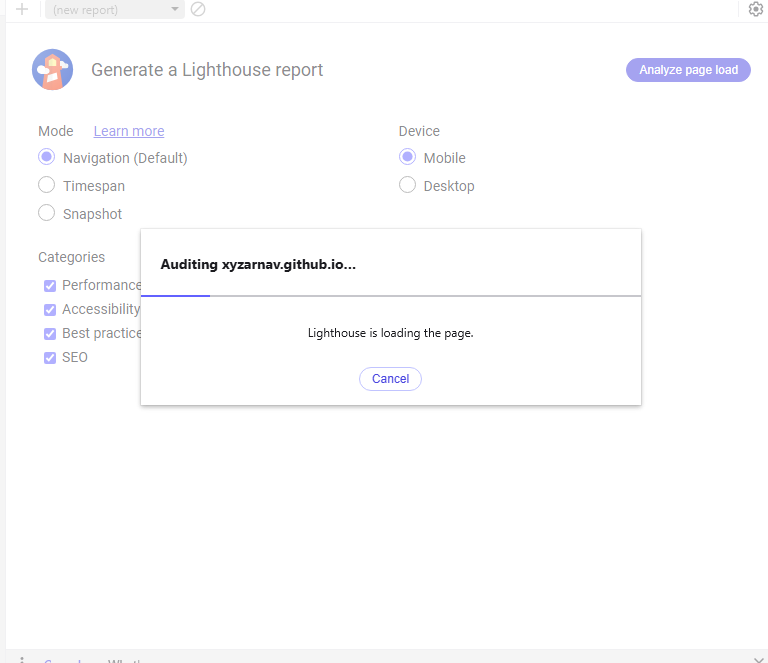






Experiment N0 9

**Experiment No 11**

****

