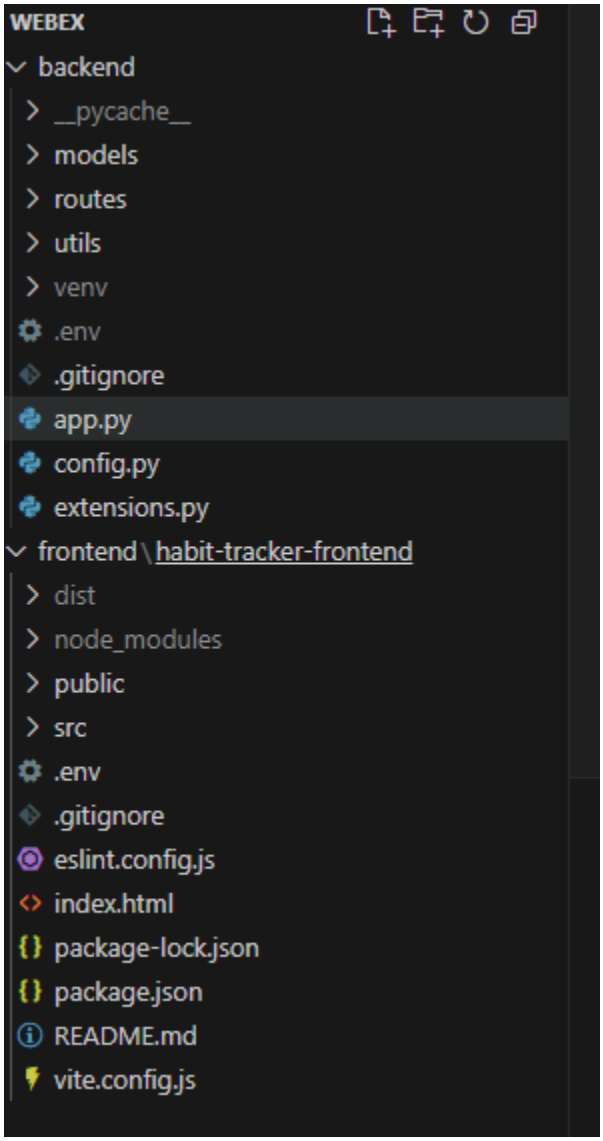
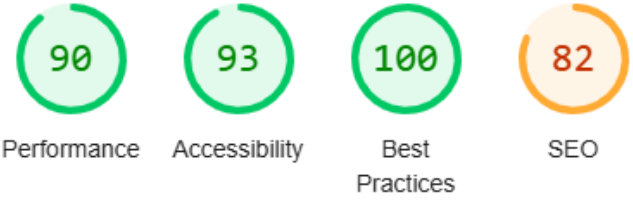
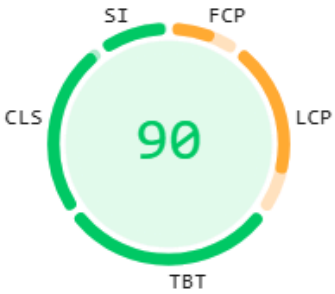
NAME - KARAN MISHRA
DIV - D15B
ROLL NO - 33
MPL & PWA EXP 11



https://sunny-axolotl-e5be37.netlify.app/dashboard





Performance

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

▲ 0–49 ■ 50–89 ● 90–100

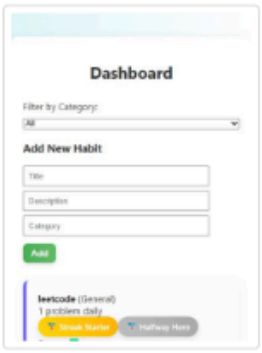
https://sunny-axolotl-e5be37.netlify.app/dashboard

90 93 100 82

Performance

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

▲ 0–49 ■ 50–89 ● 90–100



https://sunny-axolotl-e5be37.netlify.app/dashboard

90 93 100 82

First Contentful Paint
2.5 s

Largest Contentful Paint
3.0 s

Total Blocking Time
50 ms

Cumulative Layout Shift
0.064

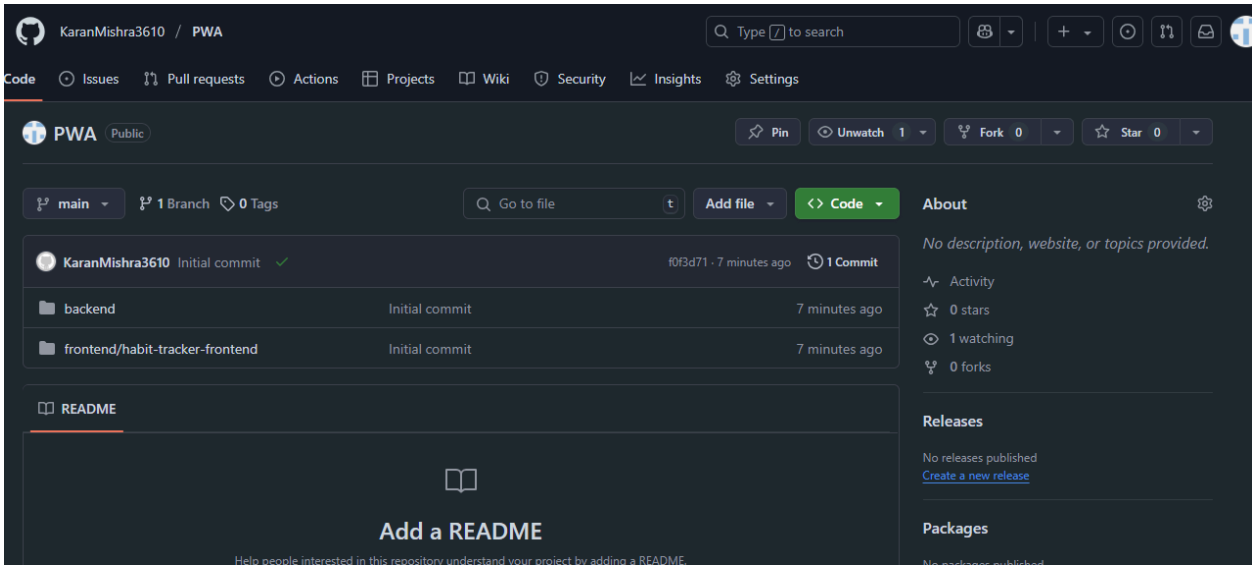
Speed Index
2.5 s

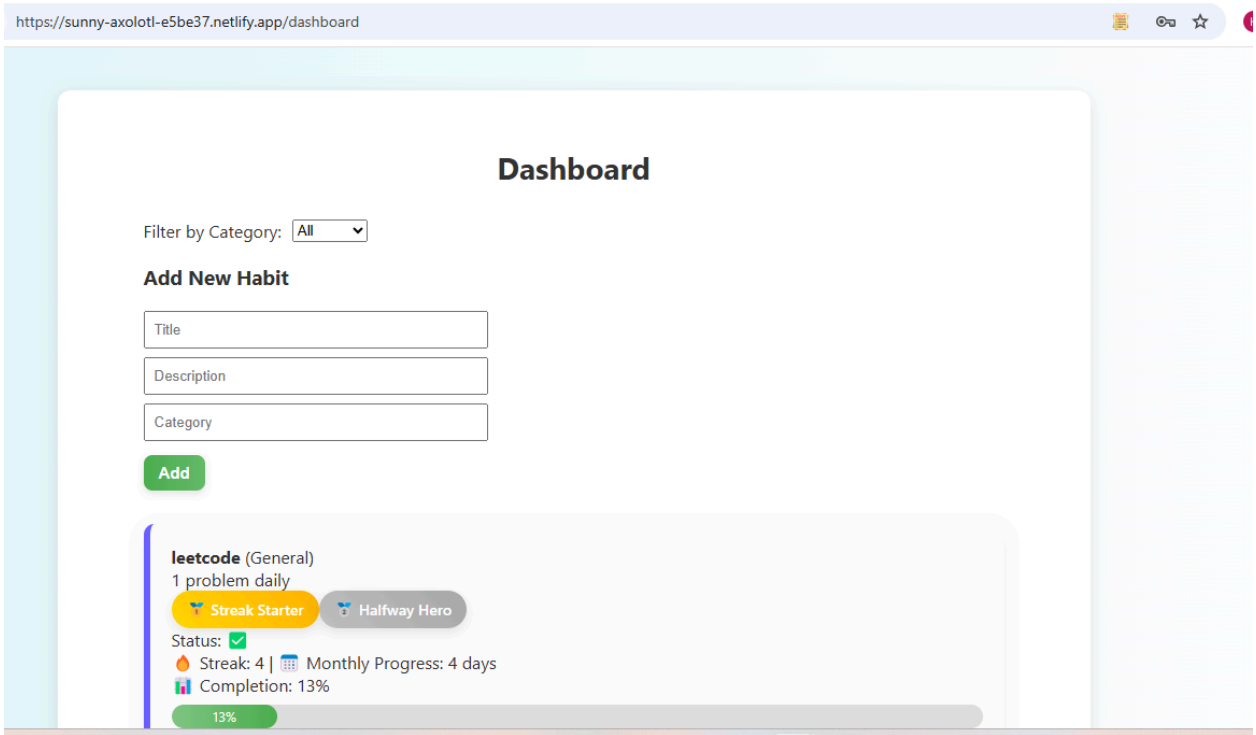
https://sunny-axolotl-e5be37.netlify.app/dashboard

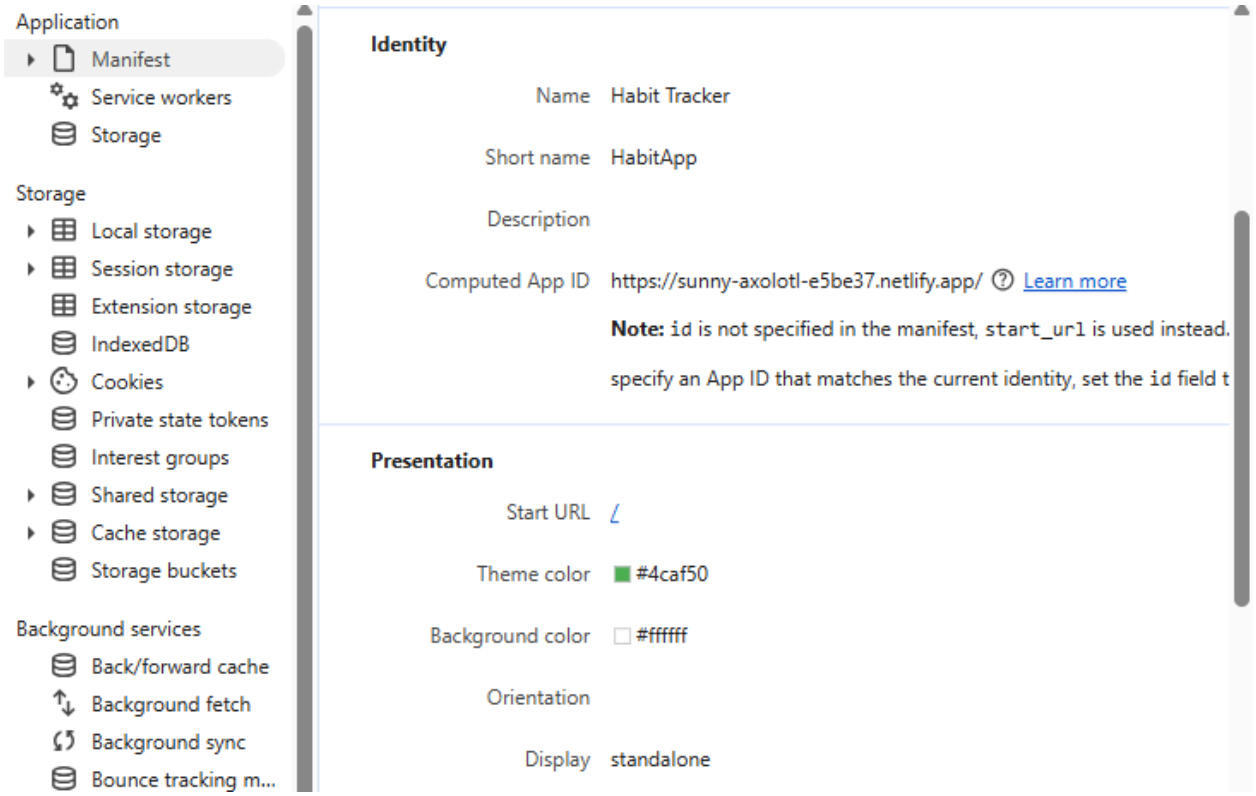
90 93 100 82

- Reduce unused JavaScript — Potential savings of 121 KiB
- Largest Contentful Paint element — 3,010 ms
- Preconnect to required origins — Potential savings of 150 ms
- Enable text compression — Potential savings of 2 KiB
- Avoid serving legacy JavaScript to modern browsers — Potential savings of 0 KiB
- Reduce unused CSS — Potential savings of 18 KiB
- Avoid large layout shifts — 1 layout shift found
- Avoid long main-thread tasks — 5 long tasks found
- Avoid non-composited animations — 2 animated elements found
- Avoid chaining critical requests — 2 chains found

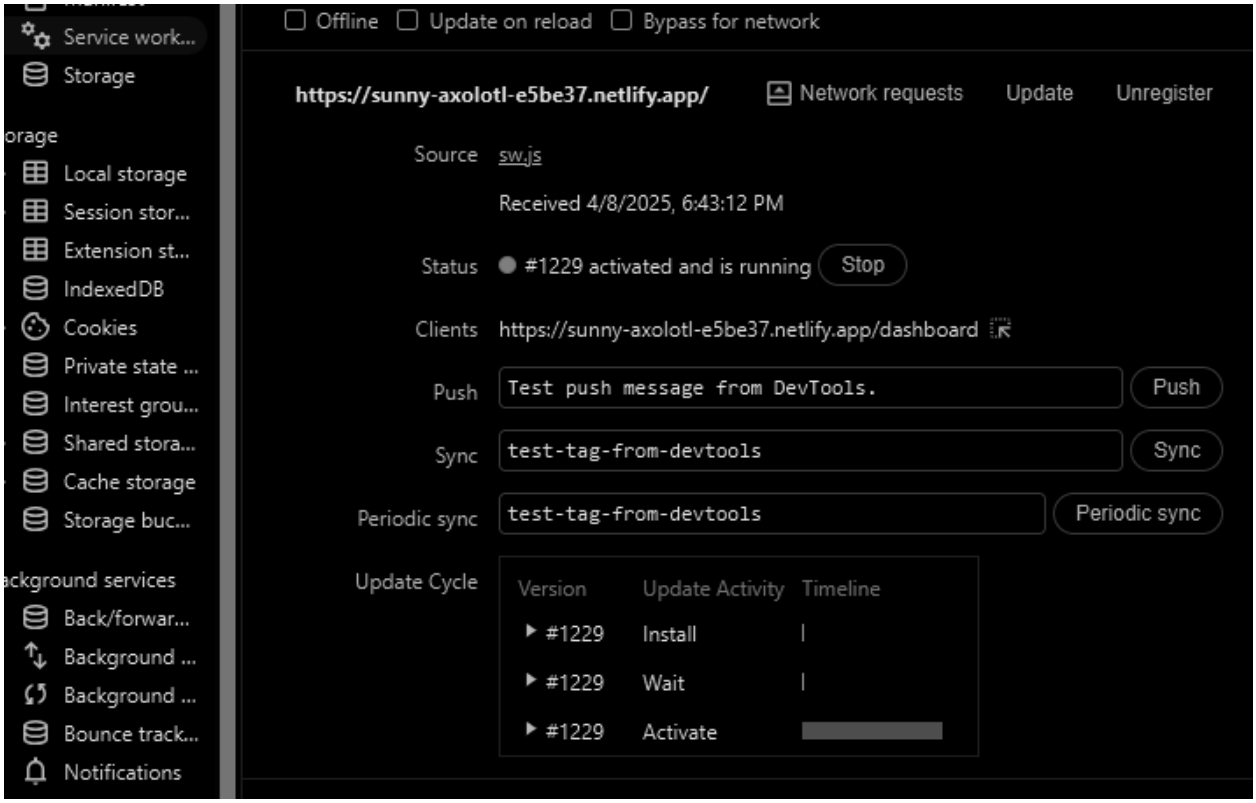
NAME - KARAN MISHRA
DIV - D15B
ROLL NO - 33
MPL & PWA EXP 10

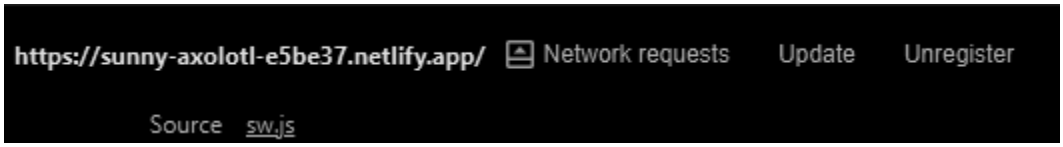



https://sunny-axolotl-e5be37.netlify.app/dashboard
Dashboard
Filter by Category:
All
Add New Habit
Title
Description
Category
Add
leetcode (General)
1 problem daily
Streak Starter
Halfway Hero
Status:
Streak: 4 | Monthly Progress: 4 days
Completion: 13%
13%

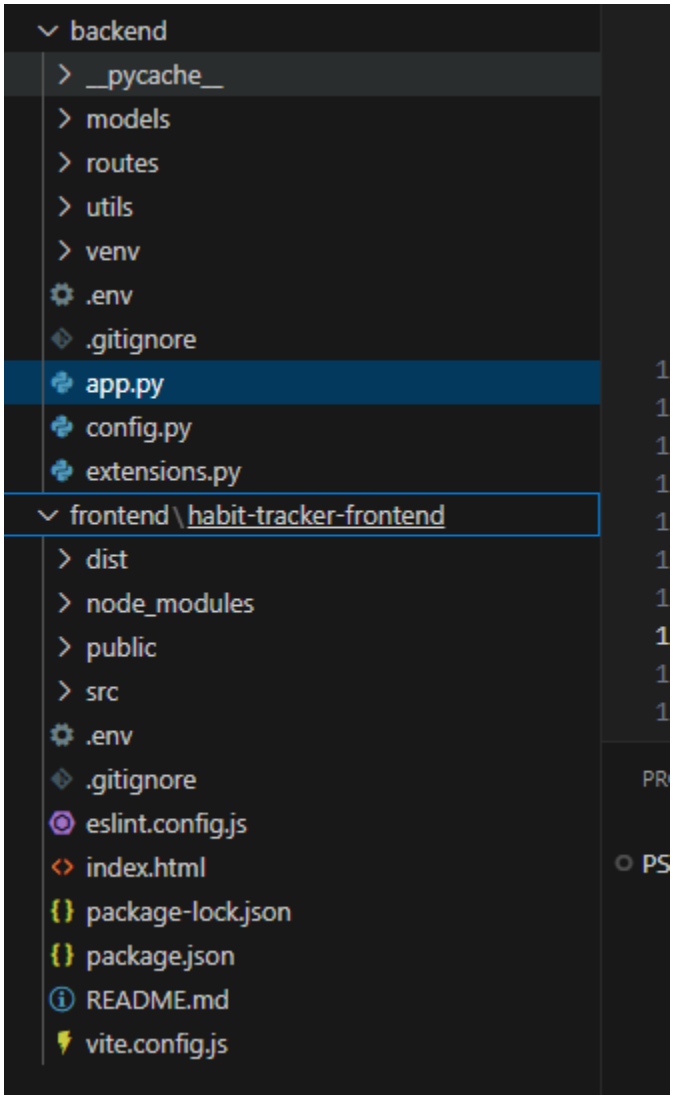

Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Storage buckets
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking m...
Identity
Name
Habit Tracker
Short name
HabitApp
Description
Computed App ID
https://sunny-axolotl-e5be37.netlify.app/
Learn more
Note: id is not specified in the manifest, start_url is used instead.
specify an App ID that matches the current identity, set the id field t
Presentation
Start URL
/
Theme color
#4caf50
Background color
#ffffff
Orientation
Display
standalone

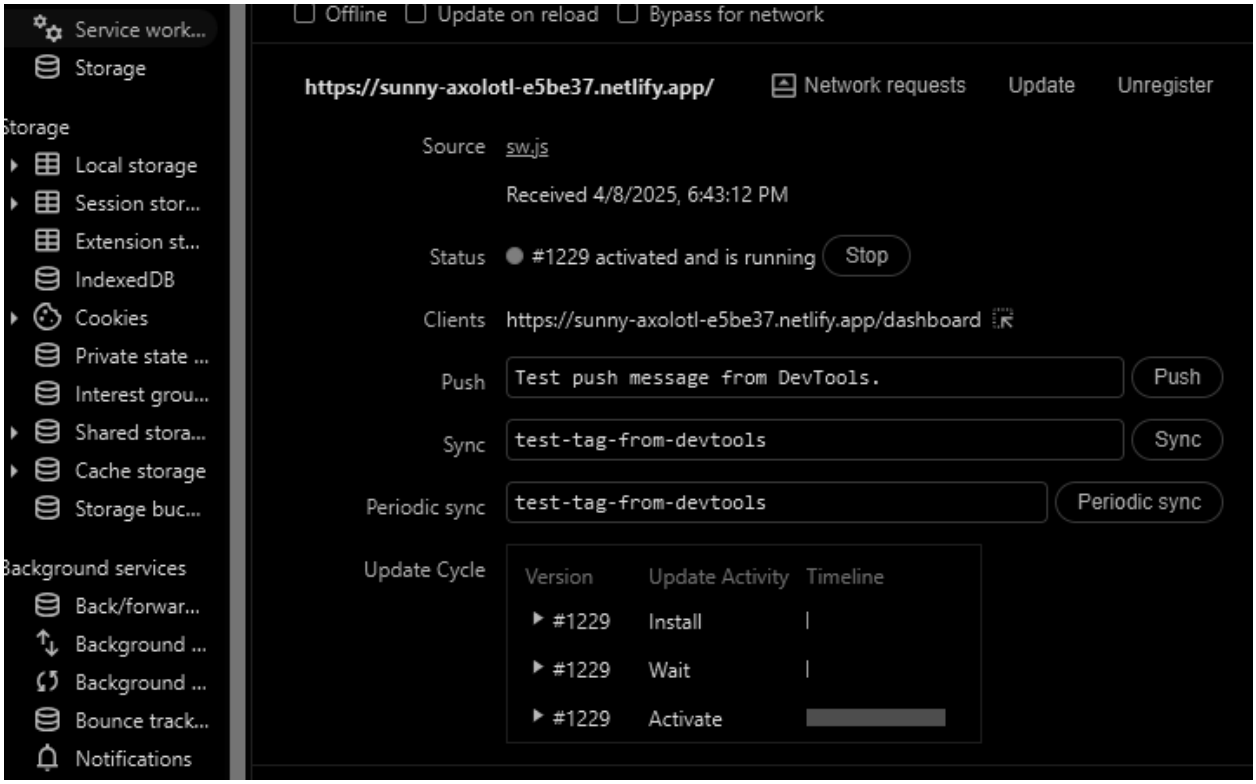
NAME - KARAN MISHRA
DIV - D15B
ROLL NO - 33
MPL & PWA EXP 9

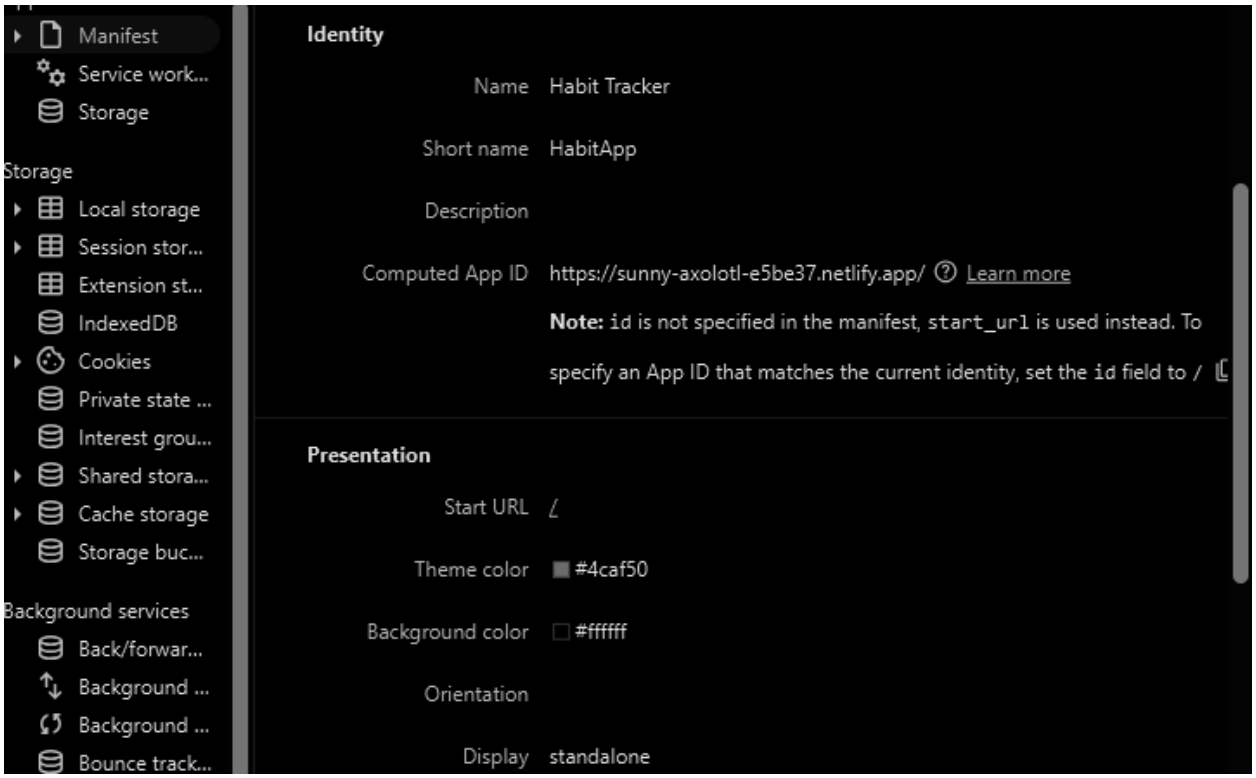




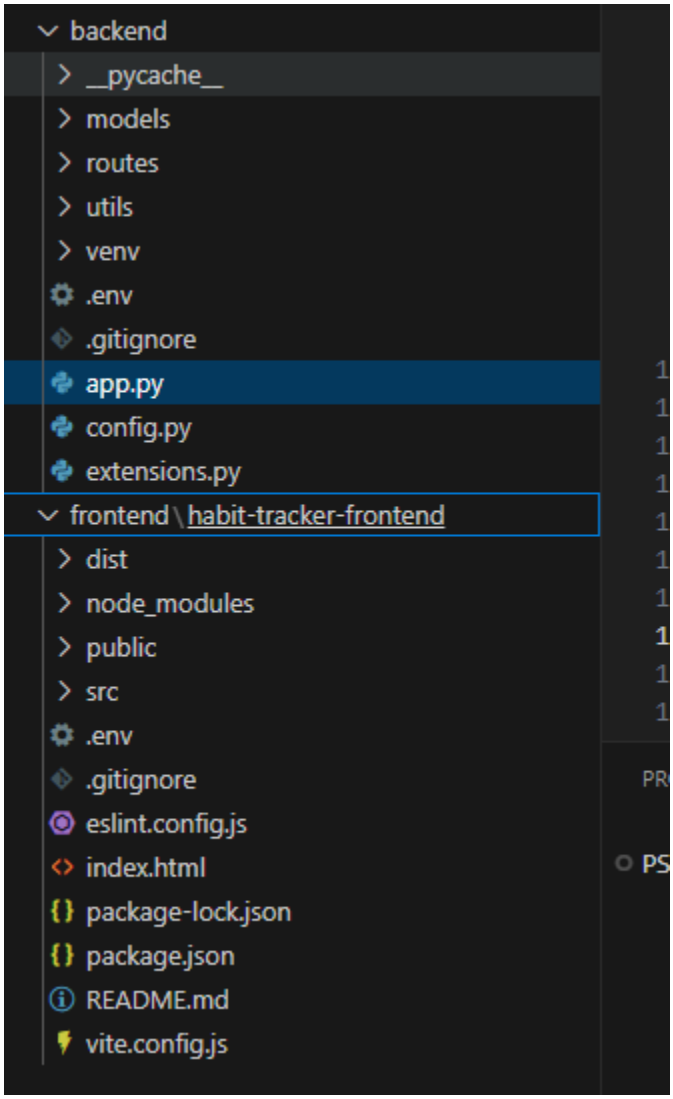
NAME - KARAN MISHRA
DIV - D15B
ROLL NO - 33
MPL & PWA EXP 8




Service work...
Storage
Storage
Local storage
Session stor...
Extension st...
IndexedDB
Cookies
Private state ...
Interest grou...
Shared stora...
Cache storage
Storage buc...
Background services
Back/forwar...
Background ...
Background ...
Bounce track...
Notifications
Offline
Update on reload
Bypass for network
https://sunny-axolotl-e5be37.netlify.app/
Network requests
Update
Unregister
Source
sw.js
Received 4/8/2025, 6:43:12 PM
Status
#1229 activated and is running
Stop
Clients
https://sunny-axolotl-e5be37.netlify.app/dashboard
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
#1229
Install
#1229
Wait
#1229
Activate


Manifest
Service work...
Storage
Storage
Local storage
Session stor...
Extension st...
IndexedDB
Cookies
Private state ...
Interest grou...
Shared stora...
Cache storage
Storage buc...
Background services
Back/forwar...
Background ...
Background ...
Bounce track...
Identity
Name
Habit Tracker
Short name
HabitApp
Description
Computed App ID
https://sunny-axolotl-e5be37.netlify.app/
Learn more
Note: id is not specified in the manifest, start_url is used instead. To specify an App ID that matches the current identity, set the id field to /
Presentation
Start URL
/
Theme color
#4caf50
Background color
#ffffff
Orientation
Display
standalone

NAME - KARAN MISHRA
DIV - D15B
ROLL NO - 33
MPL & PWA EXP 7

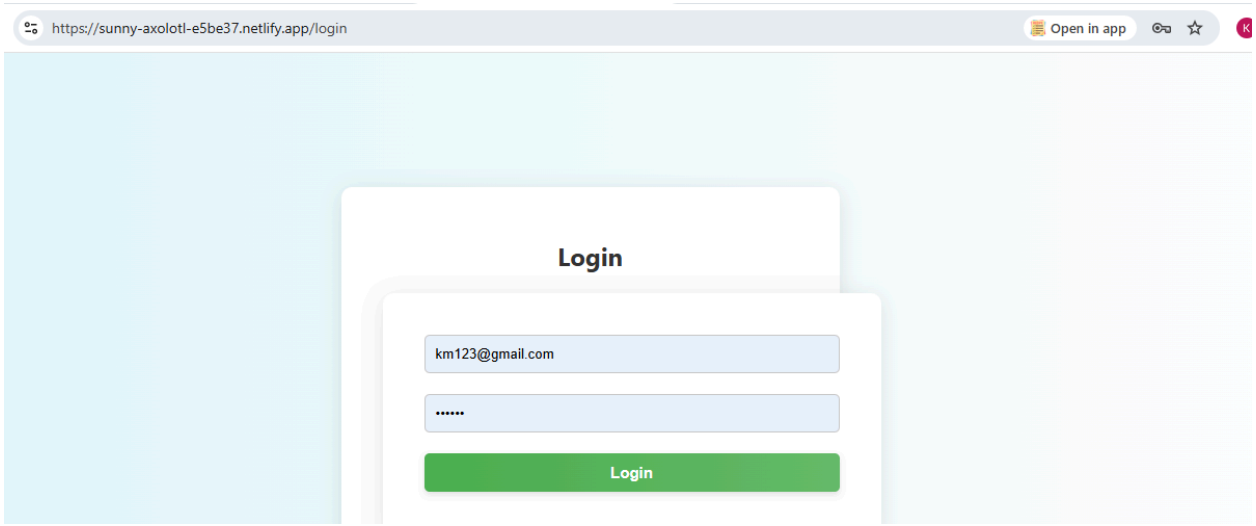


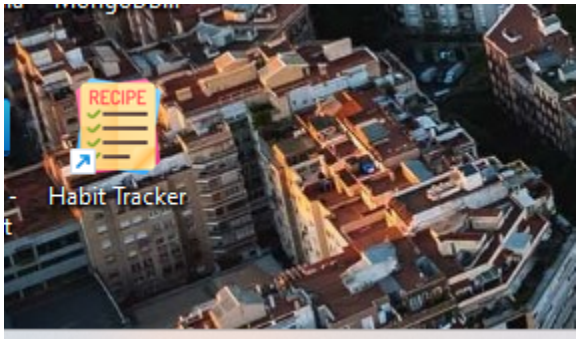
vite.config.js:-

```
import { defineConfig } from 'vite'

import react from '@vitejs/plugin-react'

import { VitePWA } from 'vite-plugin-pwa'

export default defineConfig({

plugins: [
```

```
react(),
VitePWA({
registerType: 'autoUpdate',
includeAssets: ['favicon.svg', 'robots.txt', 'apple-touch-icon.png'],
manifest: {
name: 'Habit Tracker',
short_name: 'HabitApp',
start_url: '/',
display: 'standalone',
background_color: '#ffffff',
theme_color: '#4caf50',
icons: [
{
src: 'pwa-192x192.png',
sizes: '192x192',
type: 'image/png'
},
{
src: 'pwa-512x512.png',
sizes: '512x512',
type: 'image/png'
}
]
}
```

```
})
]
})
```


https://sunny-axolotl-e5be37.netlify.app/login
Open in app
Login
km123@gmail.com
.....
Login


RECIPE
Habit Tracker

NAME - KARAN MISHRA
DIV - D15B
ROLL NO - 33
MPL EXP 5

auth_screen.dart

```
import 'dart:math';
import 'package:flutter/material.dart';
import '../services/auth_service.dart';
import '../screens/menu_screen.dart';

class AuthScreen extends StatefulWidget {
  const AuthScreen({super.key});

  @override
  _AuthScreenState createState() => _AuthScreenState();
}

class _AuthScreenState extends State<AuthScreen> {
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();
  final TextEditingController _captchaController = TextEditingController();
  final AuthService _authService = AuthService();

  final List<String> captchaWords = ["apple", "secure", "verify", "random", "trust"];
  late String captchaWord;

  @override
  void initState() {
    super.initState();
    _generateNewCaptcha();
  }

  void _generateNewCaptcha() {
    setState(() {
      captchaWord = captchaWords[Random().nextInt(captchaWords.length)];
    });
  }

  void _authenticate(bool isLogin) async {
    if (_captchaController.text.trim().toLowerCase() != captchaWord.toLowerCase()) {
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text("Incorrect CAPTCHA. Try again!")),
      );
```

```
    _generateNewCaptcha();
    return;
  }

  String? userId = isLogin
      ? await _authService.login(_emailController.text.trim(), _passwordController.text.trim())
      : await _authService.register(_emailController.text.trim(), _passwordController.text.trim());

  if (userId != null) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text(isLogin ? "Login Successful!" : "Registration Successful!")),
    );

    Future.delayed(const Duration(milliseconds: 500), () {
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(
          builder: (context) => MenuScreen(userEmail: _emailController.text.trim()),
        ),
      );
    });
  } else {
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text("Authentication Failed!")),
    );
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Stack(
      fit: StackFit.expand,
      children: [
        Image.asset("assets/images/background.png", fit: BoxFit.cover),
        Center(
          child: Padding(
            padding: const EdgeInsets.all(24.0),
            child: Container(
              padding: const EdgeInsets.all(20),
              decoration: BoxDecoration(
                color: Colors.white.withOpacity(0.9),
                borderRadius: BorderRadius.circular(15),
              ),
```

```
            child: Column(
              mainAxisSize: MainAxisSize.min,
              children: [
                const Text("Welcome!", style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold)),
                const SizedBox(height: 10),
                _buildTextField(_emailController, "Email"),
                _buildTextField(_passwordController, "Password", isPassword: true),
                const SizedBox(height: 10),
                Text('Enter the word: "$captchaWord"',
                    style: const TextStyle(fontSize: 16, fontWeight: FontWeight.bold)),
                _buildTextField(_captchaController, "Enter CAPTCHA"),
                const SizedBox(height: 10),
                _buildButton("Login", () => _authenticate(true), Colors.blue),
                _buildButton("Register", () => _authenticate(false), Colors.green),
              ],
            ),
          ),
        ),
      ),
    ],
  ),
 );
}

Widget _buildTextField(TextEditingController controller, String hint, {bool isPassword = false}) {
 return Padding(
  padding: const EdgeInsets.symmetric(vertical: 8),
  child: TextField(
   controller: controller,
   obscureText: isPassword,
   decoration: InputDecoration(
    labelText: hint,
    border: OutlineInputBorder(borderRadius: BorderRadius.circular(12)),
    fillColor: Colors.white,
    filled: true,
   ),
  ),
 );
}

Widget _buildButton(String text, VoidCallback onPressed, Color color) {
 return Padding(
  padding: const EdgeInsets.symmetric(vertical: 8),
  child: ElevatedButton(
```

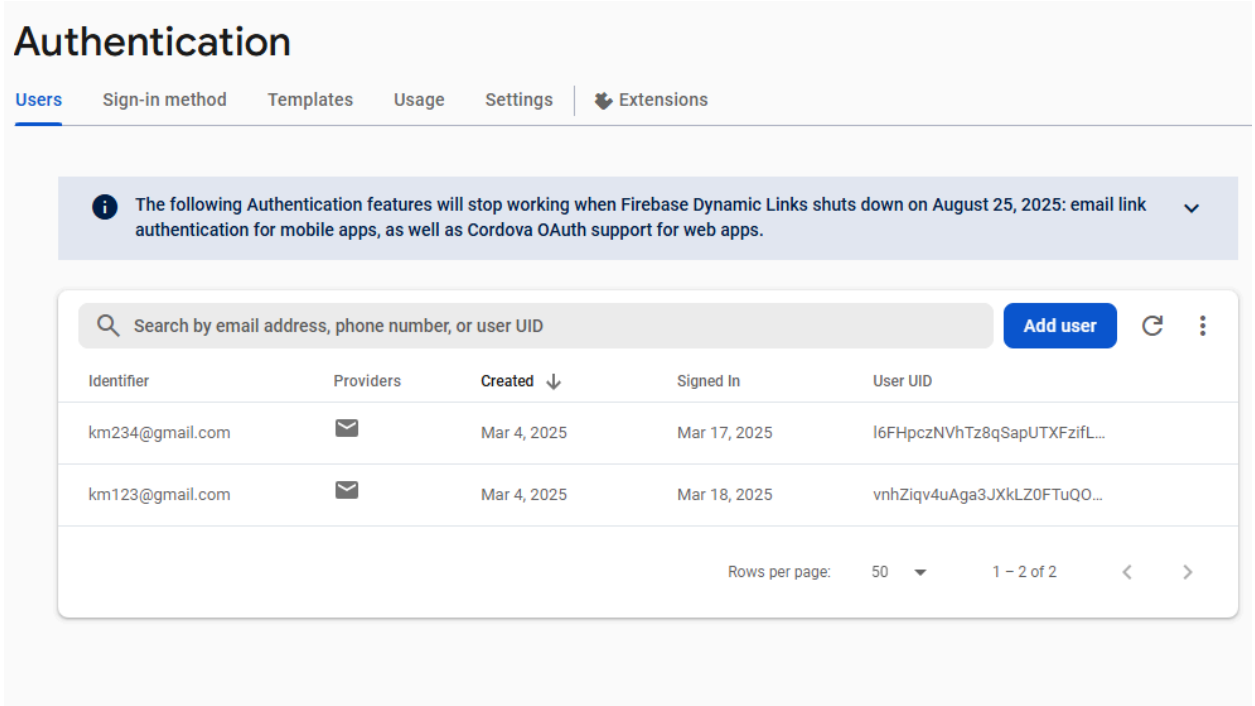
```
      style: ElevatedButton.styleFrom(backgroundColor: color, shape:
RoundedRectangleBorder(borderRadius: BorderRadius.circular(10))),
      onPressed: onPressed,
      child: Text(text, style: const TextStyle(fontSize: 18, color: Colors.white)),
    ),
  );
 }
}
```

auth.service.dart

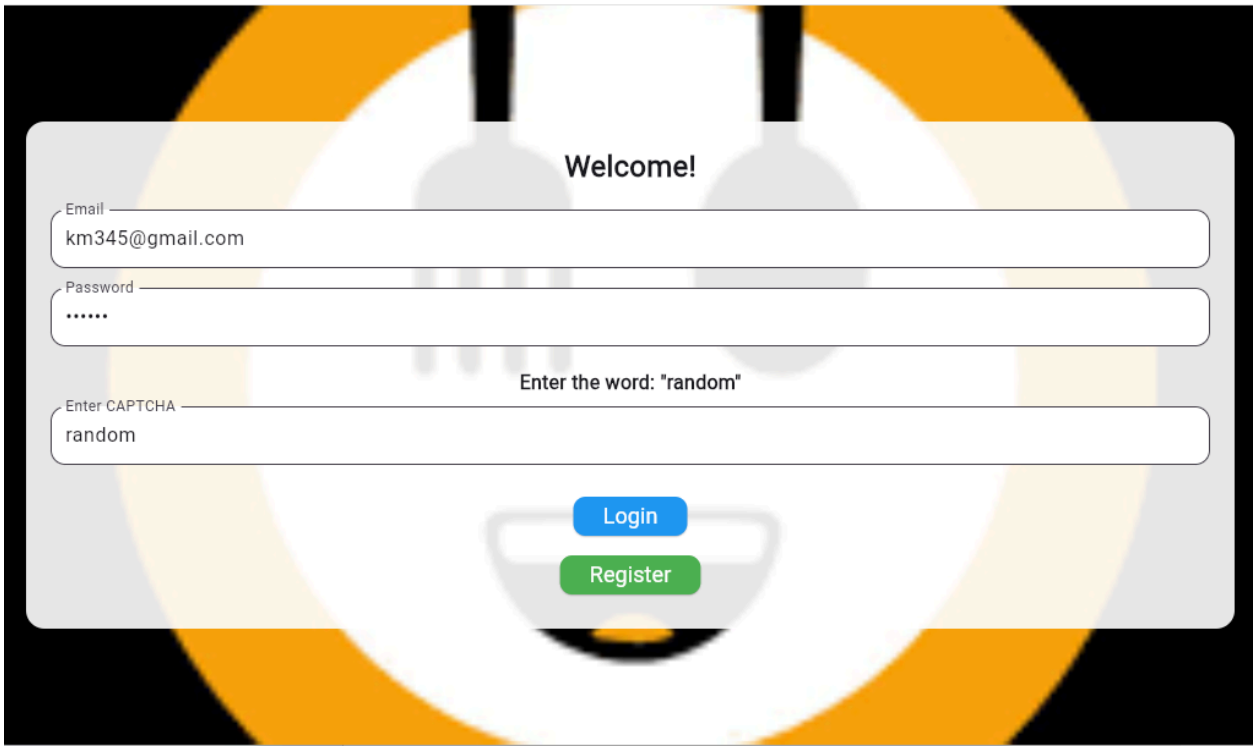
```
import 'package:firebase_auth/firebase_auth.dart';

class AuthService {
 final FirebaseAuth _auth = FirebaseAuth.instance;

 Future<String?> login(String email, String password) async {
  try {
   UserCredential userCredential = await _auth.signInWithEmailAndPassword(
    email: email,
    password: password,
   );
   return userCredential.user?.uid;
  } catch (e) {
   return null;
  }
 }

 Future<String?> register(String email, String password) async {
  try {
   UserCredential userCredential = await _auth.createUserWithEmailAndPassword(
    email: email,
    password: password,
   );
   return userCredential.user?.uid;
  } catch (e) {
   return null;
  }
 }

 Future<void> logout() async {
  await _auth.signOut();
 }
```

```
  User? getCurrentUser() {
    return _auth.currentUser;
  }
}
```

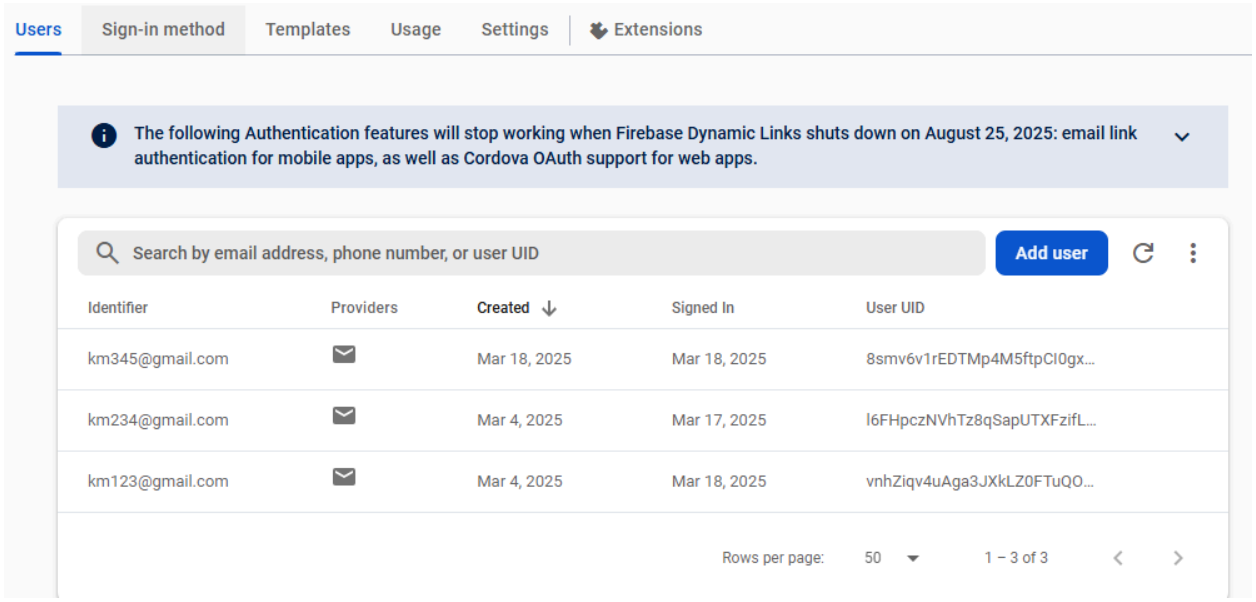
Output:-
Before registering:-



Register and Login page



After Registration:-



NAME - KARAN MISHRA
DIV - D15B
ROLL NO - 33
MPL EXP 5

Code:-

```
import 'package:flutter/material.dart';
import 'cart_screen.dart';
import '../services/order_service.dart';

class MenuScreen extends StatefulWidget {
  final String userEmail;
  const MenuScreen({required this.userEmail, super.key});

  @override
  _MenuScreenState createState() => _MenuScreenState();
}

class _MenuScreenState extends State<MenuScreen> {
  final OrderService _orderService = OrderService();

  final List<Map<String, dynamic>> menuItems = [
    {"name": "Pizza", "price": 199.0, "image": "assets/images/pizza.jfif"},
    {"name": "Burger", "price": 149.0, "image": "assets/images/burger.jfif"},
    {"name": "Pasta", "price": 180.0, "image": "assets/images/pasta.jfif"},
    {"name": "Samosa", "price": 30.0, "image": "assets/images/samosa.jfif"},
    {"name": "Vada Pav", "price": 25.0, "image": "assets/images/vada pav.jfif"},
    {"name": "Dosa", "price": 120.0, "image": "assets/images/dosa.jfif"},
    {"name": "Chole Bhature", "price": 180.0, "image": "assets/images/chole bhature.jfif"},
    {"name": "Masala Dosa", "price": 140.0, "image": "assets/images/masala_dosa.jfif"},
  ];

  Future<void> addToCart(String name, double price) async {
    await _orderService.addToCart(widget.userEmail, name, price);
    if (mounted) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text("$name added to cart")),
      );
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
```

```
appBar: AppBar(
  title: const Text("Canteen Menu"),
  actions: [
    IconButton(
      icon: const Icon(Icons.shopping_cart),
      onPressed: () => Navigator.push(
        context,
        MaterialPageRoute(builder: (context) => CartScreen(userEmail: widget.userEmail)),
      ),
    ),
  ],
),
body: LayoutBuilder(
  builder: (context, constraints) {
    double width = constraints.maxWidth;
    double aspectRatio = (width < 600) ? 0.9 : 1.2;

    return GridView.builder(
      padding: const EdgeInsets.all(16),
      gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
        crossAxisCount: 2,
        crossAxisSpacing: 12,
        mainAxisSpacing: 12,
        childAspectRatio: aspectRatio,
      ),
      itemCount: menuItems.length,
      itemBuilder: (context, index) {
        var item = menuItems[index];
        return Card(
          elevation: 5,
          shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(15)),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              ClipRRect(
                borderRadius: BorderRadius.circular(10),
                child: Image.asset(
                  item["image"],
                  height: 120, // Increased Image Size
                  width: 120,
                  fit: BoxFit.cover,
                ),
              ),
              const SizedBox(height: 25), // Increased Gap
```

```
                Text(
                  item["name"],
                  style: const TextStyle(fontSize: 18, fontWeight: FontWeight.bold), // Larger Name
Text
                ),
                const SizedBox(height: 12),
                Text(
                  "₹${item["price"].toStringAsFixed(2)}",
                  style: TextStyle(fontSize: 15, color: Colors.grey[700]),
                ),
                const SizedBox(height: 18), // Increased Gap
                ElevatedButton(
                  onPressed: () => addToCart(item["name"], item["price"]),
                  style: ElevatedButton.styleFrom(
                    backgroundColor: Colors.purple[300],
                    shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(20)),
                  ),
                  child: const Text("Add to Cart"),
                ),
              ],
            ),
          );
        },
      );
    },
  ),
 );
 }
}
```

OUTPUT:-

Canteen Menu

Pizza

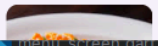
₹199.00

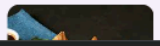
Add to Cart

Burger

₹149.00

Add to Cart





egister > lib > screens > menu_screen.dart

Cart

Burger
Price: 5.99 x 1

Dosa
Price: 120 x 1

Pizza
Price: 8.99 x 2

NAME - KARAN MISHRA

DIV - D15B

ROLL NO - 33

MPL EXP 4

```
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_core/firebase_core.dart';
void main() async {
WidgetsFlutterBinding.ensureInitialized();
await Firebase.initializeApp();
runApp(MyApp());
}
class MyApp extends StatelessWidget {
@override
Widget build(BuildContext context) {
return MaterialApp(
home: AuthScreen(),
);
}
}
class AuthScreen extends StatefulWidget {
@override
_AuthScreenState createState() => _AuthScreenState();
}
```

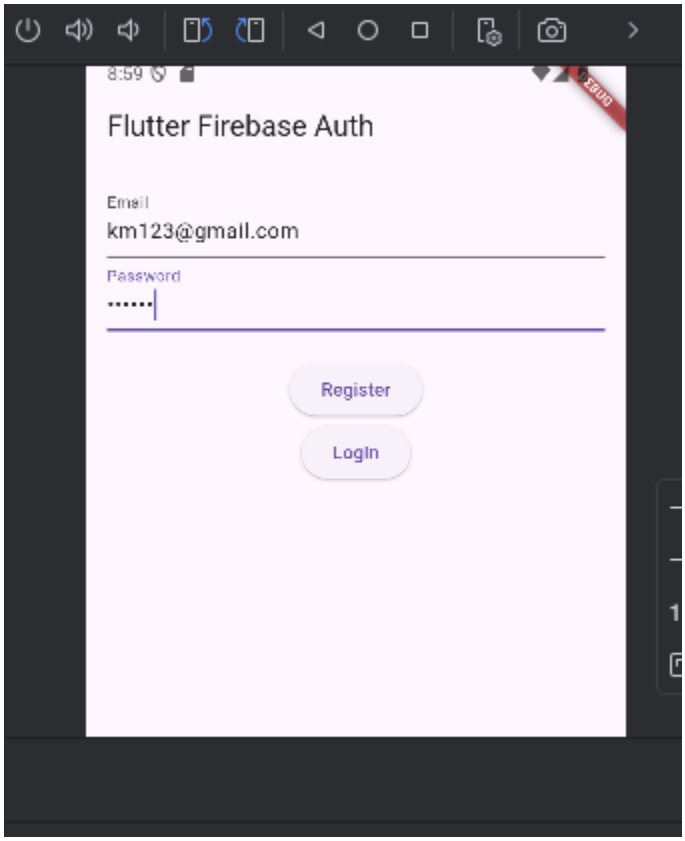
```
class _AuthScreenState extends State<AuthScreen> {
final FirebaseAuth _auth = FirebaseAuth.instance;
final TextEditingController _emailController = TextEditingController();
final TextEditingController _passwordController = TextEditingController();
Future<void> register() async {
try {
await _auth.createUserWithEmailAndPassword(
email: _emailController.text,
password: _passwordController.text,
);
ScaffoldMessenger.of(context).showSnackBar(
SnackBar(content: Text("Registration Successful")),
);
} catch (e) {
ScaffoldMessenger.of(context).showSnackBar(
SnackBar(content: Text(e.toString())),
);
}
}
Future<void> login() async {
try {
await _auth.signInWithEmailAndPassword(
email: _emailController.text,
password: _passwordController.text,
```

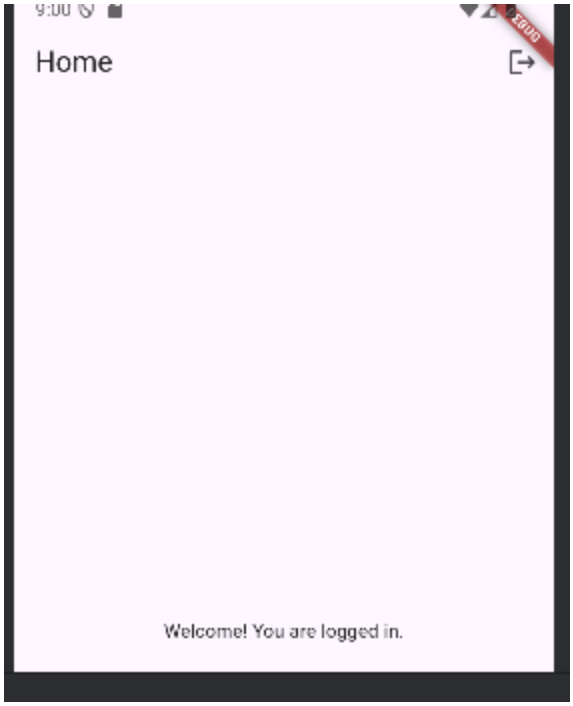
```
);
Navigator.pushReplacement(
context,
MaterialPageRoute(builder: (context) => HomeScreen()),
);
} catch (e) {
ScaffoldMessenger.of(context).showSnackBar(
SnackBar(content: Text(e.toString())),
);
}
}
@override
Widget build(BuildContext context) {
return Scaffold(
appBar: AppBar(title: Text("Flutter Firebase Auth")),
body: Padding(
padding: EdgeInsets.all(16.0),
child: Column(
children: [
TextField(
controller: _emailController,
decoration: InputDecoration(labelText: "Email"),
),
TextField(
```

```
controller: _passwordController,
decoration: InputDecoration(labelText: "Password"),
obscureText: true,
),
SizedBox(height: 20),
ElevatedButton(onPressed: register, child: Text("Register")),
ElevatedButton(onPressed: login, child: Text("Login")),
],
),
),
);
}
}
class HomeScreen extends StatelessWidget {
final FirebaseAuth _auth = FirebaseAuth.instance;
@override
Widget build(BuildContext context) {
return Scaffold(
appBar: AppBar(
title: Text("Home"),
actions: [
IconButton(
icon: Icon(Icons.logout),
onPressed: () async {
```

```
await _auth.signOut();
Navigator.pushReplacement(
context,
MaterialPageRoute(builder: (context) => AuthScreen()),
);
},
)
],
),
body: Center(
child: Text("Welcome! You are logged in."),
),
);
}
}
```

OUTPUT:-


8:59
Flutter Firebase Auth
Email
km123@gmail.com
Password
Register
LogIn



RegisterMpl

Authentication

Users | Sign-in method | Templates | Usage | Settings | Extensions

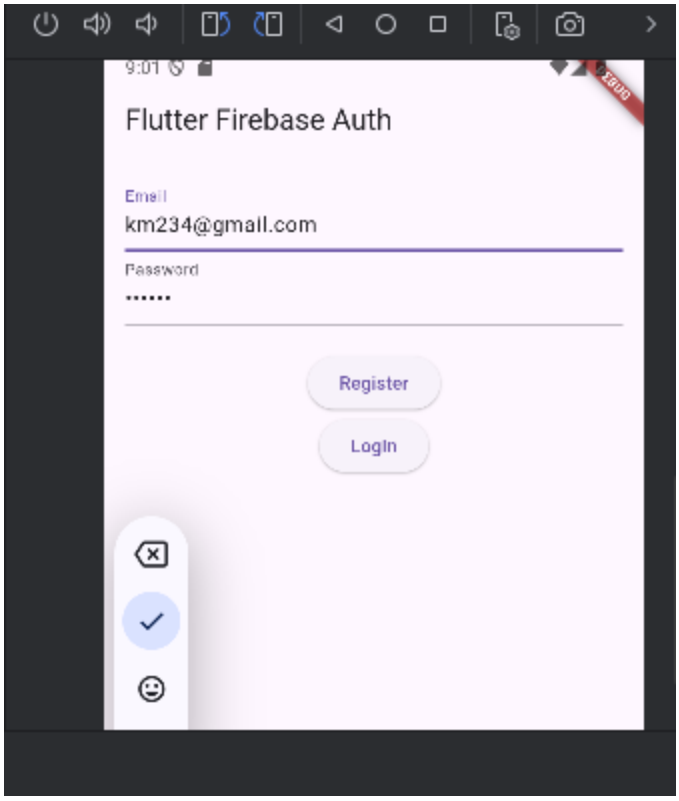
The following Authentication features will stop working when Firebase Dynamic Links shuts down on August 25, 2025: email link authentication for mobile apps, as well as Cordova OAuth support for web apps.

Search by email address, phone number, or user UID

Add user

Identifier	Providers	Created	Signed In	User UID
km123@gmail.com		Mar 4, 2025	Mar 4, 2025	vnhZiqv4uAga3JXkLZ0FTuQO...

Rows per page: 50 1 – 1 of 1



RegisterMpl

Authentication

Users | Sign-in method | Templates | Usage | Settings | Extensions

The following Authentication features will stop working when Firebase Dynamic Links shuts down on August 25, 2025: email link authentication for mobile apps, as well as Cordova OAuth support for web apps.

Search by email address, phone number, or user UID — Add user

Identifier	Providers	Created	Signed In	User UID
km234@gmail.com		Mar 4, 2025	Mar 4, 2025	l6FHpczNVhTz8qSapUTXFzifL...
km123@gmail.com		Mar 4, 2025	Mar 4, 2025	vnhZiqv4uAga3JXkLZ0FTuQO...

Rows per page: 50 — 1 – 2 of 2

NAME - KARAN MISHRA
DIV - D15B
ROLL NO - 33
MPL EXP 3
Code:-

```
// main.dart

import 'package:flutter/material.dart';

void main() {

runApp(const CanteenApp());

}

class CanteenApp extends StatelessWidget {

const CanteenApp({super.key});

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Canteen App',

theme: ThemeData(

primarySwatch: Colors.orange,

),

home: const LoginPage(),

);

}

}

class LoginPage extends StatelessWidget {

const LoginPage({super.key});

@override

Widget build(BuildContext context) {
```

```
final TextEditingController emailController = TextEditingController();
final TextEditingController passwordController = TextEditingController();
return Scaffold(
appBar: AppBar(title: const Text('Login')),
body: Padding(
padding: const EdgeInsets.all(16.0),
child: Column(
mainAxisAlignment: MainAxisAlignment.center,
children: [
TextField(
controller: emailController,
decoration: const InputDecoration(
labelText: 'Email',
border: OutlineInputBorder(),
),
keyboardType: TextInputType.emailAddress,
),
const SizedBox(height: 16),
TextField(
controller: passwordController,
decoration: const InputDecoration(
labelText: 'Password',
border: OutlineInputBorder(),
),
```

```
obscureText: true,
),
const SizedBox(height: 24),
ElevatedButton(
style: ElevatedButton.styleFrom(
backgroundColor: Colors.orange,
padding: const EdgeInsets.symmetric(horizontal: 32, vertical: 16),
),
onPressed: () {
String email = emailController.text;
String password = passwordController.text;
if (email.isNotEmpty && password.isNotEmpty) {
Navigator.push(
context,
MaterialPageRoute(builder: (context) => const MenuPage()),
);
} else {
ScaffoldMessenger.of(context).showSnackBar(
const SnackBar(content: Text('Please enter email and password')),
);
}
},
child: const Text('Login', style: TextStyle(fontSize: 18)),
),
```

```
const SizedBox(height: 16),
ElevatedButton(
style: ElevatedButton.styleFrom(
backgroundColor: Colors.grey,
padding: const EdgeInsets.symmetric(horizontal: 32, vertical: 16),
),
onPressed: () {
Navigator.push(
context,
MaterialPageRoute(builder: (context) => const MenuPage()),
);
},
child: const Text('Login as Guest', style: TextStyle(fontSize: 18)),
),
],
),
),
);
}
}
class MenuPage extends StatefulWidget {
const MenuPage({super.key});
@override
State<MenuPage> createState() => _MenuPageState();
```

```
}
class _MenuPageState extends State<MenuPage> {
final List<Map<String, String>> cartItems = [];
void addToCart(String title, String price, String imagePath) {
setState(() {
cartItems.add({'title': title, 'price': price, 'imagePath': imagePath});
});
ScaffoldMessenger.of(context).showSnackBar(
SnackBar(
content: Text('$title added to cart'),
duration: const Duration(seconds: 1),
),
);
}
@override
Widget build(BuildContext context) {
return Scaffold(
appBar: AppBar(
title: const Text('Canteen Menu'),
actions: [
IconButton(
icon: const Icon(Icons.shopping_cart),
onPressed: () {
Navigator.push(
```

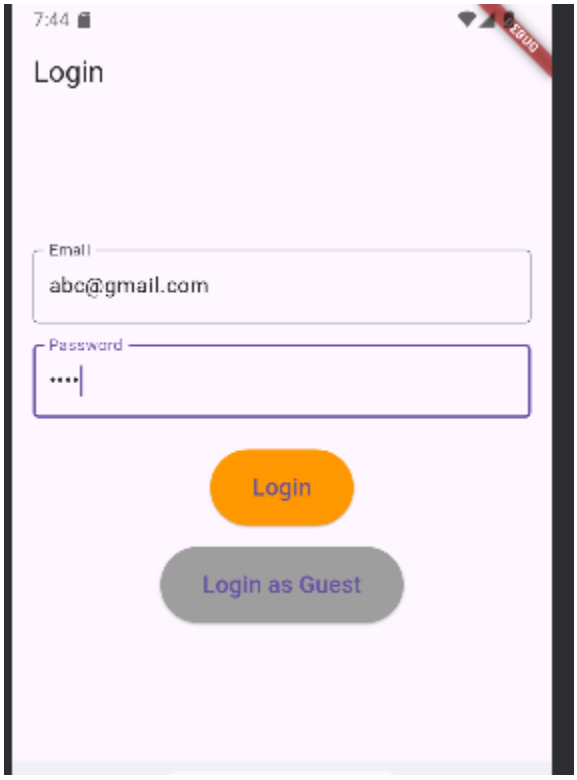
```
context,
MaterialPageRoute(
builder: (context) => CartPage(cartItems: cartItems),
),
);
},
),
],
),
body: ListView(
padding: const EdgeInsets.all(16),
children: [
_buildMenuItem('Burger', '₹50', 'lib/assets/burger.jfif'),
_buildMenuItem('Pizza', '₹120', 'lib/assets/pizza.jfif'),
_buildMenuItem('Cold Drink', '₹30', 'lib/assets/coldrink.jfif'),
_buildMenuItem('Samosa', '₹15', 'lib/assets/samosa.jfif'),
],
),
);
}
Widget _buildMenuItem(String title, String price, String imagePath) {
return Card(
shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(16)),
elevation: 5,
```

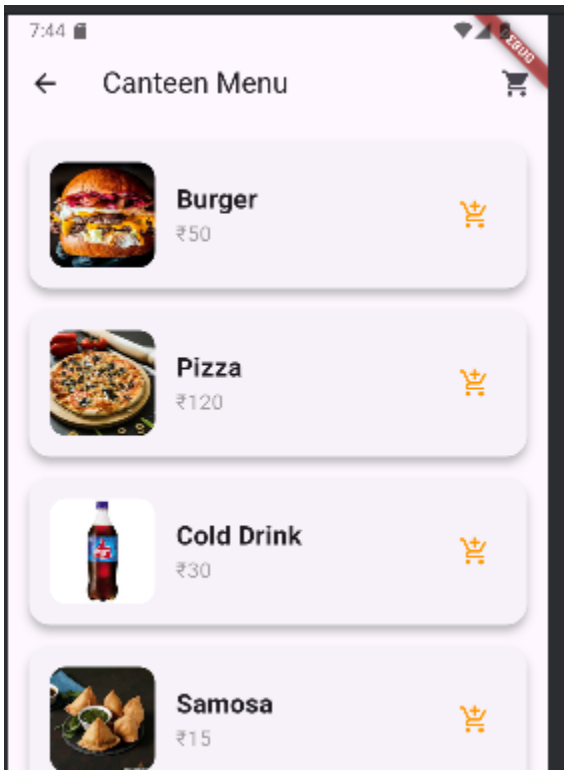
```
margin: const EdgeInsets.only(bottom: 16),
child: Padding(
padding: const EdgeInsets.all(16),
child: Row(
children: [
ClipRRect(
borderRadius: BorderRadius.circular(12),
child: Image.asset(imagePath, width: 80, height: 80, fit: BoxFit.cover),
),
const SizedBox(width: 16),
Expanded(
child: Column(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
Text(title, style: const TextStyle(fontSize: 20, fontWeight: FontWeight.bold)),
Text(price, style: const TextStyle(fontSize: 16, color: Colors.grey)),
],
),
),
IconButton(
icon: const Icon(Icons.add_shopping_cart, color: Colors.orange),
onPressed: () => addToCart(title, price, imagePath),
),
],
```

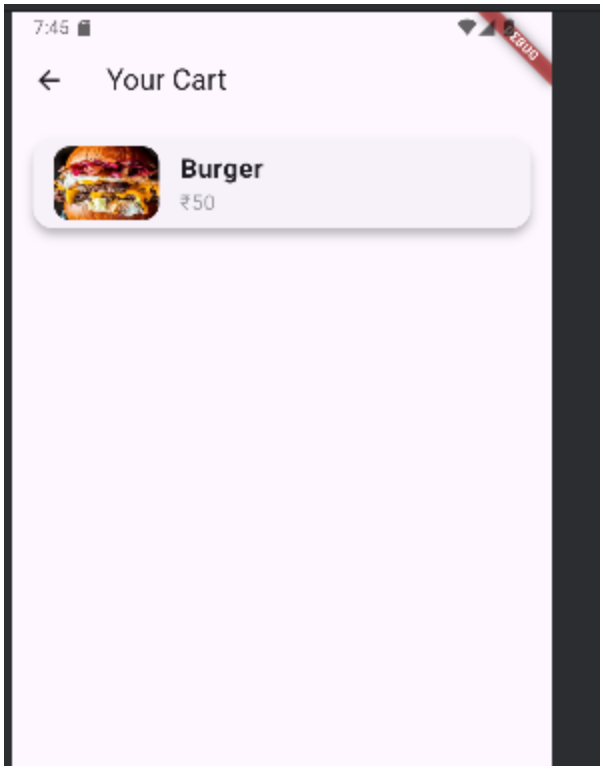
```
),
),
);
}
}
class CartPage extends StatelessWidget {
final List<Map<String, String>> cartItems;
const CartPage({super.key, required this.cartItems});
@override
Widget build(BuildContext context) {
return Scaffold(
appBar: AppBar(title: const Text('Your Cart')),
body: cartItems.isEmpty
? const Center(child: Text('Your cart is empty'))
: ListView.builder(
padding: const EdgeInsets.all(16),
itemCount: cartItems.length,
itemBuilder: (context, index) {
final item = cartItems[index];
return Card(
shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(16)),
elevation: 5,
margin: const EdgeInsets.only(bottom: 16),
child: ListTile(
```

```
leading: ClipRRect(
borderRadius: BorderRadius.circular(12),
child: Image.asset(item['imagePath']!, width: 80, height: 80, fit: BoxFit.cover),
),
title: Text(item['title']!, style: const TextStyle(fontSize: 20, fontWeight: FontWeight.bold)),
subtitle: Text(item['price']!, style: const TextStyle(fontSize: 16, color: Colors.grey)),
),
);
},
),
);
}
}
```

OUTPUT:-


7:44
Login
Email
abc@gmail.com
Password
Login
Login as Guest

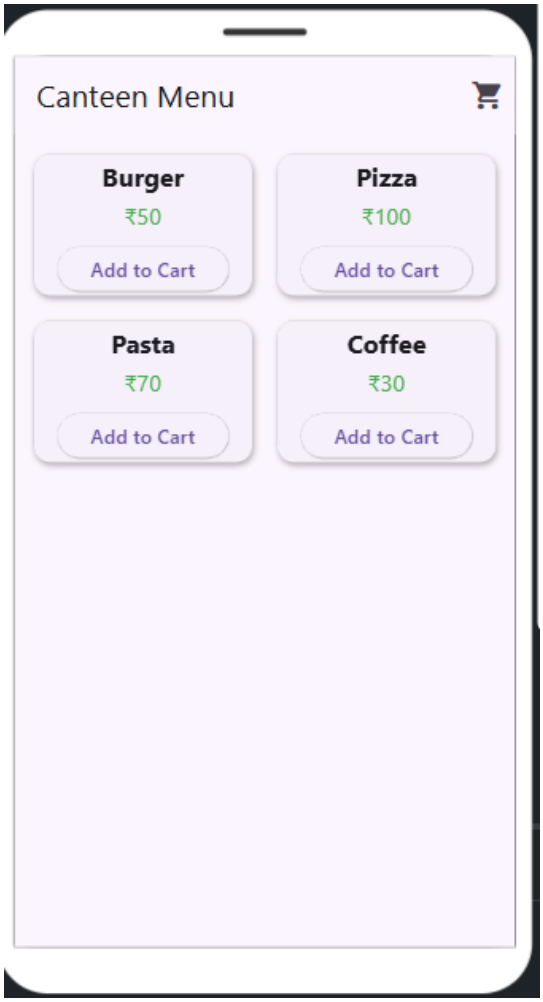

7:44
Canteen Menu
Burger
₹50
Pizza
₹120
Cold Drink
₹30
Samosa
₹15


7:45
DEBUG
Your Cart
Burger
₹50

NAME - KARAN MISHRA
DIV - D15B
ROLL NO - 33
MPL EXP 2

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

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      home: HomePage(),
    );
  }
}

class HomePage extends StatelessWidget {
  final List<Map<String, String>> menuItems = [
    {'name': 'Burger', 'price': '₹50'},
    {'name': 'Pizza', 'price': '₹100'},
    {'name': 'Pasta', 'price': '₹70'},
    {'name': 'Coffee', 'price': '₹30'},
  ];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Canteen Menu'),
        actions: [IconButton(onPressed: () {}, icon: Icon(Icons.shopping_cart))],
      ),
      body: GridView.builder(
        padding: EdgeInsets.all(10),
        gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
          crossAxisCount: 2,
          childAspectRatio: 3 / 2,
          crossAxisSpacing: 10,
          mainAxisSpacing: 10,
        ),
```

```
      itemCount: menuItems.length,
      itemBuilder: (context, index) {
        return Card(
          elevation: 4,
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Text(menuItems[index]['name']!, style: TextStyle(fontSize: 18, fontWeight:
FontWeight.bold)),
              SizedBox(height: 5),
              Text(menuItems[index]['price']!, style: TextStyle(fontSize: 16, color: Colors.green)),
              SizedBox(height: 10),
              ElevatedButton(onPressed: () {}, child: Text('Add to Cart'))
            ],
          ),
        );
      },
    ),
  );
 }
}
```


Canteen Menu
Burger
₹50
Add to Cart
Pizza
₹100
Add to Cart
Pasta
₹70
Add to Cart
Coffee
₹30
Add to Cart

Name - Tarun Mishra Div - D15B
Roll No - 33 MPL Assignment 1

Q(1)
Ans(a) Key Features and Advantages of Flutter for Mobile App Development

- Single Codebase for Multiple Platforms: One code to run Web, iOS, Android, etc.
- Hot Reload: Instantly see changes in app without losing state, speeding up development and debugging
- Rich UI with Customizable Widgets: Provides a wide range of pre-design widgets and allows full customizatio to build complex, responsive UIs
- High Performance: Built on Dart, Flutter compiles to native ARM code, eliminat the need for a Javascript bridge.
- Access to Native Features: Flutter offer plugins and platform channels to acce native device features like GPS, camera, sensors, etc.
- Google Support: Backed by Google, Flutter receives regular updates, & stability improvements and stable long-term support.

Teacher's Sign.: _____

b.) Rendering Approach: Traditional frame
rely on native components, but Flut
draws everything itself using Ski
This eliminates dependency on OEM wid
and prevents UI inconsistencies.

- Close - Platform without sacrificing
Performance: Unlike frameworks li
React Native, which use a bridge to
communication with native compon
Flutter's direct compilation to nativ
code avoids performance bottlenec

Declarative UI: Flutter uses a reacti
declarative UI approach where UI
automatically updates when app state
changes - similar to modern front-e
frameworks.

Reduced Maintenance Effort: A single
codebase means fewer bugs and eas
updates, as developers don't need to
maintain seperate code for iOS and
Android.

- Faster Development Cycles: Hot reload
and a vast library of ready to
use components drastically cut
down development and testing time

Teacher's Sign.: _____

2.)
Ans.) In Flutter, everything is a widget - buttons, text, images, layouts, and even entire app itself. These widgets are organized in a tree structure called the widget tree.

- Widget Tree: Represents the visual structure and hierarchy of the app UI. Each widget is a node in the tree, defining part of the UI. When the app runs, Flutter builds this tree and renders it on the screen.

- Widget composition: Flutter encourages building UI by combining small, reusable widgets into more complex ones. Instead of creating large, monolithic components, you compose smaller, widgets, each handling a specific responsibility.

Eg -

```
class MyApp extends StatelessWidget {
Widget build(BuildContext context) {
  return MaterialApp(
    home: Scaffold(
  appBar: AppBar(title),
    body: Column(
  children: [
    Text(),
```

Teacher's Sign.: _____

```
ElevatedButton(
  child: Text("Click Me"),
  onPressed: () {
    print("Button Pressed");
  },
)
```

b) Commonly Used Widgets and Their Roles:

1. Structural Widgets:
   1.) Scaffold: Provides a basic page structure with an app body, floating action button, etc.
   2.) App Bar: A toolbar for titles, icons and actions.
   3.) Drawer: A side navigation panel.

2. Layout Widgets:
   1.) Container: Adds padding, margins, borders and backgrounds
   2.) Row and Column: Arrange widgets horizontally and vertically.
   3.) Stack: Overlaps widgets on top of each other.
   4.) List View: Creates scrollable lists.

3. Display Widgets:
   1.) Text: Displays static or dynamic text
   2.) Image: Displays images from assets, network or memory.

Teacher's Sign.: _____

3.) Interactive Widgets:
- Elevated Button: A button with elevation
- Textfield: Input field for text entry
- Checkbox, Switch, Slider: Interactive UI elements.

```
MaterialApp
 └ Scaffold
    ├ AppBar
    │  └ Text
    └ Column
       ├ Text
       └ Image
```
(Widget Tree)

Q3:

Ans.a) State management is crucial in Flutter because it controls how data flows and how UI updates in response to changes. In Flutter, state refers to data that affects the appearance or behavior of a widget. Managing state correctly ensures that UI stays in sync with underlying data.

Why State Management is Important

- UI Reactivity: Automatically rebuild the UI when the state changes, providing a dynamic user experience.
- Performance Optimization: Efficiently rebuild

Teacher's Sign.:

only the necessary widgets / preventing unnecessary rebuilders.

1. Data Consistency : Maintain consistent data across different parts of the app even with complex UI structures.

2. Seperation of Concerns : Helps keep business logic seperate from UI to de making apps more maintainable and testable.

ss 6) Comparing State Management Approaches in Flutter :

1.) Approach - setState
Description - Built-in, simple way to manage state by calling setState to rebuild a widget.
Pros :
- Easy to implement
- Great for small, local state
Best for : Small apps, local state (eg toggling a button, updating text fields)

2.) Provider.
Description - Inherited widget - based approach to pass data down the widget tree.
Pros :
Lightweight and flexible

Teacher's Sign.:

Good for medium
Easy to learn and widely supported
Best for: Medium to large apps

3) Approach - Riverpod
Description: A more robust, compile safe state management solution that fixes Providers limitations.

Pros - No need for context lookups.
2. Supports auto-disposal
More scalable and test friendly

Best for: Complex, large-scale apps

* When to Use Each Approach
- setState:
  - For small, local state changes within a single widget.
  - Example: A counter app.
- Provider
  When you need to share state across multiple widgets.
  Example: Managing user login status, theme switching, global settings.
- Riverpod:
  - For large, complex apps with multiple independent states.

Teacher's Sign.

Ex :- E-commerce apps with cart, etc

Q4)

Ans :) Integrating Firebase with a Flutter App / Application

Steps to Integrate Firebase in Flutter

1.) Create a Firebase Project :
- Go to Firebase Console → Create a new project

2.) Add an App to Firebase :
- Choose platform.
- For Android : Download the google-services.json file and place it in android/app directory.

3.) Update Flutter Project
- Add necessary firebase packages in pubspec.yaml :
  firebase_core, firebase_auth, etc
- Run flutter pub get to install dependencies

4.) Initialize Firebase
- Add firebase initialization in main.dart :

```
import 'package:firebase_core/
firebase_core.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
```

Teacher's Sign.: _____

runApp(MyApp());

3. Configure Native Platforms:
   For Android: Update android/app/build.gradle with google services plugin

Benefits of Using Firebase :-
1) Fully Managed Backend
2) Real Time DataBase and Sync
3) Authentication and Security
4) Scalability
5) Cloud Functions

Ans 4) Commonly Used Firebase Services:
1) Firebase Authentication user login / signup or social logins
2. Cloud Firebase: NoSQL database with real-time synchronization.
3. Realtime Database: JSON based database that syncs data in real time.
4. Firebase Analytics: Tracks user interactions and app events.
5) Firebase Storage: Stores and serves user-generated content like

Teacher's Sign.:

images, videos, etc.

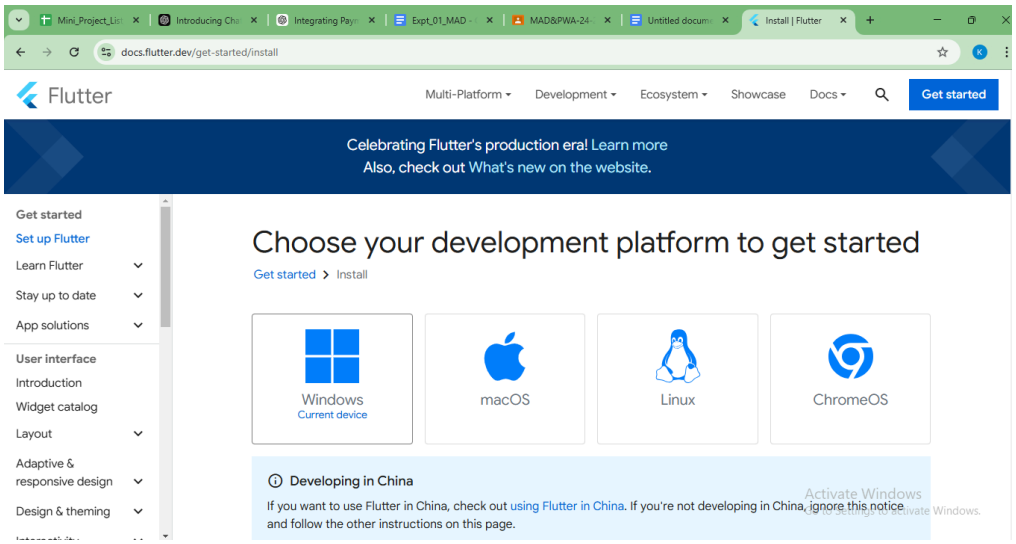
Data Synchronization in Firebase,

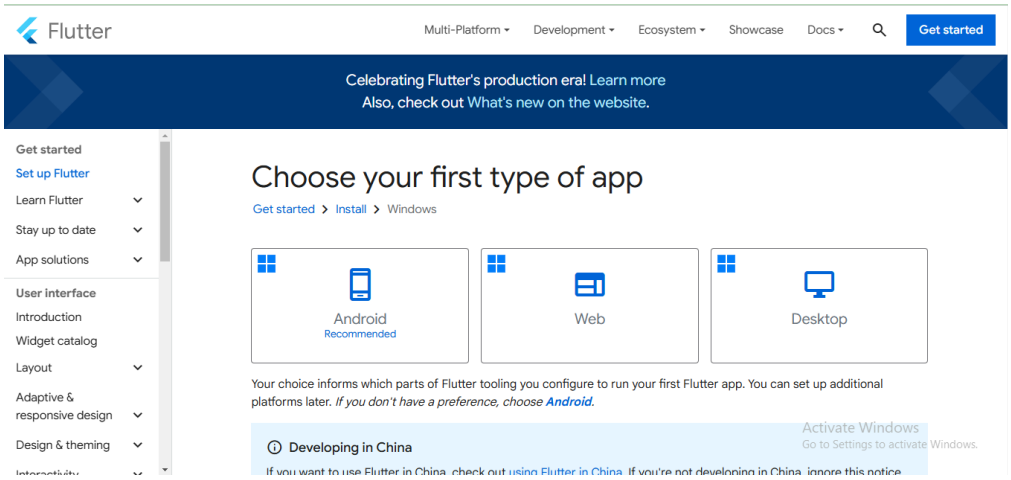
```
import 'package:cloud_firestore/cloud_firestore.dart';

StreamBuilder(
stream: FirebaseFirestore.instance.
collection('messages').snapshots(),
builder: (context, snapshot) {
if (!snapshot.hasData) return
CircularProgressIndicator();
var messages = snapshot.data!.docs;
return ListView.builder(
itemCount: messages.length,
itemBuilder: (context, index) {
return ListTile(
title: Text(messages[index]
['text']),
);
},
);
},
);
```

Snapshots() → Stream Real-time updates

Name - Karan Mishra
Roll No - 33
Div:D15B

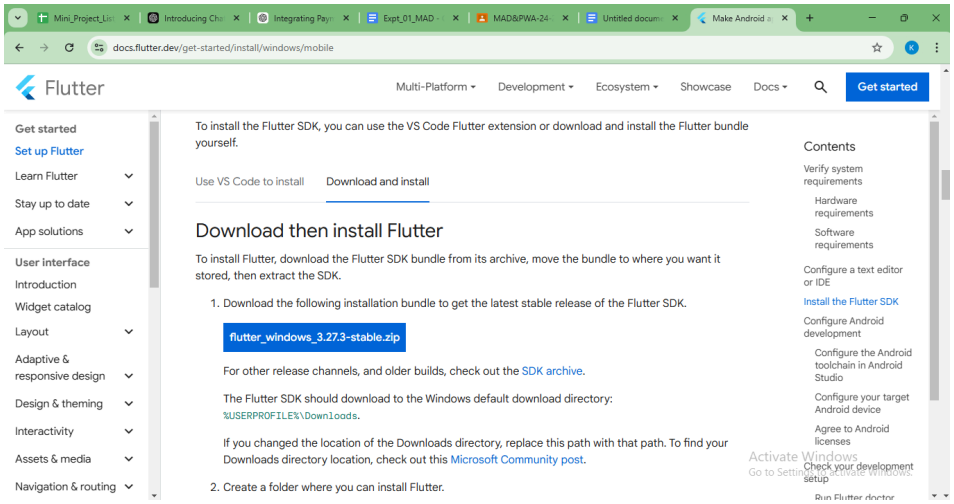
Step 1: Download the installation bundle of the Flutter Software Development Kit for windows. To download Flutter SDK, Go to its official website https://docs.flutter.dev/get-started/install , you will get the following screen.





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

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

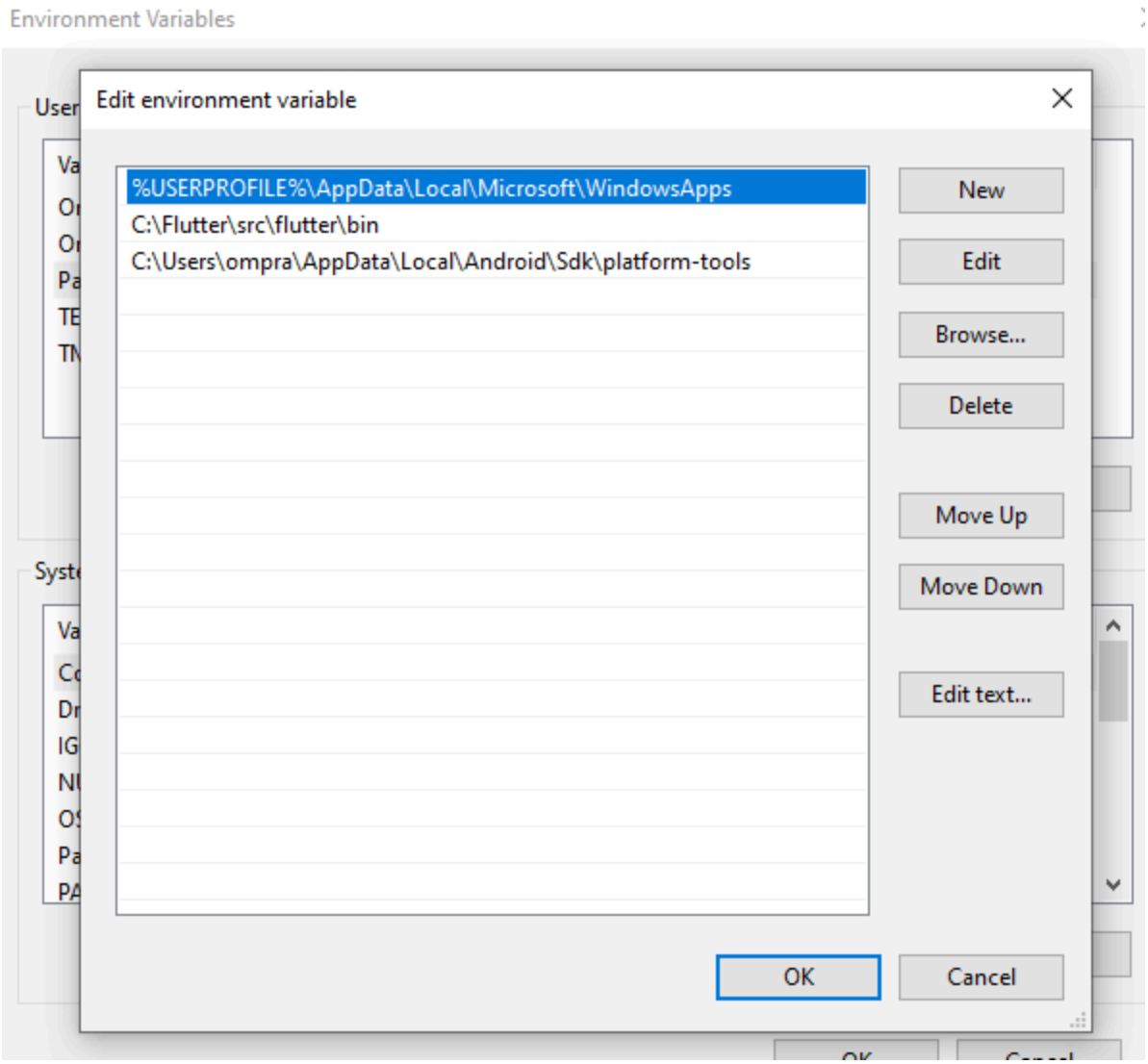


Step 4: To run the Flutter command in regular windows console, you need to update the system path to include the flutter bin directory. The following steps are required to do this:

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

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

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



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

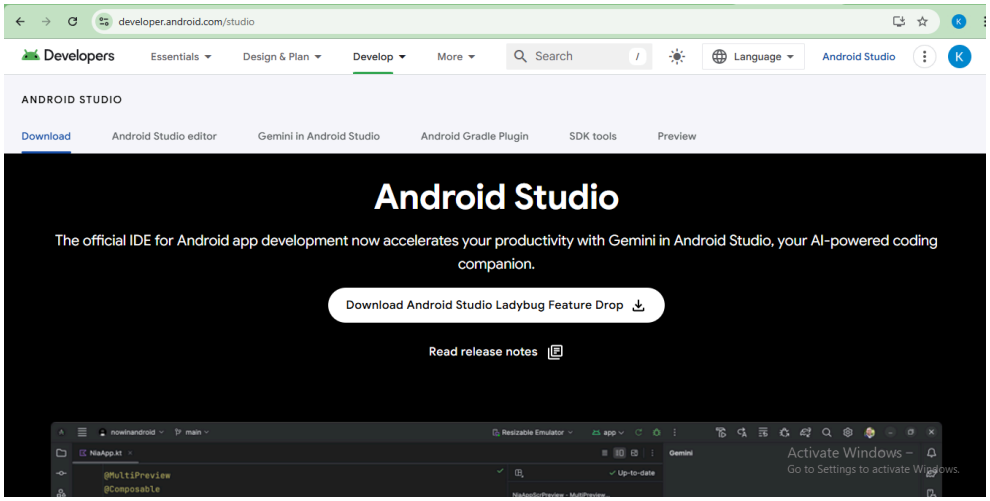
Now, run the $ flutter doctor command. This command checks for all the requirements of Flutter app development and displays a report of the status of your Flutter installation.

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

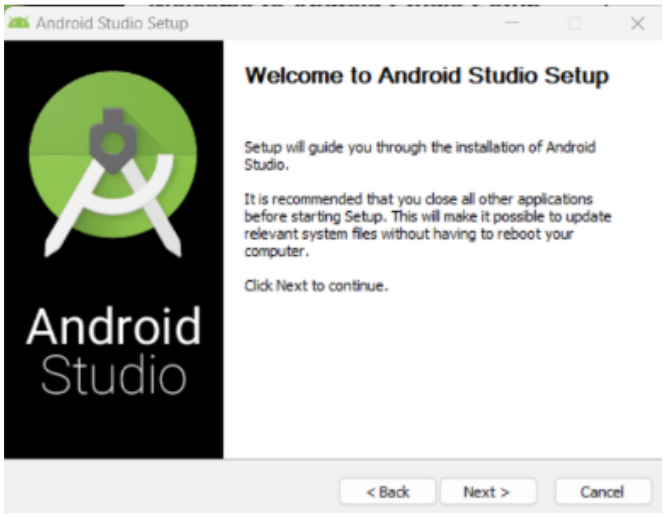
```
C:\Users\ompra>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.27.3, on Microsoft Windows [Version 10.0.19045.5247], locale en-IN)
[√] Windows Version (Installed version of Windows is version 10 or higher)
[√] Android toolchain - develop for Android devices (Android SDK version 35.0.1)
[√] Chrome - develop for the web
[X] Visual Studio - develop Windows apps
    X Visual Studio not installed; this is necessary to develop Windows apps.
      Download at https://visualstudio.microsoft.com/downloads/.
      Please install the "Desktop development with C++" workload, including all of its default components
[√] Android Studio (version 2024.2)
[√] Connected device (3 available)
[√] Network resources

! Doctor found issues in 1 category.
```

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.

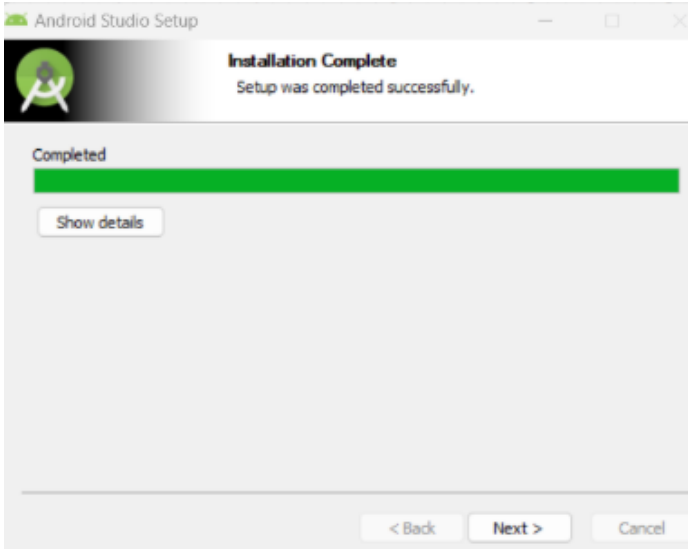
Step 7.1: Download the latest Android Studio executable or zip file from the official site.



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



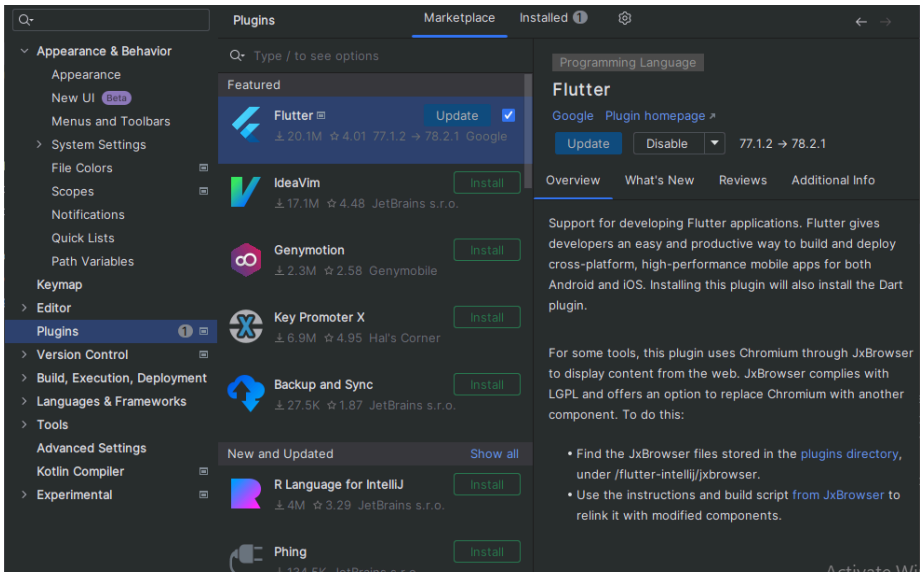
Step 7.3: Follow the steps of the installation wizard. Once the installation wizard completes, you will get the following screen.



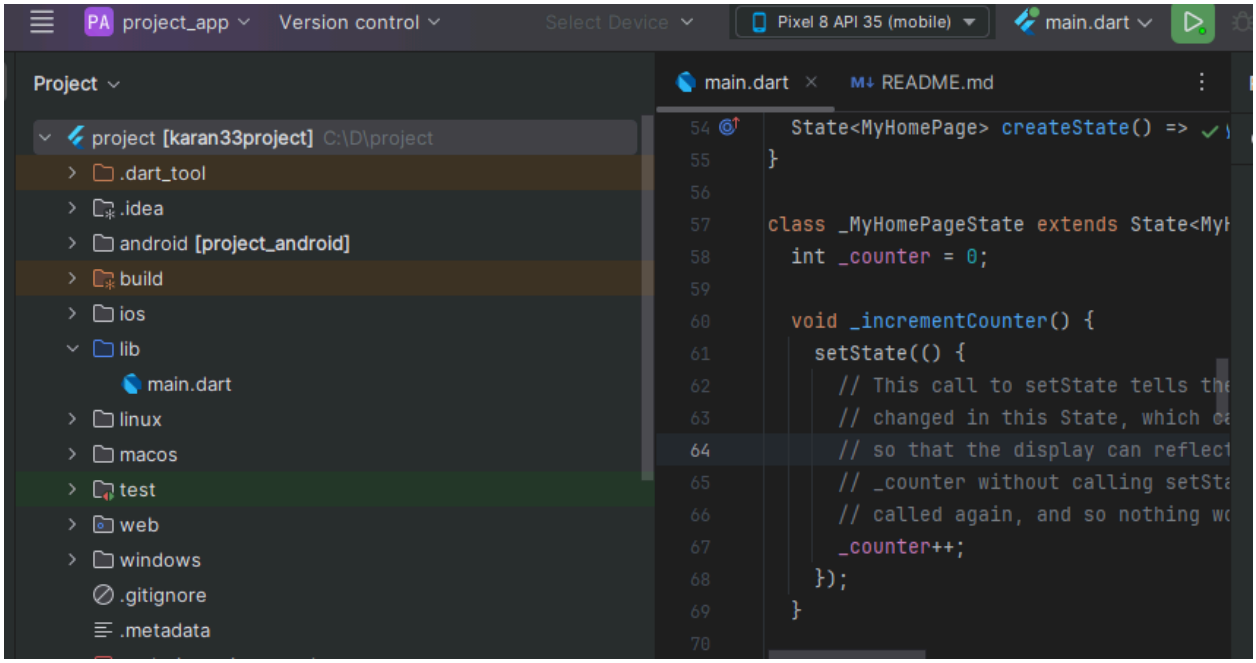
Step 7.4: In the above screen, click Next-> Finish. Once the Finish button is clicked, you need to
choose the 'Don't import Settings option' and click OK. It will start the Android Studio.

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.

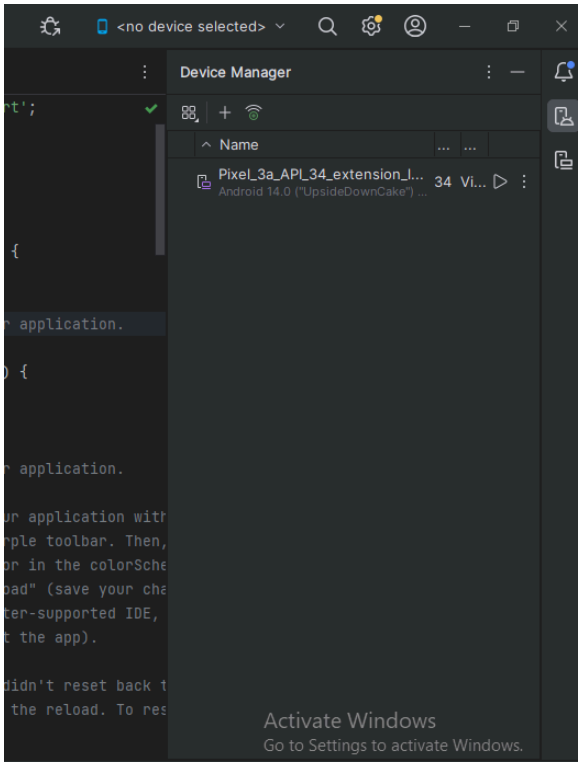


Step 8.2: Choose your device definition and click on Next.

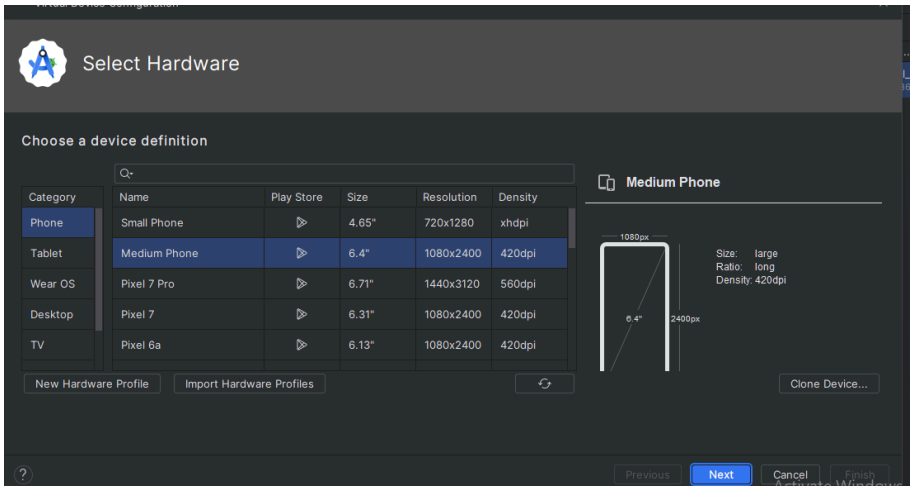


Step 8.3: Select the system image for the latest Android version and click on Next.

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



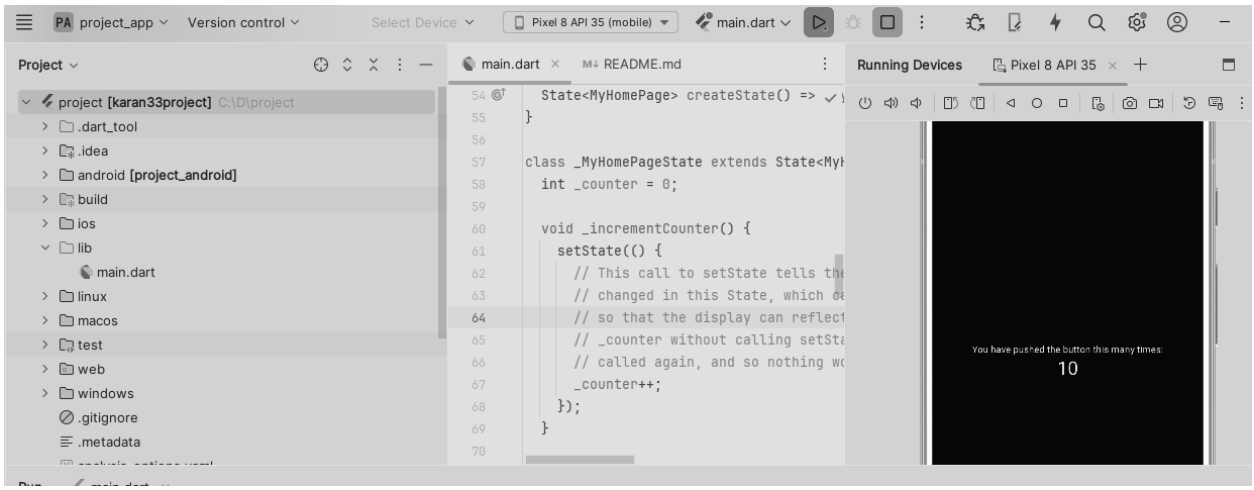
Step 9: Now, install Flutter and Dart plugin for building Flutter application in Android Studio. These plugins provide a template to create a Flutter application, give an option to run and debug Flutter application in the Android Studio itself. Do the following steps to install these plugins.

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.



```
request.onsuccess = (event) =>
  let db = event.target.re
  let transaction = db.trans
  db.transaction('users', 'readw
  let store = transaction.obje
  Store('users');
  store.put({id: 1, name: 'J
  Doe', 31);
});
```

By combining Indexed DB wi
Service workers, PWA's can
store and retrieve data e
when offline.

Comparison of Web Design Approaches

Approach - Responsive

Description - Uses CSS media queries and flexible grides to adjust layout dynamically.

Pros - Works on all devices, single codebase

Cons - May not provide pixel perfect control.

Approach - Fluid

Description - Uses percentage-based layouts that scale with screen size

Pros - Smooth scaling, better readibil

Cons - Can break at extreme screen sizes

iii) Adaptive - Approach

Description - Uses predefined layouts for specific screen sizes.

Pros - Optimized experience per device

Teacher's Sign.:

Teacher's Sign.:

	via web	via app
Performance	Faster loading due to caching	Generally faster for intensive tasks but requires de
Offline Functionality	Available through service workers	Requires explicit offline su

Ans2.) Responsive Web Design and its Importance in PWA's :-
Responsive Web Design (RWD) is a web development approach that ensures a website adapts different screen sizes and devices using flexible layouts, & fluid grids & media queries.

Importance in PWA's

- Ensures PWA's provide a seamless experience across various devices.
- Helps achieve a native-like experience on mobile, without requiring seperate designs.
- Improves accessibility and usability making PWA's more user friendly

Roll No -33 Div -D15B

MPL Assignment 2

Date
Page

Progressive Web App and Its Significance

A Progressive Web App is a type of web application that provides a native-app like experience using modern web technologies. PWAs are designed to be reliable, fast and engaging by leveraging features as service workers, caching and push notifications.

Significance :-

Cross-Platform Compatibility : PWA's work on any device with a web browser.

Offline Support - Uses service workers for caching, enabling offline access.

Performance Improvement - Faster load times due to caching and efficient asset management.

App-Like Experience : Provides home screen installation, push notifications and background updates

Feature	PWA	Traditional
Install	No App Store required	Installed via App Store
Platform Dependency	Works across all platforms with a browser	Requires platform-specific development.
Updates	Auto-updates via	Manual updates

~~Comparison~~ sons - Requires multi versions increasing complexity.

Q3.) Service Workers Lifecycle :-
Ans Services workers are background scripts that run independently of web pages and enable caching, push notifications and offline functionality. Their lifecycle consits of 3 phases:

1.) Registration
- The service worker is registered in the browser using JavaScript (navigator.serviceWorker.register()).

2.) Installation
- The browser installs the service worker and caches necessary assets.
- If successful, it moves to the activation phase, otherwise, it is discarded.

3.) Activation
- The service worker takes control of all pages in its scope.
- Old caches are cleaned up and

Teacher's Sign.: _____

new ones are initialized.
Once activated it listens for fetch events and other background tasks.

4) IndexedDB in Service Workers for Data Storage
IndexedDB is a low-level NoSQL database in the browser, allowing structured data storage.

Usage in Service Workers
Stores large data sets offline, enabling seamless user experience without an internet connection.
Works asynchronously preventing UI blocking.
Allows read/write operations even when a page is closed

Example
```
const request = indexedDB.open("PWA_DB", 1);

request.onupgradeneeded = (event) => {
  let db = event.target.result;
  db.createObjectStore("users", { keyPath: "id" });
};
```