**﻿Flutter Intern﻿ position at ﻿Together﻿ Assignment**

**TO RUN THE APPLICATION :**

1 - Run -  **“ flutter pub get “** command in terminal to get all packages.

2 - Next Select the device for test Run and Start Debugginig from Run.

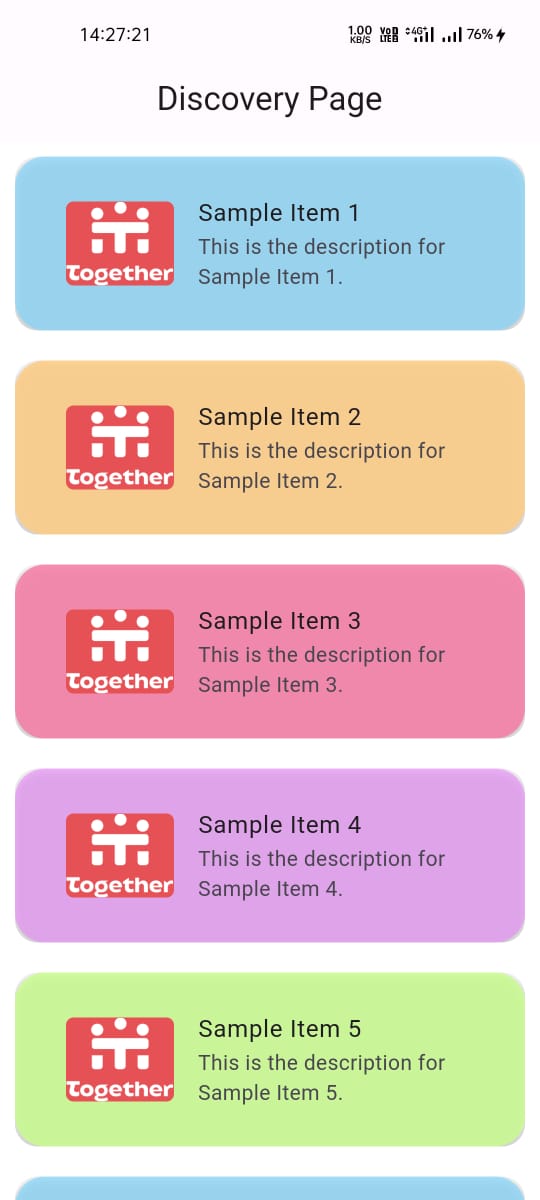
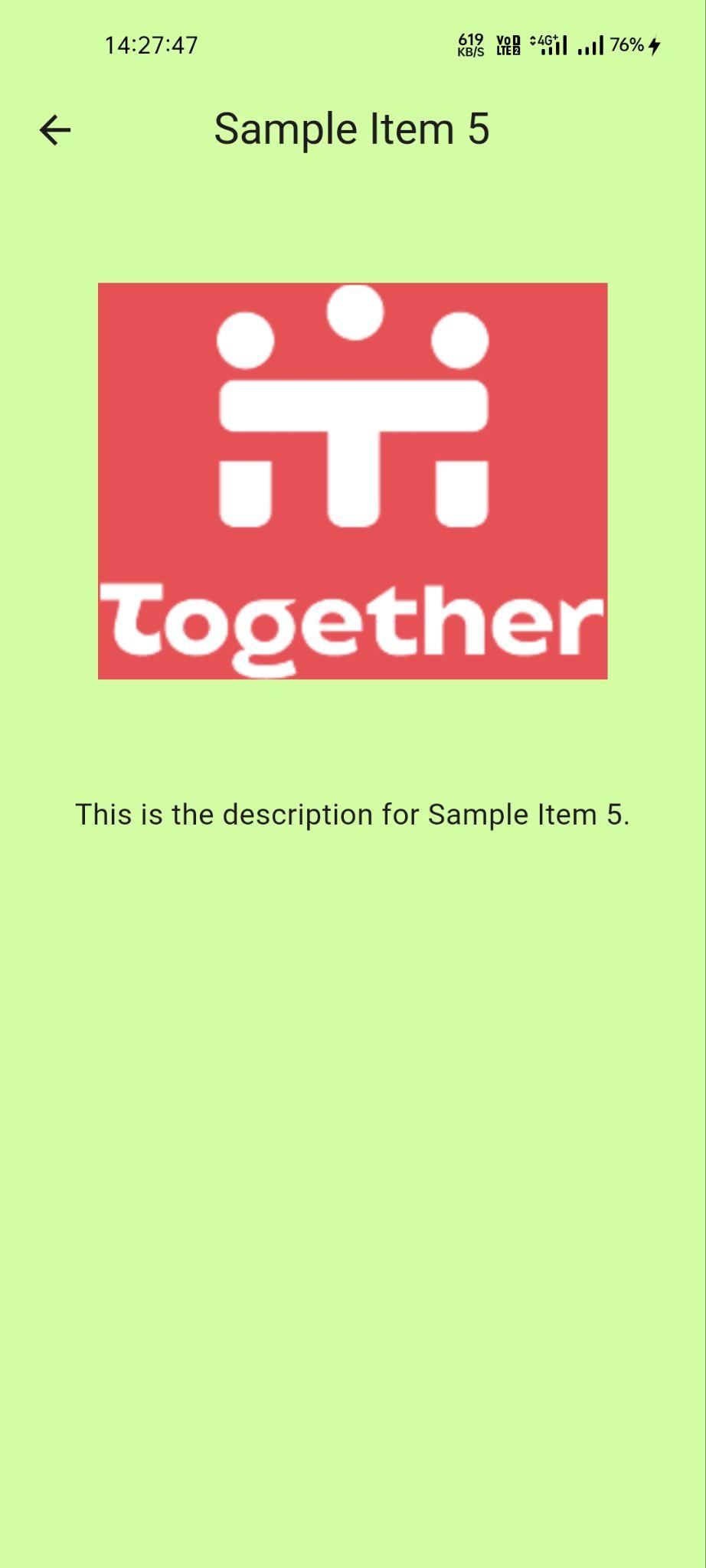
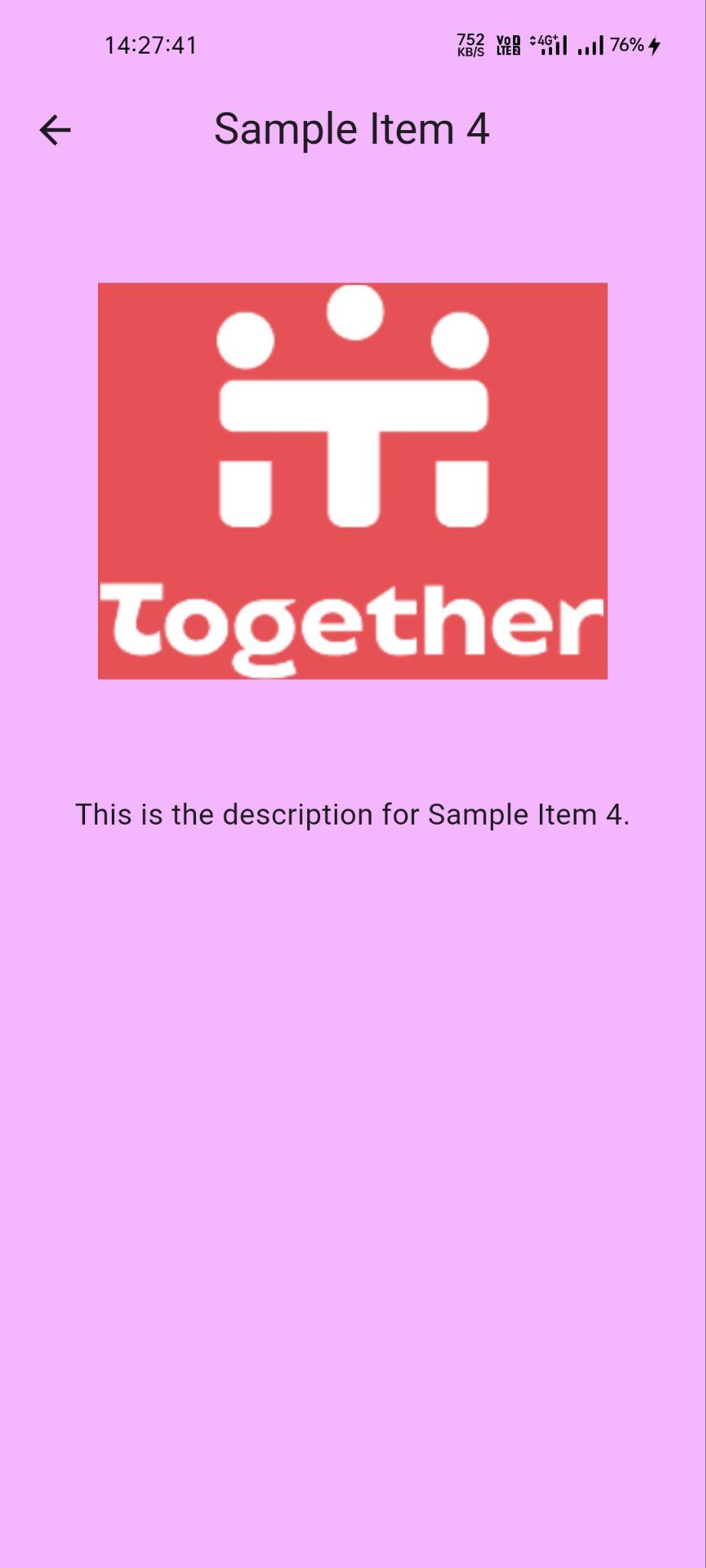
3 - Apk file is also available in below link.

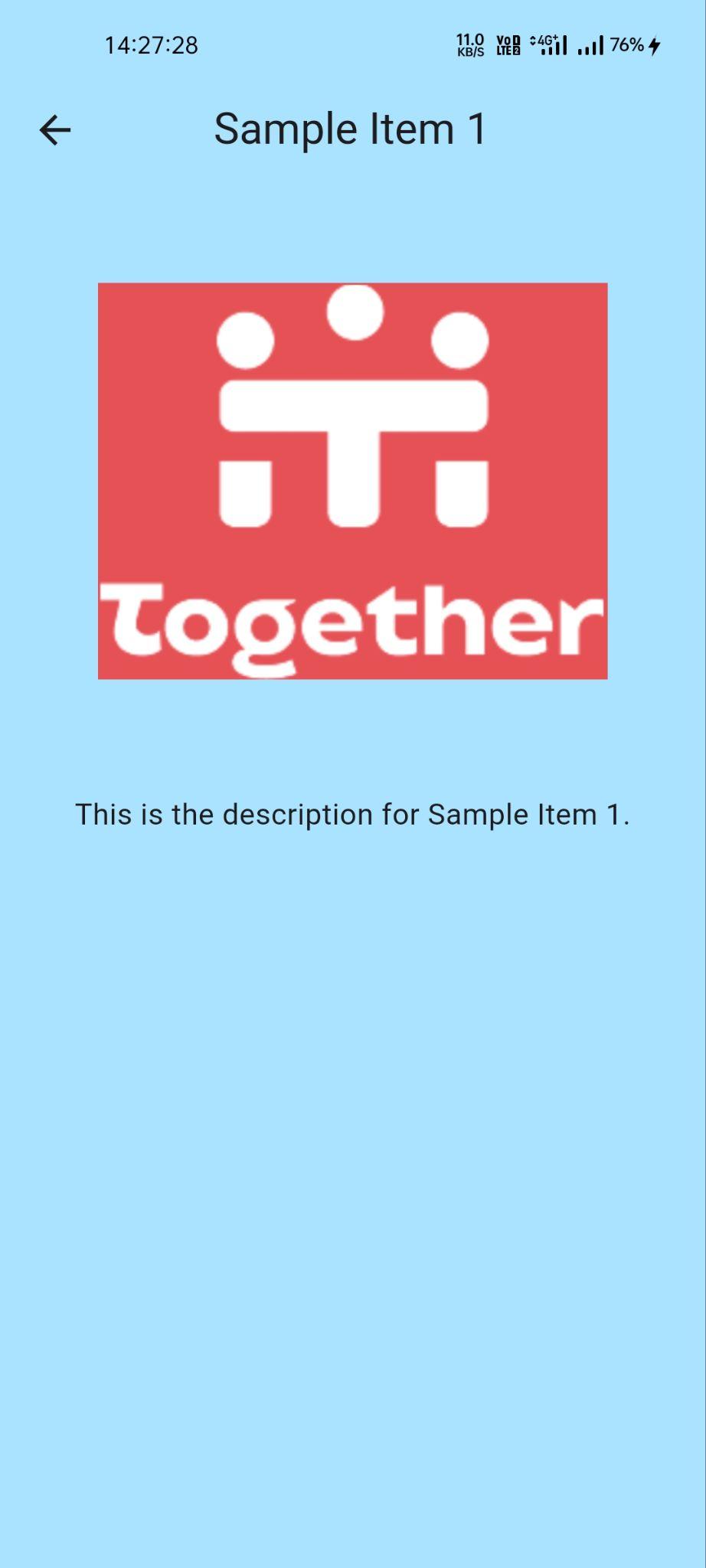
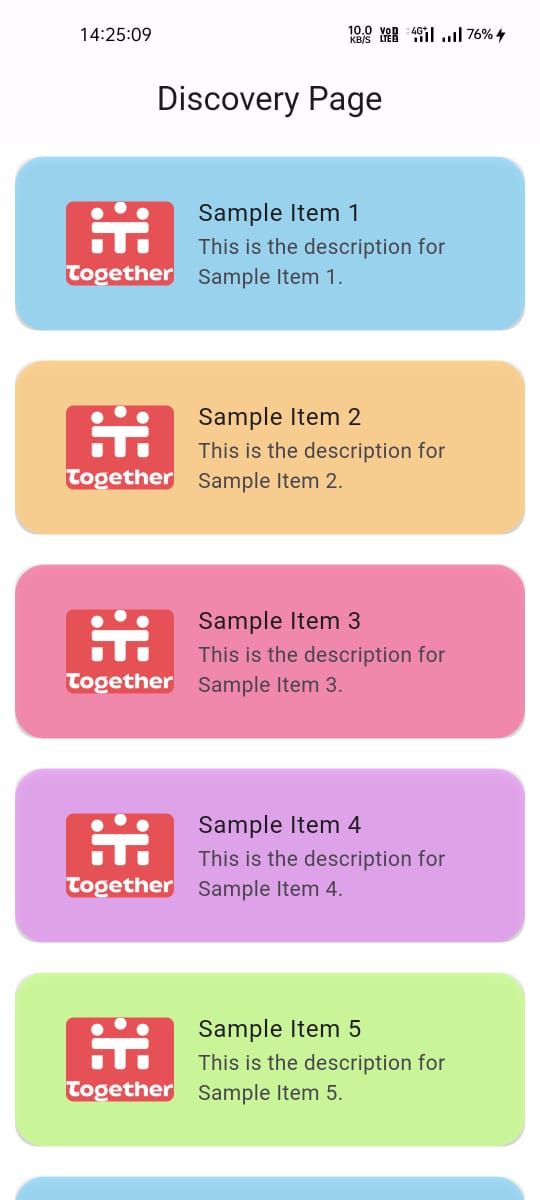
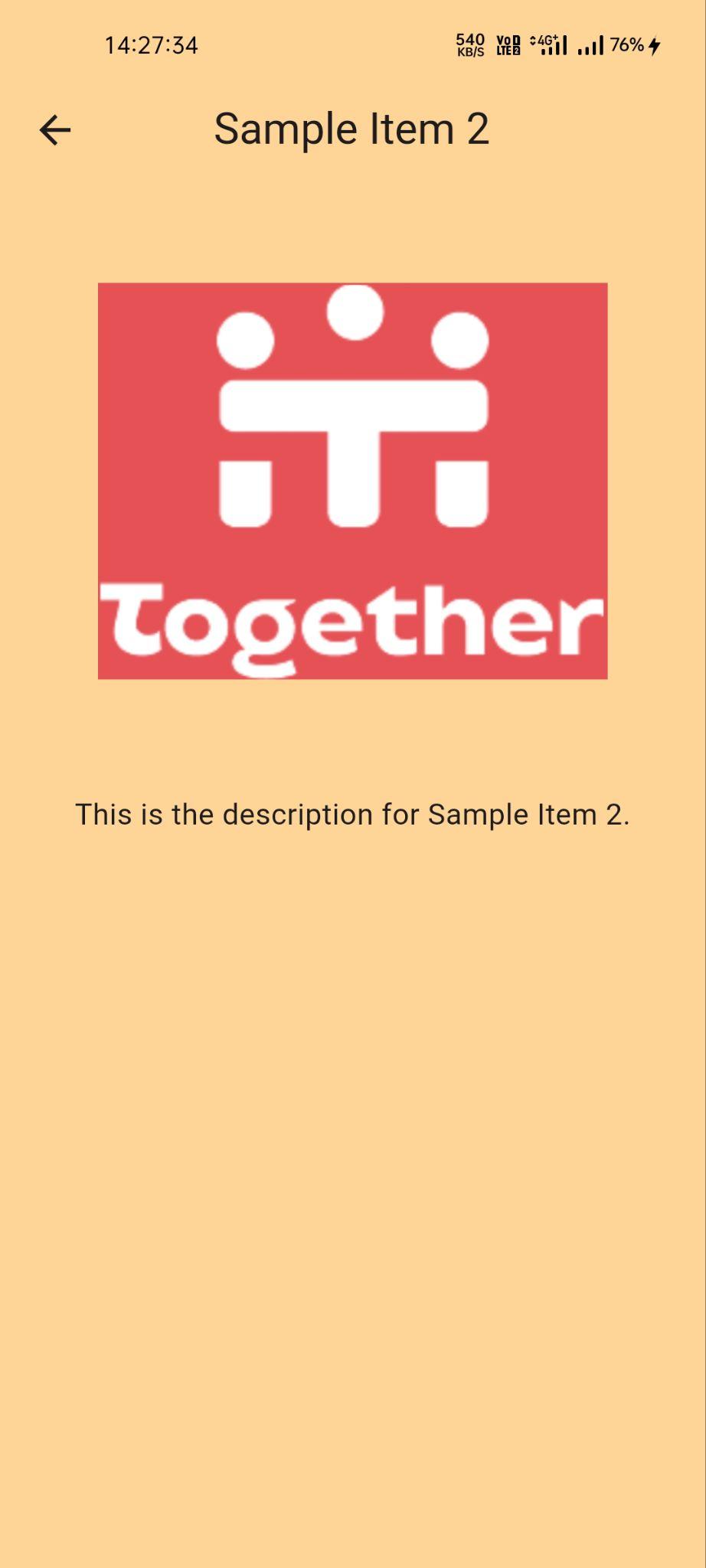
**DRIVE LINK :**

<https://drive.google.com/drive/folders/1p5fF_zv92BsTD2Reollqt012Tvao38Hn?usp=sharing>

**IMAGES**

**:**

**  **

**  **

**IMPLEMENTED CONCEPTS :**> Implemented API concepts.  
> Also implemented from Page Transition Animations.  
>I think so UI is not too Attractive but I tird my best to make it appealing.

**Code :**

**discovery.dart**

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'dart:convert';

class DiscoveryPage extends StatefulWidget {

@override

\_DiscoveryPageState createState() => \_DiscoveryPageState();

}

class \_DiscoveryPageState extends State<DiscoveryPage> {

List<dynamic> \_items = [];

int \_page = 1;

bool \_isLoading = false;

ScrollController \_scrollController = ScrollController();

@override

void get initState {

super.initState;

\_fetchData();

\_scrollController.addListener(() {

if (\_scrollController.position.pixels ==

\_scrollController.position.maxScrollExtent) {

if (!\_isLoading) {

\_fetchData();

}

}

});

}

Future<void> \_fetchData() async {

setState(() {

\_isLoading = true;

});

final response = await http.get(Uri.parse(

'https://api-stg.together.buzz/mocks/discovery?page=$\_page&limit=10'));

if (response.statusCode == 200) {

final data = json.decode(response.body);

final List<dynamic>? items = data['data'];

if (items != null) {

setState(() {

\_items.addAll(items.map((item) => {

'id': item['id'] ?? '',

'title': item['title'] ?? '',

'description': item['description'] ?? '',

'image': item['image\_url'] ?? '',

}));

\_page++;

\_isLoading = false;

});

} else {

print('Items is null');

setState(() {

\_isLoading = false;

});

}

} else {

print('Failed to load data');

setState(() {

\_isLoading = false;

});

}

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Discovery Page'),

centerTitle: true,

),

body: Container(

decoration: BoxDecoration(

gradient: LinearGradient(

begin: Alignment.topCenter,

end: Alignment.bottomCenter,

colors: [

const Color.fromARGB(255, 255, 255, 255),

Color.fromARGB(255, 255, 255, 255)

],

),

),

child: \_isLoading && \_items.isEmpty

? Center(child: CircularProgressIndicator())

: \_buildListView(),

),

);

}

Widget \_buildListView() {

final List<Color> backgroundColors = [

Color.fromARGB(162, 121, 213, 255),

Color.fromARGB(214, 255, 205, 131),

Color.fromARGB(197, 249, 115, 160),

Color.fromARGB(137, 235, 124, 255),

Color.fromARGB(199, 200, 255, 137),

// Add more colors as needed

];

return ListView.builder(

controller: \_scrollController,

itemCount: \_items.length + 1,

itemBuilder: (context, index) {

if (index < \_items.length) {

final item = \_items[index];

final Color backgroundColor = backgroundColors[

index % backgroundColors.length]; // Cycle through colors

return Padding(

padding: EdgeInsets.all(10.0),

child: MouseRegion(

onEnter: (\_) {

// Add hover animation or effects here

},

onExit: (\_) {

// Remove hover animation or effects here

},

child: ClipRRect(

borderRadius: BorderRadius.circular(15.0),

child: GestureDetector(

onTap: () {

// Navigate to the details page with container transform animation

Navigator.push(

context,

PageRouteBuilder(

transitionDuration: Duration(milliseconds: 500),

pageBuilder: (context, animation, secondaryAnimation) {

return FadeTransition(

opacity: animation,

child: DetailsPage(

item: {

'title': item['title'],

'description': item['description'],

'image': item['image'],

'backgroundColor': backgroundColor,

},

),

);

},

),

);

},

child: Container(

decoration: BoxDecoration(

color: backgroundColor,

borderRadius: BorderRadius.circular(20.0),

boxShadow: [

BoxShadow(

color: Colors.grey.withOpacity(0.5),

spreadRadius: 2,

blurRadius: 5,

offset: Offset(0, 3),

),

],

),

child: Padding(

padding: EdgeInsets.all(18.0),

child: ListTile(

title: Text(item['title'] ?? ''),

subtitle: Text(item['description'] ?? ''),

leading: ClipRRect(

borderRadius: BorderRadius.circular(5.0),

child: item['image'] != null

? Image.network(item['image'])

: null,

),

),

),

),

),

),

),

);

} else {

if (\_isLoading) {

return \_buildLoadingIndicator();

} else {

return \_items.isEmpty ? \_buildEmptyState() : \_buildEndOfList();

}

}

},

);

}

Widget \_buildLoadingIndicator() {

return Center(

child: Padding(

padding: const EdgeInsets.all(20.0),

child: CircularProgressIndicator(

valueColor: AlwaysStoppedAnimation<Color>(Colors.white),

),

),

);

}

Widget \_buildEmptyState() {

return Center(

child: Padding(

padding: const EdgeInsets.all(20.0),

child: Text(

'No items found',

style: TextStyle(color: Colors.white),

),

),

);

}

Widget \_buildEndOfList() {

return Center(

child: Padding(

padding: const EdgeInsets.all(20.0),

child: Text(

'End of list',

style: TextStyle(color: Colors.white),

),

),

);

}

@override

void dispose() {

\_scrollController.dispose();

super.dispose();

}

}

class DetailsPage extends StatelessWidget {

final dynamic item;

const DetailsPage({Key? key, required this.item}) : super(key: key);

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text(item['title'] ?? ''),

centerTitle: true,

backgroundColor: item['backgroundColor'],

),

body: Container(

decoration: BoxDecoration(

color: item['backgroundColor'],

),

child: Center(

child: Column(

// mainAxisAlignment: MainAxisAlignment.center,

children: [

if (item['image'] != null)

Padding(

padding: const EdgeInsets.all(50.0),

child: Image.network(item['image']),

),

Padding(

padding: const EdgeInsets.all(8.0),

child: Text(

item['description'] ?? '',

style: TextStyle(fontSize: 15),

),

),

],

),

),

),

);

}

}

**Thank You Together for This Opportunity.**