MPL Experiment 6

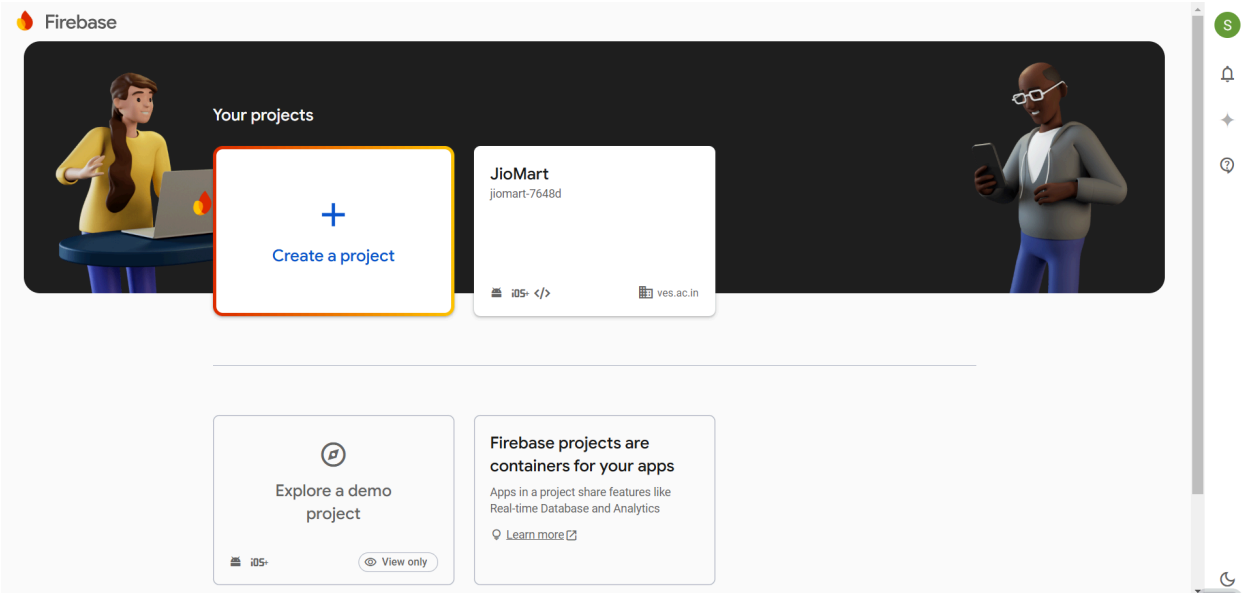
Name: Sneha Patra **Class:** D15A **Roll no:** 40

Aim: How To Set Up Firebase with Flutter for iOS and Android Apps

Steps to Set Up Firebase with Flutter:

Step 1:

Go to the Firebase Console (https://console.firebase.google.com/). Click on "Add Project" and follow the steps to create your Firebase project. Once the project is created, select the Flutter option for connecting your app with Firebase.



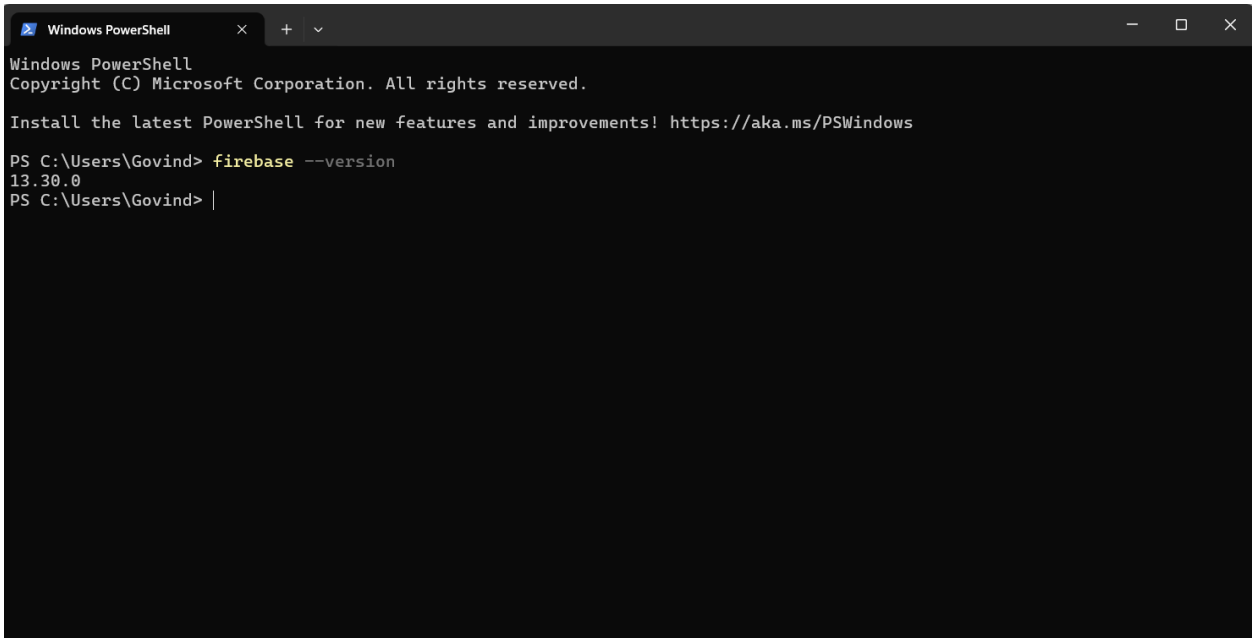
Step 2:

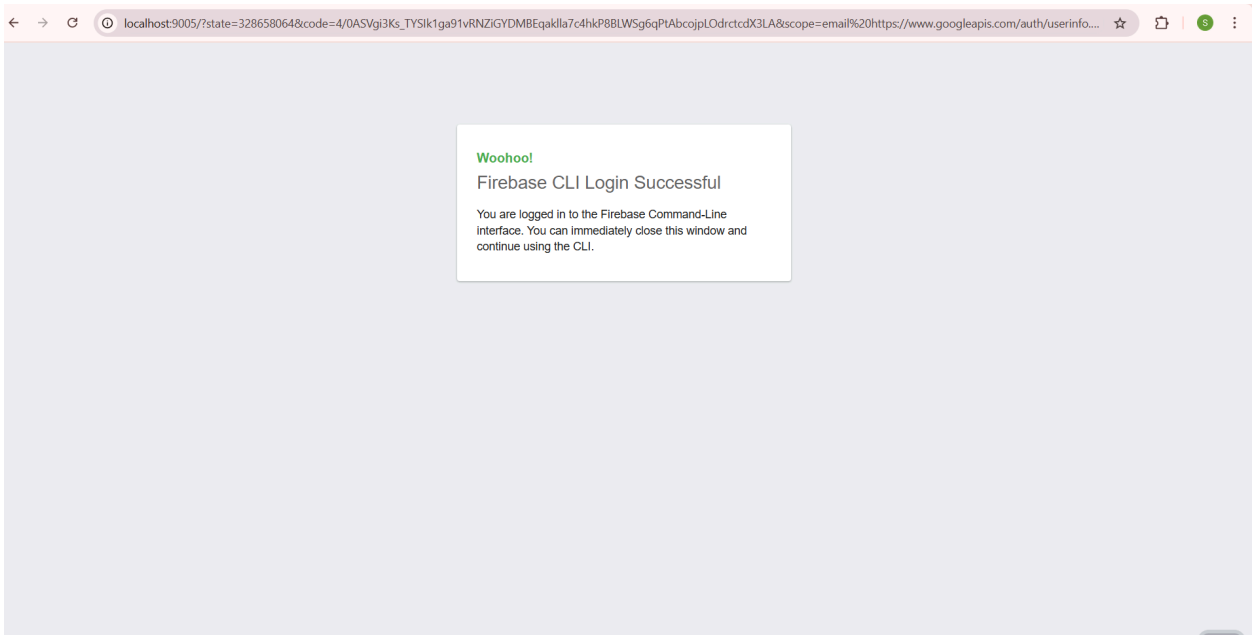
Open **Windows PowerShell** (or any terminal you prefer).

Run the following commands to install Firebase CLI and verify the installation:

```
npm install -g firebase-tools
firebase --version
firebase login
```

This will install the **Firebase CLI**, check the version, and log you into your Firebase account.





Step 3:

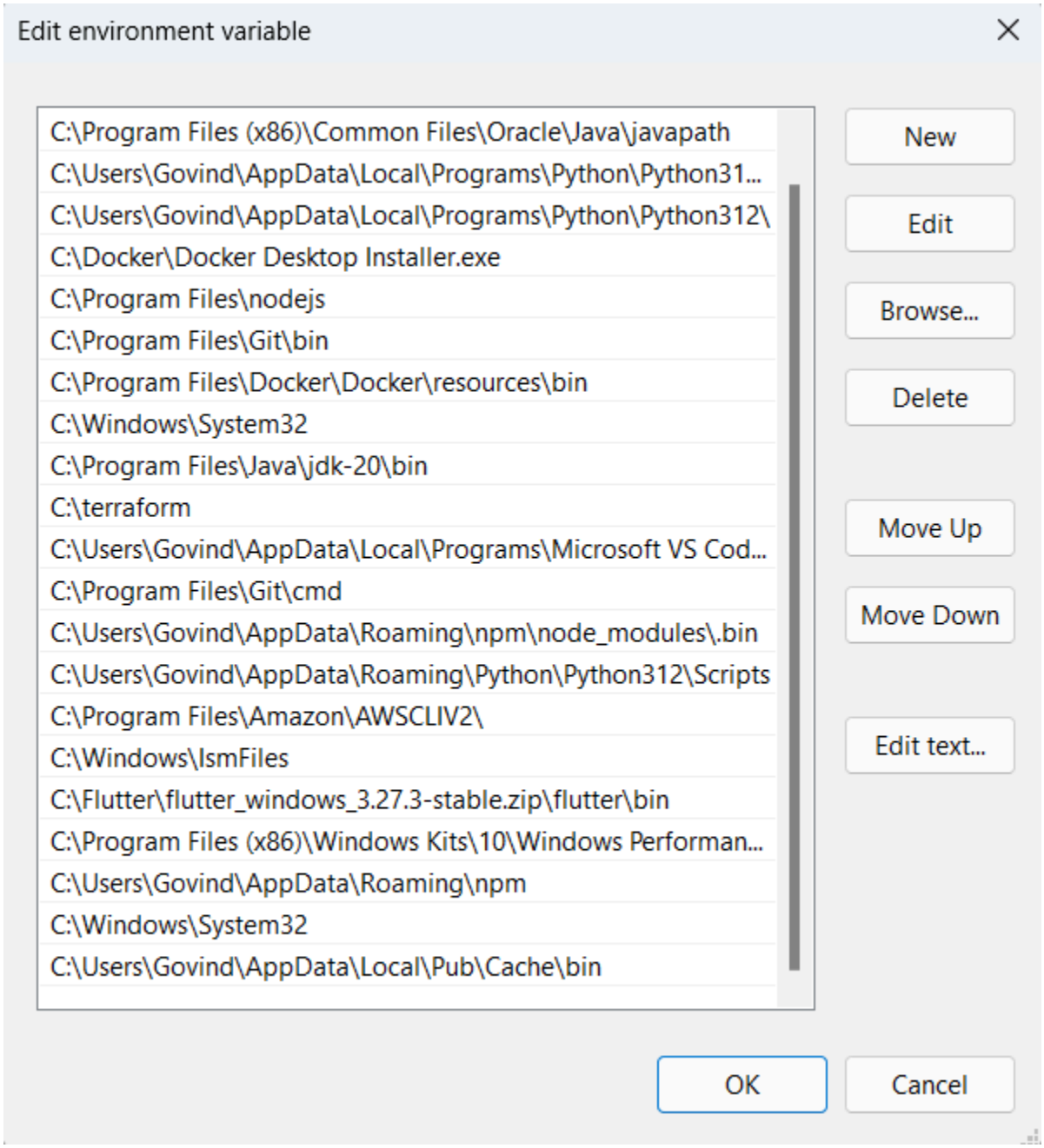
Open your Flutter app in **Android Studio**.

In the terminal of Android Studio, run the following command to activate flutterfire_cli:

```
dart pub global activate flutterfire_cli
```

Add flutterfire to your Environment Variables. You may need to restart Android Studio after

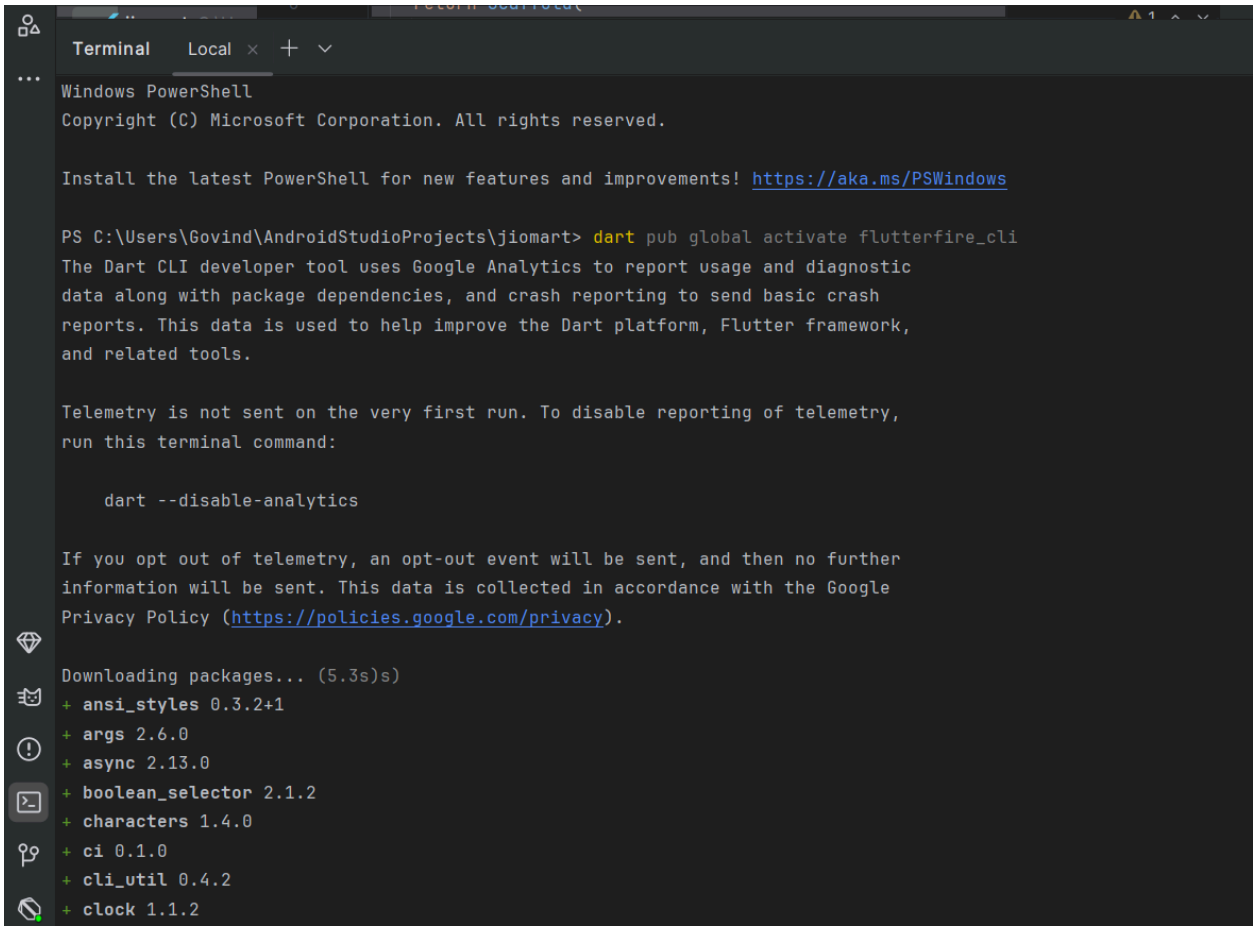
adding it.

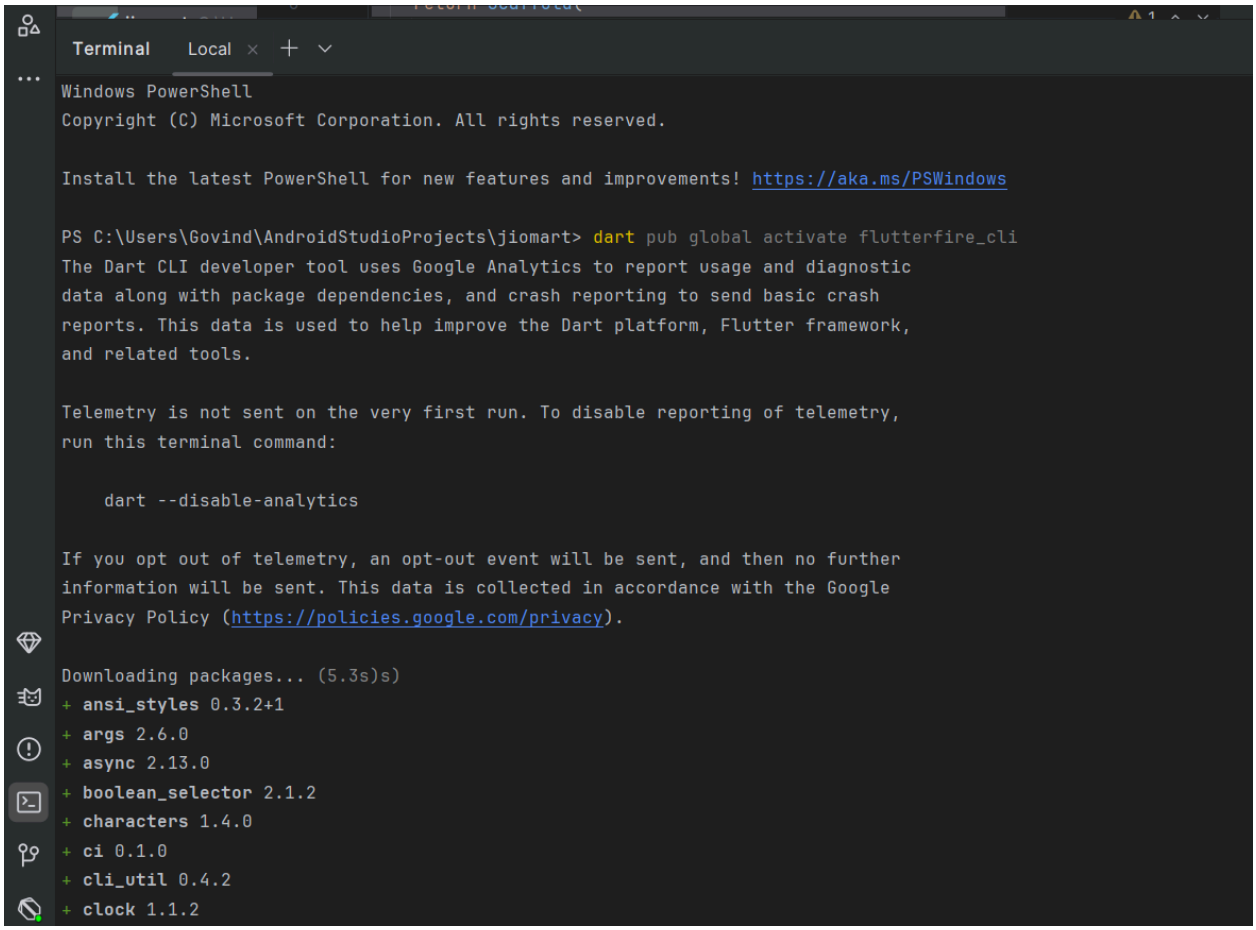


Step 4:
Run the following command to configure Firebase with your project:

```
flutterfire configure --project=dmart-c9f2c
```

Replace `dmart-c9f2c` with your Firebase project ID.

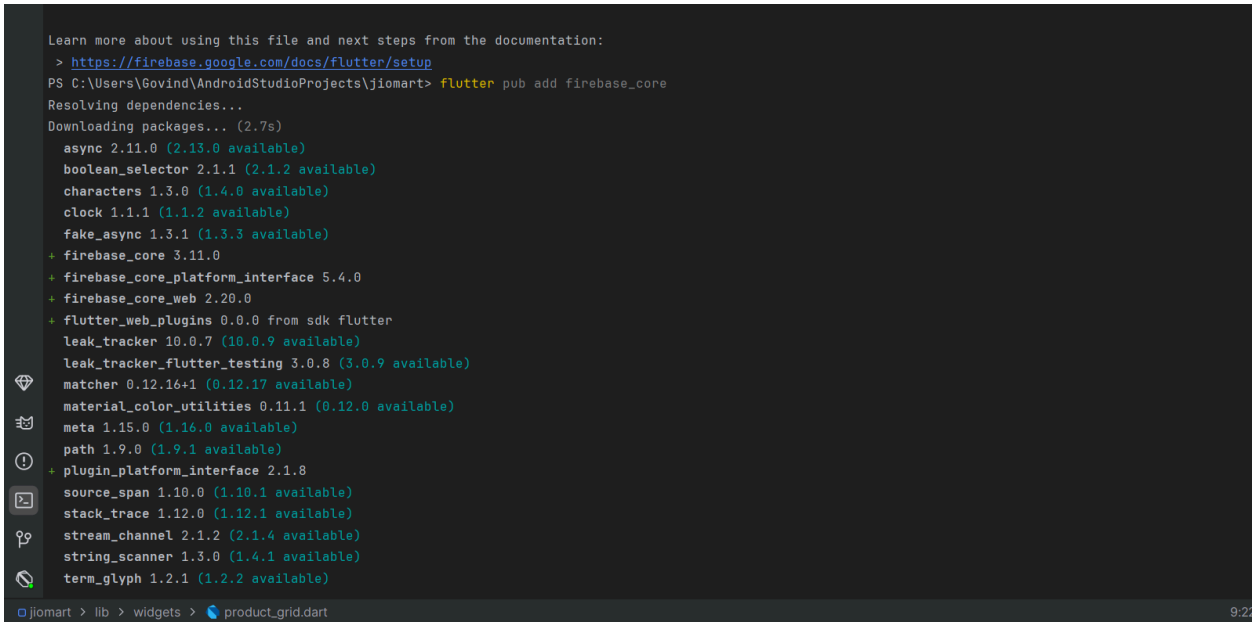

Terminal
Local
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\Users\Govind\AndroidStudioProjects\jiomart> dart pub global activate flutterfire_cli
The Dart CLI developer tool uses Google Analytics to report usage and diagnostic
data along with package dependencies, and crash reporting to send basic crash
reports. This data is used to help improve the Dart platform, Flutter framework,
and related tools.
Telemetry is not sent on the very first run. To disable reporting of telemetry,
run this terminal command:
dart --disable-analytics
If you opt out of telemetry, an opt-out event will be sent, and then no further
information will be sent. This data is collected in accordance with the Google
Privacy Policy (https://policies.google.com/privacy).
Downloading packages... (5.3s)s)
+ ansi_styles 0.3.2+1
+ args 2.6.0
+ async 2.13.0
+ boolean_selector 2.1.2
+ characters 1.4.0
+ ci 0.1.0
+ cli_util 0.4.2
+ clock 1.1.2



Step 5:

Run this command to add Firebase Core dependency to your app:

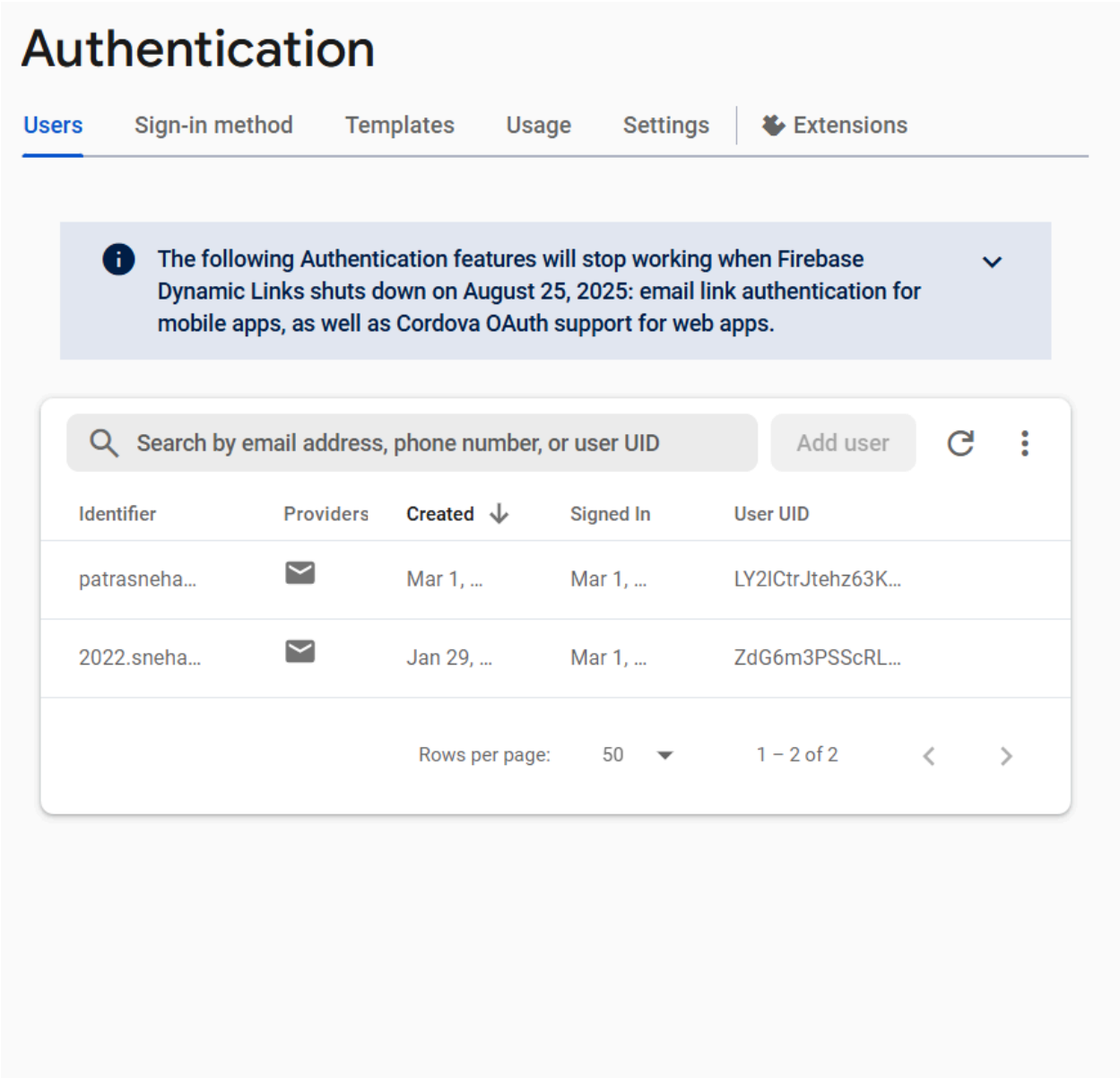
```
flutter pub add firebase_core
```



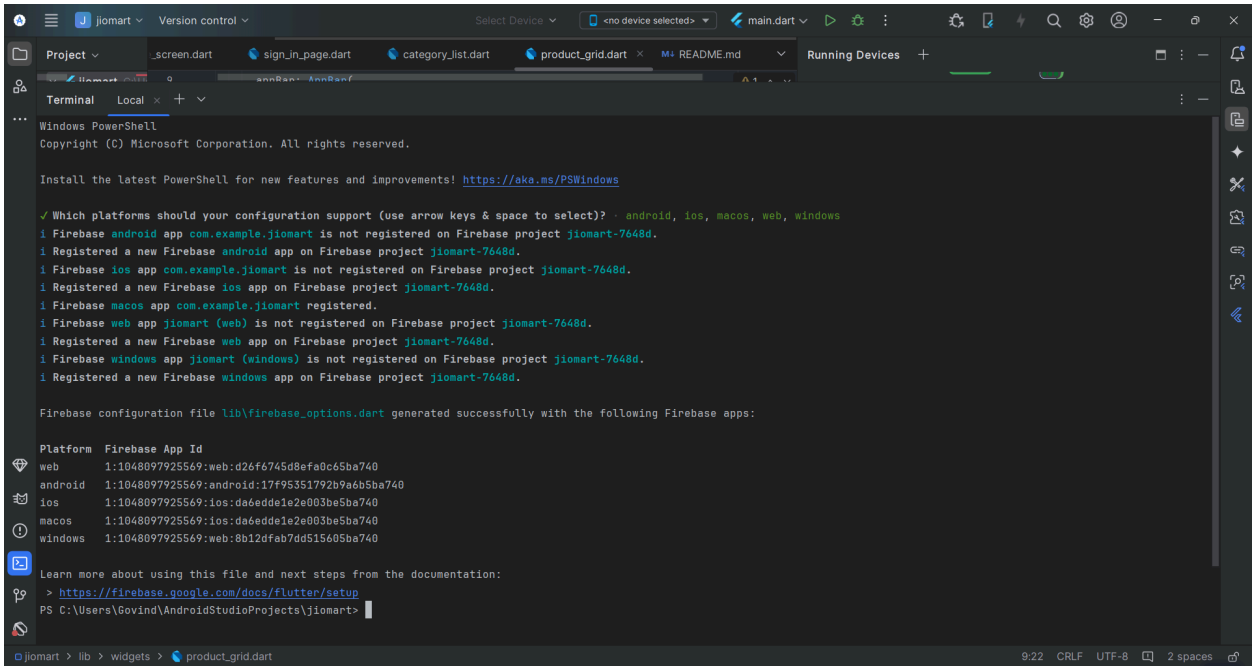
Firebase Authentication SetUp

Step 1:

Go to the Firebase Console (https://console.firebase.google.com/). In your Firebase project, navigate to **Authentication**. Under the **Sign-in method** tab, enable **Email/Password** sign-in. Once enabled, go to the **Users** section and click on **Add User**. Enter a **username** (email) and a **password** for the new user.



Step 2:
Open your app in **Android Studio**.
In the terminal, run the following command to add the Firebase Authentication dependency:
`flutter pub add firebase_auth`



Step 3:

If you encounter any issues, add the following configuration to your `android/app/build.gradle` file:

```
android {
    defaultConfig {
        minSdk = 23
        targetSdk = flutter.targetSdkVersion
        versionCode = flutter.versionCode
        versionName = flutter.versionName
    }
}
```

This ensures that the Firebase dependencies are compatible with your Android app.

Now, firebase is successfully connected to our app.

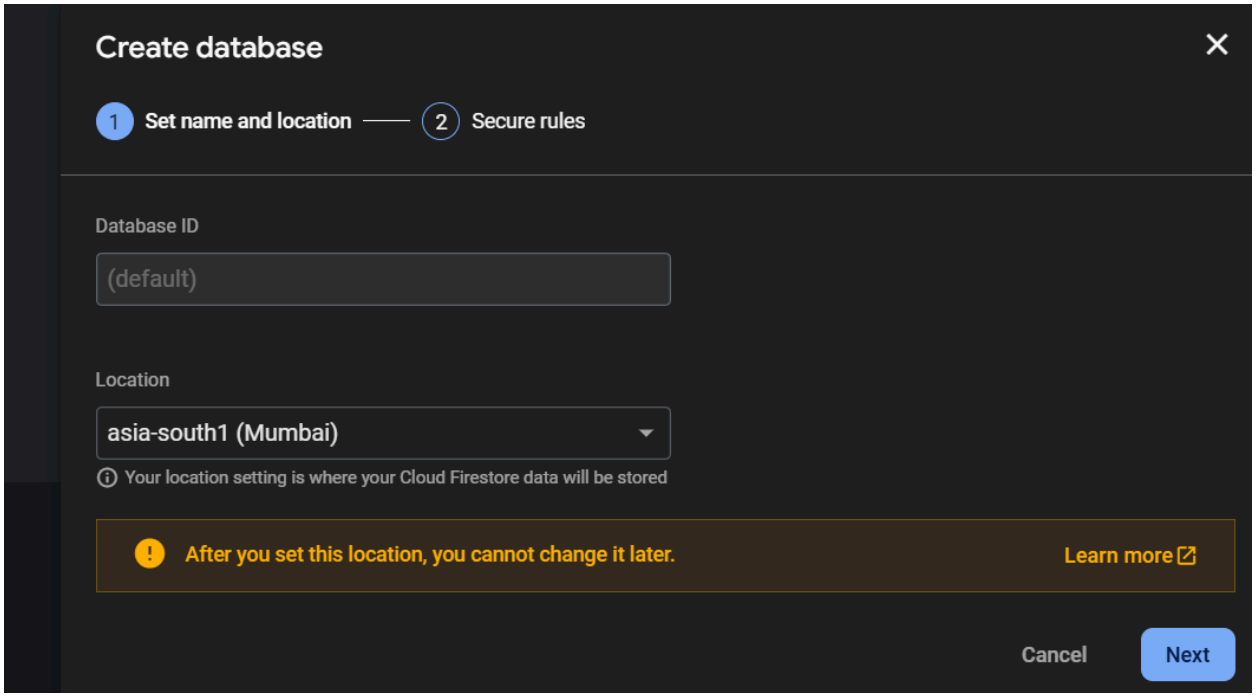
Firestore Database Setup

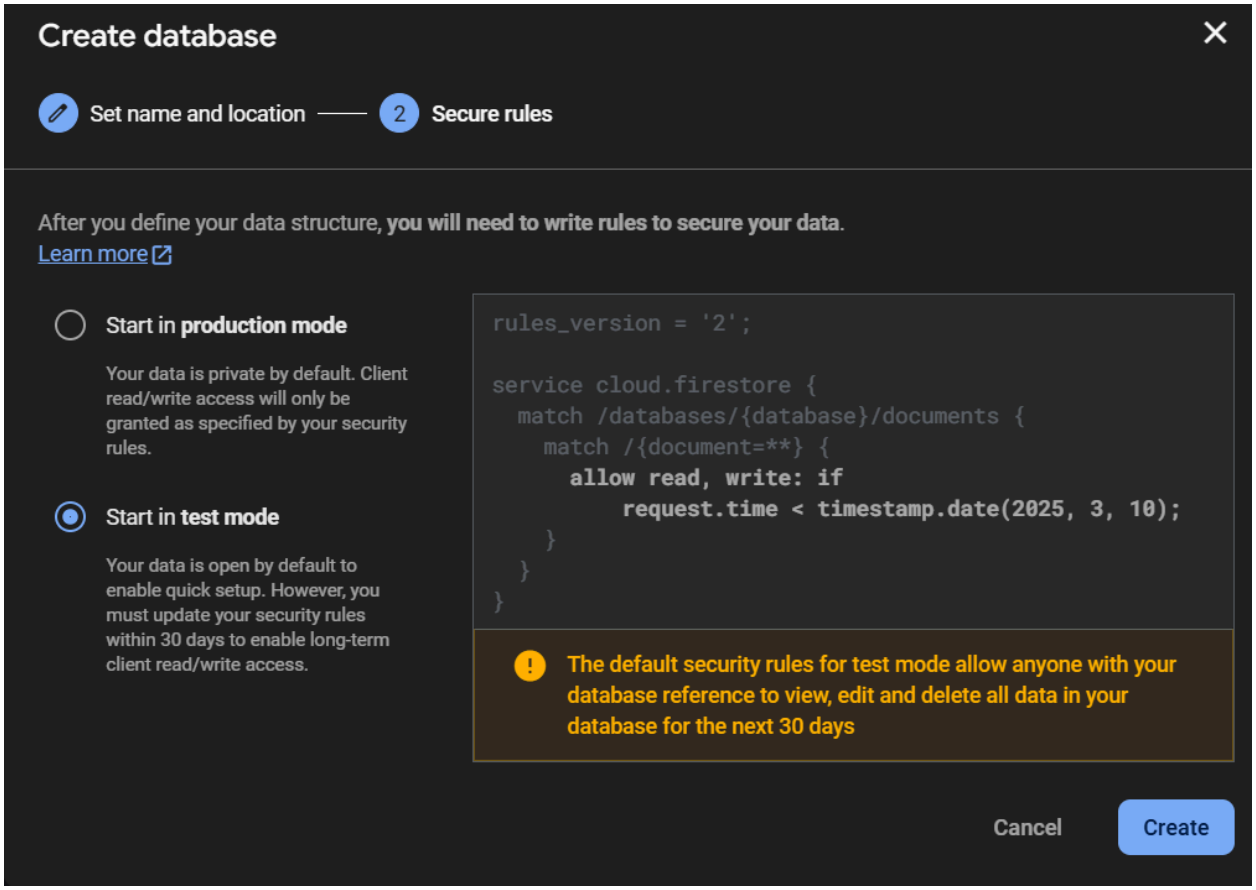
Step 1:

Select **Firestore Database** from **Build** in the sidebar.

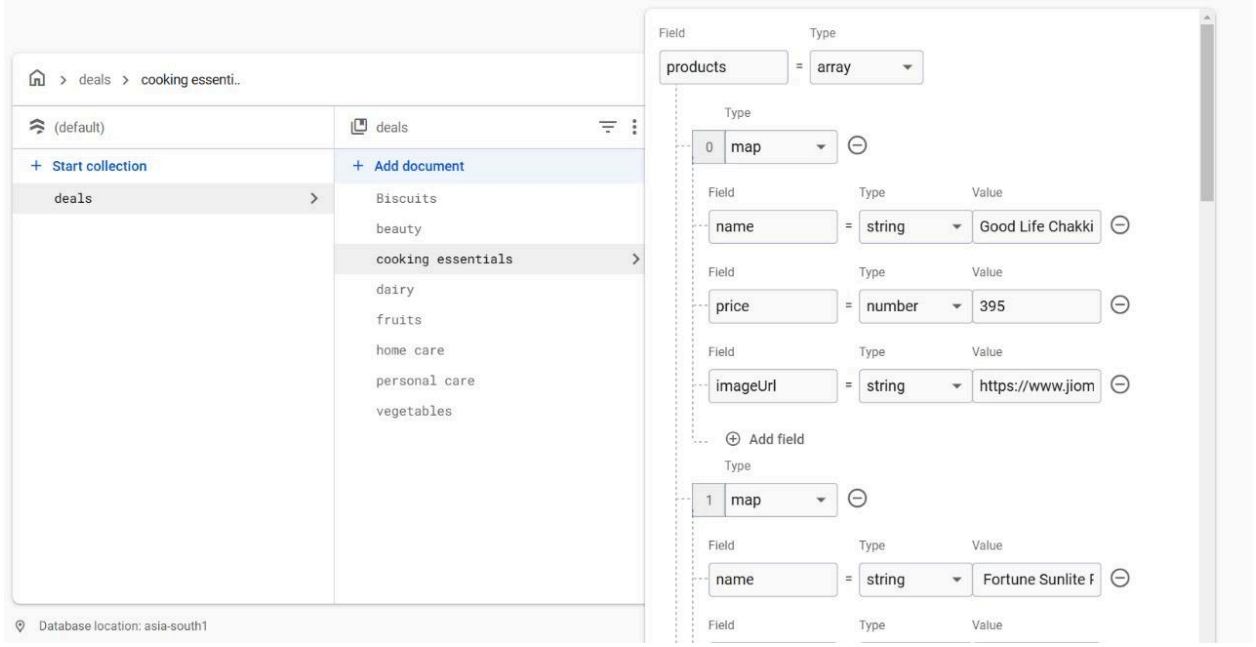
Then click on Create database. For location select asia-south1(Mumbai). Then choose **Start in test mode** (for development).

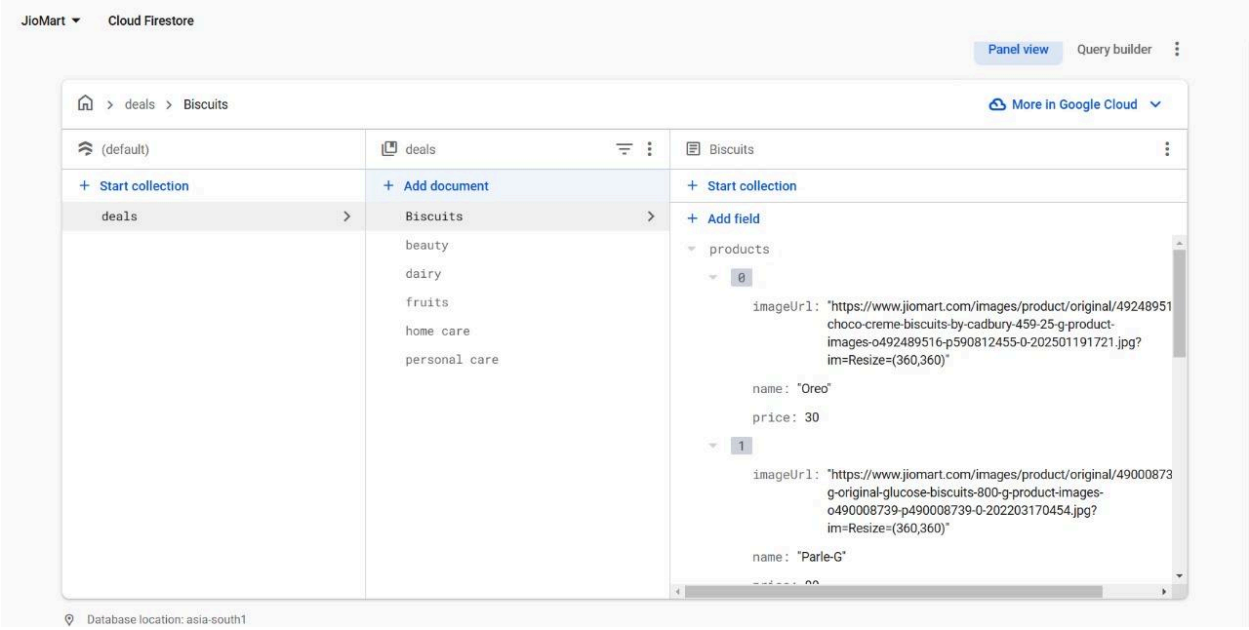
Click **Next** and choose a location, then **Enable**.





Step 2:
Start by creating a **collection**. Add a **document** within the collection. Define the **fields** as per your project requirements.





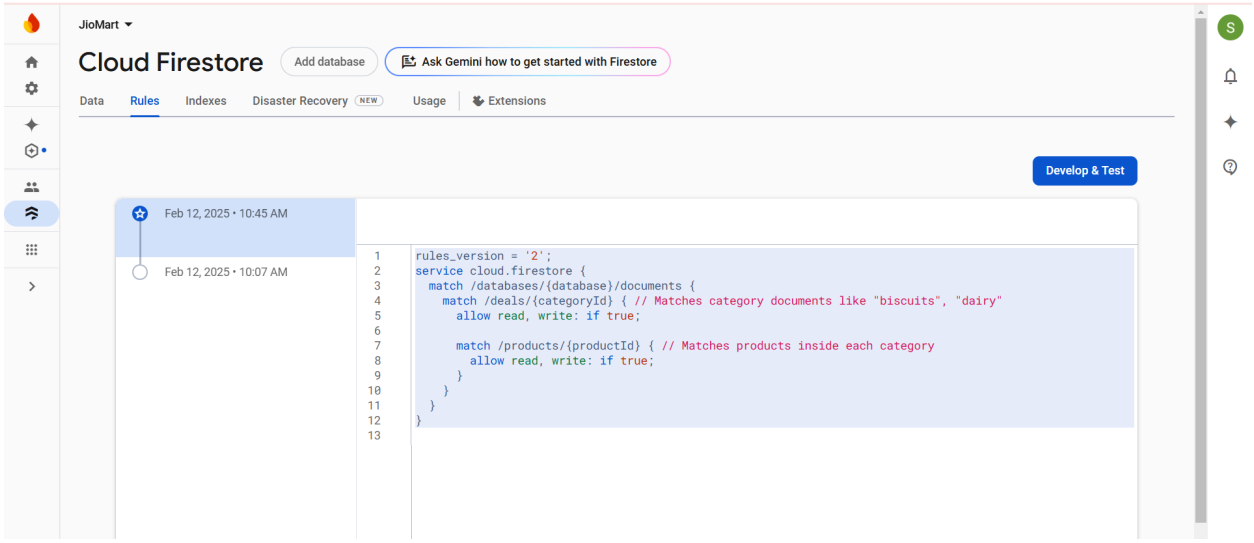
Step 3:
Run the following command to add Firestore to your Flutter app:

```
flutter pub add cloud_firestore
```

Step 4:
Go to Firebase Console → Firestore Database → Rules
Set the following rules:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /deals/{categoryId} { // Matches category documents like "biscuits", "dairy"
      allow read, write: if true;

      match /products/{productId} { // Matches products inside each category
        allow read, write: if true;
      }
    }
  }
}
```

and then click **Publish** to apply changes.



Code:

```
//category_list.dart
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import '../Screens/product_grid.dart';

class CategoryList extends StatelessWidget {
  final List<Map<String, String>> categories = [
    {"name": "Biscuits", "icon": "🍪"},
    {"name": "dairy", "icon": "🥛"},
    {"name": "fruits", "icon": "🍎"},
    {"name": "vegetables", "icon": "🥦"},
    {"name": "cooking essentials", "icon": "🧂"},
    {"name": "babycare", "icon": "🍼"},
    {"name": "stationery", "icon": "✏️"},
    {"name": "kitchenware", "icon": "🍽️"},
    {"name": "disposables", "icon": "🥡"},
  ];

  // Function to fetch products from Firestore
  Future<List<Map<String, dynamic>>> fetchProductsForCategory(String category, {bool
limitToThree = false}) async {
    List<Map<String, dynamic>> products = [];
    try {
      DocumentSnapshot categorySnapshot =
```

```
    await FirebaseFirestore.instance.collection("deals").doc(category).get();

    if (categorySnapshot.exists) {
      var categoryData = categorySnapshot.data() as Map<String, dynamic>;
      if (categoryData.containsKey('products')) {
        var productList = categoryData['products'] as List<dynamic>;
        int takeCount = limitToThree ? 3 : productList.length;

        for (var product in productList.take(takeCount)) {
          products.add({
            'name': product['name'],
            'price': product['price'],
            'imageUrl': product['imageUrl'],
          });
        }
      }
    }
  } catch (e) {
    print("Error fetching category products: $e");
  }
  return products;
 }

 // Quick Peek Modal (Shows 3 products in bottom sheet)
 void showQuickPeek(BuildContext context, String category) async {
  List<Map<String, dynamic>> products = await fetchProductsForCategory(category,
limitToThree: true);

  if (products.isEmpty) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text("No products available for $category")),
    );
    return;
  }

  showModalBottomSheet(
    context: context,
    shape: RoundedRectangleBorder(borderRadius: BorderRadius.vertical(top:
Radius.circular(20))),
    builder: (context) {
      return Container(
        padding: EdgeInsets.all(16),
        height: 280,
        child: Column(
```

```
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              "Top Picks in $category",
              style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
            ),
            SizedBox(height: 10),
            Expanded(
              child: ListView.separated(
                itemCount: products.length,
                separatorBuilder: (_, __) => Divider(),
                itemBuilder: (context, index) {
                  var product = products[index];
                  return ListTile(
                    leading: Image.network(product['imageUrl'], width: 50, height: 50, fit:
BoxFit.cover),
                    title: Text(product['name'], style: TextStyle(fontSize: 16)),
                    subtitle: Text("₹${product['price']}", style: TextStyle(color: Colors.green,
fontWeight: FontWeight.bold)),
                  );
                },
              ),
            ),
          ],
        ),
      );
    },
  );
 }

 @override
 Widget build(BuildContext context) {
   return Expanded(
     child: ListView.builder(
       itemCount: categories.length,
       itemBuilder: (context, index) {
         return GestureDetector(
           onTap: () async {
             List<Map<String, dynamic>> products = await
fetchProductsForCategory(categories[index]["name"]!);
             Navigator.push(
               context,
               MaterialPageRoute(
                 builder: (context) => ProductGrid(products: products),
```

```
    ),
  );
},
onLongPress: () {
  showQuickPeek(context, categories[index]["name"]!);
},
child: Container(
  margin: EdgeInsets.symmetric(horizontal: 12, vertical: 8),
  padding: EdgeInsets.all(16),
  decoration: BoxDecoration(
    gradient: LinearGradient(
      colors: [Colors.blueAccent, Colors.lightBlue.shade200],
      begin: Alignment.topLeft,
      end: Alignment.bottomRight,
    ),
    borderRadius: BorderRadius.circular(15),
    boxShadow: [
      BoxShadow(
        color: Colors.black26,
        blurRadius: 6,
        spreadRadius: 2,
        offset: Offset(2, 4),
      ),
    ],
  ),
  child: Row(
    children: [
      Text(
        categories[index]["icon"]!,
        style: TextStyle(fontSize: 30),
      ),
      SizedBox(width: 15),
      Expanded(
        child: Text(
          categories[index]["name"]!.toUpperCase(),
          style: TextStyle(
            color: Colors.white,
            fontSize: 18,
            fontWeight: FontWeight.bold,
          ),
        ),
      ),
      Icon(Icons.arrow_forward_ios, color: Colors.white, size: 20),
    ],
```

```
            ),
          ),
        );
      },
    ),
  );
 }
}

//product_grid.dart


import 'package:flutter/material.dart';
import 'cart_page.dart'; // Import cart_page.dart


class ProductGrid extends StatefulWidget {
 final List<Map<String, dynamic>> products;


 ProductGrid({required this.products});


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


class _ProductGridState extends State<ProductGrid> {
 Map<String, int> cartQuantities = {};


 @override
 void initState() {
  super.initState();
  // Initialize quantities from existing cart items
  for (var item in cartItems) {
   cartQuantities[item['name']] = item['quantity'];
  }
 }


 // Function to add/update product in cart
 void _updateCart(Map<String, dynamic> product, int quantity) {
```

```
  if (quantity <= 0) {
    // Remove product from cart if quantity is 0
    setState(() {
      cartItems.removeWhere((item) => item['name'] == product['name']);
      cartQuantities[product['name']] = 0;
    });
    return;
  }


  // Check if product already exists in cart
  int existingIndex = cartItems.indexWhere((item) => item['name'] == product['name']);


  setState(() {
    if (existingIndex >= 0) {
      // Update existing product quantity
      cartItems[existingIndex]['quantity'] = quantity;
    } else {
      // Add new product to cart
      cartItems.add({
        'name': product['name'],
        'price': product['price'],
        'imageUrl': product['imageUrl'],
        'quantity': quantity,
      });
    }
    // Update local quantity tracker
    cartQuantities[product['name']] = quantity;
  });
}


@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text("Products"),
      backgroundColor: Colors.blueAccent,
      actions: [
        Stack(
          alignment: Alignment.center,
          children: [
            IconButton(
```

```
            icon: Icon(Icons.shopping_cart, color: Colors.white),
            onPressed: () {
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) => CartPage()),
              ).then((_) {
                // Refresh quantities when returning from cart page
                setState(() {
                  for (var product in widget.products) {
                    String productName = product['name'];
                    int cartIndex = cartItems.indexWhere((item) => item['name'] ==
productName);
                    if (cartIndex >= 0) {
                      cartQuantities[productName] = cartItems[cartIndex]['quantity'];
                    } else {
                      cartQuantities[productName] = 0;
                    }
                  }
                });
              });
            },
          ),
          if (cartItems.isNotEmpty)
            Positioned(
              right: 8,
              top: 8,
              child: Container(
                padding: EdgeInsets.all(2),
                decoration: BoxDecoration(
                  color: Colors.red,
                  borderRadius: BorderRadius.circular(10),
                ),
                constraints: BoxConstraints(
                  minWidth: 16,
                  minHeight: 16,
                ),
                child: Text(
                  '${cartItems.length}',
                  style: TextStyle(
                    color: Colors.white,
                    fontSize: 10,
                  ),
                  textAlign: TextAlign.center,
                ),
```

```
            ),
          ),
        ],
      ),
    ],
  ),
  body: Padding(
    padding: const EdgeInsets.all(8.0),
    child: GridView.builder(
      gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
        crossAxisCount: 2,
        childAspectRatio: 0.7,
        crossAxisSpacing: 10,
        mainAxisSpacing: 10,
      ),
      itemCount: widget.products.length,
      itemBuilder: (context, index) {
        var product = widget.products[index];
        String productName = product['name'];
        cartQuantities.putIfAbsent(productName, () => 0); // Default quantity 0


        return Card(
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(10),
          ),
          elevation: 3,
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Expanded(
                child: Padding(
                  padding: const EdgeInsets.all(8.0),
                  child: Image.network(
                    product['imageUrl'],
                    fit: BoxFit.cover,
                  ),
                ),
              ),
              Text(
                productName,
                style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold),
              ),
              Text(
```

```
                  "₹${product['price']}",
                  style: TextStyle(fontSize: 14, color: Colors.green),
                ),
                SizedBox(height: 5),


                // Quantity Controller (Increase & Decrease Buttons)
                Row(
                  mainAxisAlignment: MainAxisAlignment.center,
                  children: [
                    IconButton(
                      icon: Icon(Icons.remove, color: Colors.red),
                      onPressed: () {
                        if (cartQuantities[productName]! > 0) {
                          _updateCart(product, cartQuantities[productName]! - 1);
                        }
                      },
                    ),
                    Text(
                      "${cartQuantities[productName]}",
                      style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold),
                    ),
                    IconButton(
                      icon: Icon(Icons.add, color: Colors.blue),
                      onPressed: () {
                        _updateCart(product, cartQuantities[productName]! + 1);
                      },
                    ),
                  ],
                ),
              ],
            ),
          );
        },
      ),
    ),
  );
 }
}

//Cart_page.dart

import 'package:flutter/material.dart';
import '../screens/CheckoutPage.dart'; // Import CheckoutPage
```

```
// Global cart list to store cart items
List<Map<String, dynamic>> cartItems = [];

class CartPage extends StatefulWidget {
  const CartPage({Key? key}) : super(key: key);

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

class _CartPageState extends State<CartPage> {
  // Function to remove an item from cart
  void _removeItem(int index) {
    setState(() {
      cartItems.removeAt(index);
    });
  }

  // Function to update quantity
  void _updateQuantity(int index, int change) {
    setState(() {
      cartItems[index]['quantity'] += change;
      if (cartItems[index]['quantity'] <= 0) {
        cartItems.removeAt(index);
      }
    });
  }

  // Function to clear cart
  void _clearCart() {
    setState(() {
      cartItems.clear();
    });
  }

  // Function to calculate total price
  double _calculateTotal() {
    return cartItems.fold(0, (sum, item) => sum + (item['price'] * item['quantity']));
  }

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

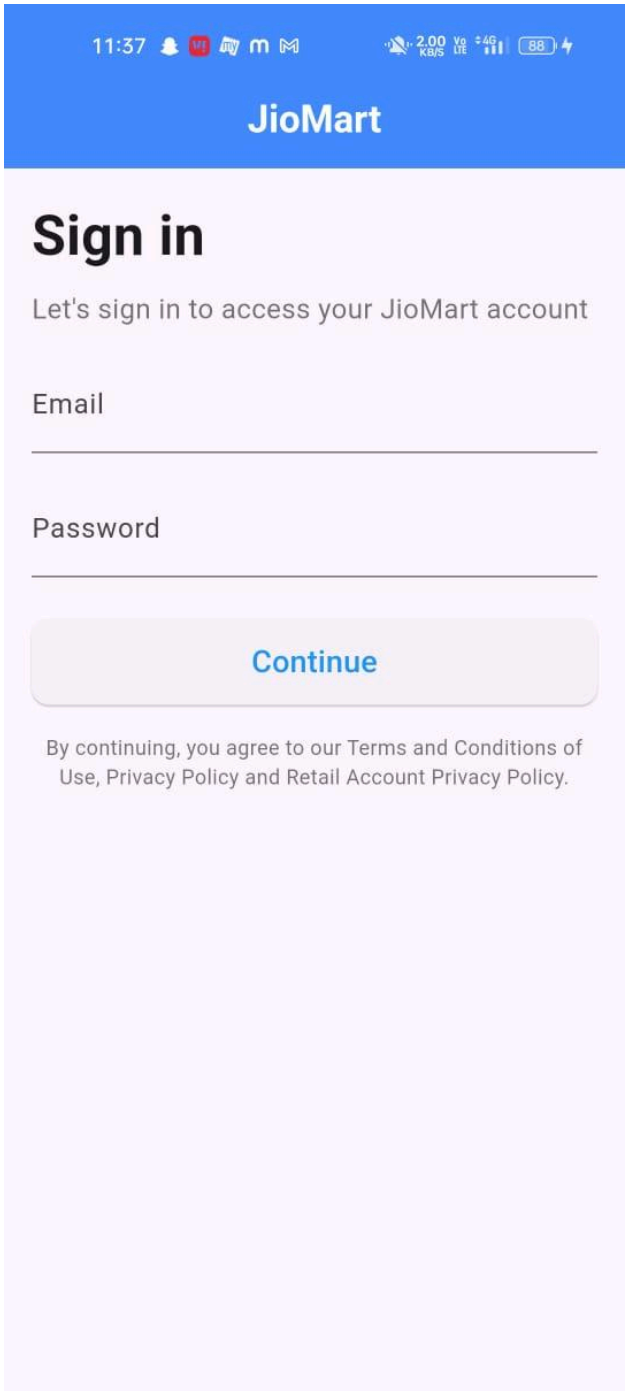
```
    appBar: AppBar(
      backgroundColor: Colors.blueAccent,
      title: const Text(
        "My Cart",
        style: TextStyle(color: Colors.black, fontWeight: FontWeight.bold),
      ),
      iconTheme: const IconThemeData(color: Colors.black),
      actions: [
        if (cartItems.isNotEmpty)
          IconButton(
            icon: const Icon(Icons.delete, color: Colors.black),
            onPressed: _clearCart,
          ),
      ],
    ),
    body: cartItems.isEmpty
        ? const Center(
      child: Text("Your cart is empty!", style: TextStyle(fontSize: 18, fontWeight:
FontWeight.bold)),
    )
        : Column(
      children: [
        Expanded(
          child: ListView.builder(
            itemCount: cartItems.length,
            itemBuilder: (context, index) {
              return _buildCartItem(index);
            },
          ),
        ),
        _buildTotalSection(),
      ],
    ),
  );
 }

 Widget _buildCartItem(int index) {
  var item = cartItems[index];
  return Padding(
    padding: const EdgeInsets.symmetric(horizontal: 10, vertical: 5),
    child: Card(
      shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(10)),
      elevation: 2,
      child: Padding(
```

```
padding: const EdgeInsets.all(8.0),
child: Row(
 children: [
  ClipRRect(
   borderRadius: BorderRadius.circular(10),
   child: Image.network(
    item['imageUrl'],
    width: 80,
    height: 80,
    fit: BoxFit.cover,
    errorBuilder: (context, error, stackTrace) =>
       Image.asset("assets/fallback_image.png", width: 80, height: 80),
   ),
  ),
  const SizedBox(width: 10),
  Expanded(
   child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
     Text(
      item['name'],
      style: const TextStyle(fontSize: 16, fontWeight: FontWeight.bold),
      overflow: TextOverflow.ellipsis,
     ),
     const SizedBox(height: 5),
     Text(
      "₹${item['price']}",
      style: const TextStyle(fontSize: 14, color: Colors.green),
     ),
     const SizedBox(height: 5),
     Row(
      children: [
       IconButton(
        icon: const Icon(Icons.remove_circle, color: Colors.red),
        onPressed: () => _updateQuantity(index, -1),
       ),
       Text(
        "${item['quantity']}",
        style: const TextStyle(fontSize: 16, fontWeight: FontWeight.bold),
       ),
       IconButton(
        icon: const Icon(Icons.add_circle, color: Colors.green),
        onPressed: () => _updateQuantity(index, 1),
       ),
```

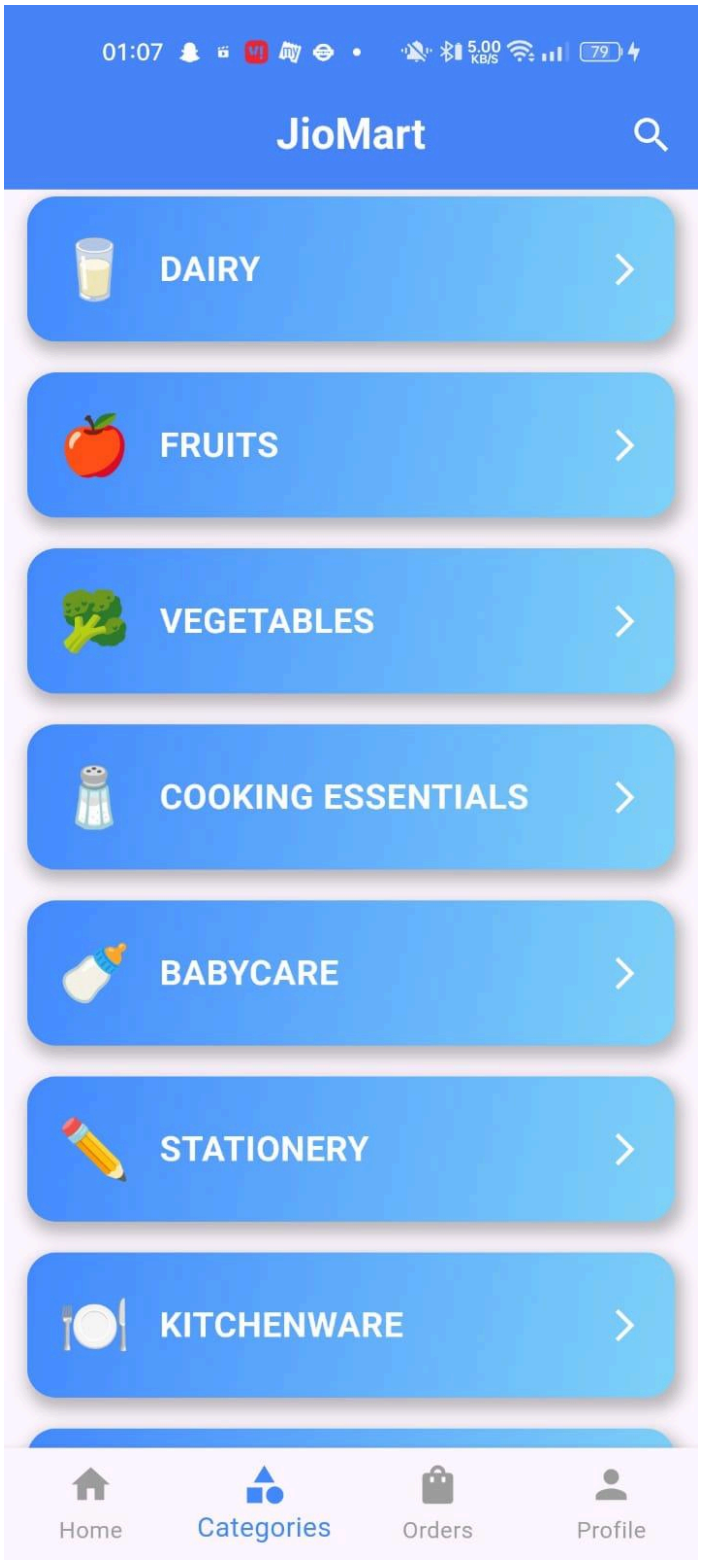
```
                  ],
                ),
              ],
            ),
          ),
          IconButton(
            icon: const Icon(Icons.delete, color: Colors.red),
            onPressed: () => _removeItem(index),
          ),
        ],
      ),
    ),
  ),
 );
}

 Widget _buildTotalSection() {
  return Container(
   padding: const EdgeInsets.all(16.0),
   decoration: BoxDecoration(
    color: Colors.white,
    boxShadow: [
     BoxShadow(color: Colors.black12, blurRadius: 5, spreadRadius: 2),
    ],
   ),
   child: Column(
    children: [
     Row(
      mainAxisAlignment: MainAxisAlignment.spaceBetween,
      children: [
       const Text(
        "Total:",
        style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
       ),
       Text(
        "₹${_calculateTotal().toStringAsFixed(2)}",
        style: const TextStyle(fontSize: 18, fontWeight: FontWeight.bold, color:
Colors.green),
       ),
      ],
     ),
     const SizedBox(height: 10),
     SizedBox(
      width: double.infinity,
```

```
            child: ElevatedButton(
              style: ElevatedButton.styleFrom(
                backgroundColor: Colors.blue,
                shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(30)),
              ),
              onPressed: () {
                // Navigate to CheckoutPage
                Navigator.push(
                  context,
                  MaterialPageRoute(builder: (context) => const CheckoutPage()),
                );
              },
              child: const Padding(
                padding: EdgeInsets.symmetric(vertical: 12),
                child: Text(
                  "Proceed to Checkout",
                  style: TextStyle(fontSize: 16, color: Colors.white),
                ),
              ),
            ),
          ),
        ],
      ),
    );
  }
}
```

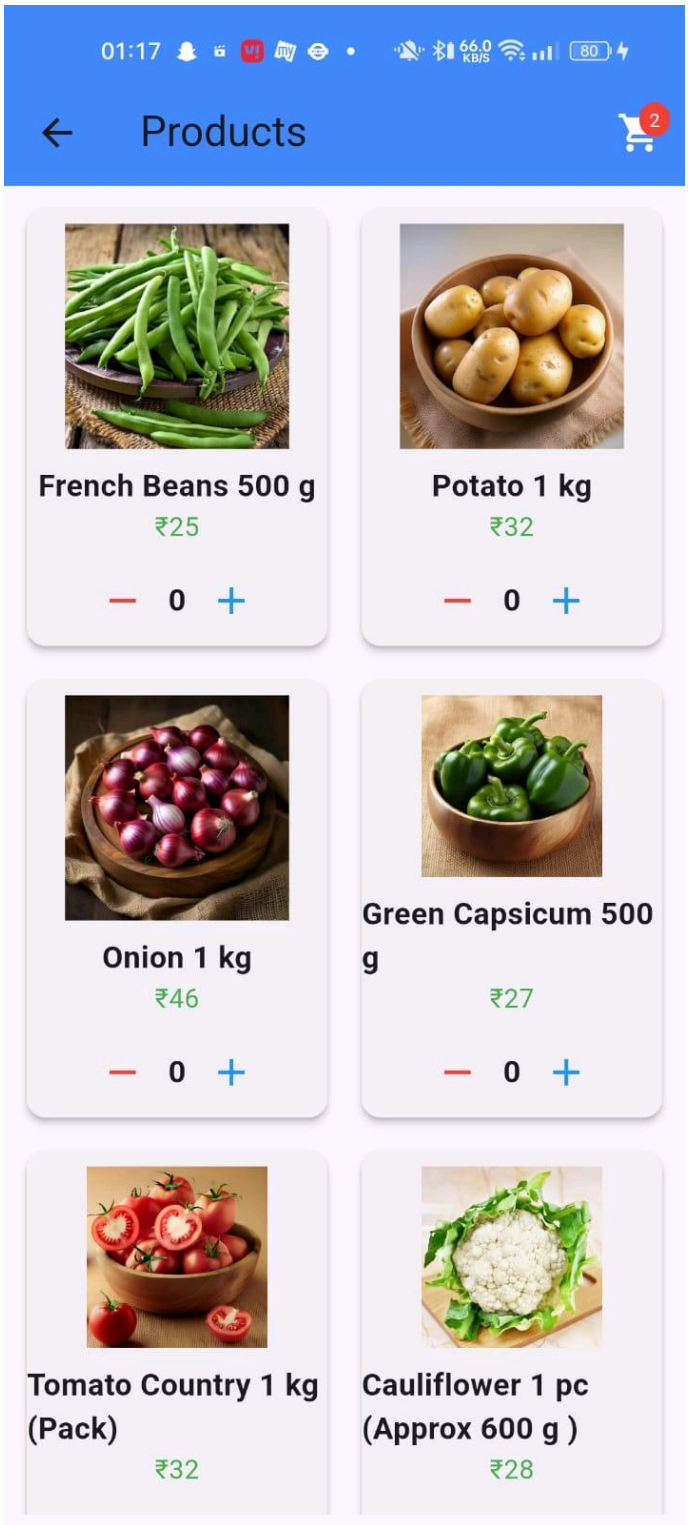
Output:
Login Page:



Category Page:



Product List Page:



01:16 16.0 KB/S 80

Products

2

Amul Pasteurised Butter 100 g

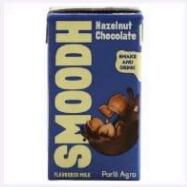
₹56

− 0 +

Amul Taaza Toned Milk 1 L

₹71

− 0 +



Hazelnut Chocolate Flavoured Milk 80 ml

₹10

− 0 +

Amul Fresh Cream 250 ml

₹64

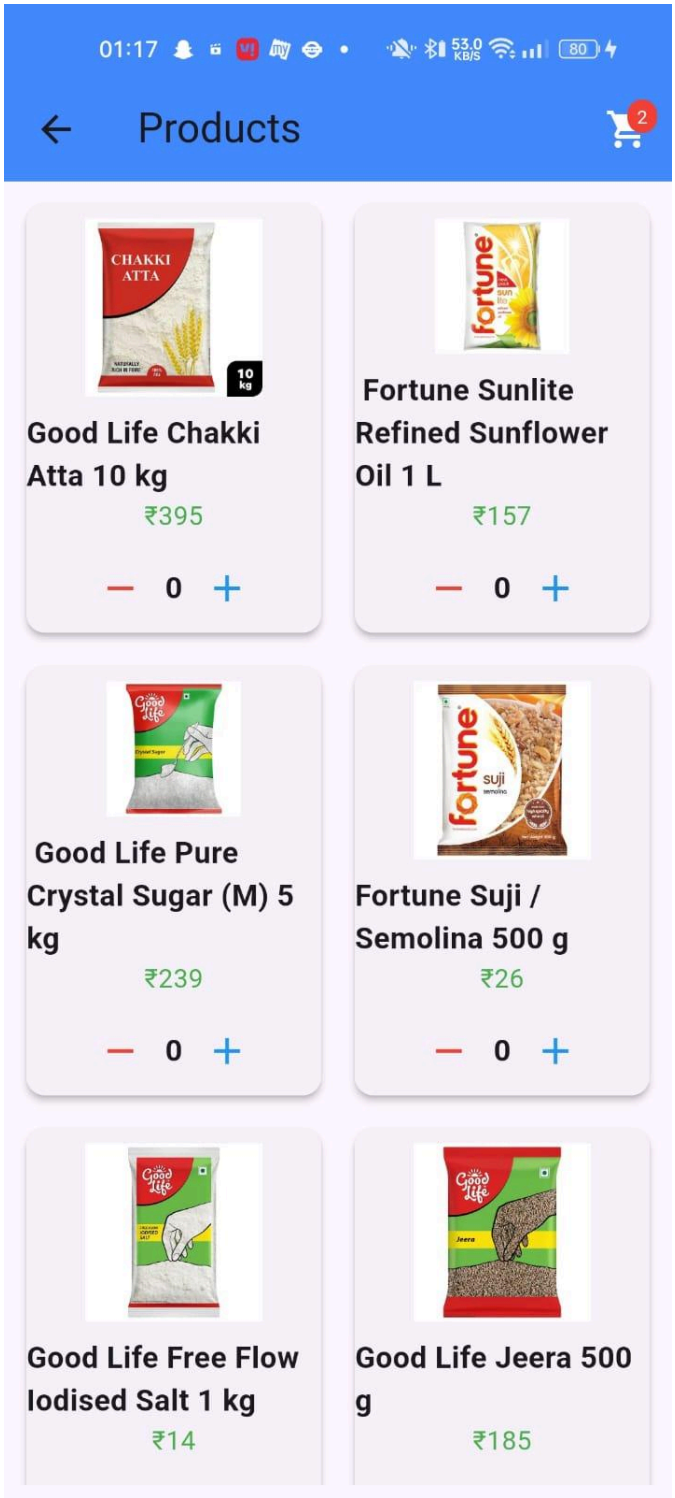
− 0 +

Amul Masti Spiced Buttermilk 200 ml

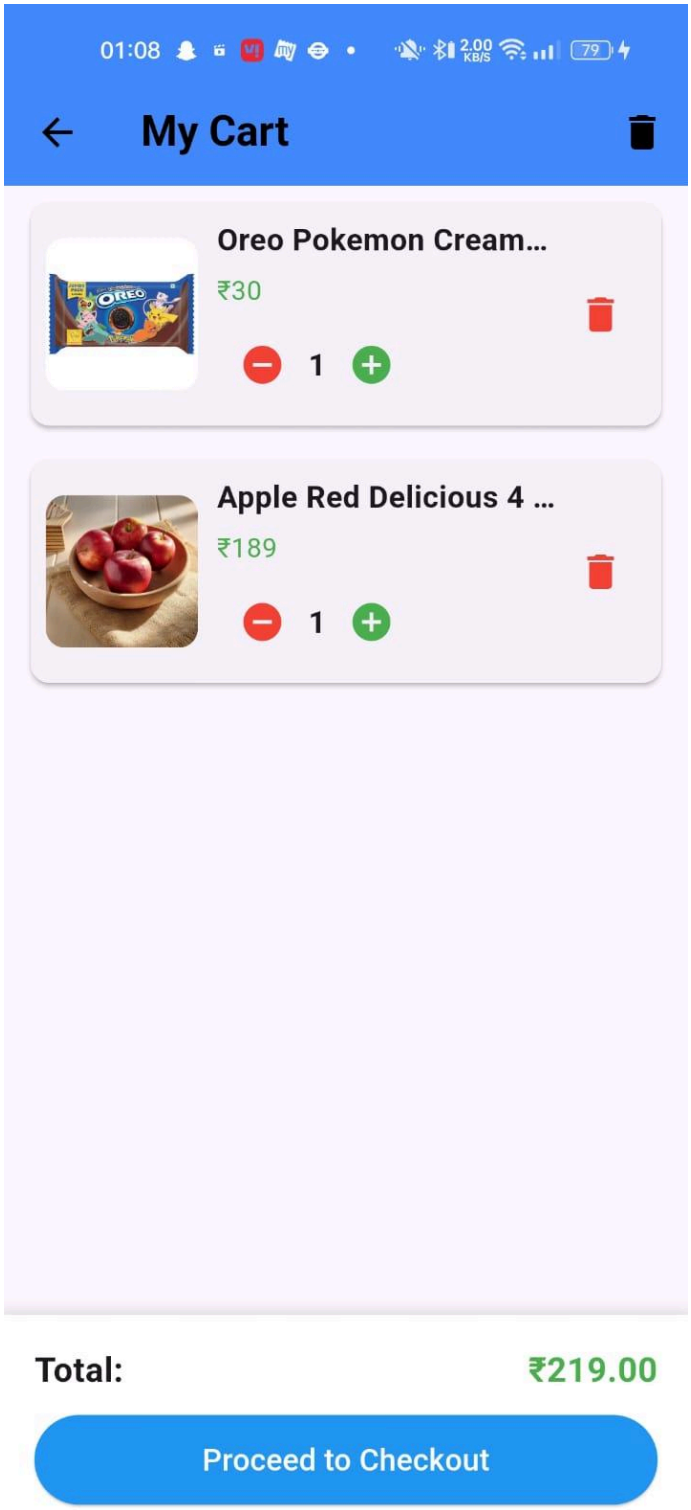
₹14

Amul Masti Dahi 200 g

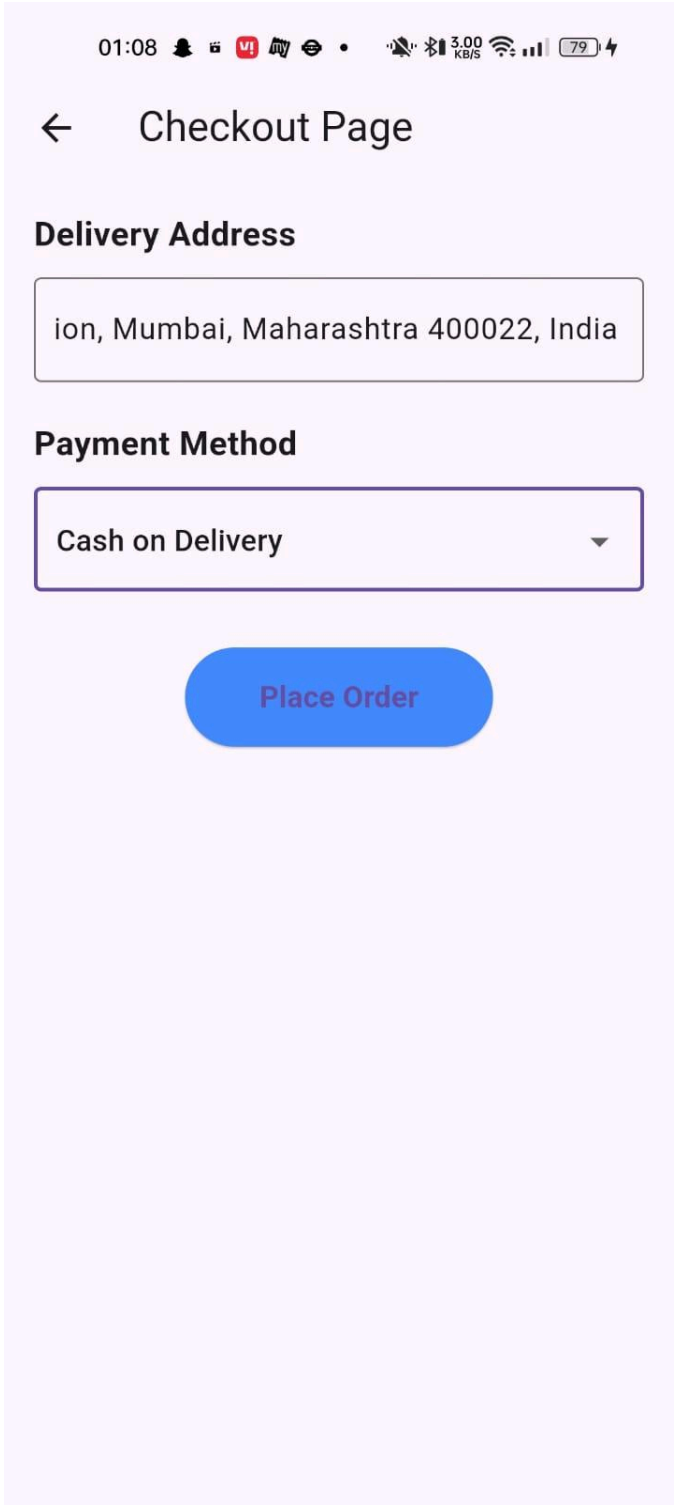
₹22


01:17
53.0 KB/S
80
Products
2
CHAKKI ATTA
10 kg
Good Life Chakki Atta 10 kg
₹395
0
fortune
Fortune Sunlite Refined Sunflower Oil 1 L
₹157
0
Good Life Pure Crystal Sugar (M) 5 kg
₹239
0
fortune
suji
Fortune Suji / Semolina 500 g
₹26
0
Good Life Free Flow Iodised Salt 1 kg
₹14
Good Life
Jeera
Good Life Jeera 500 g
₹185

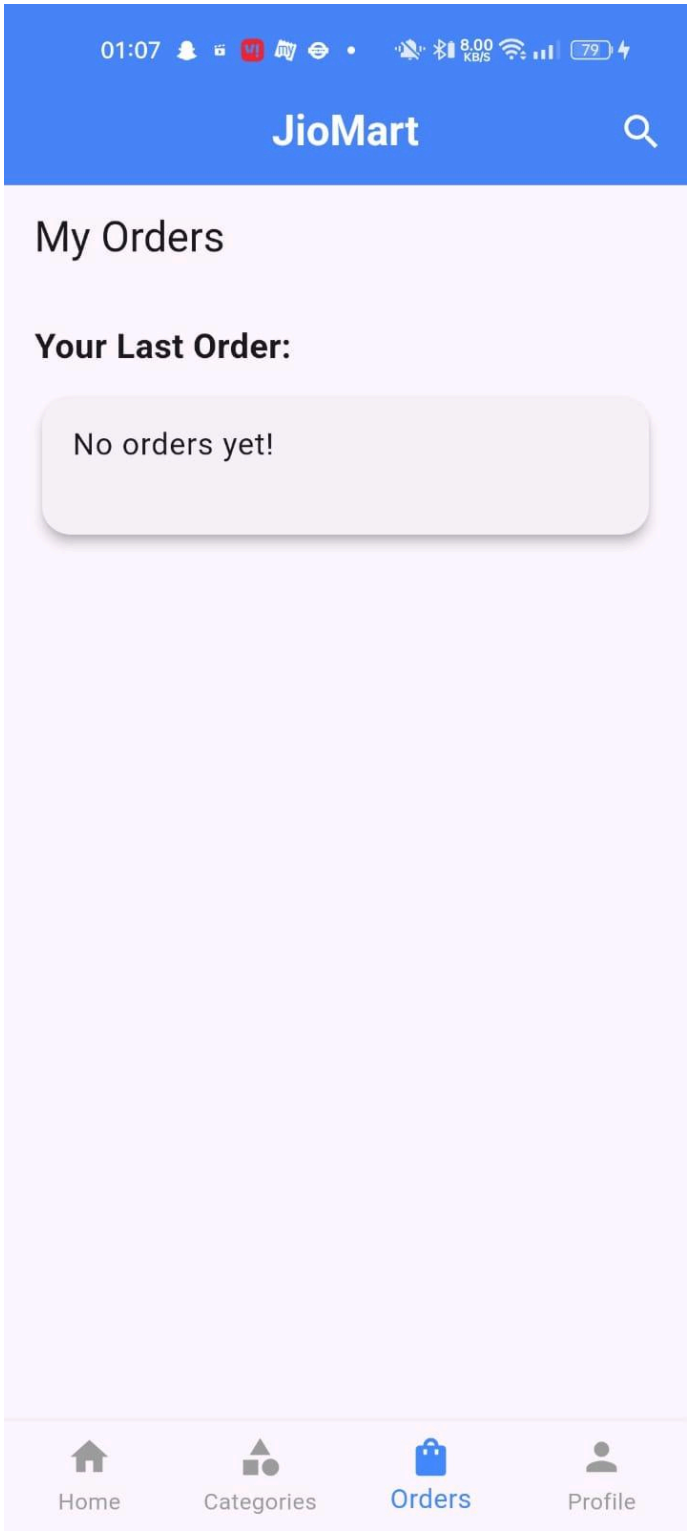
Cart Page:


01:08
My Cart
Oreo Pokemon Cream...
₹30
1
Apple Red Delicious 4 ...
₹189
1
Total:
₹219.00
Proceed to Checkout

Checkout page:



Orders Page:


01:07
JioMart
My Orders
Your Last Order:
No orders yet!
Home
Categories
Orders
Profile

Unique Feature:"Quick Peek" (Instant Product Preview)

Instead of navigating to a new screen, users can long-press a category to instantly preview a few top products in a small pop-up (bottom sheet).

How It Works?

- **Tap on a category** → Goes to the full product grid (as before).
- **Long-press on a category** → Opens a Quick Peek (bottom sheet) showing the top 3 products from that category without leaving the page.