Shilpa Pandey
42 D15A

Experiment 6

Aim: To Connect Flutter UI with firebase database

Theory:

Firebase is a Backend-as-a-Service (BaaS) app development platform that provides hosted backend services such as a realtime database, cloud storage, authentication, crash reporting, machine learning, remote configuration, and hosting for your static files.
Firebase is a Backend-as-a-Service (Baas) that provides developers with a variety of tools and services to help them:

- Real-time database: Stores and syncs data in real time in apps
- Cloud messaging: Used to send messages to users' devices in apps
- Authentication: Used to authenticate users in apps
- Database: Stores and syncs data in real-time, sometimes referred to as a real-time document store

Code:

Authentication.dart

```
import 'package:amazon_clone/model/user_details_model.dart';
import 'package:amazon_clone/resources/cloudfirestore_methods.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';

class AuthenticationMethods {
  FirebaseAuth firebaseAuth = FirebaseAuth.instance;
  CloudFirestoreClass cloudFirestoreClass = CloudFirestoreClass();

  Future<String> signUpUser(
      {required String name,
       required String address,
       required String email,
       required String password}) async {
    name.trim();
    address.trim();
    email.trim();
    password.trim();
    String output = "Something went wrong";
    if (name != "" && address != "" && email != "" && password != "") {
      try {
        await firebaseAuth.createUserWithEmailAndPassword(
            email: email, password: password);
        UserDetailsModel user = UserDetailsModel(name: name, address:
address);
        await cloudFirestoreClass.uploadNameAndAddressToDatabase(user:
user);
        output = "success";
      } on FirebaseAuthException catch (e) {
        output = e.message.toString();
      }
    } else {
      output = "Please fill up all the fields.";
    }
    return output;
  }

  Future<String> signInUser(
      {required String email, required String password}) async {
    email.trim();
    password.trim();
    String output = "Something went wrong";
    if (email != "" && password != "") {
      try {
        await firebaseAuth.signInWithEmailAndPassword(
            email: email, password: password);
        output = "success";
      } on FirebaseAuthException catch (e) {
        output = e.message.toString();
      }
    } else {
      output = "Please fill up all the fields.";
    }
    return output;
  }
}
```

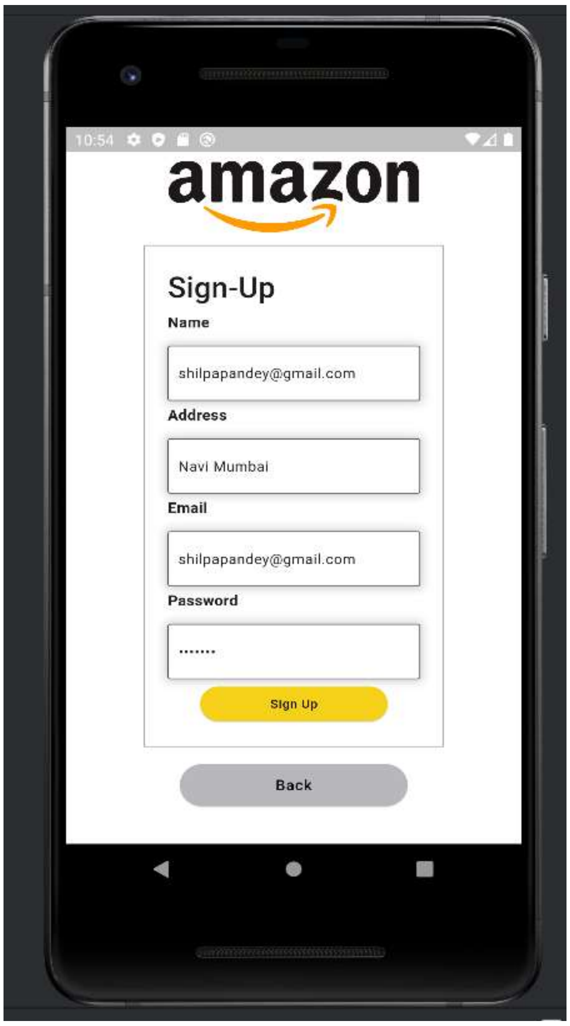
cloudFirestore.dart

```
import 'dart:typed_data';
import 'package:amazon_clone/model/order_request_model.dart';
import 'package:amazon_clone/model/product_model.dart';
import 'package:amazon_clone/model/review_model.dart';
import 'package:amazon_clone/model/user_details_model.dart';
import 'package:amazon_clone/utils/utils.dart';
import 'package:amazon_clone/widgets/simple_product_widget.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_storage/firebase_storage.dart';
import 'package:flutter/cupertino.dart';

class CloudFirestoreClass {
  FirebaseFirestore firebaseFirestore = FirebaseFirestore.instance;
  FirebaseAuth firebaseAuth = FirebaseAuth.instance;

  Future uploadNameAndAddressToDatabase(
      {required UserDetailsModel user}) async {
    await firebaseFirestore
        .collection("users")
        .doc(firebaseAuth.currentUser!.uid)
        .set(user.getJson());
  }

  Future getNameAndAddress() async {
    DocumentSnapshot snap = await firebaseFirestore
        .collection("users")
        .doc(firebaseAuth.currentUser!.uid)
        .get();

    UserDetailsModel userModel = UserDetailsModel.getModelFromJson(
      (snap.data() as dynamic),
    );

    return userModel;
  }

  Future<String> uploadProductToDatabase({
    required Uint8List? image,
    required String productName,
    required String rawCost,
    required int discount,
  required String sellerName,
  required String sellerUid,
}) async {
  productName.trim();
  rawCost.trim();
  String output = "Something went wrong";

  if (image != null && productName != "" && rawCost != "") {
    try {
      String uid = Utils().getUid();
      String url = await uploadImageToDatabase(image: image, uid: uid);
      double cost = double.parse(rawCost);
      cost = cost - (cost * (discount / 100));
      ProductModel product = ProductModel(
          url: url,
          productName: productName,
          cost: cost,
          discount: discount,
          uid: uid,
          sellerName: sellerName,
          sellerUid: sellerUid,
          rating: 5,
          noOfRating: 0);

      await firebaseFirestore
          .collection("products")
          .doc(uid)
          .set(product.getJson());
      output = "success";
    } catch (e) {
      output = e.toString();
    }
  } else {
    output = "Please make sure all the fields are not empty";
  }

  return output;
}

Future<String> uploadImageToDatabase(
    {required Uint8List image, required String uid}) async {
  Reference storageRef =
  FirebaseStorage.instance.ref().child("products").child(uid);
  UploadTask uploadToask = storageRef.putData(image);
  TaskSnapshot task = await uploadToask;
  return task.ref.getDownloadURL();
}

Future<List<Widget>> getProductsFromDiscount(int discount) async {
  List<Widget> children = [];
  QuerySnapshot<Map<String, dynamic>> snap = await firebaseFirestore
      .collection("products")
      .where("discount", isEqualTo: discount)
      .get();

  for (int i = 0; i < snap.docs.length; i++) {
    DocumentSnapshot docSnap = snap.docs[i];
    ProductModel model =
    ProductModel.getModelFromJson(json: (docSnap.data() as dynamic));
    children.add(SimpleProductWidget(productModel: model));
  }
  return children;
}
```

```
  Future uploadReviewToDatabase(
      {required String productUid, required ReviewModel model}) async {
    await firebaseFirestore
        .collection("products")
        .doc(productUid)
        .collection("reviews")
        .add(model.getJson());
    await changeAverageRating(productUid: productUid, reviewModel:
model);
  }

  Future addProductToCart({required ProductModel productModel}) async {
    await firebaseFirestore
        .collection("users")
        .doc(firebaseAuth.currentUser!.uid)
        .collection("cart")
        .doc(productModel.uid)
        .set(productModel.getJson());
  }

  Future deleteProductFromCart({required String uid}) async {
    await firebaseFirestore
        .collection("users")
        .doc(firebaseAuth.currentUser!.uid)
        .collection("cart")
        .doc(uid)
        .delete();
  }

  Future buyAllItemsInCart({required UserDetailsModel userDetails}) async
{
    QuerySnapshot<Map<String, dynamic>> snapshot = await
firebaseFirestore
        .collection("users")
        .doc(firebaseAuth.currentUser!.uid)
        .collection("cart")
        .get();

    for (int i = 0; i < snapshot.docs.length; i++) {
      ProductModel model =
      ProductModel.getModelFromJson(json: snapshot.docs[i].data());
      addProductToOrders(model: model, userDetails: userDetails);
      await deleteProductFromCart(uid: model.uid);
    }
  }

  Future addProductToOrders(
      {required ProductModel model,
        required UserDetailsModel userDetails}) async {
    await firebaseFirestore
        .collection("users")
        .doc(firebaseAuth.currentUser!.uid)
        .collection("orders")
        .add(model.getJson());
    await sendOrderRequest(model: model, userDetails: userDetails);
  }

  Future sendOrderRequest(
      {required ProductModel model,
        required UserDetailsModel userDetails}) async {
    OrderRequestModel orderRequestModel = OrderRequestModel(
        orderName: model.productName, buyersAddress: userDetails.address);
    await firebaseFirestore
```

```
      .collection("users")
      .doc(model.sellerUid)
      .collection("orderRequests")
      .add(orderRequestModel.getJson());
 }

 Future changeAverageRating(
    {required String productUid, required ReviewModel reviewModel})
async {
   DocumentSnapshot snapshot =
   await firebaseFirestore.collection("products").doc(productUid).get();
   ProductModel model =
   ProductModel.getModelFromJson(json: (snapshot.data() as dynamic));
   int currentRating = model.rating;
   int newRating = ((currentRating + reviewModel.rating) / 2).toInt();
   await firebaseFirestore
      .collection("products")
      .doc(productUid)
      .update({"rating": newRating});
 }
}
```

Output:

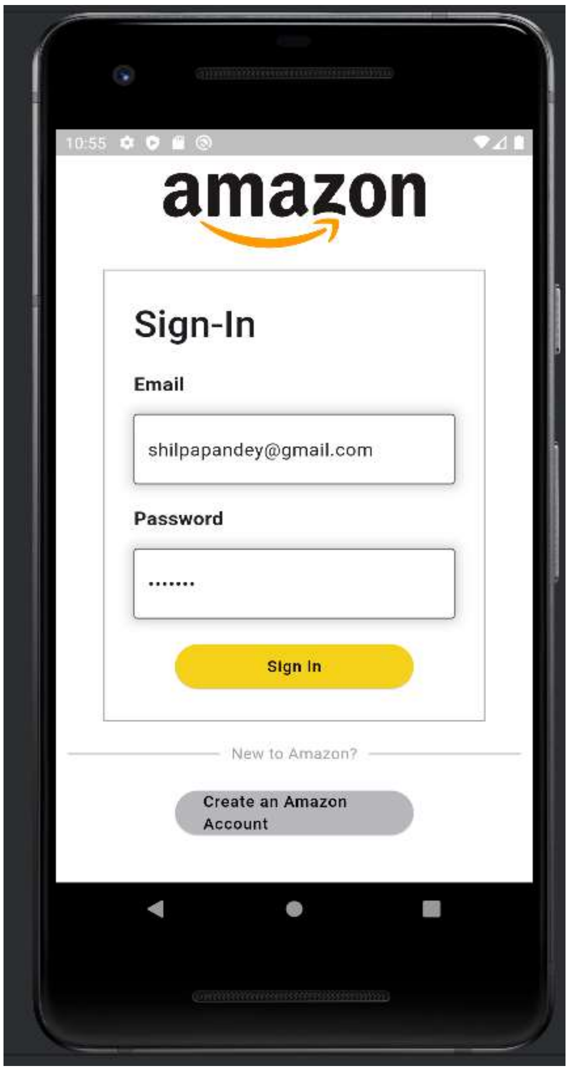

Sign-Up

Name: shilpapandey@gmail.com

Address: Navi Mumbai

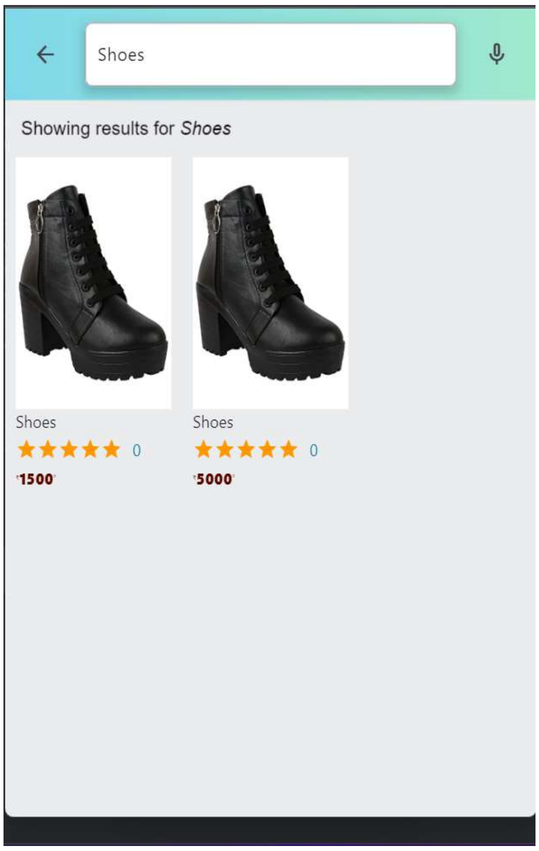
Email: shilpapandey@gmail.com

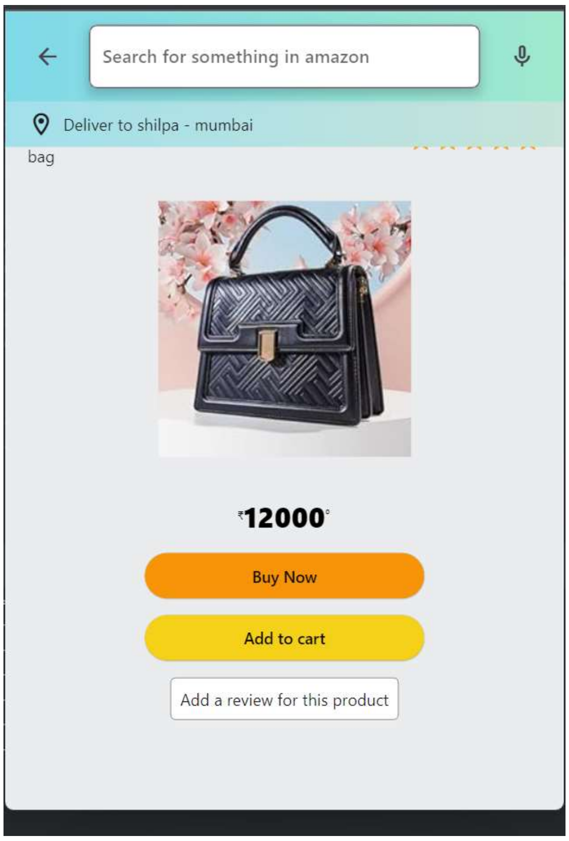
Password: •••••••

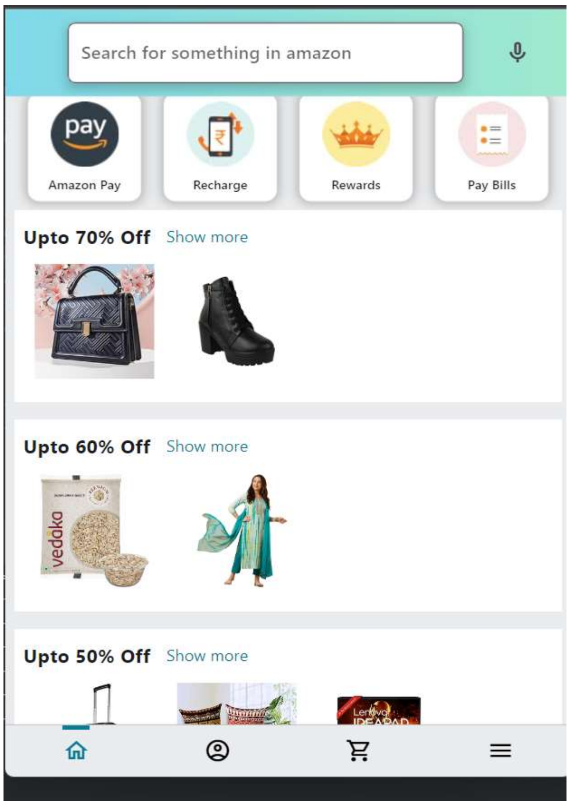

Sign-In

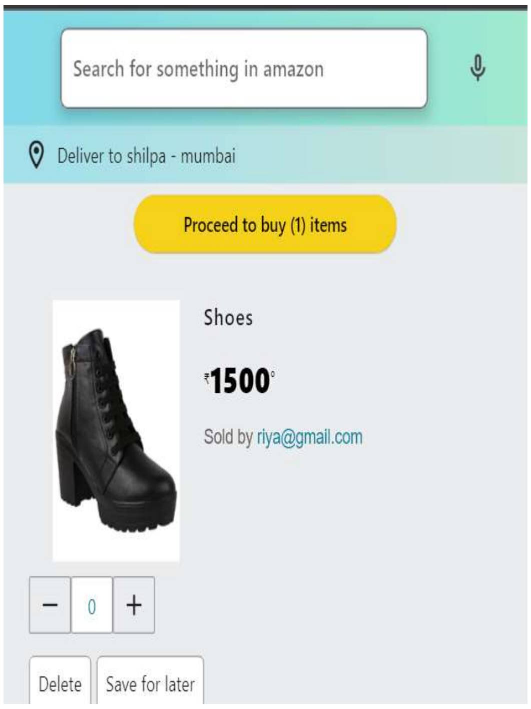
Email: shilpapandey@gmail.com

Password: •••••••

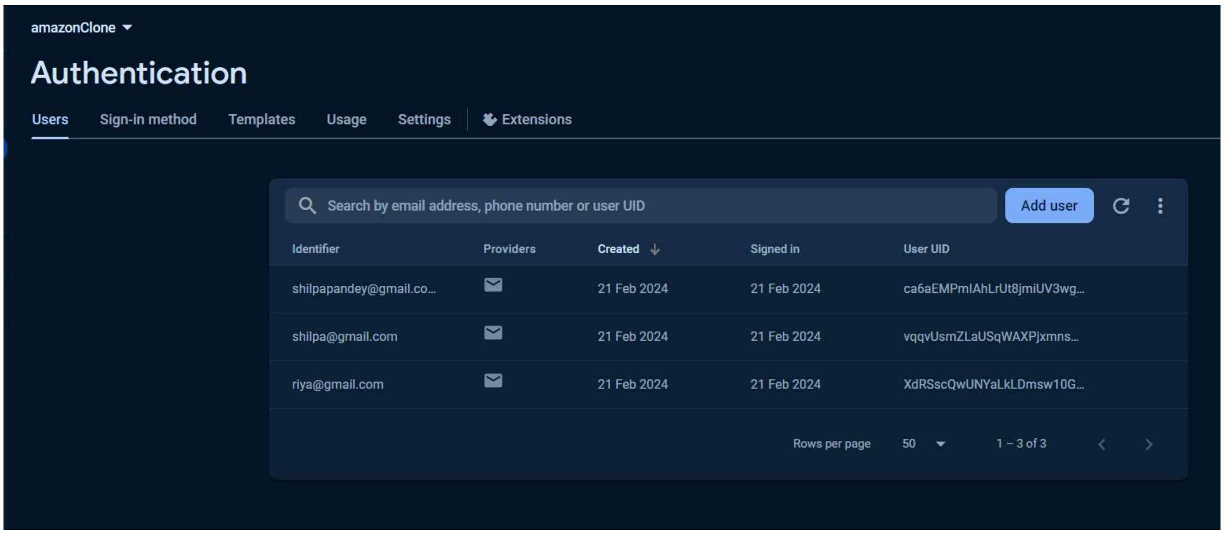

Shoes
Showing results for *Shoes*
Shoes
0
1500
Shoes
0
5000


Search for something in amazon
Deliver to shilpa - mumbai
bag
12000
Buy Now
Add to cart
Add a review for this product


Search for something in amazon
Amazon Pay
Recharge
Rewards
Pay Bills
Upto 70% Off
Show more
Upto 60% Off
Show more
Upto 50% Off
Show more


Search for something in amazon
Deliver to shilpa - mumbai
Proceed to buy (1) items
Shoes
1500
Sold by riya@gmail.com
0
Delete
Save for later

Firebase Authentication:

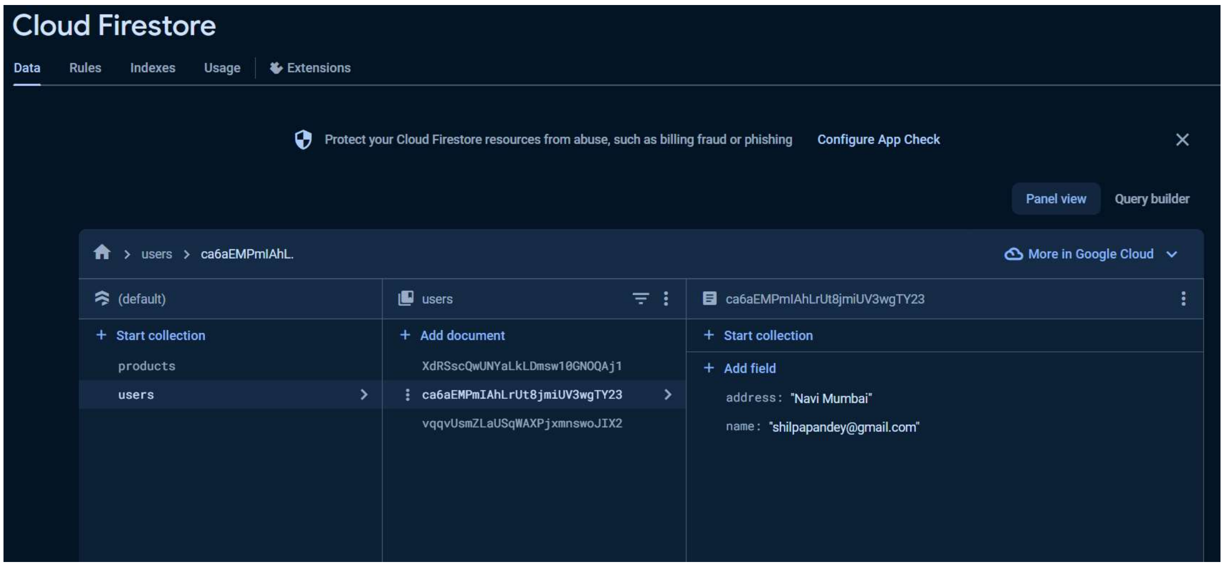

amazonClone

Authentication

Identifier	Providers	Created	Signed in	User UID
shilpapandey@gmail.co...	✉	21 Feb 2024	21 Feb 2024	ca6aEMPmIAhLrUt8jmiUV3wg...
shilpa@gmail.com	✉	21 Feb 2024	21 Feb 2024	vqqvUsmZLaUSqWAXPjxmns...
riya@gmail.com	✉	21 Feb 2024	21 Feb 2024	XdRSscQwUNYaLkLDmsw10G...

Rows per page 50 1 – 3 of 3

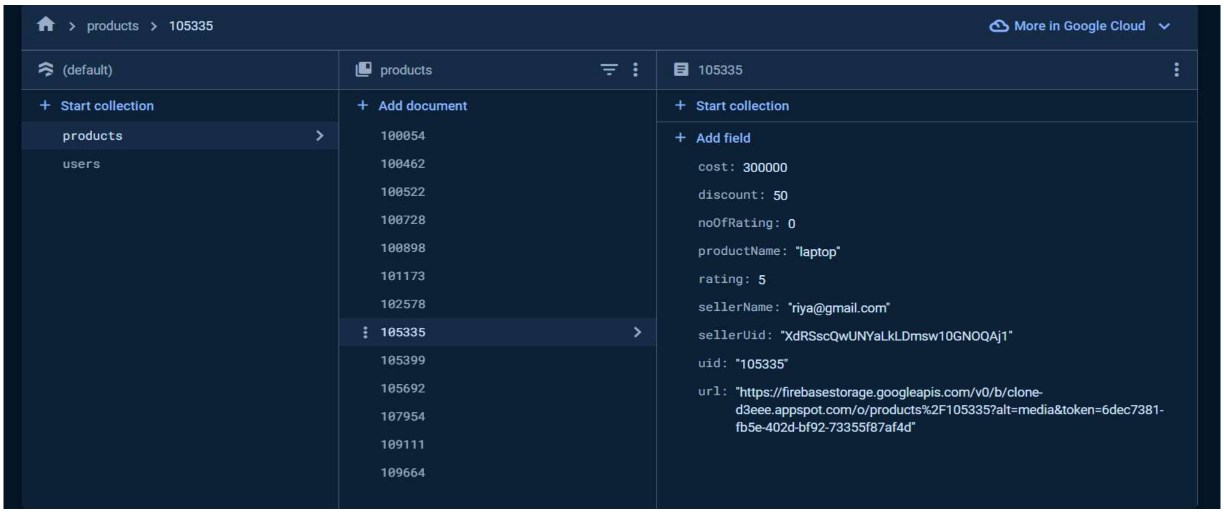
FireStore Database:


Cloud Firestore

(default)
+ Start collection
products
users

users
+ Add document
XdRSscQwUNYaLkLDmsw10GNOQAj1
ca6aEMPmIAhLrUt8jmiUV3wgTY23
vqqvUsmZLaUSqWAXPjxmnswoJIX2

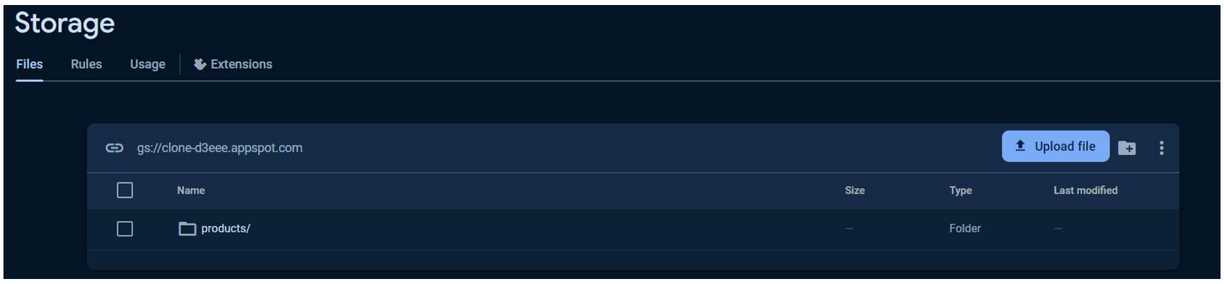
ca6aEMPmIAhLrUt8jmiUV3wgTY23
+ Start collection
+ Add field
address: "Navi Mumbai"
name: "shilpapandey@gmail.com"



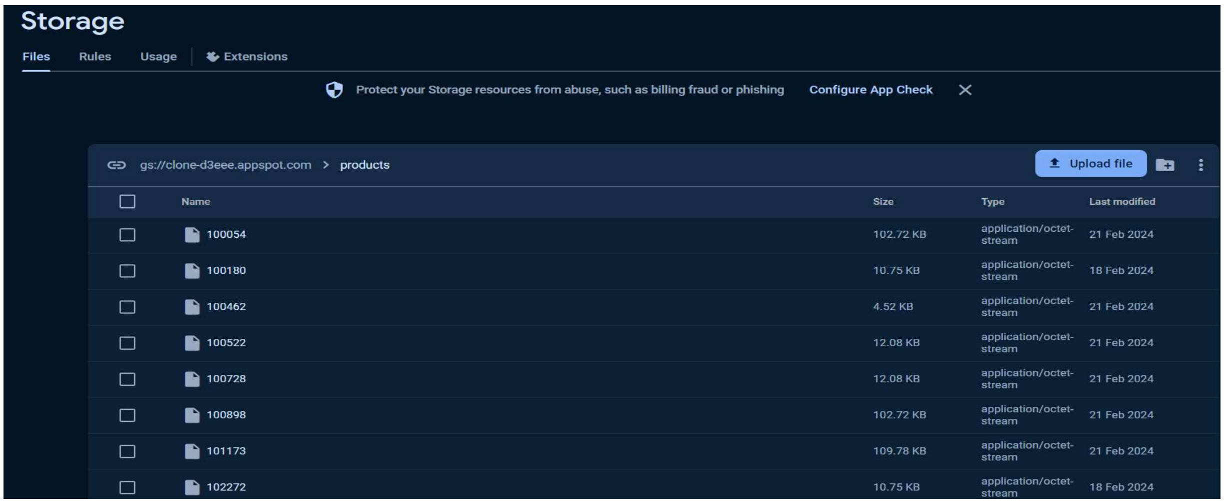
```
100054
100462
100522
100728
100898
101173
102578
105335
105399
105692
107954
109111
109664
```

```
cost: 300000
discount: 50
noOfRating: 0
productName: "laptop"
rating: 5
sellerName: "riya@gmail.com"
sellerUid: "XdRSscQwUNYaLkLDmsw10GNOQAj1"
uid: "105335"
url: "https://firebasestorage.googleapis.com/v0/b/clone-
     d3eee.appspot.com/o/products%2F105335?alt=media&token=6dec7381-
     fb5e-402d-bf92-73355f87af4d"
```

Storage:



Name	Size	Type	Last modified
products/	–	Folder	–


Storage
Files
Rules
Usage
Extensions
Protect your Storage resources from abuse, such as billing fraud or phishing
Configure App Check
gs://clone-d3eee.appspot.com > products
Upload file

Name	Size	Type	Last modified
100054	102.72 KB	application/octet-stream	21 Feb 2024
100180	10.75 KB	application/octet-stream	18 Feb 2024
100462	4.52 KB	application/octet-stream	21 Feb 2024
100522	12.08 KB	application/octet-stream	21 Feb 2024
100728	12.08 KB	application/octet-stream	21 Feb 2024
100898	102.72 KB	application/octet-stream	21 Feb 2024
101173	109.78 KB	application/octet-stream	21 Feb 2024
102272	10.75 KB	application/octet-stream	18 Feb 2024

Conclusion:

Successfully Connected Flutter UI with Firebase database.