

# 배달 앱(패캠마트)

6 프로젝트 돌아보기

## 개요 및 학습 목표 돌아보기

6.

프로젝트  
돌아보기

Flutter와 Firebase를 활용한 앱을 개발한다.

Firebase 콘솔 설정 방법을 학습한다.

Flutter 프로젝트에 Firebase 프로젝트를 추가하는 방법을 학습한다.

> **Firebase와 Flutter 프로젝트 설정방법을 학습한다.**

Firebase의 Auth 기능을 활용하여 **로그인, 회원가입 기능**을 학습한다.

Firebase 서비스의 기능을 활용하여 데이터를 저장하고 읽는 과정을 학습한다.

이미지를 Firebase의 데이터베이스에 저장하고 읽는 방법을 학습한다.

> **Firebase 내부 서비스를 활용하여 앱을 구현한다.**

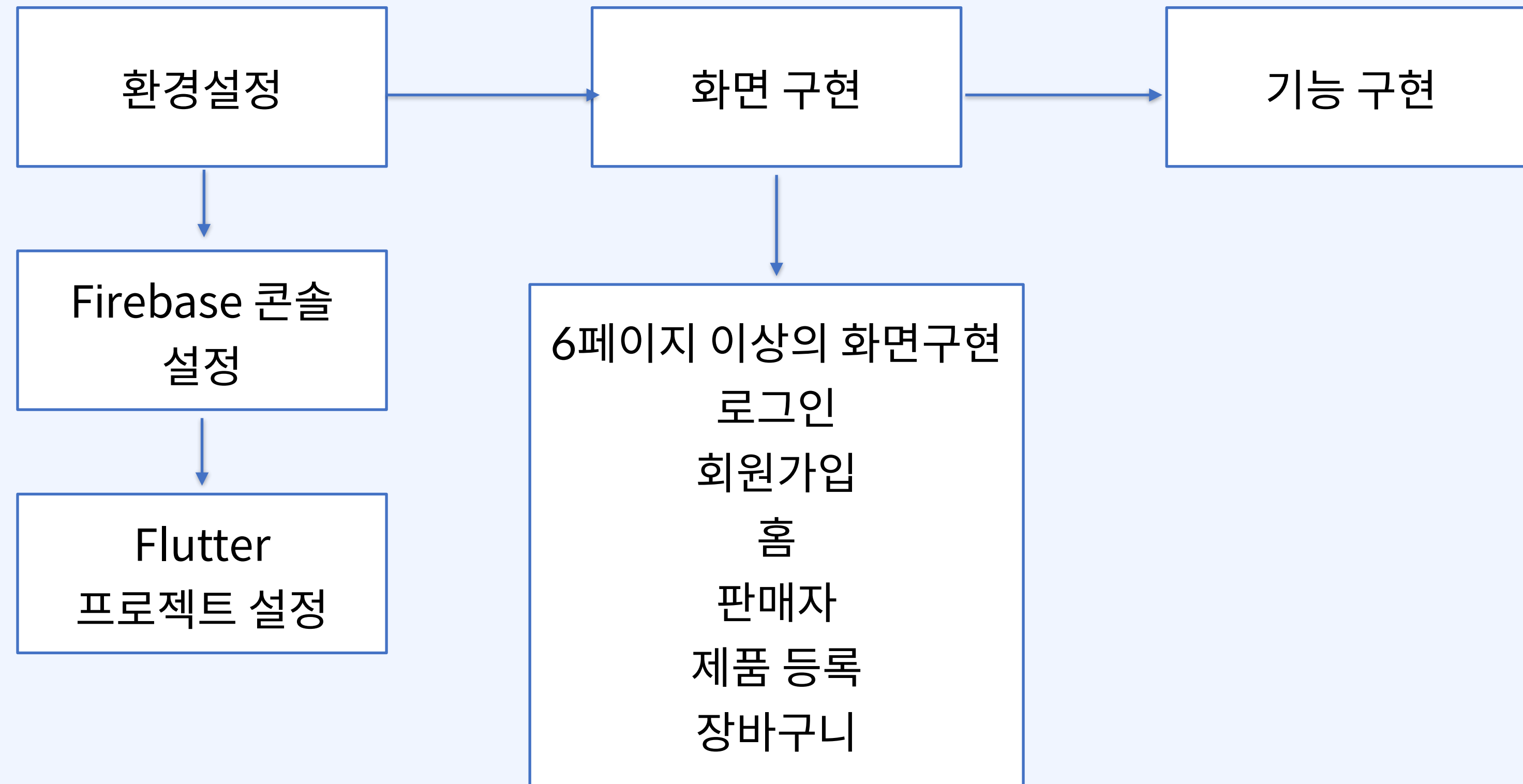
## 기획 돌아보기

## 6. 프로젝트 돌아보기

### 요구사항

1. 회원가입과 로그인 기능을 만들어주세요
2. 판매자가 상품을 등록할 수 있도록 해주세요
3. 카테고리별 상품 등록도 될 수 있어야해요
4. 할인하는 제품도 사용자가 확인할 수 있어야해요
5. 사용자는 장바구니에 상품을 담을 수 있어야해요

## 돌아보기



## 돌아보기-Firebase Firestore

### 컬렉션 설계하기

- 1.사용자 정보 >> users
- 2.카테고리 정보 >> category
- 3.제품 정보 >> products
- 4.장바구니 >> cart

컬렉션과 문서의 구성으로 설계를 진행

## 돌아보기-Firebase Firestore

### 컬렉션 설계하기

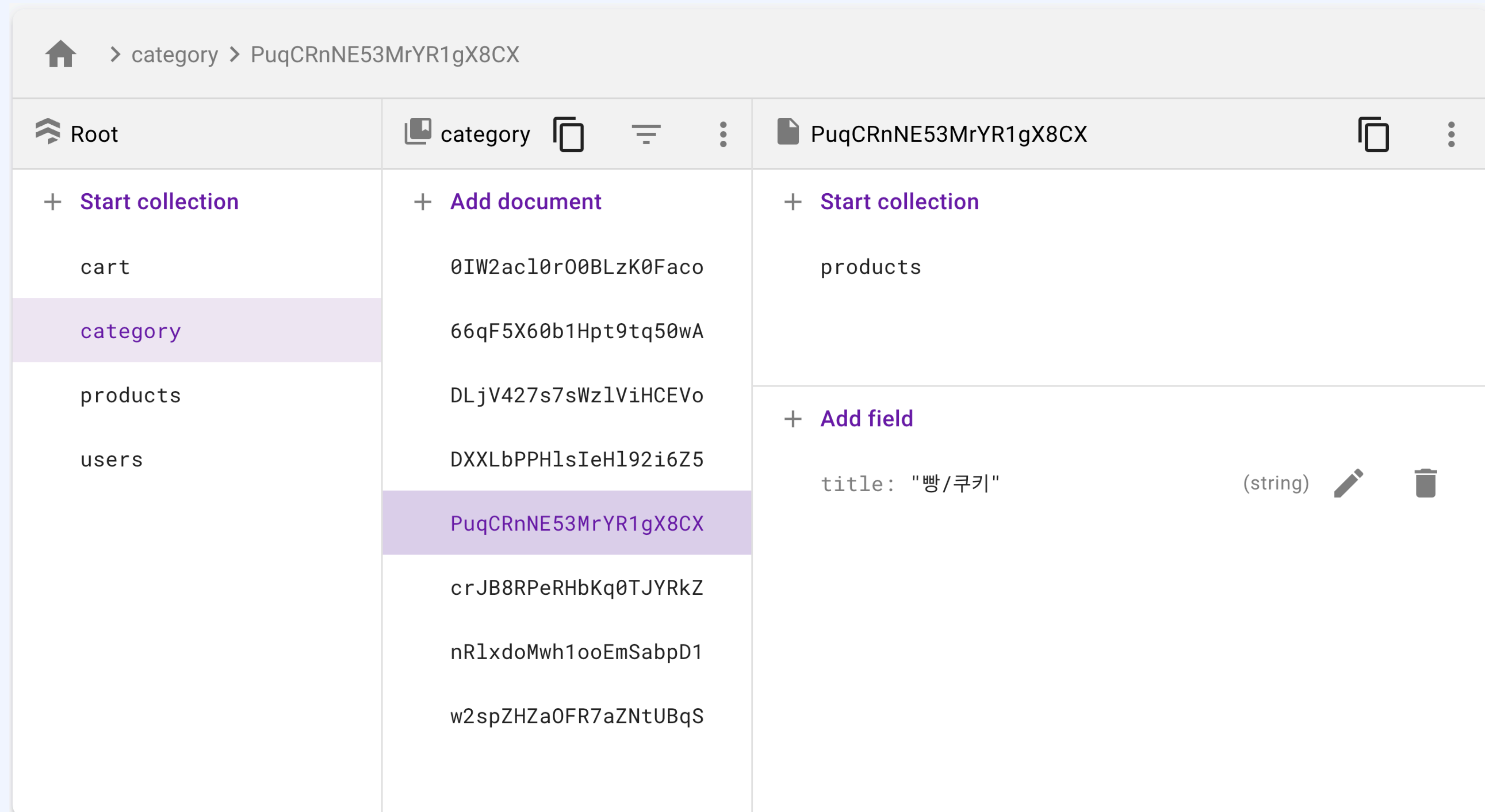
#### 1.사용자 정보

- 이메일
- Uid

🏠 > users > uPBqA7o7plyVeuL77mqC ✎		
🏠 Root	📁 users 📄 ≡ ⋮	📄 uPBqA7o7plyVeuL77mqC 📄 ⋮
+ Start collection	+ Add document	+ Start collection
cart	uPBqA7o7pIyVeuL77mqC	+ Add field
category		email: "fc@gmail.com" (string) ✎ 🗑
products		uid: "m6uGGMnfsszRLAkNUQyKxthoWr...(string) ✎ 🗑
users		

## 돌아보기-Firebase Firestore

### 컬렉션 설계하기 2. 카테고리 정보



## 돌아보기-Firebase Firestore

### 컬렉션 설계하기 2. 카테고리 정보

🏠 > category > PuqCRnNE53MrYR1gX8CX > products > 5kKCMzFXcEY7XfEUatEI		
📁 PuqCRnNE53MrYR1gX8CX	📁 products	📁 5kKCMzFXcEY7XfEUatEI
+ Start collection	+ Add document	+ Start collection
products	5kKCMzFXcEY7XfEUatEI	+ Add field
	CoBmLcGrh7JRiw333kf7	docId: "Gm2LdJBaFK3kWmcrlAS" (string) ✎ 🗑
+ Add field title: (string) ✎ 🗑	FcNxIhbwB62wzUV8U8tS	
	QdQ1ZqbB7h2KTdqaHr2S	
	XrQthZMCMEs9Ng0RwqJB	
	e3Kj1BuWQq6SXHPnNsNX	
	hJFz7WJ1j39umqsrDpm1	
	kt91RgLMpGGr3SWM6EQb	
	tJtPNapxa16ymbW3WgKK	



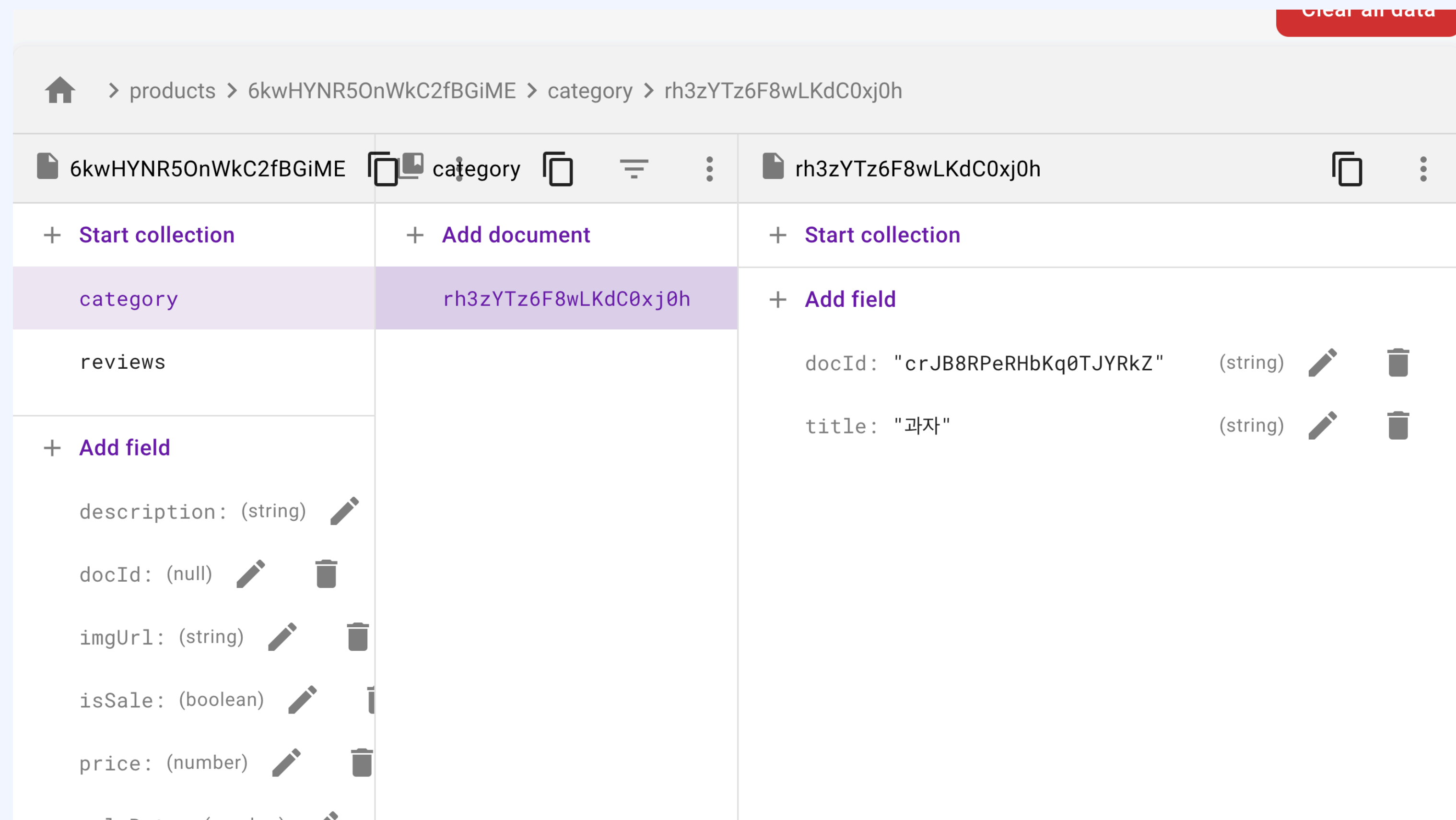
## 돌아보기-Firebase Firestore

### 컬렉션 설계하기 3. 제품정보

🏠 > products > 6kwHYNR5OnWkC2fBGiME		
🏠 Root	📁 products 📄 ⚙️ ⋮	📄 6kwHYNR5OnWkC2fBGiME 📄 ⋮
+ Start collection	+ Add document	+ Start collection
cart	3n1TPjk1pSakP0t0tYPk	category
category	6kwHYNR5OnWkC2fBGiME	reviews
products	6qhAS2n0k0gBuEE9c8aw	+ Add field
users	8zhGEuoJV0vtjwDwx7uE	description: "" (string) ✎ 🗑
	96ox031pJ4yDdDUxKP0Y	docId: null (null) ✎ 🗑
	Gm2LdJBaFK3kWmcrlAS	imgUrl: "http://10.0.2.2:9199/v0...(string) ✎ 🗑
	Kh7bwFF1ZCggQoCkwLX0	isSale: true (boolean) ✎ 🗑
	PVNc27ExzzcgaKJv0Umx	price: 1500 (number) ✎ 🗑
	Qk1vQyJJjFEPlboFXgzu	saleRate: 95 (number) ✎ 🗑

## 돌아보기-Firebase Firestore

### 컬렉션 설계하기 3. 제품정보



## 돌아보기-Firebase Firestore

### 컬렉션 설계하기 3. 제품정보

The screenshot shows the Firebase Firestore console interface. The breadcrumb path is: `products > 6kwHYNR5OnWkC2fBGiME > reviews > VcHKXUZpK1yXHNPwouys`.

The interface is divided into three main sections:

- Left Panel (Collection 6kwHYNR5OnWkC2fBGiME):**
  - Buttons: `+ Start collection`, `+ Add field`.
  - Fields: `category`, `reviews` (highlighted), `description: (string)`, `docId: (null)`, `imgUrl: (string)`, `isSale: (boolean)`, `price: (number)`, `saleRate: (number)`.
- Middle Panel (Collection reviews):**
  - Buttons: `+ Add document`.
  - Document ID: `VcHKXUZpK1yXHNPwouys` (highlighted).
- Right Panel (Document VcHKXUZpK1yXHNPwouys):**
  - Buttons: `+ Start collection`, `+ Add field`.
  - Fields: `datetime: Wed Jul 26 2023 21... (timestamp)`, `review: "lol" (string)`, `score: 3 (number)`, `timestamp: Wed Jul 26 2023 2... (timestamp)`, `uid: "m6uGGMnfsszRLAkNUQyKxthoWr... (string)`.

## 돌아보기-Firebase Firestore

### 컬렉션 설계하기

### 4. 장바구니

🏠 > cart > kTUoTmfupgC2yaATJ4dM		
🏠 Root	📁 cart 📄 📄 ⋮	📄 kTUoTmfupgC2yaATJ4dM 📄 ⋮
+ Start collection	+ Add document	+ Start collection
cart	kTUoTmfupgC2yaATJ4dM	+ Add field
category		count: 1 (number) ✎ 🗑
products		email: "fc@gmail.com" (string) ✎ 🗑
users		▼ product (map) + 🗑
		description: "" (string) ✎ 🗑
		docId: "6kwHYNR50nWkC2fBGiME" (string) ✎ 🗑
		imgUrl: "http://10.0.2.2:919..." (string) ✎ 🗑
		isSale: true (boolean) ✎ 🗑
		price: 1500 (number) ✎ 🗑

## 돌아보기 - 구현

### Firestore 로그인

```
Future<UserCredential?> signIn(String emailAddress, String password) async {  
  try {  
    final credential = await FirebaseAuth.instance  
      .signInWithEmailAndPassword(email: emailAddress, password: password);  
    print(credential);  
    return credential;  
  } on FirebaseAuthException catch (e) {  
    if (e.code == 'user-not-found') {  
      print('No user found for that email.');    } else if (e.code == 'wrong-password') {  
      print('Wrong password provided for that user.');    }  
  }  
}
```

## 돌아보기 - 구현

Firebase  
회원가입

firestore에  
저장하기

```
Future<bool> signUp(String emailAddress, String password) async {  
  try {  
    final credential =  
      await FirebaseAuth.instance.createUserWithEmailAndPassword(  
        email: emailAddress,  
        password: password,  
      );  
    await FirebaseFirestore.instance.collection("users").add({  
      "uid": credential.user?.uid ?? "",  
      "email": credential.user?.email ?? ""  
    });  
    return true;  
  } on FirebaseAuthException catch (e) {  
    if (e.code == 'weak-password') {  
      print('The password provided is too weak.');    } else if (e.code == 'email-already-in-use') {  
      print('The account already exists for that email.');    }  
    return false;  
  } catch (e) {  
    print(e);  
    return false;  
  }  
}
```



## 돌아보기 - 구현

### 이미지 압축

```
Future<Uint8List> imageCompressList(Uint8List list) async {  
    var result = await FlutterImageCompress.compressWithList(  
        list,  
        quality: 50,  
    );  
    return result;  
}
```

## 돌아보기 - 구현

### Storage 올리기 다운로드 링크 얻기

```
final storageRef = storage.ref().child(
    "${DateTime.now().millisecondsSinceEpoch}_${image?.name ?? "${DateTime.now()}.jpg"}");
final compressedData = await imageCompressList(imageData!);
await storageRef.putData(compressedData);
final downloadLink = await storageRef.getDownloadURL();
```



## 돌아보기 - 구현

### Firestore 데이터 쓰기

```
final doc = await db.collection("products").add(sampleData.toJson());
await doc.collection("category").add(selectedCategory?.toJson() ?? {});
final catRef = db.collection("category").doc(selectedCategory?.docId);
await catRef.collection("products").add({"docId": doc.id});
```

## 돌아보기 - 구현

### Firestore 실시간으로 데이터 받아오기 Stream

```
Stream<QuerySnapshot<Map<String, dynamic>>> fetchCartItems() {  
  return FirebaseFirestore.instance  
    .collection("cart")  
    .where("email", isEqualTo: widget.uid)  
    .orderBy("timestamp")  
    .snapshots();  
}
```

## 돌아보기 - 구현

Firestore 실시간으로 데이터 받아오기  
Stream

```
StreamBuilder(  
  stream: fetchCartItems(),  
  builder: (BuildContext context,  
    AsyncSnapshot<QuerySnapshot<Map<String, dynamic>>> snapshot) {
```

## 더 나아가기 Next Step

## 6. 프로젝트 돌아보기

1. 다양한 로그인 방법 붙여보기
2. 주문하기 화면 구현해보기
3. 리뷰 화면 추가 구현해보기

구현이란 100% 정답이 없습니다.  
많이 만들어보고 **반복**해보면서 시스템에 맞는 여러분만의 적합한 방법을 찾아보세요!  
**끝까지 고생많으셨습니다. 화이팅!**