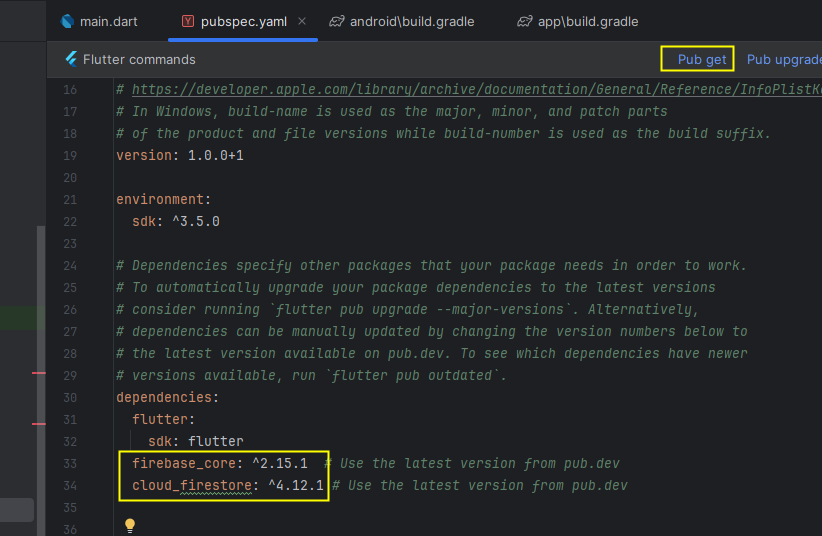
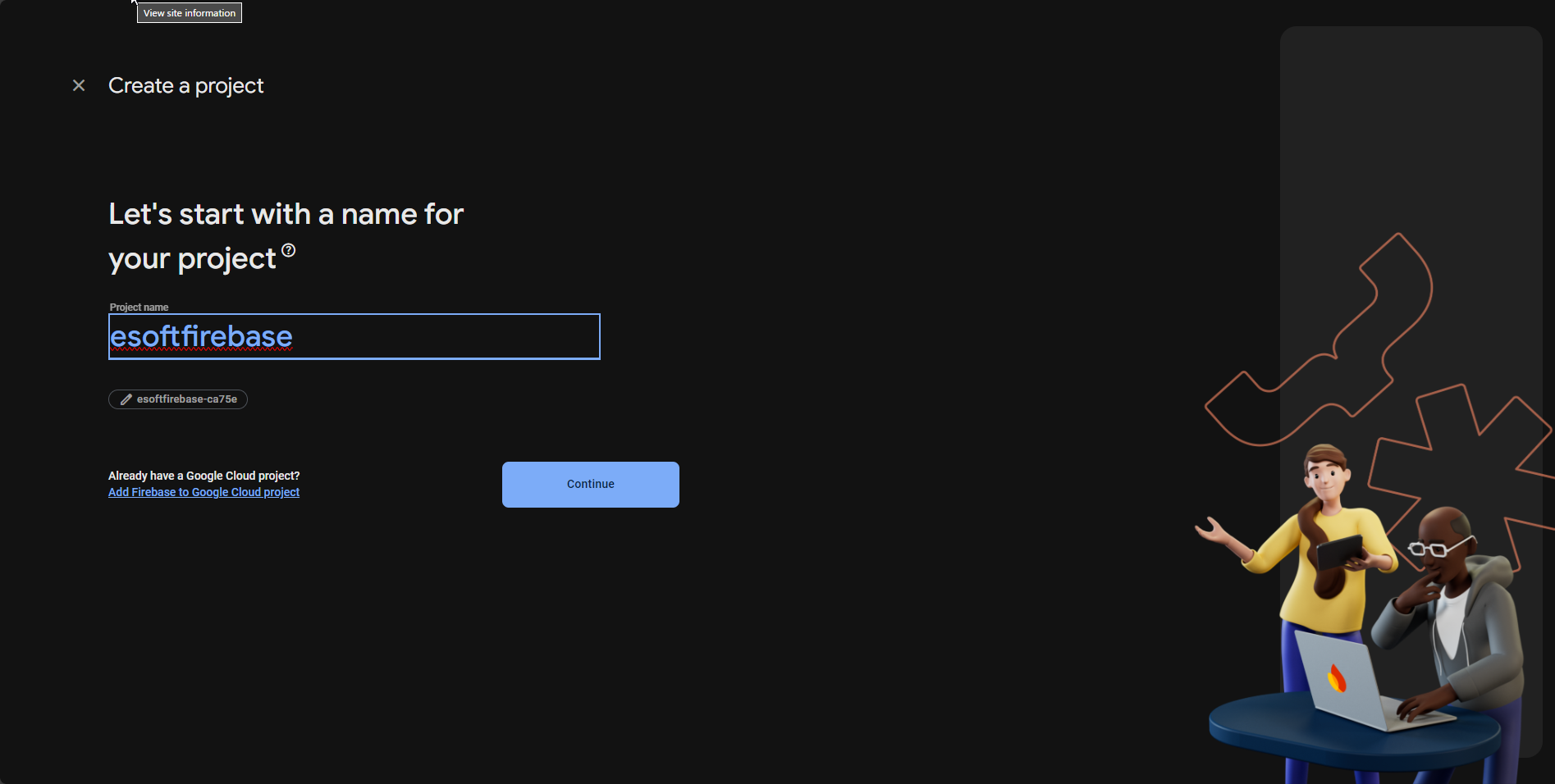
**Step 1: Set Up Flutter Project**

1. Add dependencies in pubspec.yaml



**Step 2: Set Up Firebase Project**

1. Go to Firebase Console.
2. Create a new Firebase project.



1. Add an Android app to the project:
   * Provide the package name (e.g., com.example.myapp).
   * Download the google-services.json file and place it in the android/app directory.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

1. *Add an iOS app (optional):*
   * *Download the GoogleService-Info.plist file and add it to your iOS project's root directory.*
2. Update android/build.gradle file with plug-in:



1. Update android/app/build.gradle file with plug-in:

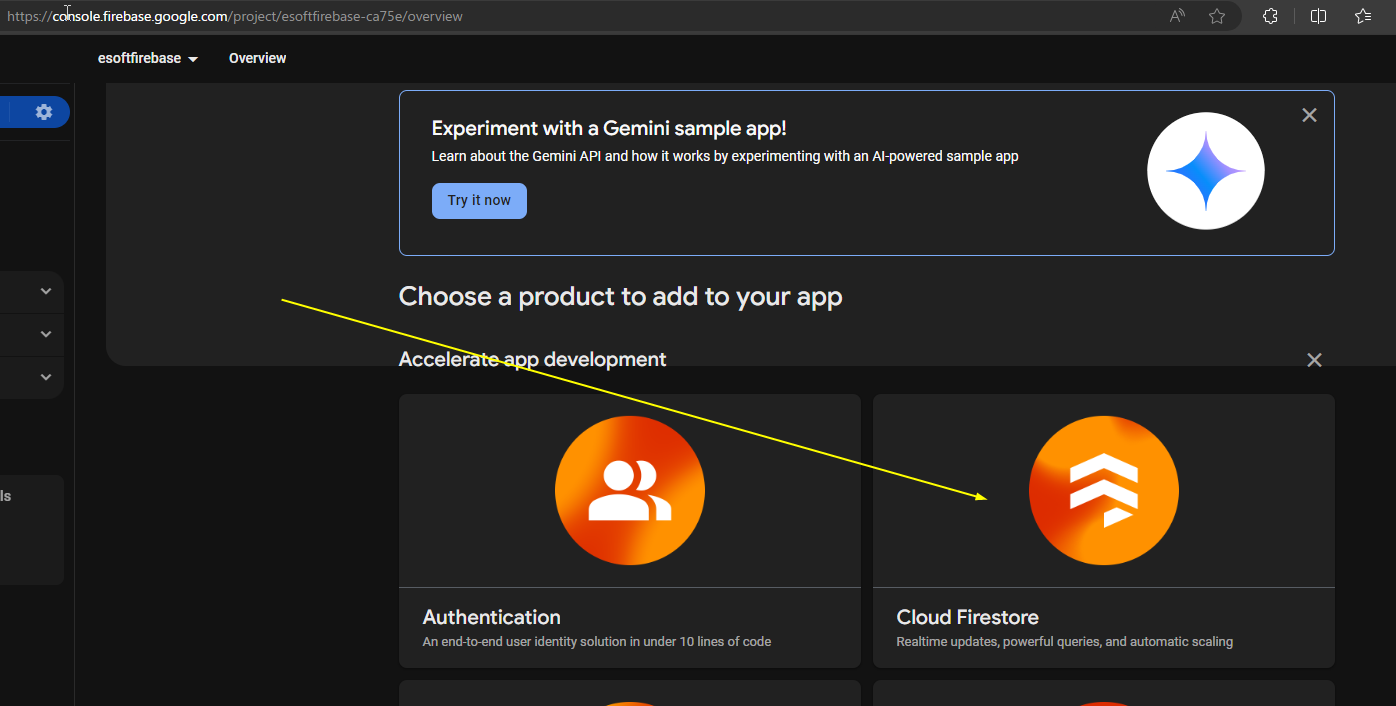
A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

1. Enable **Cloud Firestore** in the Firebase Console:



* + Go to **Firestore Database** → **Create database** → Select the appropriate mode (start with test mode for development).

A screen shot of a computer

Description automatically generated

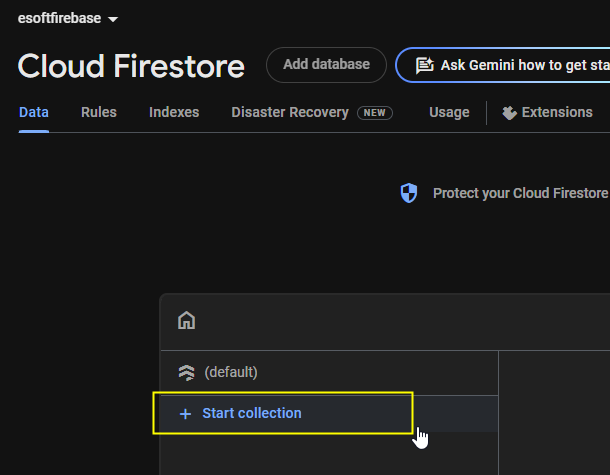
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* Create a collection:



* + Firestore Structure:
    - Collection: users

A screenshot of a computer

Description automatically generated

* + - Document: Auto-generated

A screenshot of a computer

Description automatically generated

* + - Fields: name: String

A screenshot of a computer

Description automatically generated

A screenshot of a computer

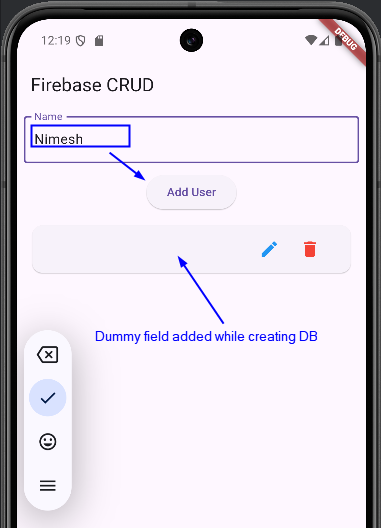
Description automatically generated

**Step 3: Initialize Firebase in Your Flutter App and Create UI**

1. Update the main.dart file:

|  |
| --- |
| import 'package:flutter/material.dart'; **import 'package:firebase\_core/firebase\_core.dart';** **import 'package:cloud\_firestore/cloud\_firestore.dart';**  void main() async {  WidgetsFlutterBinding.*ensureInitialized*();  **await Firebase.*initializeApp*(**  **options: FirebaseOptions(**  **apiKey: 'AIzaSyA111F3ZsF234mpme123rxj5Fix9kpYcpw',**  **appId: '1:501113539183:android:56222aa14000031c272e1a',**  **messagingSenderId: 'sendid',**  **projectId: 'esoftfirebase-ca75e',**  **storageBucket: 'esoftfirebase-ca75e.firebasestorage.app',**  **)**  **);**  runApp(const MyApp()); }  class MyApp extends StatelessWidget {  const MyApp({super.key});   @override  Widget build(BuildContext context) {  return MaterialApp(  title: 'Firebase CRUD',  theme: ThemeData(primarySwatch: Colors.*blue*),  home: HomePage(),  );  } }  class HomePage extends StatefulWidget {  @override  \_HomePageState createState() => \_HomePageState(); }  class \_HomePageState extends State<HomePage> {  final TextEditingController *nameController* = TextEditingController();  final CollectionReference users = **FirebaseFirestore.*instance*.collection('*users*');**   // Add user  Future<void> addUser(String name) async {  try {  await users.add({'name': name});  ScaffoldMessenger.*of*(context).showSnackBar(  const SnackBar(content: Text('User added successfully!')),  );  } catch (e) {  ScaffoldMessenger.*of*(context).showSnackBar(  SnackBar(content: Text('Failed to add user: $e')),  );  }  }   // Update user  Future<void> updateUser(String docId, String newName) async {  try {  await users.doc(docId).update({'name': newName});  ScaffoldMessenger.*of*(context).showSnackBar(  const SnackBar(content: Text('User updated successfully!')),  );  } catch (e) {  ScaffoldMessenger.*of*(context).showSnackBar(  SnackBar(content: Text('Failed to update user: $e')),  );  }  }   // Delete user  Future<void> deleteUser(String docId) async {  try {  await users.doc(docId).delete();  ScaffoldMessenger.*of*(context).showSnackBar(  const SnackBar(content: Text('User deleted successfully!')),  );  } catch (e) {  ScaffoldMessenger.*of*(context).showSnackBar(  SnackBar(content: Text('Failed to delete user: $e')),  );  }  }   @override  void dispose() {  nameController.dispose();  super.dispose();  }   @override  Widget build(BuildContext context) {  return Scaffold(  appBar: AppBar(title: const Text('Firebase CRUD')),  body: Padding(  padding: const EdgeInsets.all(8.0),  child: Column(  children: [  *// TextField for input*  TextField(  controller: nameController,  decoration: const InputDecoration(  labelText: 'Name',  border: OutlineInputBorder(),  ),  ),  const SizedBox(height: 10),  ElevatedButton(  **onPressed: () {**  **if (nameController.text.trim().isEmpty) {**  **ScaffoldMessenger.*of*(context).showSnackBar(**  **const SnackBar(content: Text('Name cannot be empty')),**  **);**  **return;**  **}**  **addUser(nameController.text.trim());**  **nameController.clear();**  **},**  child: const Text('Add User'),  ),  const SizedBox(height: 10),  Expanded(  *// Real-time data display*  child: StreamBuilder(  stream: users.snapshots(),  builder: (context, AsyncSnapshot<QuerySnapshot> snapshot) {  if (snapshot.hasError) {  return const Center(  child: Text('Error loading data.'),  );  }  if (snapshot.connectionState == ConnectionState.waiting) {  return const Center(  child: CircularProgressIndicator(),  );  }  final data = snapshot.data!.docs;  if (data.isEmpty) {  return const Center(child: Text('No users found.'));  }  return ListView.builder(  itemCount: data.length,  itemBuilder: (context, index) {  final doc = data[index];  final name = doc['name'];  return Card(  margin: const EdgeInsets.symmetric(  vertical: 5, horizontal: 10),  child: ListTile(  title: Text(name),  trailing: Row(  mainAxisSize: MainAxisSize.min,  children: [  *// Edit button*  IconButton(  icon: const Icon(Icons.*edit*, color: Colors.*blue*),  onPressed: () {  final newNameController =  TextEditingController(text: name);  showDialog(  context: context,  builder: (\_) => AlertDialog(  title: const Text('Update Name'),  content: TextField(  controller: newNameController,  decoration: const InputDecoration(  labelText: 'New Name',  border: OutlineInputBorder(),  ),  ),  actions: [  TextButton(  onPressed: () {  Navigator.*pop*(context);  },  child: const Text('Cancel'),  ),  ElevatedButton(  **onPressed: () {**  **if (newNameController.text.trim().isEmpty) {**  **ScaffoldMessenger.*of*(context).showSnackBar(**  **const SnackBar(**  **content: Text('Name cannot be empty')),**  **);**  **return;**  **}**  **updateUser(**  **doc.id, newNameController.text.trim());**  **Navigator.*pop*(context);**  **},**  child: const Text('Update'),  ),  ],  ),  );  },  ),  *// Delete button*  IconButton(  icon: const Icon(Icons.*delete*, color: Colors.*red*),  **onPressed: () => deleteUser(doc.id),**  ),  ],  ),  ),  );  },  );  },  ),  ),  ],  ),  ),  );  } } |

**Step 4: Run the Project**

 A screenshot of a phone

Description automatically generated

A screenshot of a computer

Description automatically generated

Possible solutions for issues

* Kotlin/ Android version compatibility issue:

