Experiment 6

Aim : To connect Flutter UI with Firebase Database.

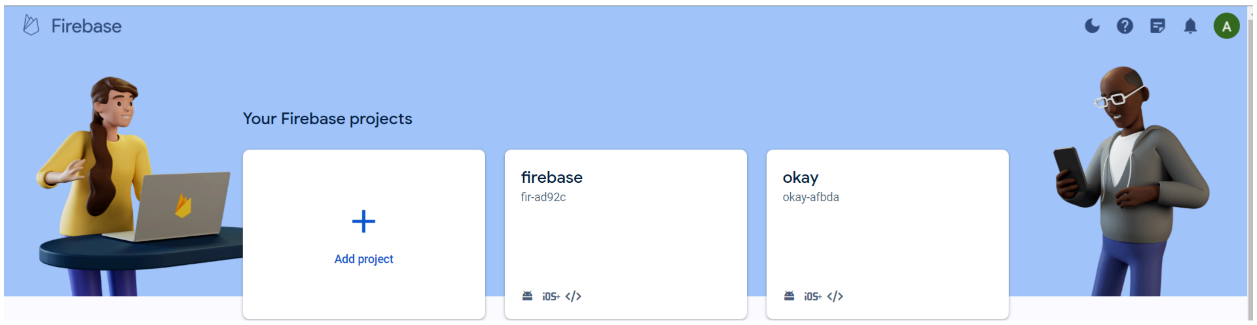
Theory :

Prerequisites :

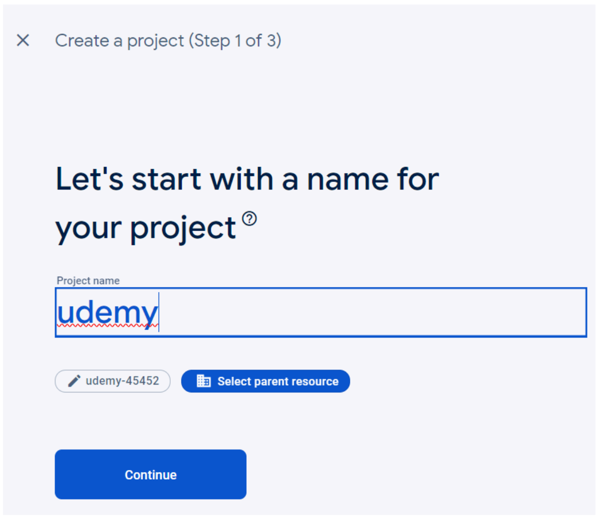
To complete this tutorial, you will need:

- A Google account to use Firebase.
- Developing for iOS will require XCode.
- To download and install Flutter.
- To download and install Android Studio and Visual Studio Code.
- It is recommended to install plugins for your code editor:
    - Flutter and Dart plugins installed for Android Studio.
    - Flutter extension installed for Visual Studio Code.

Creating a New Firebase Project :



Adding Android support:

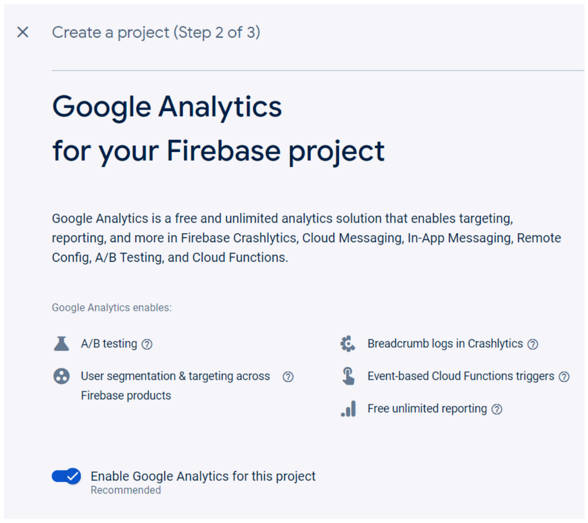


Create a project (Step 1 of 3)

Let's start with a name for your project

Project name

udemy



Create a project (Step 2 of 3)

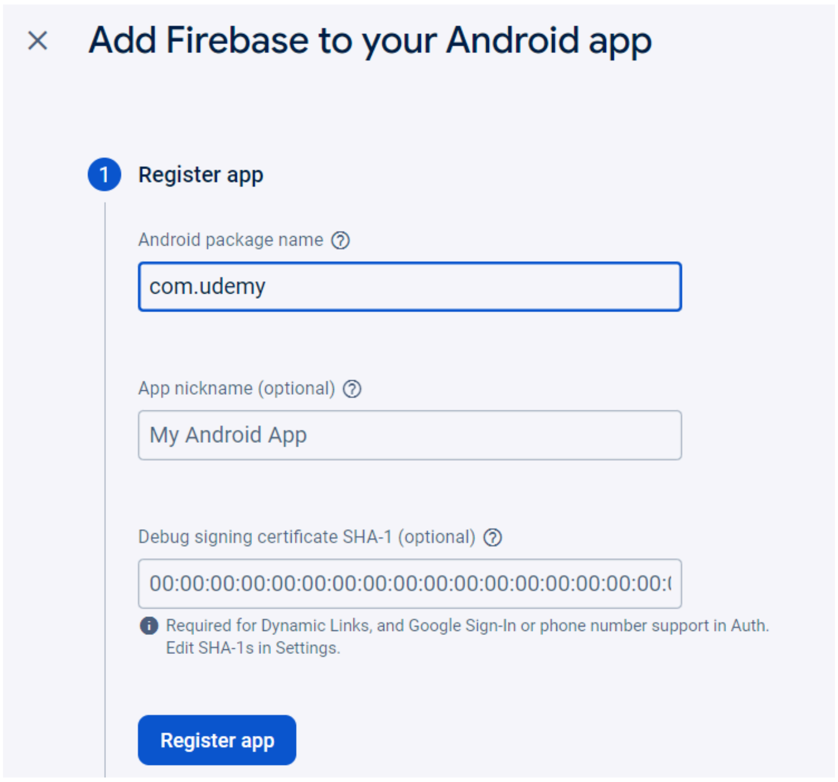
Google Analytics for your Firebase project

Google Analytics is a free and unlimited analytics solution that enables targeting, reporting, and more in Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing, and Cloud Functions.

Google Analytics enables:

- A/B testing
- User segmentation & targeting across Firebase products
- Breadcrumb logs in Crashlytics
- Event-based Cloud Functions triggers
- Free unlimited reporting

Enable Google Analytics for this project ✓
Recommended



Add Firebase to your Android app

1 Register app

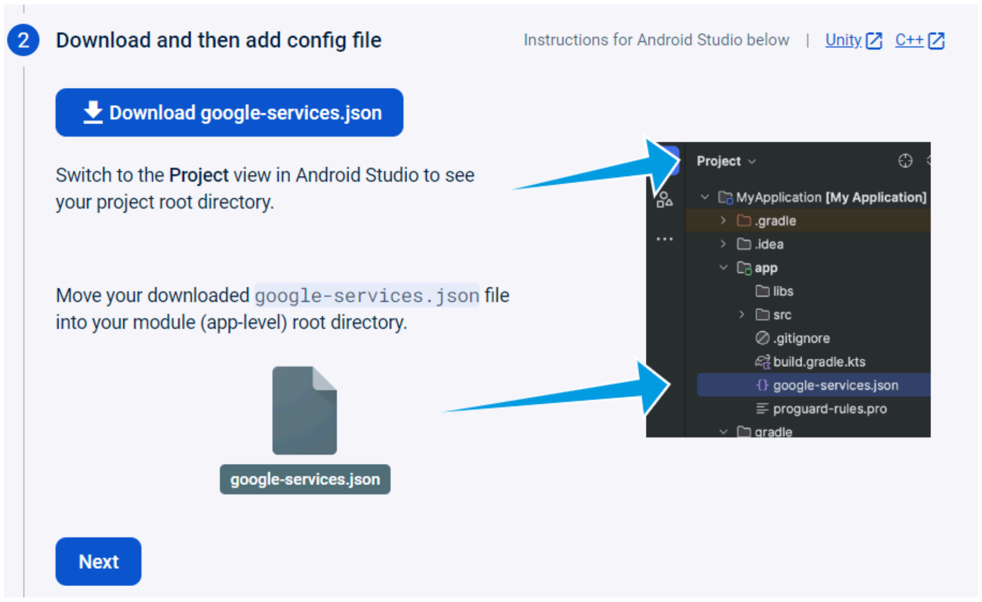
Android package name: com.udemy

App nickname (optional): My Android App

Debug signing certificate SHA-1 (optional): 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:(

Required for Dynamic Links, and Google Sign-In or phone number support in Auth. Edit SHA-1s in Settings.

Downloading the Config File:


2 Download and then add config file

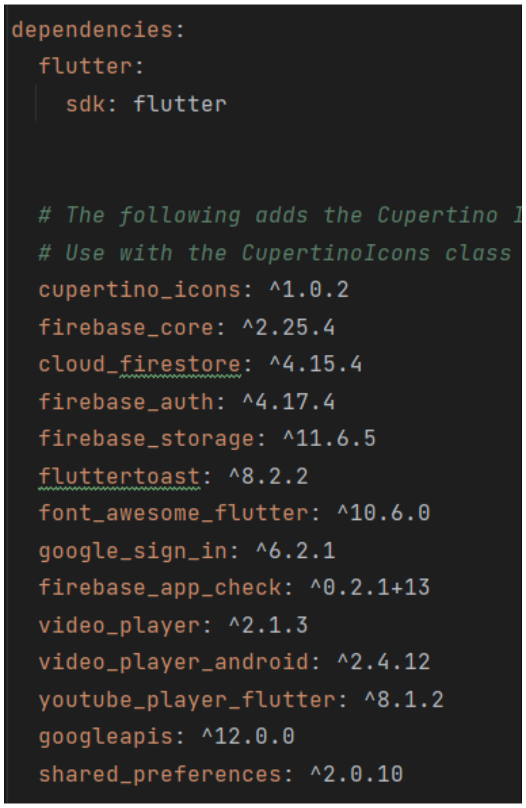
Instructions for Android Studio below | Unity C++

Download google-services.json

Switch to the **Project** view in Android Studio to see your project root directory.

Move your downloaded `google-services.json` file into your module (app-level) root directory.

Adding dependencies to the project:


```
dependencies:
  flutter:
    sdk: flutter


  # The following adds the Cupertino I
  # Use with the CupertinoIcons class
  cupertino_icons: ^1.0.2
  firebase_core: ^2.25.4
  cloud_firestore: ^4.15.4
  firebase_auth: ^4.17.4
  firebase_storage: ^11.6.5
  fluttertoast: ^8.2.2
  font_awesome_flutter: ^10.6.0
  google_sign_in: ^6.2.1
  firebase_app_check: ^0.2.1+13
  video_player: ^2.1.3
  video_player_android: ^2.4.12
  youtube_player_flutter: ^8.1.2
  googleapis: ^12.0.0
  shared_preferences: ^2.0.10
```

Code:

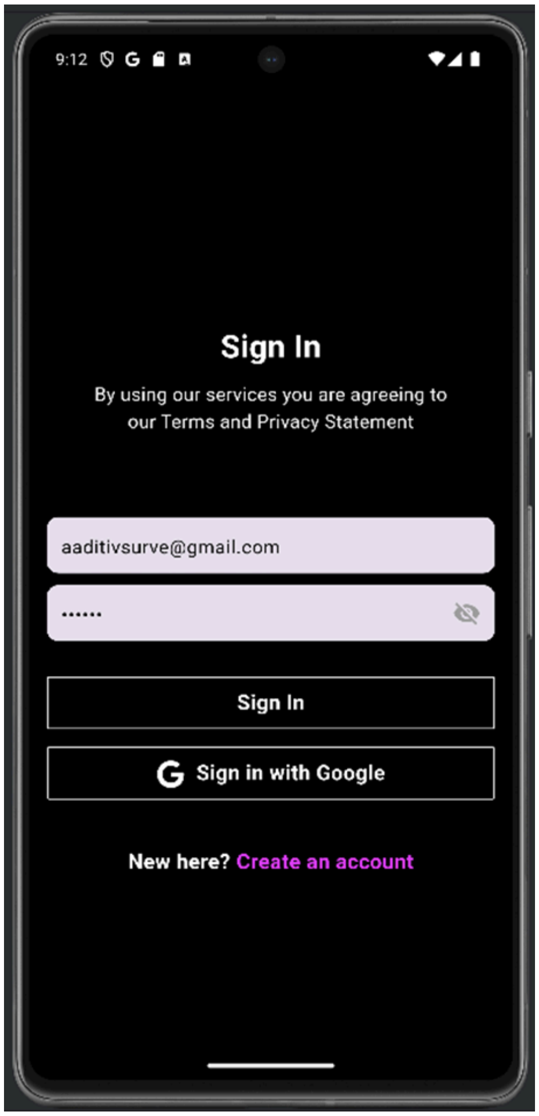
```
//login_page
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:flutter_firebase/features/user_auth/presentation/pages/sign_up_page.dart';
import 'package:flutter_firebase/features/user_auth/presentation/widgets/form_container_widget.dart';
import 'package:flutter_firebase/global/common/toast.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:google_sign_in/google_sign_in.dart';
import 'package:firebase_app_check/firebase_app_check.dart';
// import 'package:firebase_app_check_debug/firebase_app_check_debug.dart';
import '../../firebase_auth_implementation/firebase_auth_services.dart';


class LoginPage extends StatefulWidget {
  const LoginPage({super.key});
  @override
  State<LoginPage> createState() => _LoginPageState();
}
class _LoginPageState extends State<LoginPage> {
  bool _isSigning = false;
  final FirebaseAuthService _auth = FirebaseAuthService();
  final FirebaseAuth _firebaseAuth = FirebaseAuth.instance;
  TextEditingController _emailController = TextEditingController();
  TextEditingController _passwordController = TextEditingController();
  }
  @override
  void dispose() {
    _emailController.dispose();
    _passwordController.dispose();
    super.dispose();
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.black,
        automaticallyImplyLeading: false,
      ),
      body: Container(
        color: Colors.black, // Set the background color to white
        padding: EdgeInsets.symmetric(horizontal: 15),
        child: Column(
mainAxisAlignment: MainAxisAlignment.center,
children: [
  Text(
    "Sign In",
    style: TextStyle(
        fontSize: 27,
        fontWeight: FontWeight.bold,
        color: Colors.white),
  ),
  SizedBox(height: 10), // Add SizedBox for spacing
  Padding(
    padding: EdgeInsets.symmetric(horizontal: 32), // Adjust the left padding as needed
    child: Center(
      child: Text(
        "By using our services you are agreeing to our Terms and Privacy Statement",
        textAlign: TextAlign.center,
        style: TextStyle(
          fontSize: 16,
          color: Colors.white,
        ),
      ),
    ),
  ),
  SizedBox(height: 70),
  FormContainerWidget(
    controller: _emailController,
    hintText: "Email",
    isPasswordField: false,
  ),
  SizedBox(height: 10),
  FormContainerWidget(
    controller: _passwordController,
    hintText: "Password",
    isPasswordField: true,
  ),
  SizedBox(height: 30),
  GestureDetector(
    onTap: () {
      _signIn();
    },
    child: Container(
      width: double.infinity,
      height: 45,
      decoration: BoxDecoration(
      color: Colors.black,
      borderRadius: BorderRadius.circular(2),
      border: Border.all(color: Colors.white),
    ),
    child: Center(
      child: _isSigning
          ? CircularProgressIndicator(color: Colors.white)
          : Text(
        "Sign In",
        style: TextStyle(
          color: Colors.white,
          fontSize: 18.0,
          fontWeight: FontWeight.bold,
        ),
      ),
    ),
  ),
),
SizedBox(height: 15),
GestureDetector(
  onTap: () {
    _signInWithGoogle();
  },
  child: Container(
    width: double.infinity,
    height: 45,
    decoration: BoxDecoration(
      color: Colors.black,
      borderRadius: BorderRadius.circular(2),
      border: Border.all(color: Colors.white),
    ),
    child: Center(
      child: Row(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Icon(FontAwesomeIcons.google, color: Colors.white),
          SizedBox(width: 10),
          Text(
            "Sign in with Google",
            style: TextStyle(
              color: Colors.white,
              fontSize: 18.0,
              fontWeight: FontWeight.bold,
            ),
                    ),
                  ],
                ),
              ),
            ),
          ),
          SizedBox(height: 40),
          Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Text(
                "New here?",
                style: TextStyle(
                    color: Colors.white,
                    fontWeight: FontWeight.bold,
                    fontSize: 18.0
                ),
              ),
              SizedBox(width: 5),
              GestureDetector(
                onTap: () {
                  Navigator.pushAndRemoveUntil(
                    context,
                    MaterialPageRoute(builder: (context) => SignUpPage()),
                        (route) => false,
                  );
                },
                child: Text(
                  "Create an account",
                  style: TextStyle(
                      color: Colors.purpleAccent,
                      fontWeight: FontWeight.bold,
                      fontSize: 18.0
                  ),
                ),
              ),
            ],
          ),
        ],
      ),
    ),
  );
}
void _signIn() async {
    setState(() {
      _isSigning = true;
    });
    String email = _emailController.text;
    String password = _passwordController.text;
    User? user = await _auth.signInWithEmailAndPassword(email, password);
    setState(() {
      _isSigning = false;
    });
    if (user != null) {
      showToast(message: "User is successfully signed in");
      Navigator.pushNamed(context, "/home");
    } else {
      showToast(message: "Some error occurred");
    }
  }
  _signInWithGoogle() async {
    final GoogleSignIn _googleSignIn = GoogleSignIn();
    try {
      final GoogleSignInAccount? googleSignInAccount =
      await _googleSignIn.signIn();
      if (googleSignInAccount != null) {
        final GoogleSignInAuthentication googleSignInAuthentication =
        await googleSignInAccount.authentication;
        final AuthCredential credential = GoogleAuthProvider.credential(
          idToken: googleSignInAuthentication.idToken,
          accessToken: googleSignInAuthentication.accessToken,
        );
        await _firebaseAuth.signInWithCredential(credential);
        Navigator.pushNamed(context, "/home");
      }
    } catch (e) {
      showToast(message: "Some error occurred $e");
    }
  }
}

//sign_up_page
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import
'package:flutter_firebase/features/user_auth/firebase_auth_implementation/firebase_auth_services.dart';
import 'package:flutter_firebase/features/user_auth/presentation/pages/login_page.dart';
import 'package:flutter_firebase/features/user_auth/presentation/widgets/form_container_widget.dart';
import 'package:flutter_firebase/global/common/toast.dart';
class SignUpPage extends StatefulWidget {
  const SignUpPage({super.key});
  @override
  State<SignUpPage> createState() => _SignUpPageState();
}
class _SignUpPageState extends State<SignUpPage> {
  final FirebaseAuthService _auth = FirebaseAuthService();
  TextEditingController _usernameController = TextEditingController();
  TextEditingController _emailController = TextEditingController();
  TextEditingController _passwordController = TextEditingController();
  bool isSigningUp = false;
  @override
  void dispose() {
    _usernameController.dispose();
    _emailController.dispose();
    _passwordController.dispose();
    super.dispose();
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        automaticallyImplyLeading: false,
        backgroundColor: Colors.black,
      ),
      backgroundColor: Colors.black,
      body: Center(
        child: Padding(
          padding: const EdgeInsets.symmetric(horizontal: 15),
          child: Column(
           mainAxisAlignment: MainAxisAlignment.center,
           children: [
            Text(
             "Create an account",
             style: TextStyle(
               fontSize: 27,
               fontWeight: FontWeight.bold,
               color: Colors.white),
            ),
            SizedBox(
             height: 20,
            ),
            Text(
    "By clicking on 'Create account' you are agreeing to our Terms of Use and Privacy Policy",
    textAlign: TextAlign.center,
    style: TextStyle(fontSize: 16, color: Colors.white),
),
SizedBox(
    height: 80,
),
FormContainerWidget(
    controller: _usernameController,
    hintText: "Username",
    isPasswordField: false,
),
SizedBox(
    height: 10,
),
FormContainerWidget(
    controller: _emailController,
    hintText: "Email",
    isPasswordField: false,
),
SizedBox(
    height: 10,
),
FormContainerWidget(
    controller: _passwordController,
    hintText: "Password",
    isPasswordField: true,
),
SizedBox(
    height: 30,
),
GestureDetector(
    onTap: () {
        _signUp();
    },
    child: Container(
        width: double.infinity,
        height: 45,
        decoration: BoxDecoration(
           color: Colors.white,
           borderRadius: BorderRadius.circular(10),
        ),
        child: Center(
             child: isSigningUp
```

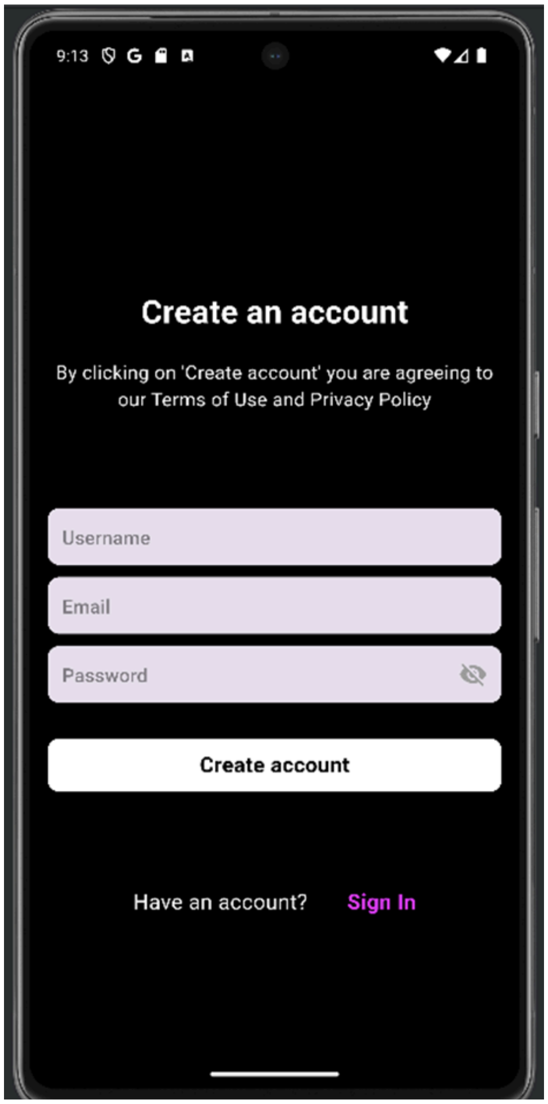
```
              ? CircularProgressIndicator(
                  color: Colors.white,
                )
              : Text(
                  "Create account",
                  style: TextStyle(
                      color: Colors.black,
                      fontWeight: FontWeight.bold,
                      fontSize: 18.0),
                )),
      ),
    ),
    SizedBox(
      height: 80,
    ),
    Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Text(
          "Have an account?",
          style: TextStyle(
              color: Colors.white,
              fontSize: 18.0),
        ),
        SizedBox(
          width: 5,
        ),
        GestureDetector(
            onTap: () {
              Navigator.pushAndRemoveUntil(
                  context,
                  MaterialPageRoute(
                      builder: (context) => LoginPage()),
                  (route) => false);
            },
            child: Text(
              "     Sign In",
              style: TextStyle(
                  color: Colors.purpleAccent, fontWeight: FontWeight.bold, fontSize: 18.0),
            ))
      ],
    )
  ],
),
        ),
      ),
    );
  }
  void _signUp() async {
    setState(() {
      isSigningUp = true;
    });
    String username = _usernameController.text;
    String email = _emailController.text;
    String password = _passwordController.text;
    User? user = await _auth.signUpWithEmailAndPassword(email, password);
    setState(() {
      isSigningUp = false;
    });
    if (user != null) {
      showToast(message: "User is successfully created");
      Navigator.pushNamed(context, "/home");
    } else {
      showToast(message: "Some error happend");
    }
  }
}

//wishlist_page
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';

class WishlistPage extends StatefulWidget {
  @override
  _WishlistPageState createState() => _WishlistPageState();
}

class _WishlistPageState extends State<WishlistPage> {
  late List<Map<String, dynamic>> wishlist;

  @override
  void initState() {
    super.initState();
    wishlist = []; // Initialize wishlist as an empty list
    loadWishlist(); // Load wishlist data
  }

  Future<void> loadWishlist() async {
```

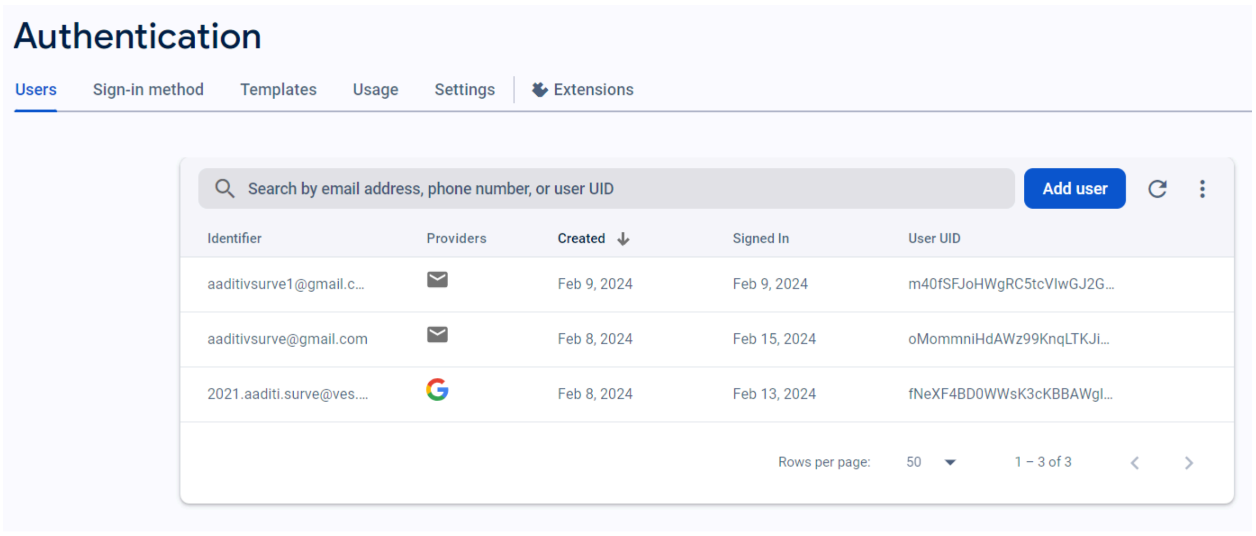
```
 try {
   CollectionReference wishlistCollection =
   FirebaseFirestore.instance.collection('Wishlist');
   QuerySnapshot querySnapshot = await wishlistCollection.get();
   List<Map<String, dynamic>> loadedWishlist =
   []; // Temporary list to hold the loaded wishlist
   querySnapshot.docs.forEach((doc) {
     Map<String, dynamic>? data =
     doc.data() as Map<String, dynamic>?; // Get document data as Map
     if (data != null &&
         data.containsKey('price') &&
         data.containsKey('ratings')) {
       // Check if 'price' and 'ratings' fields exist in the document data
       final price = data['price'] as num; // Assuming price is of type num
       final ratings = data['ratings'] as double? ??
           0.0; // Assuming ratings is of type double
       if (ratings != 0.0) {
         // Add course to wishlist only if ratings are not 0
         loadedWishlist.add({
           'id': doc.id,
           'course_name': data['course_name'],
           'author': data['author'],
           'price': price,
           'ratings': ratings,
         });
       }
     }
   });
   setState(() {
     wishlist =
         loadedWishlist; // Update the state variable with the loaded wishlist
   });
 } catch (e) {
   print('Failed to load wishlist: $e');
   ScaffoldMessenger.of(context).showSnackBar(
     SnackBar(
         content: Text(
             'Failed to load wishlist: $e')), // Display a more descriptive error message
   );
 }
}

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

```
return Scaffold(
  appBar: AppBar(
    title: Text('Wishlist', style: TextStyle(color: Colors.white)),
    backgroundColor: Colors.black,
  ),
  backgroundColor: Colors.black,
  body: ListView.builder(
    itemCount: wishlist.length,
    itemBuilder: (context, index) {
      final double? ratings = wishlist[index]['ratings'] as double?;
      return ListTile(
        // leading: Image.asset('assets/images/course_image.jpg'),
        title: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(wishlist[index]['course_name'],
                style: TextStyle(color: Colors.white, fontWeight: FontWeight.bold, fontSize: 18.0)),
            SizedBox(height: 5),
            Text('${wishlist[index]['author']}',
                style: TextStyle(color: Colors.grey.shade300, fontSize: 15.0)),
            SizedBox(height: 5),
            Row(
              children: [
                Text(
                  '\$${wishlist[index]['price']}',
                  style: TextStyle(color: Colors.white, fontWeight: FontWeight.bold),
                ),
                SizedBox(width: 10), // Add some space between price and star icons
                Row(
                  children: List.generate(5, (starIndex) {
                    // Generate star icons based on ratings
                    final bool isFilledStar = starIndex < (ratings ?? 0).floor();
                    return Icon(
                      isFilledStar ? Icons.star : Icons.star_border,
                      color: Colors.yellow,
                      size: 20.0,
                    );
                  }),
                ),
                SizedBox(width: 5), // Add some space between star icons and number of ratings
                Text(
                  ratings != 0.0 ? '(${ratings!.toStringAsFixed(1)} ratings)' : '', // Display ratings if not 0
                  style: TextStyle(color: Colors.white),
                ),
              ],
            ),
          ],
        ),
        trailing: IconButton(
          icon: Icon(Icons.delete, color: Colors.white),
          onPressed: () {
            removeFromWishlist(context, wishlist[index]['id']);
          },
        ),
      );
    },

   ),
  );
 }

 void removeFromWishlist(BuildContext context, String documentID) async {
  try {
   CollectionReference wishlistCollection =
   FirebaseFirestore.instance.collection('Wishlist');
   await wishlistCollection.doc(documentID).delete();
   ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(content: Text('Course removed from wishlist')),
   );
   loadWishlist();
  } catch (e) {
   ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(content: Text('Failed to remove course from wishlist: $e')),
   );
  }
 }
}
```

Output:

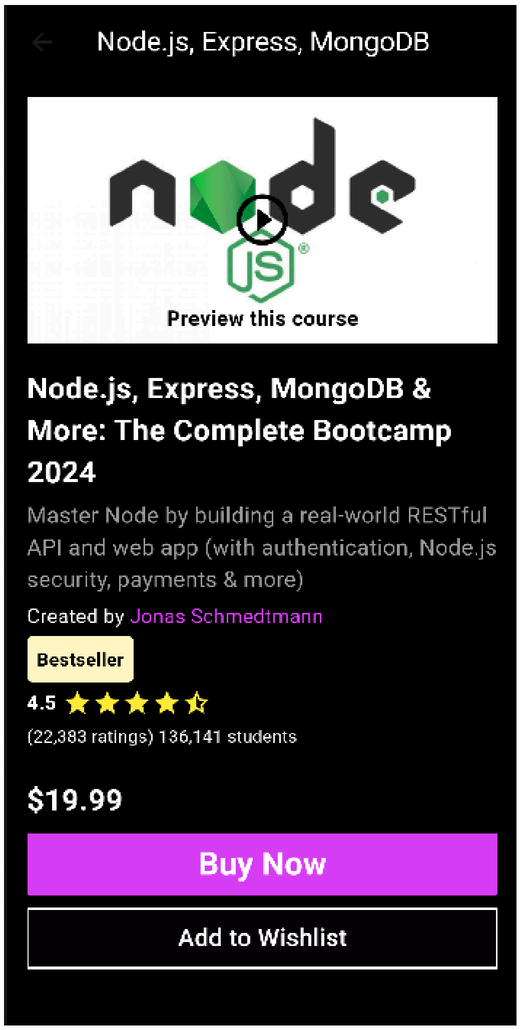

9:12
Sign In
By using our services you are agreeing to our Terms and Privacy Statement
aaditivsurve@gmail.com
••••••
Sign In
Sign in with Google
New here? Create an account


9:13
Create an account
By clicking on 'Create account' you are agreeing to our Terms of Use and Privacy Policy
Username
Email
Password
Create account
Have an account? Sign In



Identifier	Providers	Created	Signed In	User UID
aaditivsurve1@gmail.c...	✉	Feb 9, 2024	Feb 9, 2024	m40fSFJoHWgRC5tcVlwGJ2G...
aaditivsurve@gmail.com	✉	Feb 8, 2024	Feb 15, 2024	oMommniHdAWz99KnqLTKJi...
2021.aaditi.surve@ves....	G	Feb 8, 2024	Feb 13, 2024	fNeXF4BD0WWsK3cKBBAWgl...

Rows per page: 50 1 – 3 of 3



Node.js, Express, MongoDB

Node.js, Express, MongoDB & More: The Complete Bootcamp 2024

Master Node by building a real-world RESTful API and web app (with authentication, Node.js security, payments & more)

Created by Jonas Schmedtmann

Bestseller

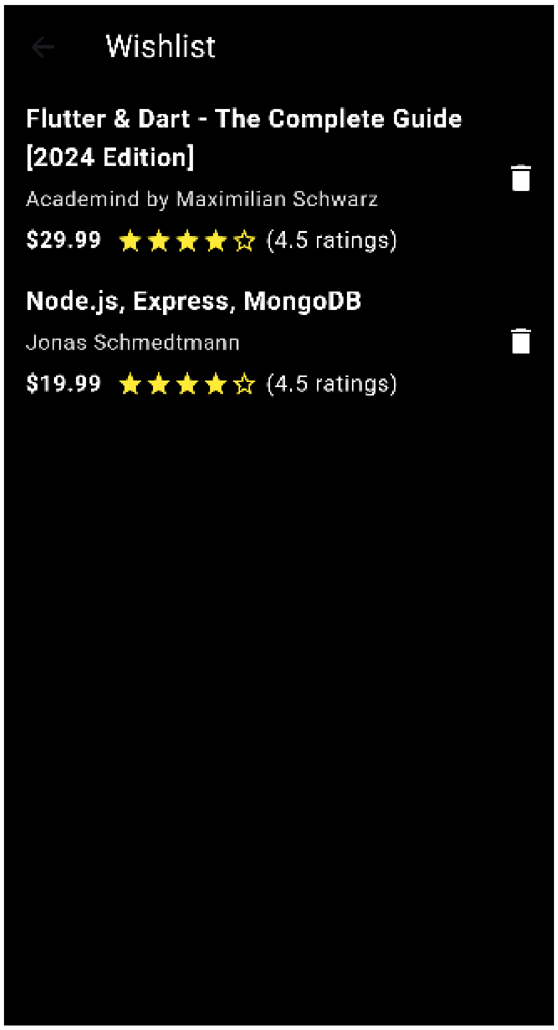
4.5 ★★★★☆

(22,383 ratings) 136,141 students

$19.99

Buy Now

Add to Wishlist



Wishlist

Flutter & Dart - The Complete Guide [2024 Edition]

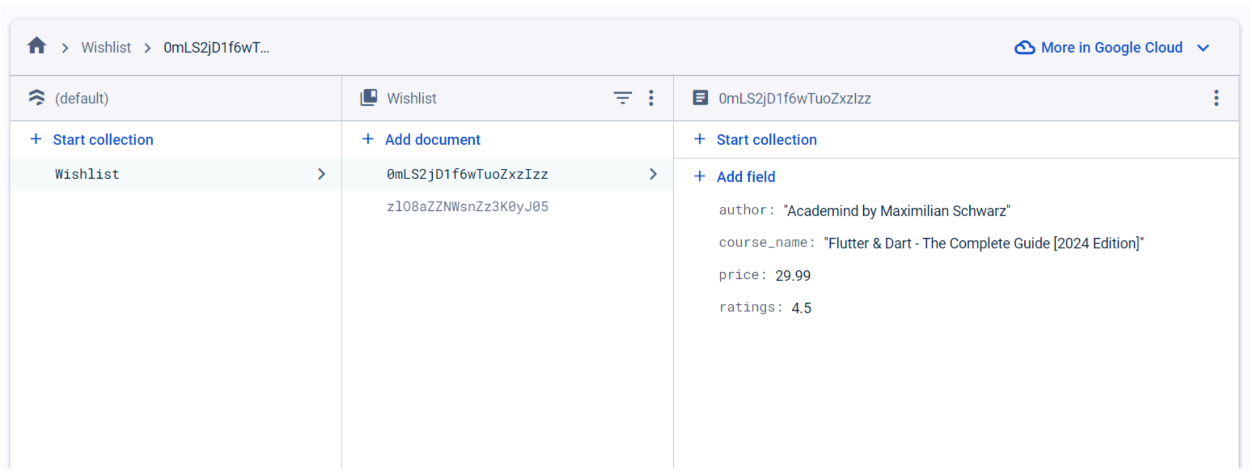
Academind by Maximilian Schwarz

$29.99 ★★★★☆ (4.5 ratings)

Node.js, Express, MongoDB

Jonas Schmedtmann

$19.99 ★★★★☆ (4.5 ratings)



Wishlist > 0mLS2jD1f6wT... More in Google Cloud

(default)	Wishlist	0mLS2jD1f6wTuoZxzIzz
+ Start collection	+ Add document	+ Start collection
Wishlist	0mLS2jD1f6wTuoZxzIzz	+ Add field
	zl08aZZNWsnZz3K0yJ05	author: "Academind by Maximilian Schwarz"
		course_name: "Flutter & Dart - The Complete Guide [2024 Edition]"
		price: 29.99
		ratings: 4.5