Nayaab Jindani D15C Rollno. 20

EXP 6

Aim: To connect flutter UI with firebase database

Theory:

Connecting Flutter UI with Firebase allows real-time cloud-based data management, enabling users to store, retrieve, and update data seamlessly from the app interface.

Key Concepts:

- Firebase Integration: The firebase_core package is used to initialize Firebase, while cloud_firestore connects the app to Firestore, Firebase's NoSQL cloud database.
- Data Structure: Firestore organizes data as collections (e.g., "journalEntries") and documents (each journal entry). This flexible, document-based structure suits dynamic apps like journals, chats, and to-do lists.
- Data Operations:
  - add() is used to insert new documents.
  - get() retrieves data.
  - update() modifies existing data.
  - delete() removes data from the collection.
- Real-Time Updates: Firestore supports real-time data syncing. Any changes made in the database can instantly reflect in the app using listeners like StreamBuilder.
- Security & Rules: Firebase provides authentication and Firestore security rules to control read/write access based on user roles or login status.
- Scalability: Being a cloud solution, Firebase scales automatically and is ideal for apps needing consistent performance and online syncing.

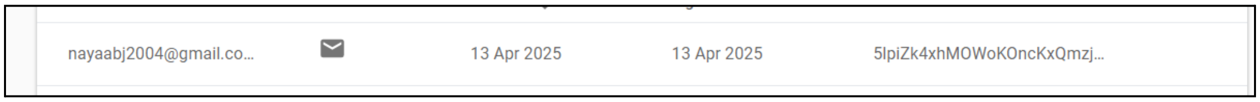
Code:

Signup screen:

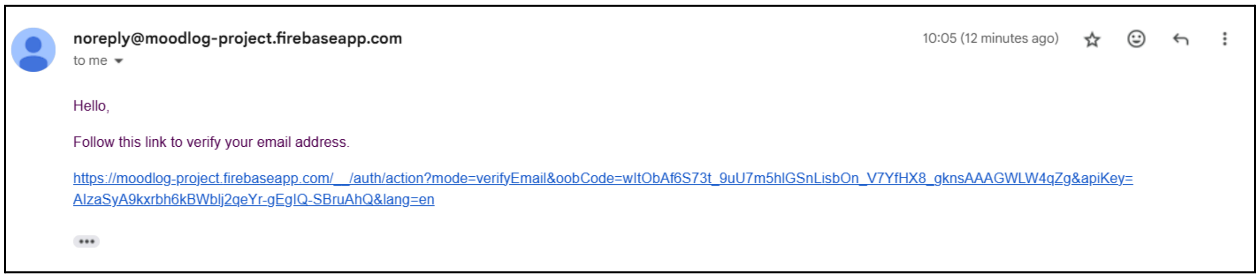
```
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart'; // Import Firebase Authentication package

class SignupScreen extends StatefulWidget {
  const SignupScreen({super.key});

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

class _SignupScreenState extends State<SignupScreen> {
  // Controllers for TextFields
  final TextEditingController _nameController = TextEditingController();
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();

  // FirebaseAuth instance
  final FirebaseAuth _auth = FirebaseAuth.instance;

  // Sign Up method
  Future<void> _signUp() async {
    try {
      // Create user with email and password
      final UserCredential userCredential = await _auth.createUserWithEmailAndPassword(
        email: _emailController.text,
        password: _passwordController.text,
      );

      // Get the user object
      User? user = userCredential.user;

      // Send a verification email
      await user?.sendEmailVerification();

      // Notify the user
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(
          content: Text("Verification email sent! Please check your inbox."),
        ),
      );

      // Optionally navigate to the login screen
      Navigator.pop(context);
    } catch (e) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text("Error: ${e.toString()}"),
        ),
      );
    }
  }
```

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Padding(
      padding: const EdgeInsets.all(24.0),
      child: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text(
              "Create an Account",
              style: TextStyle(
                fontSize: 28,
                fontWeight: FontWeight.bold,
                color: Colors.pinkAccent,
              ),
            ),
            const SizedBox(height: 30),

            // Full Name Field
            TextField(
              controller: _nameController,
              decoration: InputDecoration(
                labelText: "Full Name",
                border: OutlineInputBorder(),
              ),
            ),
            const SizedBox(height: 16),

            // Email Field
            TextField(
              controller: _emailController,
              decoration: InputDecoration(
                labelText: "Email",
                border: OutlineInputBorder(),
              ),
            ),
            const SizedBox(height: 16),

            // Password Field
            TextField(
              controller: _passwordController,
              obscureText: true,
              decoration: InputDecoration(
                labelText: "Password",
                border: OutlineInputBorder(),
              ),
            ),
            const SizedBox(height: 24),

            // Sign Up Button
            ElevatedButton(
              onPressed: _signUp,
              style: ElevatedButton.styleFrom(
                backgroundColor: Colors.pinkAccent,
                foregroundColor: Colors.white,
                padding: const EdgeInsets.symmetric(vertical: 14, horizontal: 50),
                shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.circular(20),
                ),
              ),
              child: Text("Sign Up", style: TextStyle(fontSize: 18)),
            ),

            const SizedBox(height: 20),

            // Navigate to Login Screen
            TextButton(
              onPressed: () {
                Navigator.pop(context); // Go back to Login screen
              },
              child: Text(
                "Already have an account? Login",
                style: TextStyle(color: Colors.pinkAccent),
              ),
            ),
          ],
        ),
      ),
    ),
  );
}
}
```

Output:



nayaabj2004@gmail.co...   13 Apr 2025   13 Apr 2025   5lpiZk4xhMOWoKOncKxQmzj...

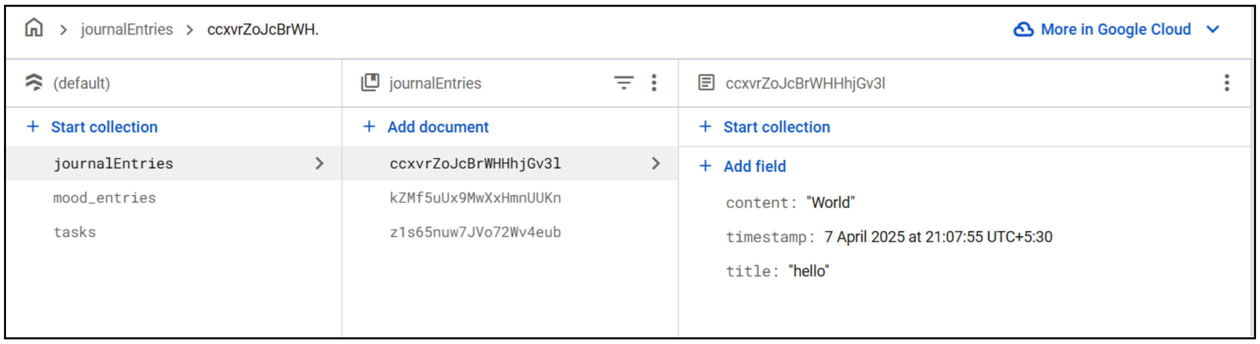


noreply@moodlog-project.firebaseapp.com
to me
10:05 (12 minutes ago)

Hello,

Follow this link to verify your email address.

https://moodlog-project.firebaseapp.com/__/auth/action?mode=verifyEmail&oobCode=wItObAf6S73t_9uU7m5hlGSnLisbOn_V7YfHX8_gknsAAAGWLW4qZg&apiKey=AIzaSyA9kxrbh6kBWblj2qeYr-gEgIQ-SBruAhQ&lang=en



journalEntries
mood_entries
tasks

ccxvrZoJcBrWHHhjGv3l
kZMf5uUx9MwXxHmnUUKn
z1s65nuw7JVo72Wv4eub

content: "World"
timestamp: 7 April 2025 at 21:07:55 UTC+5:30
title: "hello"

Conclusion:
By connecting Flutter UI with Firebase Database, the app achieves real-time data management and secure user authentication. Using Firebase Authentication ensures safe login and registration, while Firestore enables efficient storage and retrieval of user data. This integration simplifies backend operations and enhances the responsiveness and interactivity of the Flutter application.