Name : Vivek Gupta
Div : D15B
Roll No : 19

**MPL Practical 06**

**Aim: To integrate Firebase Authentication in a Flutter app.**

**Theory:**

**Introduction**
Firebase provides a backend-as-a-service (BaaS), making it easy to handle user authentication and real-time database operations. Firebase Authentication allows secure user login and signup using email/password. With Firebase, we can:

- Register users and store their credentials.
- Authenticate users securely using FirebaseAuth.
- Verify emails to ensure valid registrations.

Flutter interacts with Firebase using the firebase_auth package, which provides methods for signup, login, logout, and authentication state management.

**Implementation in Our Code**

1. Signup Page (signup_screen.dart)
    - Allows users to create an account using email and password.
    - Sends an email verification after signup.
    - Displays messages to inform users about signup success or errors.

2. Login Page (login_screen.dart)
    - Allows users to log in if their credentials are correct.
    - Checks if the email is verified before granting access.
    - Displays proper error messages in case of invalid credentials.

3. Authentication Service (auth_service.dart)
    - Manages signup, login, logout, and email verification.
    - Uses Firebase's authentication API for secure user management.

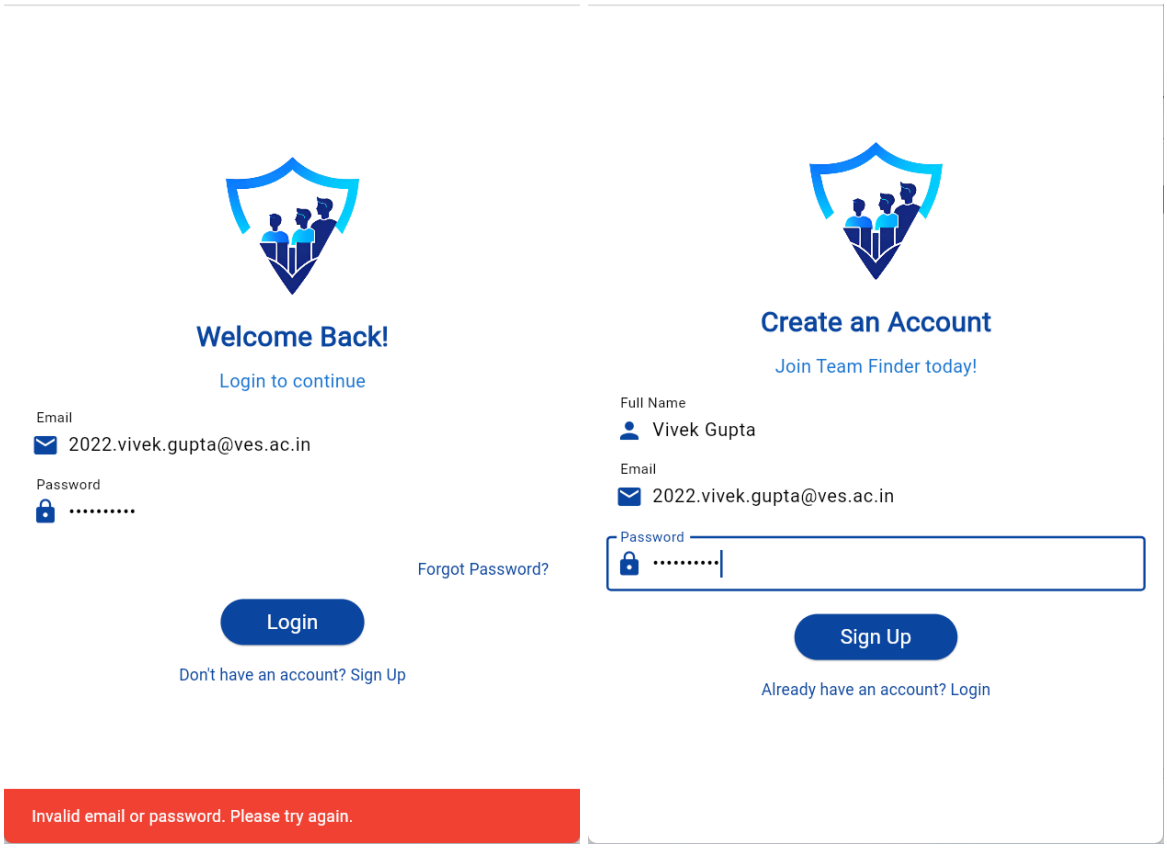
**auth_service.dart**

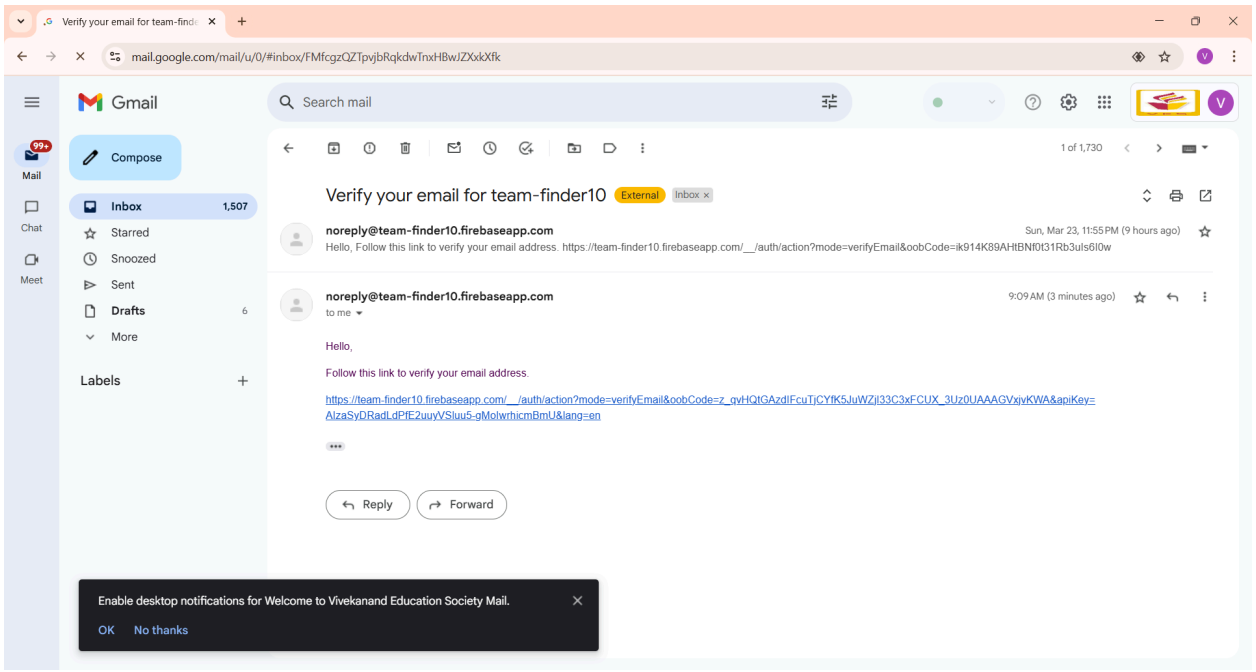
```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:google_sign_in/google_sign_in.dart';

class AuthService {
 final FirebaseAuth _auth = FirebaseAuth.instance;
 final GoogleSignIn _googleSignIn = GoogleSignIn(
 );

 // Sign Up with Email & Password + Store User in Firestore
```

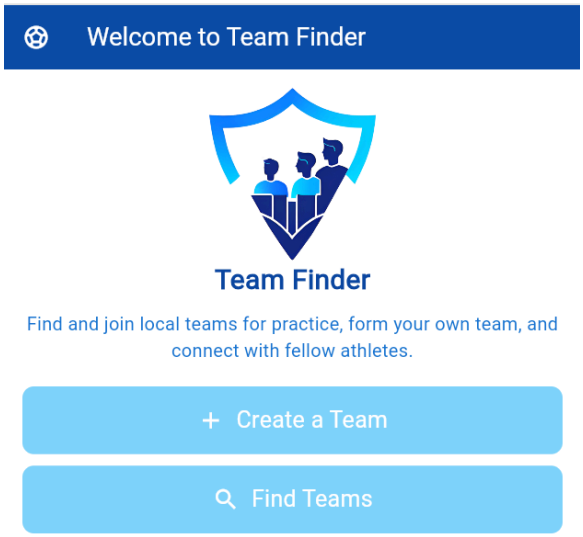
```
Future<User?> signUp(String email, String password, String fullName) async {
 try {
  UserCredential userCredential = await _auth.createUserWithEmailAndPassword(
   email: email,
   password: password,
  );

  User? user = userCredential.user;
  if (user != null) {
   await FirebaseFirestore.instance.collection('users').doc(user.uid).set({
    'uid': user.uid,
    'fullName': fullName,
    'email': email,
    'joinedTeams': [],
   });

   await user.sendEmailVerification();
   print("Verification email sent to: ${user.email}");
  }
  return user;
 } catch (e) {
  print("Sign Up Error: $e");
  return null;
 }
}

// Login with Email & Password (Only If Verified)
Future<User?> signIn(String email, String password) async {
 try {
  UserCredential userCredential = await _auth.signInWithEmailAndPassword(
   email: email,
   password: password,
  );

  if (!userCredential.user!.emailVerified) {
   print("Email not verified. Please check your inbox.");
   return null;
  }

  print("Login Successful!");
  return userCredential.user;
```

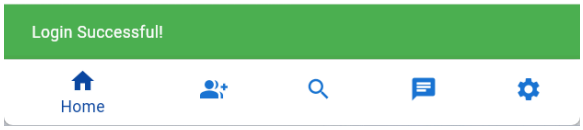
```
  } catch (e) {
    print("Login Error: $e");
    return null;
  }
 }

 // Get current user
 User? getCurrentUser() {
  return _auth.currentUser;
 }

 // Reset Password
 Future<bool> resetPassword(String email) async {
  try {
    await _auth.sendPasswordResetEmail(email: email);
    return true;
  } catch (e) {
    print("Password Reset Error: $e");
    return false;
  }
 }

 // Google Sign-In + Store User in Firestore if New
 Future<User?> signInWithGoogle() async {
  try {
    final GoogleSignInAccount? googleUser = await _googleSignIn.signIn();
    if (googleUser == null) return null;

    final GoogleSignInAuthentication googleAuth = await googleUser.authentication;
    final AuthCredential credential = GoogleAuthProvider.credential(
      accessToken: googleAuth.accessToken,
      idToken: googleAuth.idToken,
    );

    UserCredential userCredential = await _auth.signInWithCredential(credential);
    User? user = userCredential.user;

    if (user != null) {
      // Check if user already exists in Firestore
                                                DocumentSnapshot userDoc = await
FirebaseFirestore.instance.collection('users').doc(user.uid).get();
```

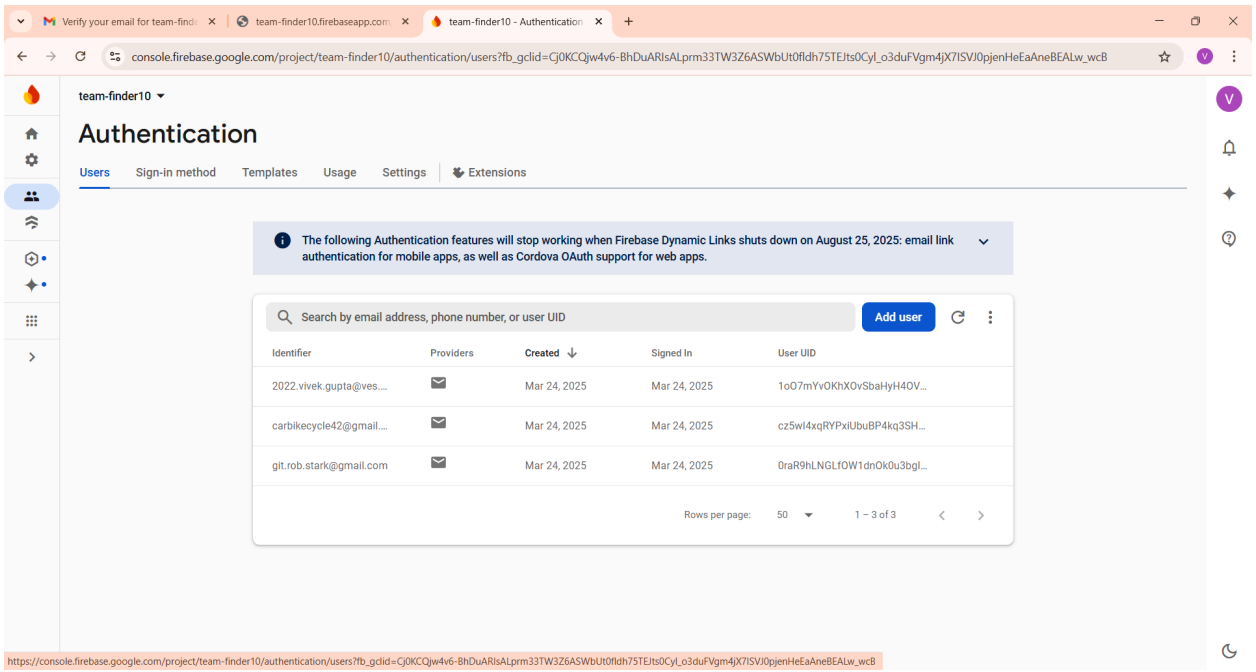
```
      if (!userDoc.exists) {
        // If new user, store their details in Firestore
        await FirebaseFirestore.instance.collection('users').doc(user.uid).set({
          'uid': user.uid,
          'fullName': user.displayName ?? "No Name",
          'email': user.email,
          'joinedTeams': [],
        });
      }
    }
    return user;
  } catch (e) {
    print("Google Sign-In Error: $e");
    return null;
  }
 }

 // Sign Out (Google & Email)
 Future<void> signOut() async {
  await _auth.signOut();
  await _googleSignIn.signOut();
 }
}
```


Welcome Back!
Login to continue
Email
2022.vivek.gupta@ves.ac.in
Password
Forgot Password?
Login
Don't have an account? Sign Up
Create an Account
Join Team Finder today!
Full Name
Vivek Gupta
Email
2022.vivek.gupta@ves.ac.in
Password
Sign Up
Already have an account? Login
Invalid email or password. Please try again.


Verify your email for team-finder10
mail.google.com/mail/u/0/#inbox/FMfcgzQZTpvjbRqkdwTnxHBwJZXxkXfk
Gmail
Search mail
Compose
Inbox 1,507
Starred
Snoozed
Sent
Drafts 6
More
Labels
Verify your email for team-finder10 External Inbox
noreply@team-finder10.firebaseapp.com
Sun, Mar 23, 11:55 PM (9 hours ago)
Hello, Follow this link to verify your email address. https://team-finder10.firebaseapp.com/__/auth/action?mode=verifyEmail&oobCode=ik914K89AHtBNf0t31Rb3uls6l0w
noreply@team-finder10.firebaseapp.com
9:09 AM (3 minutes ago)
to me
Hello,
Follow this link to verify your email address.
https://team-finder10.firebaseapp.com/__/auth/action?mode=verifyEmail&oobCode=z_qvHQtGAzdIFcuTjCYfK5JuWZjl33C3xFCUX_3Uz0UAAAGVxjvKWA&apiKey=AIzaSyDRadLdPfE2uuyVSluu5-gMolwrhicmBmU&lang=en
Reply
Forward
Enable desktop notifications for Welcome to Vivekanand Education Society Mail.
OK
No thanks




**Conclusion:**

In this experiment, we successfully implemented navigation using named routes and integrated gestures for user interaction in the Team Finder app. Initially, we faced issues with incorrect route navigation and unresponsive gestures, which we resolved by debugging route names and ensuring GestureDetector was properly wrapped around interactive widgets.