**Models Folder**

**Booikng**

class Booking {

  final String bookingDate;

  final String eventId;

  final String paymentMethod;

  final String paymentType;

  final String status;

  final String transactionId;

  final String userId;

  Booking({

    required this.bookingDate,

    required this.eventId,

    required this.paymentMethod,

    required this.paymentType,

    required this.status,

    required this.transactionId,

    required this.userId,

  });

  Map<String, dynamic> toMap() {

    return {

      'booking\_date': bookingDate,

      'event\_id': eventId,

      'payment\_method': paymentMethod,

      'payment\_type': paymentType,

      'status': status,

      'transaction\_id': transactionId,

      'user\_id': userId,

    };

  }

}

**User\_mode**

class UserModel {

  final String uid;

  final String email;

  final String role;

  final String? address;

  final String? department;

  final String? name;

  final String? phone;

  final String? semester;

  final String? studentId;

  UserModel({

    required this.uid,

    required this.email,

    this.role = 'user',

    this.address,

    this.department,

    this.name,

    this.phone,

    this.semester,

    this.studentId,

  });

  factory UserModel.fromMap(Map<String, dynamic> data) {

    return UserModel(

      uid: data['uid'],

      email: data['email'],

      role: data['role'] ?? 'user',

      address: data['address'],

      department: data['department'],

      name: data['name'],

      phone: data['phone'],

      semester: data['semester'],

      studentId: data['student\_id'],

    );

  }

  Map<String, dynamic> toMap() {

    return {

      'uid': uid,

      'email': email,

      'role': role,

      'address': address,

      'department': department,

      'name': name,

      'phone': phone,

      'semester': semester,

      'student\_id': studentId,

    };

  }

}

**Auth folder**

**Auth raper**

import 'package:flutter/material.dart';

import '../../services/auth\_service.dart';

import '../home/home\_screen.dart';

import '../login\_screen.dart';

class AuthWrapper extends StatelessWidget {

  const AuthWrapper({super.key});

  @override

  Widget build(BuildContext context) {

    return FutureBuilder<String?>(

      future: AuthService.getUID(),

      builder: (context, snapshot) {

        if (snapshot.connectionState == ConnectionState.waiting) {

          return const Center(child: CircularProgressIndicator());

        }

        if (snapshot.hasData && snapshot.data != null) {

          return HomeScreen(uid: snapshot.data!);

        }

        return const LoginScreen();

      },

    );

  }

}

**Forget\_password\_screen**

import 'package:flutter/material.dart';

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:fluttertoast/fluttertoast.dart';

class ForgotPasswordScreen extends StatefulWidget {

  const ForgotPasswordScreen({super.key});

  @override

  State<ForgotPasswordScreen> createState() => \_ForgotPasswordScreenState();

}

class \_ForgotPasswordScreenState extends State<ForgotPasswordScreen> {

  final \_emailController = TextEditingController();

  final \_formKey = GlobalKey<FormState>();

  bool \_isLoading = false;

  Future<void> \_resetPassword() async {

    if (!\_formKey.currentState!.validate()) return;

    setState(() => \_isLoading = true);

    try {

      await FirebaseAuth.instance.sendPasswordResetEmail(

        email: \_emailController.text.trim(),

      );

      if (!mounted) return;

      ScaffoldMessenger.of(context).showSnackBar(

        const SnackBar(

          content: Text("Password reset email sent! Check your inbox."),

          backgroundColor: Colors.green,

        ),

      );

      Navigator.pop(context);

    } on FirebaseAuthException catch (e) {

      if (!mounted) return;

      ScaffoldMessenger.of(context).showSnackBar(

        SnackBar(

          content: Text(e.message ?? "An error occurred"),

          backgroundColor: Colors.red,

        ),

      );

    } finally {

      if (mounted) setState(() => \_isLoading = false);

    }

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('Reset Password'),

        backgroundColor: Colors.purple[200],

      ),

      body: Padding(

        padding: const EdgeInsets.all(16.0),

        child: Form(

          key: \_formKey,

          child: Column(

            mainAxisAlignment: MainAxisAlignment.center,

            children: [

              Text(

                'Enter your email to reset password',

                style: TextStyle(fontSize: 16, color: Colors.grey[700]),

              ),

              const SizedBox(height: 24),

              TextFormField(

                controller: \_emailController,

                keyboardType: TextInputType.emailAddress,

                decoration: InputDecoration(

                  labelText: 'Email',

                  border: OutlineInputBorder(

                    borderRadius: BorderRadius.circular(12),

                  ),

                  prefixIcon: const Icon(Icons.email),

                ),

                validator: (value) {

                  if (value == null || value.isEmpty) {

                    return 'Please enter your email';

                  }

                  if (!value.contains('@')) {

                    return 'Please enter a valid email';

                  }

                  return null;

                },

              ),

              const SizedBox(height: 24),

              SizedBox(

                width: double.infinity,

                height: 50,

                child: ElevatedButton(

                  onPressed: \_isLoading ? null : \_resetPassword,

                  style: ElevatedButton.styleFrom(

                    backgroundColor: Colors.blue,

                    foregroundColor: Colors.white,

                    shape: RoundedRectangleBorder(

                      borderRadius: BorderRadius.circular(12),

                    ),

                  ),

                  child:

                      \_isLoading

                          ? const CircularProgressIndicator(color: Colors.white)

                          : const Text(

                            'Send Reset Link',

                            style: TextStyle(fontSize: 16),

                          ),

                ),

              ),

            ],

          ),

        ),

      ),

    );

  }

  @override

  void dispose() {

    \_emailController.dispose();

    super.dispose();

  }

}

**Splash\_screen**

import 'dart:async';

import 'dart:ui';

import 'package:flutter/material.dart';

import 'login\_screen.dart';

import 'onboarding\_screen.dart';

import 'auth/auth\_wrapper.dart';

class SplashScreen extends StatefulWidget {

  const SplashScreen({super.key});

  @override

  State<SplashScreen> createState() => \_SplashScreenState();

}

class \_SplashScreenState extends State<SplashScreen>

    with SingleTickerProviderStateMixin {

  late AnimationController \_controller;

  late Animation<double> \_animation;

  double \_progress = 0.0;

  @override

  void initState() {

    super.initState();

    // Initialize animation controller

    \_controller = AnimationController(

      duration: const Duration(seconds: 2),

      vsync: this,

    );

    // Create scale animation

    \_animation = Tween<double>(

      begin: 0.5,

      end: 1.0,

    ).animate(CurvedAnimation(parent: \_controller, curve: Curves.easeInOut));

    // Start animation

    \_controller.forward();

    // Animate progress bar

    Timer.periodic(const Duration(milliseconds: 50), (timer) {

      if (\_progress < 1.0) {

        setState(() {

          \_progress += 0.02;

        });

      } else {

        timer.cancel();

        \_navigateToNextScreen();

      }

    });

  }

  Future<void> \_navigateToNextScreen() async {

    // Check if it's first time launch

    bool isFirstTime = true; // You can use shared preferences to check this

    if (!mounted) return;

    Navigator.pushReplacement(

      context,

      MaterialPageRoute(

        builder:

            (context) =>

                isFirstTime ? const OnboardingScreen() : const AuthWrapper(),

      ),

    );

  }

  @override

  void dispose() {

    \_controller.dispose();

    super.dispose();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: Container(

        decoration: const BoxDecoration(

          image: DecorationImage(

            image: AssetImage('assets/images/Splashscreen.png'),

            fit: BoxFit.cover,

          ),

        ),

        child: Container(

          decoration: BoxDecoration(

            gradient: LinearGradient(

              begin: Alignment.topCenter,

              end: Alignment.bottomCenter,

              colors: [

                Colors.black.withOpacity(0.7),

                Colors.black.withOpacity(0.5),

              ],

            ),

          ),

          child: Center(

            child: Column(

              mainAxisAlignment: MainAxisAlignment.center,

              children: [

                ScaleTransition(

                  scale: \_animation,

                  child: Container(

                    width: 120,

                    height: 120,

                    decoration: BoxDecoration(

                      shape: BoxShape.circle,

                      image: const DecorationImage(

                        image: AssetImage('assets/icons/diu\_logo.jpg'),

                        fit: BoxFit.cover,

                      ),

                      boxShadow: [

                        BoxShadow(

                          color: Colors.white.withOpacity(0.2),

                          spreadRadius: 5,

                          blurRadius: 15,

                        ),

                      ],

                    ),

                  ),

                ),

                const SizedBox(height: 30),

                ClipRRect(

                  borderRadius: BorderRadius.circular(20),

                  child: BackdropFilter(

                    filter: ImageFilter.blur(sigmaX: 10, sigmaY: 10),

                    child: Container(

                      padding: const EdgeInsets.symmetric(

                        horizontal: 25,

                        vertical: 15,

                      ),

                      decoration: BoxDecoration(

                        borderRadius: BorderRadius.circular(20),

                        color: Colors.white.withOpacity(0.15),

                      ),

                      child: const Text(

                        "DIU Event Manager",

                        style: TextStyle(

                          fontSize: 32,

                          fontWeight: FontWeight.bold,

                          color: Colors.white,

                          letterSpacing: 1.5,

                          shadows: [

                            Shadow(

                              offset: Offset(2, 2),

                              blurRadius: 3.0,

                              color: Color.fromRGBO(0, 0, 0, 0.5),

                            ),

                          ],

                        ),

                      ),

                    ),

                  ),

                ),

                const SizedBox(height: 40),

                SizedBox(

                  width: 200,

                  child: ClipRRect(

                    borderRadius: BorderRadius.circular(10),

                    child: LinearProgressIndicator(

                      value: \_progress,

                      backgroundColor: Colors.white.withOpacity(0.2),

                      valueColor: const AlwaysStoppedAnimation<Color>(

                        Colors.white,

                      ),

                      minHeight: 6,

                    ),

                  ),

                ),

              ],

            ),

          ),

        ),

      ),

    );

  }

}

**Services**

**Auth\_service**

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:google\_sign\_in/google\_sign\_in.dart';

import '../models/user\_model.dart';

import 'package:shared\_preferences/shared\_preferences.dart';

class AuthService {

  final FirebaseAuth \_auth = FirebaseAuth.instance;

  final GoogleSignIn \_googleSignIn = GoogleSignIn();

  final FirebaseFirestore \_firestore = FirebaseFirestore.instance;

  static const String \_uidKey = 'uid';

  // Email & Password Sign In

  Future<User?> signInWithEmail(String email, String password) async {

    try {

      UserCredential result = await \_auth.signInWithEmailAndPassword(

        email: email,

        password: password,

      );

      if (result.user != null) {

        await saveUID(result.user!.uid);

      }

      return result.user;

    } catch (e) {

      print("Sign-in error: ${e.toString()}");

      return null;

    }

  }

  // Store User Data in Firestore

  Future<void> storeUserData(UserModel user) async {

    try {

      await \_firestore.collection('users').doc(user.uid).set(user.toMap());

    } catch (e) {

      print("Firestore write error: ${e.toString()}");

      throw Exception('Failed to store user data: ${e.toString()}');

    }

  }

  // Email & Password Registration

  Future<User?> registerWithEmail(String email, String password) async {

    try {

      UserCredential result = await \_auth.createUserWithEmailAndPassword(

        email: email,

        password: password,

      );

      if (result.user != null) {

        await saveUID(result.user!.uid);

      }

      return result.user;

    } catch (e) {

      print("Registration error: ${e.toString()}");

      return null;

    }

  }

  // Google Sign In

  Future<User?> signInWithGoogle() async {

    try {

      final GoogleSignInAccount? googleUser = await \_googleSignIn.signIn();

      if (googleUser == null) return null;

      final GoogleSignInAuthentication googleAuth =

          await googleUser.authentication;

      if (googleAuth.accessToken == null || googleAuth.idToken == null) {

        throw Exception('Failed to get Google authentication tokens');

      }

      final AuthCredential credential = GoogleAuthProvider.credential(

        accessToken: googleAuth.accessToken!,

        idToken: googleAuth.idToken!,

      );

      UserCredential result = await \_auth.signInWithCredential(credential);

      if (result.user != null) {

        await saveUID(result.user!.uid);

      }

      return result.user;

    } catch (e) {

      print("Google Sign-in error: ${e.toString()}");

      return null;

    }

  }

  // Sign Out

  Future<void> signOut() async {

    try {

      await clearUID();

      await \_auth.signOut();

      await \_googleSignIn.signOut();

    } catch (e) {

      print("Sign-out error: ${e.toString()}");

      throw Exception('Failed to sign out: ${e.toString()}');

    }

  }

  // Save UID to SharedPreferences

  static Future<void> saveUID(String uid) async {

    try {

      final prefs = await SharedPreferences.getInstance();

      await prefs.setString(\_uidKey, uid);

    } catch (e) {

      print("Error saving UID: ${e.toString()}");

      throw Exception('Failed to save UID: ${e.toString()}');

    }

  }

  // Get saved UID

  static Future<String?> getUID() async {

    try {

      final prefs = await SharedPreferences.getInstance();

      return prefs.getString(\_uidKey);

    } catch (e) {

      print("Error getting UID: ${e.toString()}");

      return null;

    }

  }

  // Clear saved UID (for logout)

  static Future<void> clearUID() async {

    try {

      final prefs = await SharedPreferences.getInstance();

      await prefs.remove(\_uidKey);

    } catch (e) {

      print("Error clearing UID: ${e.toString()}");

      throw Exception('Failed to clear UID: ${e.toString()}');

    }

  }

}

**Cloundianary\_service**

import 'dart:io';

import 'package:http/http.dart' as http;

import 'package:path/path.dart';

import 'package:mime/mime.dart';

import 'package:http\_parser/http\_parser.dart';

import 'package:flutter\_dotenv/flutter\_dotenv.dart';

class CloudinaryService {

  late final String cloudName;

  late final String apiKey;

  late final String apiSecret;

  CloudinaryService() {

    dotenv.load(fileName: ".env"); // ✅ Load environment variables here

    cloudName = dotenv.env['CLOUDINARY\_CLOUD\_NAME']!;

    apiKey = dotenv.env['CLOUDINARY\_API\_KEY']!;

    apiSecret = dotenv.env['CLOUDINARY\_SECRET']!;

  }

  Future<String?> uploadImage(File imageFile) async {

    try {

      final url = Uri.parse(

        "https://api.cloudinary.com/v1\_1/$cloudName/image/upload",

      );

      var request = http.MultipartRequest("POST", url);

      request.fields["upload\_preset"] = "ml\_default";

      request.fields["api\_key"] = apiKey;

      request.fields["timestamp"] =

          DateTime.now().millisecondsSinceEpoch.toString();

      var mimeType = lookupMimeType(imageFile.path) ?? "image/jpeg";

      var fileStream = http.ByteStream(imageFile.openRead());

      var fileLength = await imageFile.length();

      var multipartFile = http.MultipartFile(

        "file",

        fileStream,

        fileLength,

        filename: basename(imageFile.path),

        contentType: MediaType.parse(mimeType),

      );

      request.files.add(multipartFile);

      var response = await request.send();

      var responseBody = await response.stream.bytesToString();

      if (response.statusCode == 200) {

        return responseBody; // ✅ Successfully uploaded

      } else {

        print("Failed to upload image: ${response.reasonPhrase}");

        return null;

      }

    } catch (e) {

      print("Error uploading image: $e");

      return null;

    }

  }

}