|  |  |
| --- | --- |
| **Experiment** | 8 |
| **Aim** | Implement user authentication and registration using database with flutter |
| **Objective** | * Create Registration page * Create database * Store, fetch, delete values from database. * User Authentication from database |
| **Name** | Atharva Vasant Angre |
| **UCID** | 2024510001 |
| **Class** | FYMCA |
| **Batch** | A |
| **Date of Submission** | 21.04.2025 |

|  |  |
| --- | --- |
| **Technology used** | **Flutter, Firbase** |
| **Task** | Create a basic registration page and store data n database. Fetch and also delete few entries from database. You can use either firebase database or postgresql. |
| **Code with proper label** | Main.dart import 'package:flutter/material.dart'; import 'package:get/get.dart'; import 'package:firebase\_core/firebase\_core.dart'; import 'auth\_controller.dart'; import 'login\_page.dart'; import 'firebase\_options.dart';  void main() async {  WidgetsFlutterBinding.*ensureInitialized*();  await Firebase.*initializeApp*(  options: DefaultFirebaseOptions.*currentPlatform*,  );  Get.put(AuthController());   runApp(const MyApp()); }  class MyApp extends StatelessWidget {  const MyApp({super.key});  @override  Widget build(BuildContext context) {  return GetMaterialApp(  debugShowCheckedModeBanner: false,  initialBinding: BindingsBuilder(() {  Get.lazyPut<AuthController>(() => AuthController());  }),  title: 'Auth App',  theme: ThemeData(  useMaterial3: true,  colorSchemeSeed: Colors.*deepPurple*,  fontFamily: 'Poppins',  ),  home: const LoginPage(),  );  } }  authController.dart import 'package:cloud\_firestore/cloud\_firestore.dart'; import 'package:firebase\_auth/firebase\_auth.dart'; import 'package:get/get.dart'; import 'package:google\_sign\_in/google\_sign\_in.dart'; import 'package:lab\_8/home\_page.dart'; import 'package:shared\_preferences/shared\_preferences.dart'; import 'login\_page.dart';  class AuthController extends GetxController {  static AuthController get *to* => Get.find();   final auth = FirebaseAuth.*instance*;  final firestore = FirebaseFirestore.*instance*;  late SharedPreferences \_prefs;   @override  void onInit() {  super.onInit();  // Initialize SharedPreferences  \_initPrefs();  // Listen to auth state changes  auth.authStateChanges().listen((User? user) {  if (user != null) {  // User is signed in  \_saveLoginStatus(true);  if (Get.currentRoute != '/home') {  Get.offAll(() => const HomePage());  }  } else {  // User is signed out  \_saveLoginStatus(false);  if (Get.currentRoute != '/login') {  Get.offAll(() => const LoginPage());  }  }  });  }   Future<void> \_initPrefs() async {  try {  \_prefs = await SharedPreferences.*getInstance*();  // Check if user is already logged in  checkLoginStatus();  } catch (e) {  print("Error initializing SharedPreferences: $e");  }  }   Future<void> checkLoginStatus() async {  try {  final isLoggedIn = \_prefs.getBool('isLoggedIn') ?? false;    if (isLoggedIn && auth.currentUser != null) {  if (Get.currentRoute != '/home') {  Get.offAll(() => const HomePage());  }  } else {  if (Get.currentRoute != '/login') {  Get.offAll(() => const LoginPage());  }  }  } catch (e) {  print("Error checking login status: $e");  }  }   Future<void> \_saveLoginStatus(bool status) async {  try {  await \_prefs.setBool('isLoggedIn', status);  } catch (e) {  print("Error saving login status: $e");  }  }   Future<void> signInWithGoogle() async {  try {  // Trigger the authentication flow  final GoogleSignInAccount? googleUser = await GoogleSignIn().signIn();   if (googleUser == null) {  Get.snackbar("Cancelled", "Google sign-in cancelled");  return;  }   // Obtain the auth details from the request  final GoogleSignInAuthentication googleAuth = await googleUser.authentication;   // Create a new credential  final credential = GoogleAuthProvider.*credential*(  accessToken: googleAuth.accessToken,  idToken: googleAuth.idToken,  );   // Sign in to Firebase with the Google [UserCredential]  await auth.signInWithCredential(credential);  await \_saveLoginStatus(true);   Get.snackbar("Success", "Logged in with Google!");  Get.offAll(() => const HomePage());  } catch (e) {  print("signInWithGoogle: $e");  Get.snackbar("Error", e.toString());  }  }   void registerUser(String name, String email, String password) async {  try {  UserCredential userCred = await auth.createUserWithEmailAndPassword(  email: email, password: password);   await firestore.collection("users").doc(userCred.user!.uid).set({  "name": name,  "email": email,  "createdAt": DateTime.now(),  });   await \_saveLoginStatus(true);   Get.snackbar("Success", "Account created successfully!");  Get.offAll(() => const HomePage());   } catch (e) {  print("registerUser: $e");  Get.snackbar("Error", e.toString());  }  }   void loginUser(String email, String password) async {  try {  await auth.signInWithEmailAndPassword(email: email, password: password);  await \_saveLoginStatus(true);    Get.snackbar("Success", "Login successful");  Get.offAll(() => const HomePage());  } catch (e) {  print("loginUser: $e");  Get.snackbar("Error", e.toString());  }  }   void logout() async {  try {  await auth.signOut();  await \_saveLoginStatus(false);  Get.offAll(() => const LoginPage());  } catch (e) {  print("logout: $e");  Get.snackbar("Error", e.toString());  }  } }  registerPage.dart import 'package:flutter/material.dart'; import 'package:get/get.dart'; import 'package:google\_fonts/google\_fonts.dart'; import 'package:flutter\_animate/flutter\_animate.dart'; import 'auth\_controller.dart'; import 'login\_page.dart';  class RegisterPage extends StatelessWidget {  const RegisterPage({super.key});   @override  Widget build(BuildContext context) {  final nameController = TextEditingController();  final emailController = TextEditingController();  final passwordController = TextEditingController();   return Scaffold(  body: SafeArea(  child: SingleChildScrollView(  child: Padding(  padding: const EdgeInsets.all(24.0),  child: Column(  crossAxisAlignment: CrossAxisAlignment.stretch,  children: [  const SizedBox(height: 40),  Text(  "Create Account",  style: GoogleFonts.*poppins*(  fontSize: 32,  fontWeight: FontWeight.*bold*,  color: Theme.*of*(context).primaryColor,  ),  ).animate().fadeIn().slideX(),  const SizedBox(height: 8),  Text(  "Sign up to get started",  style: GoogleFonts.*poppins*(  fontSize: 16,  color: Colors.*grey*[600],  ),  ).animate().fadeIn().slideX(),  const SizedBox(height: 40),  TextField(  controller: nameController,  decoration: InputDecoration(  labelText: "Full Name",  hintText: "Enter your full name",  prefixIcon: const Icon(Icons.*person\_outline*),  border: OutlineInputBorder(  borderRadius: BorderRadius.circular(12),  ),  enabledBorder: OutlineInputBorder(  borderRadius: BorderRadius.circular(12),  borderSide: BorderSide(color: Colors.*grey*.shade300),  ),  ),  ).animate().fadeIn().slideX(),  const SizedBox(height: 16),  TextField(  controller: emailController,  decoration: InputDecoration(  labelText: "Email",  hintText: "Enter your email",  prefixIcon: const Icon(Icons.*email\_outlined*),  border: OutlineInputBorder(  borderRadius: BorderRadius.circular(12),  ),  enabledBorder: OutlineInputBorder(  borderRadius: BorderRadius.circular(12),  borderSide: BorderSide(color: Colors.*grey*.shade300),  ),  ),  ).animate().fadeIn().slideX(),  const SizedBox(height: 16),  TextField(  controller: passwordController,  obscureText: true,  decoration: InputDecoration(  labelText: "Password",  hintText: "Enter your password",  prefixIcon: const Icon(Icons.*lock\_outline*),  border: OutlineInputBorder(  borderRadius: BorderRadius.circular(12),  ),  enabledBorder: OutlineInputBorder(  borderRadius: BorderRadius.circular(12),  borderSide: BorderSide(color: Colors.*grey*.shade300),  ),  ),  ).animate().fadeIn().slideX(),  const SizedBox(height: 24),  ElevatedButton(  onPressed: () {  AuthController.*to*.registerUser(  nameController.text.trim(),  emailController.text.trim(),  passwordController.text.trim(),  );  },  style: ElevatedButton.*styleFrom*(  padding: const EdgeInsets.symmetric(vertical: 16),  shape: RoundedRectangleBorder(  borderRadius: BorderRadius.circular(12),  ),  ),  child: Text(  "Create Account",  style: GoogleFonts.*poppins*(  fontSize: 16,  fontWeight: FontWeight.*w600*,  ),  ),  ).animate().fadeIn().slideX(),  const SizedBox(height: 16),  TextButton(  onPressed: () => Get.back(),  child: Text(  "Already have an account? Sign In",  style: GoogleFonts.*poppins*(  color: Theme.*of*(context).primaryColor,  ),  ),  ).animate().fadeIn().slideX(),  const SizedBox(height: 24),  Row(  children: [  Expanded(child: Divider(color: Colors.*grey*.shade300)),  Padding(  padding: const EdgeInsets.symmetric(horizontal: 16),  child: Text(  "OR",  style: GoogleFonts.*poppins*(  color: Colors.*grey*[600],  ),  ),  ),  Expanded(child: Divider(color: Colors.*grey*.shade300)),  ],  ).animate().fadeIn(),  const SizedBox(height: 24),  OutlinedButton.icon(  onPressed: () => AuthController.*to*.signInWithGoogle(),  icon: Image.asset(  "assets/images/google.png",  height: 24,  ),  label: Text(  "Sign up with Google",  style: GoogleFonts.*poppins*(  fontSize: 16,  fontWeight: FontWeight.*w500*,  ),  ),  style: OutlinedButton.*styleFrom*(  padding: const EdgeInsets.symmetric(vertical: 16),  shape: RoundedRectangleBorder(  borderRadius: BorderRadius.circular(12),  ),  side: BorderSide(color: Colors.*grey*.shade300),  ),  ).animate().fadeIn().slideX(),  ],  ),  ),  ),  ),  );  } }  loginPage.dart import 'package:flutter/material.dart'; import 'package:get/get.dart'; import 'package:google\_fonts/google\_fonts.dart'; import 'package:flutter\_animate/flutter\_animate.dart'; import 'auth\_controller.dart'; import 'register\_page.dart';  class LoginPage extends StatelessWidget {  const LoginPage({super.key});   @override  Widget build(BuildContext context) {  final emailController = TextEditingController();  final passwordController = TextEditingController();   return Scaffold(  body: SafeArea(  child: SingleChildScrollView(  child: Padding(  padding: const EdgeInsets.all(24.0),  child: Column(  crossAxisAlignment: CrossAxisAlignment.stretch,  children: [  const SizedBox(height: 40),  Text(  "Welcome!",  style: GoogleFonts.*poppins*(  fontSize: 32,  fontWeight: FontWeight.*bold*,  color: Theme.*of*(context).primaryColor,  ),  ).animate().fadeIn().slideX(),  const SizedBox(height: 8),  Text(  "Sign in to continue",  style: GoogleFonts.*poppins*(  fontSize: 16,  color: Colors.*grey*[600],  ),  ).animate().fadeIn().slideX(),  const SizedBox(height: 40),  TextField(  controller: emailController,  decoration: InputDecoration(  labelText: "Email",  hintText: "Enter your email",  prefixIcon: const Icon(Icons.*email\_outlined*),  border: OutlineInputBorder(  borderRadius: BorderRadius.circular(12),  ),  enabledBorder: OutlineInputBorder(  borderRadius: BorderRadius.circular(12),  borderSide: BorderSide(color: Colors.*grey*.shade300),  ),  ),  ).animate().fadeIn().slideX(),  const SizedBox(height: 16),  TextField(  controller: passwordController,  obscureText: true,  decoration: InputDecoration(  labelText: "Password",  hintText: "Enter your password",  prefixIcon: const Icon(Icons.*lock\_outline*),  border: OutlineInputBorder(  borderRadius: BorderRadius.circular(12),  ),  enabledBorder: OutlineInputBorder(  borderRadius: BorderRadius.circular(12),  borderSide: BorderSide(color: Colors.*grey*.shade300),  ),  ),  ).animate().fadeIn().slideX(),  const SizedBox(height: 24),  ElevatedButton(  onPressed: () {  AuthController.*to*.loginUser(  emailController.text.trim(),  passwordController.text.trim(),  );  },  style: ElevatedButton.*styleFrom*(  padding: const EdgeInsets.symmetric(vertical: 16),  shape: RoundedRectangleBorder(  borderRadius: BorderRadius.circular(12),  ),  ),  child: Text(  "Sign In",  style: GoogleFonts.*poppins*(  fontSize: 16,  fontWeight: FontWeight.*w600*,  ),  ),  ).animate().fadeIn().slideX(),  const SizedBox(height: 16),  TextButton(  onPressed: () => Get.to(() => const RegisterPage()),  child: Text(  "Don't have an account? Register",  style: GoogleFonts.*poppins*(  color: Theme.*of*(context).primaryColor,  ),  ),  ).animate().fadeIn().slideX(),  const SizedBox(height: 24),  Row(  children: [  Expanded(child: Divider(color: Colors.*grey*.shade300)),  Padding(  padding: const EdgeInsets.symmetric(horizontal: 16),  child: Text(  "OR",  style: GoogleFonts.*poppins*(  color: Colors.*grey*[600],  ),  ),  ),  Expanded(child: Divider(color: Colors.*grey*.shade300)),  ],  ).animate().fadeIn(),  const SizedBox(height: 24),  OutlinedButton.icon(  onPressed: () => AuthController.*to*.signInWithGoogle(),  icon: Image.asset(  "assets/images/google.png",  height: 24,  ),  label: Text(  "Sign in with Google",  style: GoogleFonts.*poppins*(  fontSize: 16,  fontWeight: FontWeight.*w500*,  ),  ),  style: OutlinedButton.*styleFrom*(  padding: const EdgeInsets.symmetric(vertical: 16),  shape: RoundedRectangleBorder(  borderRadius: BorderRadius.circular(12),  ),  side: BorderSide(color: Colors.*grey*.shade300),  ),  ).animate().fadeIn().slideX(),  ],  ),  ),  ),  ),  );  } }  homePage.dart import 'package:flutter/material.dart'; import 'package:cloud\_firestore/cloud\_firestore.dart'; import 'package:firebase\_auth/firebase\_auth.dart'; import 'package:get/get.dart'; import 'package:google\_fonts/google\_fonts.dart'; import 'package:flutter\_animate/flutter\_animate.dart'; import 'auth\_controller.dart';  class HomePage extends StatelessWidget {  const HomePage({super.key});   @override  Widget build(BuildContext context) {  final user = FirebaseAuth.*instance*.currentUser;  if (user == null) {  return const Scaffold(  body: Center(  child: Text('User not found'),  ),  );  }   return Scaffold(  appBar: AppBar(  title: Text(  'My Notes',  style: GoogleFonts.*poppins*(  fontSize: 24,  fontWeight: FontWeight.*w600*,  ),  ),  centerTitle: true,  elevation: 0,  actions: [  IconButton(  icon: const Icon(Icons.*logout*),  onPressed: () => AuthController.*to*.logout(),  tooltip: 'Logout',  ),  ],  ),  body: StreamBuilder<QuerySnapshot>(  stream: FirebaseFirestore.*instance* .collection('notes')  .where('userId', isEqualTo: user.uid)  .snapshots(),  builder: (context, snapshot) {  if (snapshot.hasError) {  return Center(child: Text('Error: ${snapshot.error}'));  }   if (snapshot.connectionState == ConnectionState.waiting) {  return const Center(child: CircularProgressIndicator());  }   if (!snapshot.hasData || snapshot.data!.docs.isEmpty) {  return Center(  child: Text(  'No notes yet',  style: GoogleFonts.*poppins*(  fontSize: 18,  color: Colors.*grey*,  ),  ),  );  }   return ListView.builder(  padding: const EdgeInsets.all(16),  itemCount: snapshot.data!.docs.length,  itemBuilder: (context, index) {  final doc = snapshot.data!.docs[index];  final data = doc.data() as Map<String, dynamic>;    return Card(  margin: const EdgeInsets.only(bottom: 16),  elevation: 2,  child: ListTile(  contentPadding: const EdgeInsets.all(16),  title: Text(  data['title'] ?? 'No Title',  style: GoogleFonts.*poppins*(  fontSize: 18,  fontWeight: FontWeight.*w600*,  ),  ),  subtitle: Text(  data['content'] ?? 'No Content',  style: GoogleFonts.*poppins*(  fontSize: 14,  color: Colors.*grey*[600],  ),  ),  trailing: Row(  mainAxisSize: MainAxisSize.min,  children: [  IconButton(  icon: const Icon(Icons.*edit*),  onPressed: () => \_showEditDialog(context, doc),  color: Colors.*blue*,  ),  IconButton(  icon: const Icon(Icons.*delete*),  onPressed: () => \_deleteNote(doc.id),  color: Colors.*red*,  ),  ],  ),  ),  ).animate().fadeIn().slideX();  },  );  },  ),  floatingActionButton: FloatingActionButton(  onPressed: () => \_showAddDialog(context),  child: const Icon(Icons.*add*),  ).animate().scale(),  );  }   void \_showAddDialog(BuildContext context) {  final titleController = TextEditingController();  final contentController = TextEditingController();  final user = FirebaseAuth.*instance*.currentUser;   showDialog(  context: context,  builder: (context) => AlertDialog(  title: Text(  'Add Note',  style: GoogleFonts.*poppins*(fontWeight: FontWeight.*w600*),  ),  content: Column(  mainAxisSize: MainAxisSize.min,  children: [  TextField(  controller: titleController,  decoration: const InputDecoration(  labelText: 'Title',  border: OutlineInputBorder(),  ),  ),  const SizedBox(height: 16),  TextField(  controller: contentController,  decoration: const InputDecoration(  labelText: 'Content',  border: OutlineInputBorder(),  ),  maxLines: 3,  ),  ],  ),  actions: [  TextButton(  onPressed: () => Navigator.*pop*(context),  child: const Text('Cancel'),  ),  ElevatedButton(  onPressed: () {  if (titleController.text.isNotEmpty && user != null) {  FirebaseFirestore.*instance*.collection('notes').add({  'title': titleController.text,  'content': contentController.text,  'userId': user.uid,  'timestamp': FieldValue.*serverTimestamp*(),  });  Navigator.*pop*(context);  }  },  child: const Text('Add'),  ),  ],  ),  );  }   void \_showEditDialog(BuildContext context, DocumentSnapshot doc) {  final data = doc.data() as Map<String, dynamic>;  final titleController = TextEditingController(text: data['title']);  final contentController = TextEditingController(text: data['content']);  final user = FirebaseAuth.*instance*.currentUser;   showDialog(  context: context,  builder: (context) => AlertDialog(  title: Text(  'Edit Note',  style: GoogleFonts.*poppins*(fontWeight: FontWeight.*w600*),  ),  content: Column(  mainAxisSize: MainAxisSize.min,  children: [  TextField(  controller: titleController,  decoration: const InputDecoration(  labelText: 'Title',  border: OutlineInputBorder(),  ),  ),  const SizedBox(height: 16),  TextField(  controller: contentController,  decoration: const InputDecoration(  labelText: 'Content',  border: OutlineInputBorder(),  ),  maxLines: 3,  ),  ],  ),  actions: [  TextButton(  onPressed: () => Navigator.*pop*(context),  child: const Text('Cancel'),  ),  ElevatedButton(  onPressed: () {  if (titleController.text.isNotEmpty && user != null) {  FirebaseFirestore.*instance*.collection('notes').doc(doc.id).update({  'title': titleController.text,  'content': contentController.text,  'userId': user.uid,  'timestamp': FieldValue.*serverTimestamp*(),  });  Navigator.*pop*(context);  }  },  child: const Text('Update'),  ),  ],  ),  );  }   void \_deleteNote(String docId) {  showDialog(  context: Get.context!,  builder: (context) => AlertDialog(  title: Text(  'Delete Note',  style: GoogleFonts.*poppins*(fontWeight: FontWeight.*w600*),  ),  content: const Text('Are you sure you want to delete this note?'),  actions: [  TextButton(  onPressed: () => Navigator.*pop*(context),  child: const Text('Cancel'),  ),  ElevatedButton(  onPressed: () {  FirebaseFirestore.*instance*.collection('notes').doc(docId).delete();  Navigator.*pop*(context);  },  style: ElevatedButton.*styleFrom*(  backgroundColor: Colors.*red*,  foregroundColor: Colors.*white*,  ),  child: const Text('Delete'),  ),  ],  ),  );  } } |
| **Screenshots** |  |
| **Question and Answers** | Answer the following Questions:   1. Write the connection steps, step-by-step for database connectivity.   Create a Firebase project   * Go to the Firebase console and click Add project. * Enter a project name, disable Google Analytics if you like, and click Create projecy * Once created, you’ll be taken to the project overview.   Enable Realtime Database   * In the Firebase console sidebar, select Realtime Database, then click Create Database. * Choose Test mode (world‑read/write) for initial development, pick a region, and click Enable   Register your Android app   * In project overview, click Add app → Android. * Enter your Android package name (e.g., com.example.myapp), provide an app nickname, and click Register app. * Download the generated google-services.json and place it in your Flutter project under android/app/   (Optional) Register your iOS app   * Back in the console, click Add app → iOS, enter your Bundle Identifier, and register. * Download GoogleService-Info.plist and add it to ios/Runner/ in Xcode   Install FlutterFire CLI and configure your app   * Install the Firebase CLI if you haven’t already:   + npm install -g firebase-tools   + firebase login * Activate the FlutterFire CLI:   + dart pub global activate flutterfire\_cli * From your Flutter project root, run:   + flutterfire configure   Select your Firebase project and platforms; this generates lib/firebase\_options.dart.  Add FlutterFire plugins   * In your project directory, add the core and database plugins: * flutter pub add firebase\_core * flutter pub add firebase\_database   Re-run flutterfire configure to ensure any required Gradle plugins are added  Apply Android Gradle settings   * In android/build.gradle, under dependencies, add:   + classpath 'com.google.gms:google-services:4.3.10' * In android/app/build.gradle, at the bottom add:   + apply plugin: 'com.google.gms.google-services'   Apply iOS CocoaPods settings   * Ensure your ios/Podfile targets at least iOS 10.0. * Run:   + cd ios   + pod install   Initialize Firebase in Dart   * In lib/main.dart import and initialize Firebase before runApp:   import 'package:firebase\_core/firebase\_core.dart';  import 'firebase\_options.dart';  void main() async {  WidgetsFlutterBinding.ensureInitialized();  await Firebase.initializeApp(  options: DefaultFirebaseOptions.currentPlatform,  );  runApp(const MyApp());  }  Verify the connection   * Rebuild and run your app:   + flutter run * In the Firebase console’s Realtime Database section, you should see reads/writes reflected in real time |
| **Conclusion** | From this session, I learned how quickly you can get a Flutter app talking to Firebase without writing any backend code. I saw the power of the FlutterFire CLI in automating setup and configuration for Android. I discovered how easy it is to add and initialize the firebase\_core and firebase\_database packages to perform real‑time CRUD operations. Overall, I realized that with just a few steps, Firebase takes care of data syncing, letting me focus on building user‑facing features. |