

EXPERIMENT NO: - 06

Name : Anushka Shahane

Class : D15A

Roll:No : 54

AIM: To connect flutter UI with firebase database.

Introduction to Firebase and Flutter Integration

Firebase is a comprehensive platform developed by Google, designed to help developers build high-quality applications for both mobile and web. It provides essential services such as real-time databases, authentication, cloud storage, hosting, and much more. One of the most widely used Firebase services is the Firebase Realtime Database, which is a NoSQL cloud database that allows data to be stored and synced in real-time across all connected devices.

Flutter, on the other hand, is an open-source UI software development kit created by Google, which allows developers to build natively compiled applications for mobile, web, and desktop from a single codebase. Its rich set of pre-designed widgets and powerful tools makes Flutter an attractive option for developing visually appealing and performant applications.

Integrating Firebase with Flutter allows developers to leverage the full potential of Firebase services in their applications. By using Firebase's Realtime Database, Flutter apps can achieve features such as real-time data synchronization, secure authentication, and cloud-based storage. This combination enables developers to create powerful, scalable, and feature-rich mobile and web applications.

Firebase Realtime Database Overview

Firebase Realtime Database is a cloud-hosted NoSQL database that stores data in a JSON-like format. The key characteristic of this database is its real-time synchronization feature, meaning that any changes made to the database are instantly reflected on all clients (i.e., devices) connected to it. This makes it an ideal solution for applications that require frequent updates and need to maintain synchronized data across multiple users or devices, such as messaging apps, social media platforms, or collaborative tools.

The Firebase Realtime Database is structured as a tree of data, where each node in the tree can contain key-value pairs. This structure allows for easy data retrieval and modification. Firebase's real-time capabilities enable apps to immediately receive updates to the data whenever it changes, without the need to refresh or reload the page. Additionally, the database supports offline data persistence, meaning that even if the user's device loses its internet connection, the app can still function by using the locally cached data.

Setting Up Firebase in Flutter :

To connect a Flutter app with Firebase, the following steps are typically followed:

1. **Creating a Firebase Project:** To start using Firebase with Flutter, the first step is to create a Firebase project in the Firebase Console. Once the project is created, developers can associate their Flutter app with the Firebase project by following the platform-specific instructions for Android or iOS. This usually involves configuring API keys, downloading configuration files, and adding them to the Flutter project.
2. **Integrating Firebase SDK in Flutter:** After the Firebase project is set up, developers need to integrate Firebase's SDK into the Flutter app. This involves adding the necessary dependencies to the Flutter project's pubspec.yaml file. For Firebase's Realtime Database, the package firebase_database is used. Additionally, Firebase's core SDK (firebase_core) must also be included to initialize Firebase services.
3. **Initializing Firebase:** Before any Firebase functionality can be used, it is essential to initialize Firebase in the Flutter app. This is done by calling Firebase.initializeApp() in the main entry point of the app (usually in the main.dart file). Firebase needs to be initialized before interacting with any Firebase services, such as the Realtime Database, Cloud Firestore, or Authentication.

Connecting Firebase to a Flutter app enables developers to create robust, scalable, and real-time applications with ease. Firebase's Realtime Database offers a powerful, cloud-based solution for managing data in real-time, while Firebase Authentication ensures secure access control. By integrating Firebase with Flutter, developers can take advantage of real-time data synchronization, offline support, and a wide range of other Firebase features, allowing them to build feature-rich apps that meet modern user expectations.

Code: -authentication.dart

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

```
class AuthMethod {
    final FirebaseFirestore
_firestore =
FirebaseFirestore.instance;
    final FirebaseAuth _auth =
FirebaseAuth.instance;
```

```
    // Sign Up User using email
and password only.
```

```
        Future<String>
signUpUser({
    required String email,
    required String password,
}) async {
    String res = "Some error
occurred";
```

```
    try {
        // Check that both email
and password are provided.
```

```
        if (email.isNotEmpty &&
password.isNotEmpty) {
```

```
            // Register user with
Firebase Auth.
```

```
            UserCredential cred =
await
_auth.createUserWithEmailAndPassword(
```

```
                email: email,
                password: password,
            );
```

```
            // Add user details to
Firestore.
            await
```

```
        _firestore.collection("users").
doc(cred.user!.uid).set({
            'uid': cred.user!.uid,
            'email': email,
        });
        res = "success";
    } else {
        res = "Please enter all the
fields";
    }
    } catch (err) {
        res = err.toString();
    }
    return res;
}
```

```
    // Log In User using email
and password.
```

```
        Future<String> loginUser({
    required String email,
    required String password,
}) async {
    String res = "Some error
occurred";
    try
```

```
        // Sign out the current user.
        Future<void> signOut()
```

```
async {
    await _auth.signOut();
}
}
```

```
Colors.grey[100], appBar:
AppBar(
```

```
    backgroundColor:
Colors.blue[900], title: Text(
    'Booking.co
m', style:
TextStyle(
```

```
        color: Colors.white, fontWeight:
FontWeight.bold, fontSize: 18),
    ),
```

```

actions: [
  IconButton(
    icon: Icon(Icons.notifications, color:
    Colors.white), onPressed: () {}),
  ),
  worldwide"),
    _buildDiscountCard(
      "10% off rental cars", "Save on select
      rental cars"),
    _buildDiscountCard(
      "Exclusive flight deals", "Best fares for
members

    SizedBox(height: 15),

    // **Deals Section**
    Text(
      "Deals for the weekend",

    Text(subtitle, style: TextStyle(fontSize: 10)),
  ],
),
);

// Log In User using email and password.
Future<String> loginUser({
  required String email,
  required String password,
}) async {
  String res = "Some error occurred";
  try {
    // Ensure both fields are filled.
    if (email.isNotEmpty && password.isNotEmpty
y) {
      await _auth.signInWithEmailAndPassword(
        email: email,
        password: password,
      );

```

Hotel_model.dart

```
// lib/models/hotel_model.dart
class Hotel {
  final String name;
  final String address;
  final String price;
  final String imageUrl;

  Hotel({
    required this.name,
    required this.address,
    required this.price,
    required this.imageUrl,
  });

  factory Hotel.fromJson(Map<String, dynamic> json) {
    return Hotel(
      name: json['hotel_name'] ?? 'No Name',
      address: json['address'] ?? 'No Address',
      price: json['price']?.toString() ?? 'N/A',
      imageUrl: json['image_url'] ?? "",
    );
  }
}
```

Hotel_service.dart

```
// lib/services/hotel_service.dart
import 'dart:convert';
import 'package:http/http.dart' as http;
import '../models/destination_model.dart';

class HotelService {
  // Updated endpoint using your curl command
  static const String _endpoint =
    "https://booking-com15.p.rapidapi.com/api/v1/hotels/searchDestination";

  static Future<List<Destination>> fetchDestinations({
    required String destinationQuery,
    required String checkIn, // include if supported by your API
    required String checkOut, // include if supported by your API
    required String rooms, // include if supported by your API
  }) async {
```

```

final Uri url = Uri.parse(_endpoint).replace(queryParameters: {
  'query': destinationQuery,
  'checkin': checkIn,
  'checkout': checkOut,
  'rooms': rooms,
});

// Headers required by the API. (Remember to secure your API key in production.)
final headers = {
  'x-rapidapi-host': 'booking-com15.p.rapidapi.com',
  'x-rapidapi-key': 'your-api-key',
  'Content-Type': 'application/json',
};

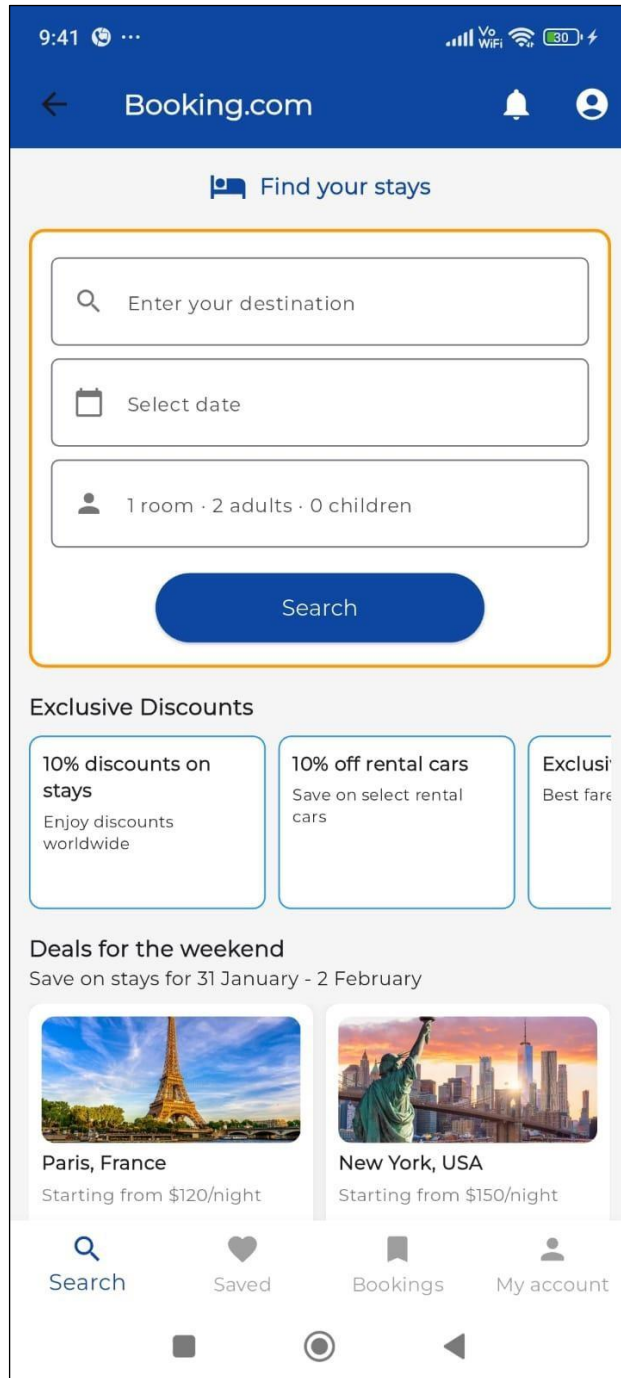
final response = await http.get(url, headers: headers);

if (response.statusCode == 200) {
  f

inal Map<String, dynamic> jsonResponse = json.decode(response.body);
  if (jsonResponse['status'] == true) {

    final List<dynamic> data = jsonResponse['data'];
    return data.map((item) => Destination.fromJson(item)).toList();
  } else {
    throw Exception("API error: ${jsonResponse['message']}");
  }
} else {
  throw Exception(
    "Failed to load destinations. HTTP Status: ${response.statusCode}");
}

```



2:21

Vg WiFi 93%



Booking.com

Register here

We'll use this to create an account if you don't have one yet.

Continue

OR



Continue with Google



Continue with Facebook

[Already have an account? Sign In](#)

2:29



Booking.com

Sign in

Email address

anushka3204@gmail.com

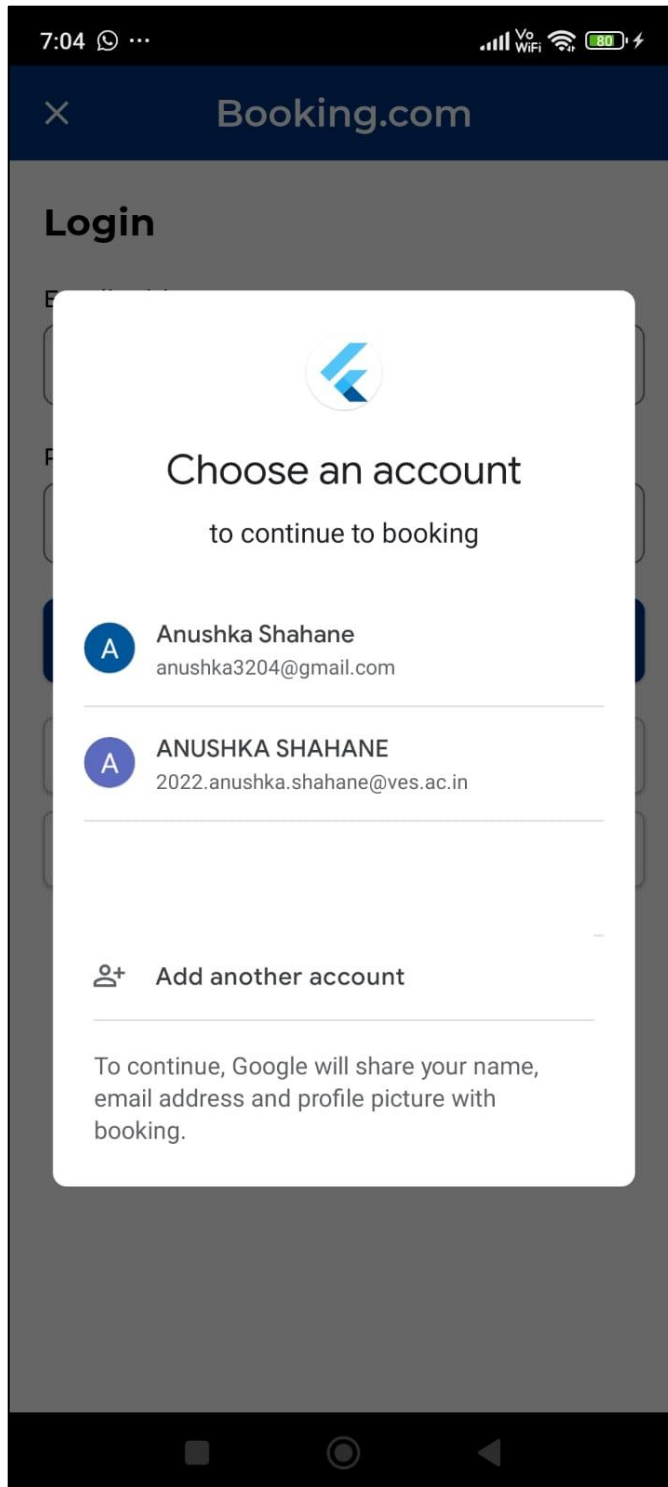
Password

.....











Sign In





Search by email address, phone number, or user UID







Add user

Identifier	Providers	Created ↓	Signed In	User UID
anu@gmail.com		Feb 16, 2025	Feb 16, 2025	M3LafJOudGbWDLYPoxycJM...
anushkasha21@gmail...		Feb 16, 2025	Feb 16, 2025	sY10HZtoaiXqN7XIPKRi5av4x...
anushkashahane221@...		Feb 16, 2025	Feb 16, 2025	YJ879IAVAnVEFB83h5lqxwx6...
anushkashahane22@g...		Feb 16, 2025	Feb 16, 2025	v6PgabeojZNueLjHjuLU6TcW...
saurabh@gmail.com		Feb 16, 2025	Feb 16, 2025	FXrfYu0vLDRaf7iGdTFi17Z6U...
2022.anushkaq@ves.ac...		Feb 16, 2025	Feb 16, 2025	L6mY900lw2ZYSVeLitPTQ8w...
2022.anushka@ves.ac.in		Feb 16, 2025	Feb 16, 2025	SVSjwjeAbilhTW0zU9zVhCOuh...
anushka3204@gmail.c...		Feb 16, 2025	Feb 16, 2025	vueOTV0Y9ZPB9zMCXQVMM...

Rows per page:

50

1 – 8 of 8

 (default)	 users  	 Di6rS39pvhNRkghyq2yTXELm08r1 
+ Start collection	+ Add document	+ Start collection
users >	Di6rS39pvhNRkghyq2y... > M3LafJOudGbWDLYPoxy... sY10HZtoaiXqN7X1PKR... v6PgabeojZNueLjHjuL... vue0TV0Y9ZPB9zMCXQV...	+ Add field email: "abc@gmail.com" uid: "Di6rS39pvhNRkghyq2yTXELm08r1"

11:57

Vo WiFi 74



paris · 2025-02-16 - 2025-02-18



Paris

Ile de France, France

Hotels Available: 7588

City: Paris

Destination Type: city

View Hotels

