# EXPERIMENT 6

**AIM:** To connect a Flutter-based Expense Tracker UI with Firebase for user authentication and expense data storage.

## THEORY:

Firebase is a comprehensive backend-as-a-service (BaaS) platform developed by Google. It provides a wide range of tools and services such as authentication, real-time databases, cloud storage, and analytics to help developers build and scale modern mobile and web applications.  
  
Integrating Firebase with Flutter allows developers to enhance their apps with powerful backend capabilities while focusing on building rich user interfaces using Flutter.  
  
Why Use Firebase with Flutter?  
- Real-time Functionality: Firebase offers real-time database and cloud-based data synchronization, making it ideal for dynamic data apps like Expense Trackers.  
- User Authentication: Secure sign-in and sign-up using email/password or social providers.  
- Data Storage: Store structured expense data in Firestore.  
- Notifications: Send push notifications using Firebase Cloud Messaging (if needed).  
- Analytics & Crash Reporting: Monitor user behavior and app performance in real-time.  
  
Common Firebase Services Used:  
- Firebase Authentication: For managing user login and identity verification.  
- Cloud Firestore: A scalable NoSQL database used to store and sync expenses.  
- Firebase Storage: (Optional) For uploading receipts or images.  
- Firebase Cloud Messaging: (Optional) For sending reminders.  
  
Benefits of Using Firebase with Flutter:  
- Cross-Platform Support  
- Scalability  
- Built-in Security Rules  
- Faster Development

## CODE SNIPPETS:

### main.dart:

void main() async {  
 WidgetsFlutterBinding.ensureInitialized();  
 await Firebase.initializeApp();  
 runApp(const MyApp());  
}

### auth\_provider.dart:

Future<void> login(String email, String password) async {  
 final credential = await FirebaseAuth.instance.signInWithEmailAndPassword(  
 email: email,  
 password: password,  
 );  
 user = credential.user;  
 notifyListeners();  
}

### login\_screen.dart:

ElevatedButton(  
 onPressed: login,  
 child: const Text("Login"),  
)

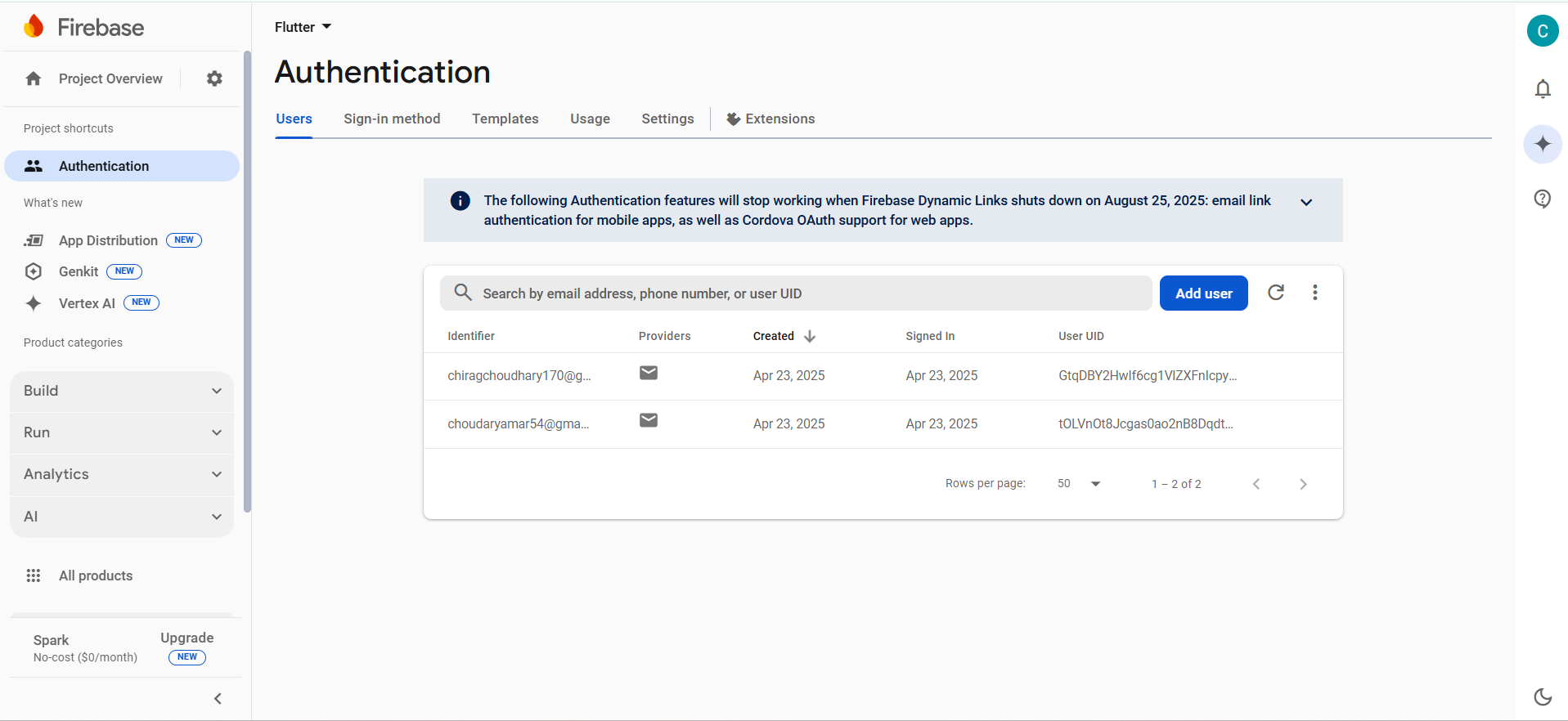
### Firestore Usage in expense\_provider.dart (if applicable):

FirebaseFirestore.instance.collection('expenses').add({  
 'amount': expense.amount,  
 'category': expense.category,  
 'date': expense.date.toIso8601String(),  
 'userId': user.uid,  
});

### build.gradle:

dependencies {  
 implementation platform('com.google.firebase:firebase-bom:33.9.0')  
 implementation 'com.google.firebase:firebase-auth'  
 implementation 'com.google.firebase:firebase-firestore'  
}

## OUTPUT:



## CONCLUSION:

Integrating Firebase with Flutter enabled secure login and real-time data storage in the Expense Tracker app. The Firebase Auth SDK handled user sessions, while Firestore was used to store and retrieve categorized expenses per user.