Firebase

What is Firebase?

BaaS ?

- Developed by Google.
- Solves common backend challenges so you can focus on the frontend.

Core Firebase Services

Firebase offers a suite of services, each designed to handle a specific part of your app's backend needs.

Authentication

Easy user sign-up and login with email, Google, phone, and more.

Firestore & Realtime Database

Flexible, scalable NoSQL databases for storing and syncing app data.

Cloud Storage

Store user-generated content like photos and videos securely.

setup

1) download node=> https://nodejs.org/en/download

2) node -v npm -v

3) npm install -g firebase-tools

4) firebase --version

5) firebase login

6) dart pub global activate flutterfire_cli

7)flutterfire --version

edit sys environment

8)create a project on firebase website

9)flutterfire configure

10)flutter pub add firebase_core

11)flutter pub add firebase_auth

what is the states and the logic of Auth feature?

firebase code

1) convert main from sync to async

```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
await Firebase.initializeApp(options: DefaultFirebaseOptions.currentPlatform);
runApp(const MyApp());
}
```

2)FirebaseClass

`final FirebaseAuth _auth = FirebaseAuth.instance;` =>dealing with authentication

`final FirebaseFirestore _firestore = FirebaseFirestore.instance;` =>dealing with cloudStorage

`await _auth.createUserWithEmailAndPassword( email: email, password: password );` => to create an email

`await _auth.signInWithEmailAndPassword( email: email, password: password );` => cheack if email exist and login

saveData

```
_firestore.collection('users').doc(user.uid).set(appUser.toJson()); =>add data to firebase
```

```
Future<AppUser> signUp(String name, String email, String password) async {
  UserCredential result = await _auth.createUserWithEmailAndPassword(
    email: email,
    password: password,
  );

  User user = result.user!;

  AppUser appUser = AppUser(
    id: user.uid,
    name: name,
    email: email,
    favorites: [],
  );

  try {
    await _firestore.collection('users').doc(user.uid).set(appUser.toJson());
    print(" Data saved to Firestore for user ${user.uid}");
  } catch (e) {
    print(" Firestore error: $e");
  }

  return appUser;
}
```

retrive data

`DocumentSnapshot doc = await _firestore.collection('users').doc(user.uid).get();`

```
// Login
Future<AppUser> login(String email, String password) async {
  UserCredential result = await _auth.signInWithEmailAndPassword(
    email: email,
    password: password,
  );

  User user = result.user!;

  DocumentSnapshot doc = await _firestore
      .collection('users')
      .doc(user.uid)
      .get();

  return AppUser.fromJson(doc.data() as Map<String, dynamic>);
}

// Logout
Future<void> logout() async {
  await _auth.signOut();
}
```

Project Requirements – Flutter App

1. General Requirements

- The app must be **Responsive** (support multiple screen sizes: Mobile / Tablet).
- Follow (core- features) **Architecture** .
- Use **Cubit (BLoC pattern)** for state management.
- save the data(hive _firebase _ apis)

2. Functional Requirements

a.Authentication Module

- User sign up and login.
- User logout.

b.CRUD Operations (Create, Read, Update, Delete).

3. Deliverables

a.complete project

b. Flutter project source code on github (all team should commits) .

c. README file including: