4.User Sign In

Ab ham user ko sign in ki function dengy !

Hamen firebase se 2 things required hen :

getAuth , siginIn methods

signInWithEmailAndPassword Firebase Authentication ka ek method hai jo email aur password ka use karke user ko authenticate karta hai. Ye method asyncronous hai aur ye promise return karta hai.

Kya Karta Hai

1. Email aur password ko Firebase Authentication system ke against verify karta hai.
2. Agar credentials sahi hote hain, to user ko successfully sign-in kar diya jata hai.
3. Agar credentials galat hote hain, ya koi aur issue hota hai (jaise user exist nahi karta), to error throw karta hai.

Kya Leta Hai

signInWithEmailAndPassword method ko 2 arguments pass kiye jaate hain:

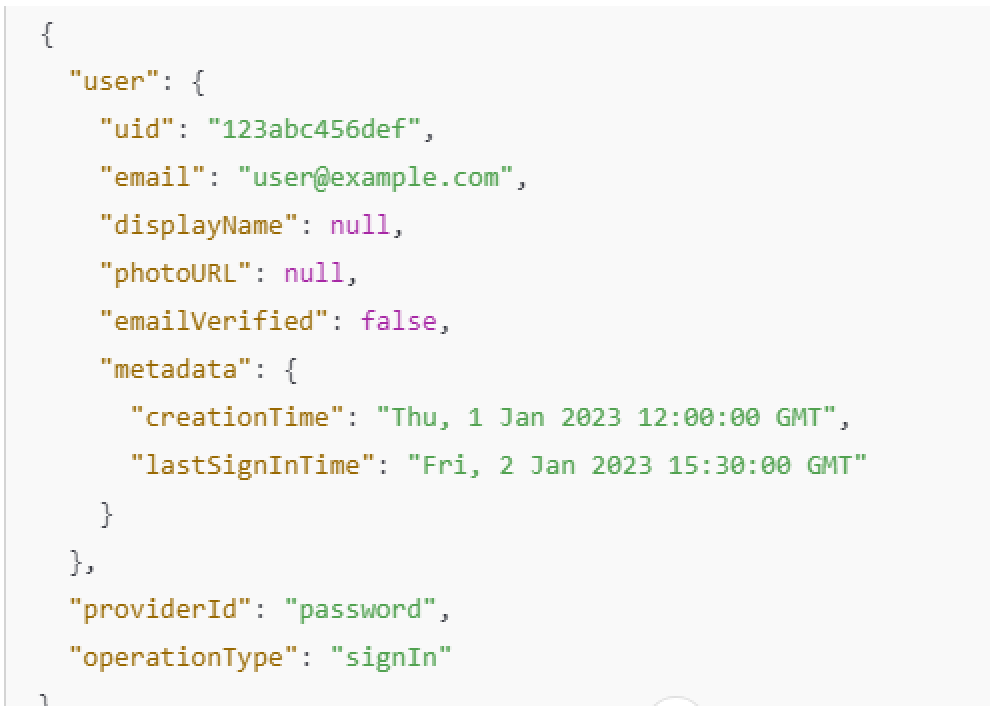
1. **auth**: Firebase ka authentication object jo initialize kiya gaya hota hai.
2. **email**: User ka email address (string format).
3. **password**: User ka password (string format).

Kya Return Karta Hai

Agar sign-in successful hota hai, to ye method ek **Promise** return karta hai jo resolve hota hai **UserCredential** object ke saath.

UserCredential object ke andar ye properties hoti hain:

1. **user**: Ek object jo authenticated user ki details ko hold karta hai.
    - uid: User ka unique ID.
    - email: User ka email address.
    - displayName: (Agar set kiya gaya ho to).
    - photoURL: (Agar set ki gayi ho to).
    - emailVerified: Boolean value, kya email verify ki gayi hai ya nahi.
    - metadata: User ke account creation aur last login ke timestamps.
2. **providerId**: Authentication provider ki information (e.g., "password" agar email-password use kiya gaya ho).
3. **operationType**: String, batata hai ki ye "signIn" operation hai.



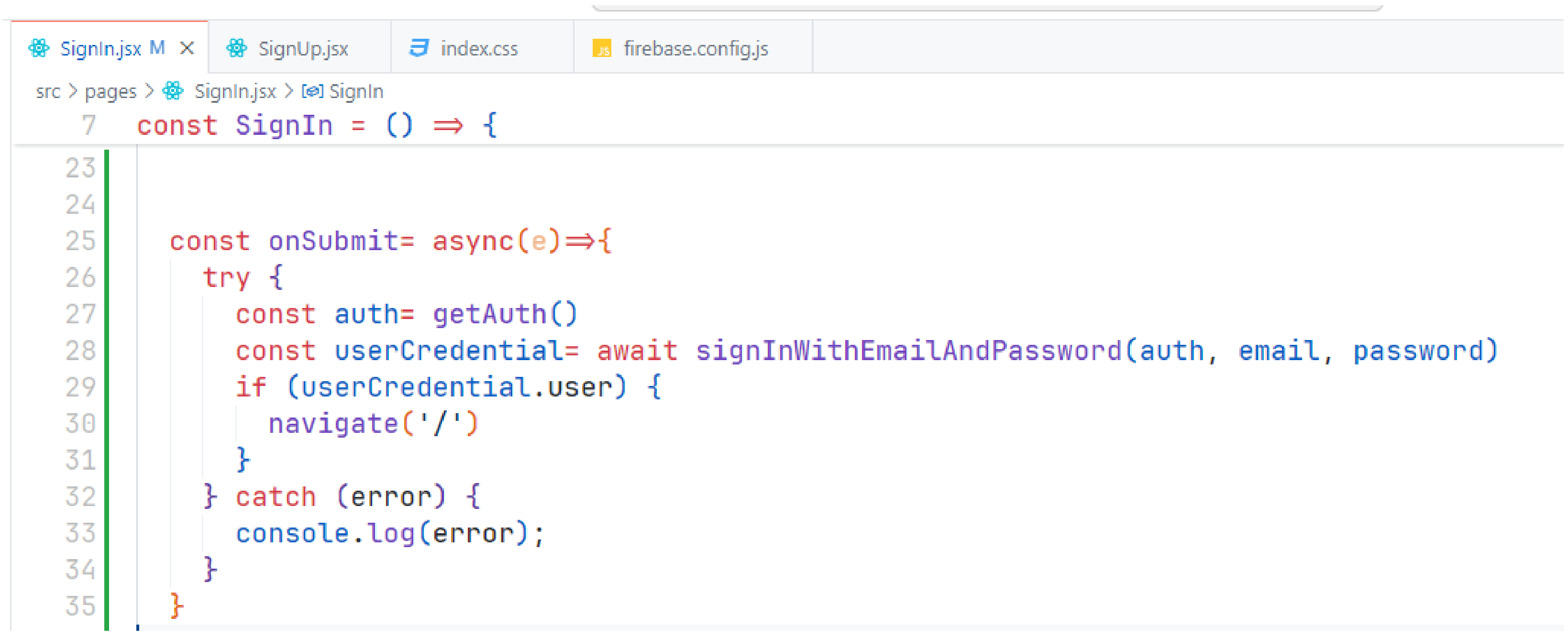
```
{
  "user": {
    "uid": "123abc456def",
    "email": "user@example.com",
    "displayName": null,
    "photoURL": null,
    "emailVerified": false,
    "metadata": {
      "creationTime": "Thu, 1 Jan 2023 12:00:00 GMT",
      "lastSignInTime": "Fri, 2 Jan 2023 15:30:00 GMT"
    }
  },
  "providerId": "password",
  "operationType": "signIn"
}
```

Error Handling

Agar koi error hoti hai, to catch block me ek error object return hota hai. Common errors hain:

- auth/user-not-found: Agar email register nahi hai.
- auth/wrong-password: Agar password galat hai.
- auth/too-many-requests: Agar ek hi user par bohot zyada login attempts ki gayi hain.

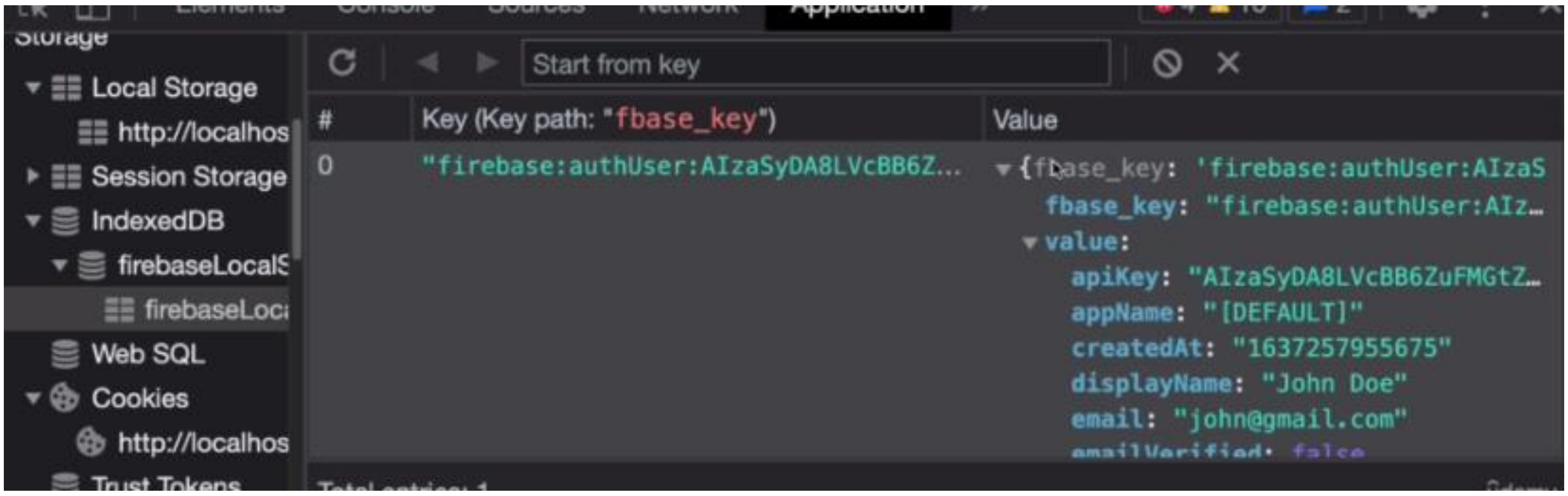
So in sign in compt:



```
const SignIn = () => {


  const onSubmit= async(e)=>{
    try {
      const auth= getAuth()
      const userCredential= await signInWithEmailAndPassword(auth, email, password)
      if (userCredential.user) {
        navigate('/')
      }
    } catch (error) {
      console.log(error);
    }
  }
```

Firebase tamam details ko IndexDB me store krta ha:



```
#   Key (Key path: "fbase_key")                 Value
0   "firebase:authUser:AIzaSyDA8LVcBB6Z...    ▾{fbase_key: 'firebase:authUser:AIzaS
                                                 fbase_key: "firebase:authUser:AIz…
                                                ▾value:
                                                   apiKey: "AIzaSyDA8LVcBB6ZuFMGtZ…
                                                   appName: "[DEFAULT]"
                                                   createdAt: "1637257955675"
                                                   displayName: "John Doe"
                                                   email: "john@gmail.com"
                                                   emailVerified: false
Total entries: 1
```

****

getAuth Firebase Authentication ka ek method hai jo aapke app ke liye ek authentication instance ko initialize karta hai. Ye instance aapke app aur Firebase Authentication service ke beech ka connection manage karta hai.

---

getAuth Kya Hota Hai?

getAuth ek function hai jo Firebase authentication ka ek instance return karta hai. Ye instance:

1. Firebase ke backend se connect hota hai.
2. Authentication operations ko manage karta hai, jaise user sign-in, sign-up, aur sign-out.
3. User ka state track karta hai (e.g., user login hai ya logout).

Is instance ko aap har authentication-related operation (e.g., signInWithEmailAndPassword, createUserWithEmailAndPassword, onAuthStateChanged) ke liye use karte hain.

Kyu Use Hota Hai?

getAuth ko use karne ka primary reason hai Firebase Authentication ke functions ko enable karna. Agar aap getAuth use nahi karenge, to Firebase ke authentication methods ka use nahi kar paenge. Ye ek centralized point provide karta hai jahan se:

1. Aapke app ka Firebase ke saath connection bana rahe.
2. User ka authentication state maintain ho.

Kaise Use Karte Hain?

getAuth ko use karne ke liye sabse pehle Firebase app ko initialize karna padta hai.

Firebase Initialization kro or getauth lelo firebase se .

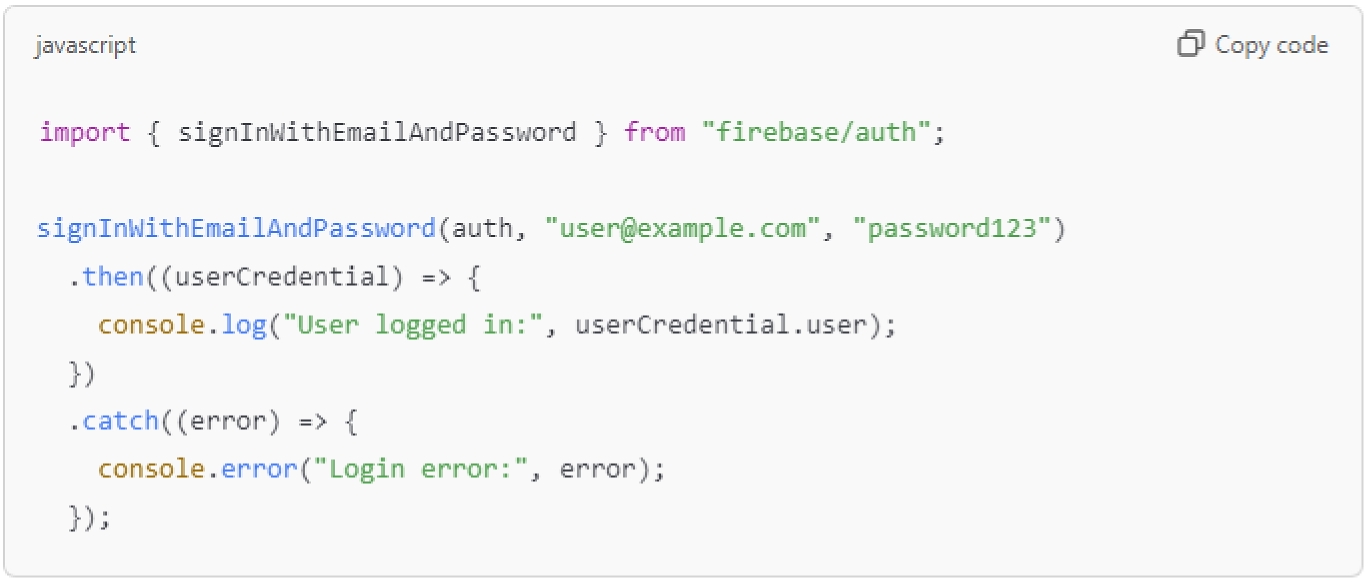
getAuth Se Kya Return Hota Hai?

getAuth ek **auth instance** return karta hai jo Firebase Authentication service ke saath linked hota hai. Ye instance multiple functionalities enable karta hai, jaise:

1. **User Authentication**
   - Sign-in (Email/Password, Google, Facebook, etc.)
   - Sign-up
   - Sign-out
2. **User State Management**
   - Current user ki information retrieve karna.
   - User state changes ko track karna (e.g., login ya logout hone par).
3. **Security**
   - Access tokens manage karna.
   - Backend API ko securely interact karna.

Auth Instance Ka Use

1. **Login Example:**



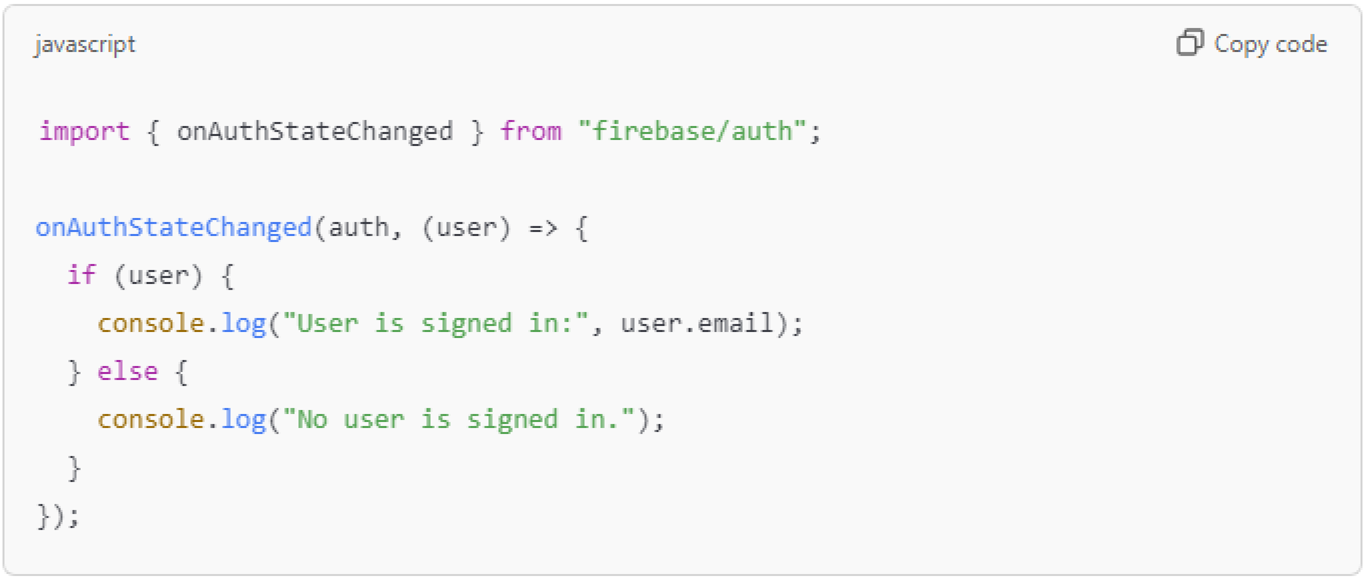
javascript

Copy code

```javascript
import { signInWithEmailAndPassword } from "firebase/auth";

signInWithEmailAndPassword(auth, "user@example.com", "password123")
  .then((userCredential) => {
    console.log("User logged in:", userCredential.user);
  })
  .catch((error) => {
    console.error("Login error:", error);
  });
```

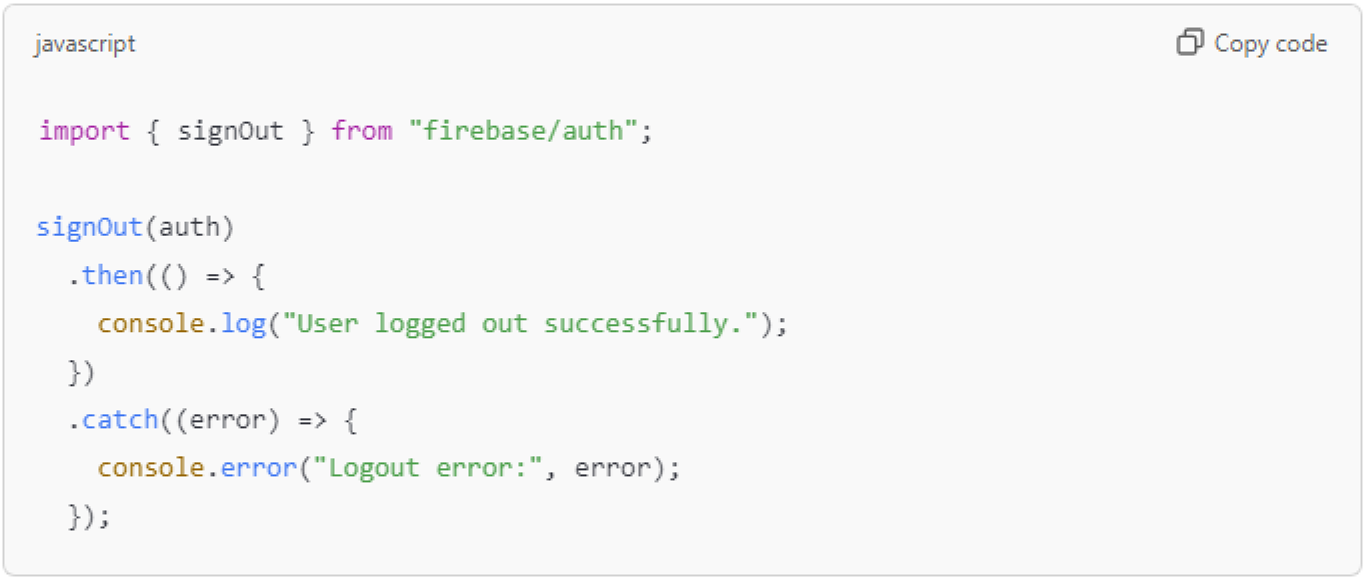
2. Track User State:



```javascript
import { onAuthStateChanged } from "firebase/auth";

onAuthStateChanged(auth, (user) => {
  if (user) {
    console.log("User is signed in:", user.email);
  } else {
    console.log("No user is signed in.");
  }
});
```

3. Logout Example:



```javascript
import { signOut } from "firebase/auth";

signOut(auth)
  .then(() => {
    console.log("User logged out successfully.");
  })
  .catch((error) => {
    console.error("Logout error:", error);
  });
```

Key Points

1. getAuth ek single authentication instance ko manage karta hai. Aapko har operation ke liye ek hi auth instance ka use karna chahiye.
2. Iske bina Firebase Authentication methods ka use nahi kiya ja sakta.
3. Ye Firebase app ke saath tightly integrated hota hai, aur aapke app ke authentication needs ko backend se fulfill karta hai.

getAuth ke through aap current user ki details bhi retrieve kar sakte hain. Jab aap Firebase Authentication ka instance banate hain (auth), to uska **auth.currentUser** property aapko **currently logged-in user** ki details deti hai.

Kaise Current User Milega?

Firebase me, auth.currentUser ke zariye aapko currently logged-in user ki information milti hai. Agar koi user logged-in nahi hai, to auth.currentUser **null** return karega.

Example:

```
import { getAuth } from "firebase/auth";

const auth = getAuth();
const currentUser = auth.currentUser;

if (currentUser) {
  console.log("Logged in user:", currentUser.email);
} else {
  console.log("No user is currently logged in.");
}
```

currentUser Me Kya Hota Hai?

auth.currentUser ek **User** object return karta hai, jisme user ke details hoti hain:

1. **uid**: User ka unique identifier.
2. **email**: User ka email address.
3. **emailVerified**: Boolean value (email verify hua ya nahi).
4. **displayName**: User ka display name (agar set hai).
5. **photoURL**: User ki profile picture URL (agar set hai).
6. **metadata**: User ke creation aur last login ke timestamps.
7. **providerData**: Array, jo user ke authentication providers ke details batata hai (e.g., Google, Facebook, etc.).
8. **refreshToken**: User ka refresh token, jo authentication session ko maintain karta hai.

```
{
  "uid": "abc123xyz",
  "email": "user@example.com",
  "emailVerified": true,
  "displayName": "John Doe",
  "photoURL": "https://example.com/profile.jpg",
  "metadata": {
    "creationTime": "Wed, 01 Jan 2023 10:00:00 GMT",
    "lastSignInTime": "Thu, 02 Jan 2023 12:30:00 GMT"
  },
  "providerData": [
    {
      "providerId": "password",
      "email": "user@example.com",
      "displayName": null,
      "photoURL": null
    }
  ]
}
```

Important Points

1. **auth.currentUser is temporary:**
   - Ye tabhi available hota hai jab user signed-in ho aur app refresh ke baad state restore ho chuki ho.
   - Agar app reload hota hai, to ye Firebase ke asynchronous initialization ke baad populate hota hai.
2. **Use onAuthStateChanged for Real-Time Updates:** Agar aapko ensure karna ho ki user state har waqt accurate ho, to aapko **onAuthStateChanged** listener ka use karna chahiye.

```
import { onAuthStateChanged, getAuth } from "firebase/auth";

const auth = getAuth();

onAuthStateChanged(auth, (user) => {
  if (user) {
    console.log("User is signed in:", user.email);
  } else {
    console.log("No user is signed in.");
  }
});
```

Error Cases

1. **null** return hoga agar:
    - Koi user sign-in nahi hai.
    - Firebase abhi tak initialize nahi hua hai.
    - User ka session expire ho chuka hai.
2. **Token Issues**: Agar current user ka session expire ho gaya hai, to auth.currentUser valid nahi rahega, aur aapko user ko reauthenticate karna padega.

****

Firebase Authentication ka **auth** object console me ek JavaScript object ke form me dikhta hai. Ye Firebase app ke authentication service se linked hota hai aur uski properties aur methods ko encapsulate karta hai.

Agar aap console me auth ko log karenge (e.g., console.log(auth)), to ye kuch is tarah dikhega:

Typical Console Output for auth

```
AuthImpl {
  app: FirebaseAppImpl,
  currentUser: UserImpl,
  config: { apiKey: "AIzaSy1234...", authDomain: "your-app.firebaseapp.com", ... },
  name: "[DEFAULT]",
  tenantId: null,
  emulatorConfig: null,
  ...
}
```

Important Properties in auth:

1. **app**: Firebase app instance jo is auth object se linked hai.
    - **name**: App ka naam.
    - **options**: Firebase configuration details, jaise apiKey, authDomain, etc.
2. **currentUser**:
    - Agar user logged in hai, to ek user object deta hai.
    - Agar user logged in nahi hai, to **null** hota hai.
3. **config**: Firebase Authentication configuration, jaise:
    - apiKey: Firebase project ka unique API key.
    - authDomain: Authentication domain.

4. **emulatorConfig**: Emulator ka configuration agar use ho raha ho.
5. **tenantId**: Multi-tenancy support ke liye tenant ID (default: null).
6. **Methods**: Is object me multiple authentication methods available hote hain, jaise:
    - signInWithEmailAndPassword
    - createUserWithEmailAndPassword
    - signOut
    - onAuthStateChanged, etc.

****