2.1 firebase

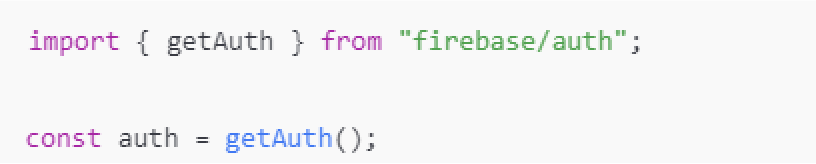
Firebase authentication ka kaam user ko signup, login, logout aur authentication-related tasks ko handle karna hota hai. Aapke question ke context mein, main **getAuth**, **createUserWithEmailAndPassword**, **userCredential**, aur **updateProfile** methods ko detail mein samjhaunga.

---

1. getAuth:

- **Role:** getAuth Firebase Authentication service ko initialize karta hai.
- **Purpose:** Agar aapko authentication-related operations karni hain (like signup, login), to sabse pehle getAuth function ko use karke Firebase Authentication instance lena hota hai.

Syntax:



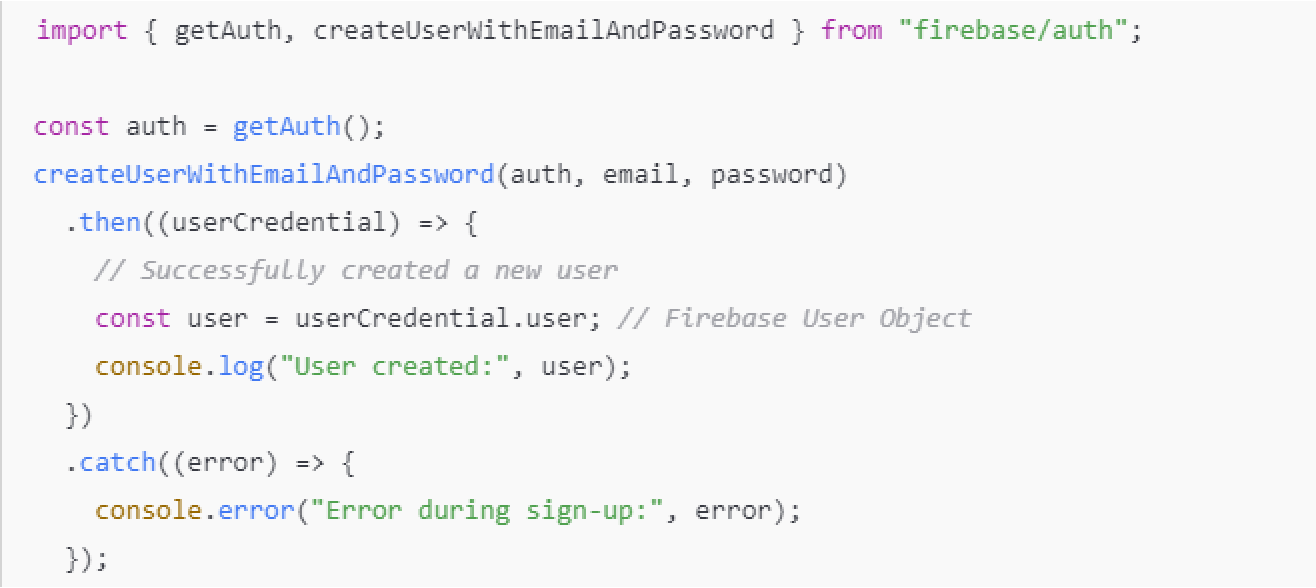
```
import { getAuth } from "firebase/auth";

const auth = getAuth();
```

□ Yahan auth ek instance hai, jo aap Firebase Authentication ke saath interact karne ke liye use karte hain.

□ Agar Firebase ko initialize nahi kiya hoga, to ye method error throw karega. (Firebase initialization ke bina getAuth kaam nahi karega.)

2. createUserWithEmailAndPassword:

- **Role:** Ye function ek naye user ko signup karne ke liye use hota hai, jisme user ka email aur password diya jata hai.
- **Purpose:**
  - Firebase database me ek naye user account ko create karta hai.
  - Agar email pehle se registered hai, to error throw karega (e.g., "email already in use").
  - Ye promise-based function hai, aur successful hone par ek **userCredential** object return karta hai.



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

const auth = getAuth();
createUserWithEmailAndPassword(auth, email, password)
  .then((userCredential) => {
    // Successfully created a new user
    const user = userCredential.user; // Firebase User Object
    console.log("User created:", user);
  })
  .catch((error) => {
    console.error("Error during sign-up:", error);
  });
```

Parameters:

1. **auth**: Authentication instance jo aapne getAuth se banaya hai.

2. **email**: Naye user ka email address.
3. **password**: Naye user ka password.

Return Value:

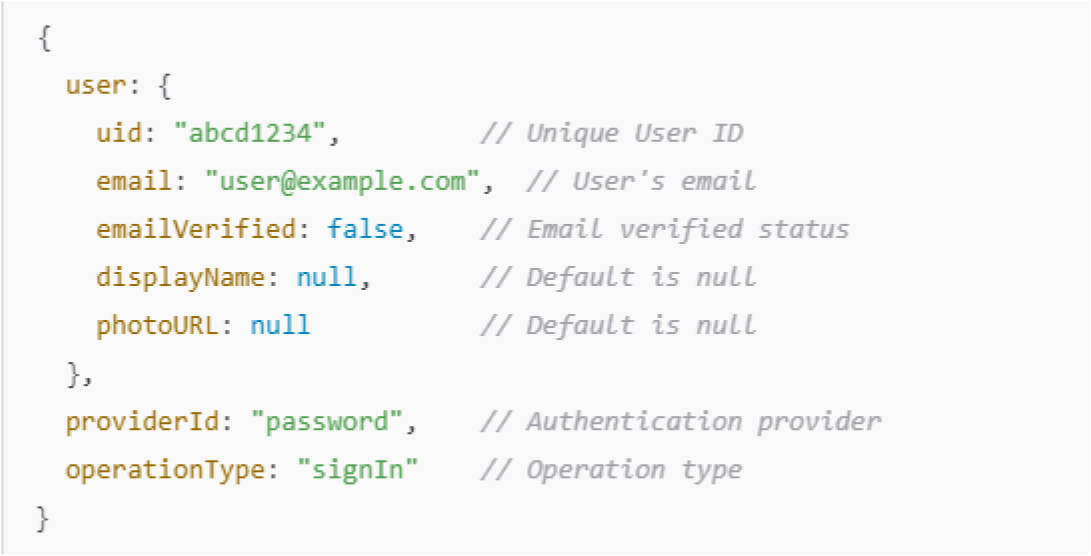
Agar user successfully create hota hai, to ye ek **userCredential** object return karta hai.

3. userCredential:

- **Role:** Ye object createUserWithEmailAndPassword aur kuch doosri Firebase authentication functions ke result ke taur pe return hota hai.
- **Purpose:** Iske andar naye user ke authentication-related details hoti hain.

Structure of userCredential:

1. **user**:
   - Ye ek Firebase **User Object** hai, jo naye user ki details store karta hai.
   - Example properties:
     - **uid**: Unique ID of the user.
     - **email**: User's email address.
     - **emailVerified**: Whether the email is verified (true/false).
     - **displayName**: User ka naam (default me null hota hai).
     - **photoURL**: User ka profile photo URL.
2. **providerId**:
   - Authentication provider ka ID (e.g., "password" for email-password authentication).
3. **operationType**:
   - Kis type ka operation hua hai (e.g., "signIn" ya "signUp").

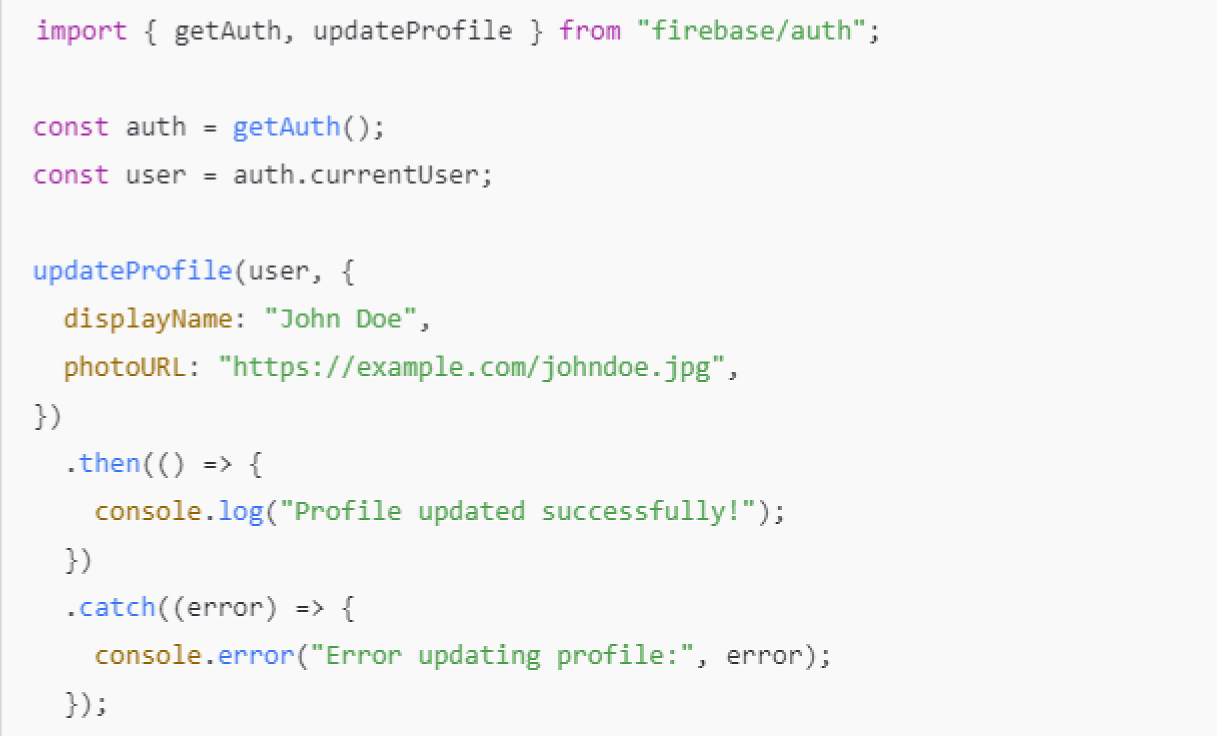


```
{
  user: {
    uid: "abcd1234",          // Unique User ID
    email: "user@example.com",  // User's email
    emailVerified: false,     // Email verified status
    displayName: null,        // Default is null
    photoURL: null            // Default is null
  },
  providerId: "password",     // Authentication provider
  operationType: "signIn"     // Operation type
}
```

4. updateProfile:

- **Role:** updateProfile Firebase User Object ki details ko update karne ke liye use hota hai.
- **Purpose:** Is method ke zariye aap user ka **displayName** aur **photoURL** update kar sakte hain.

Syntax



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

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

updateProfile(user, {
  displayName: "John Doe",
  photoURL: "https://example.com/johndoe.jpg",
})
  .then(() => {
    console.log("Profile updated successfully!");
  })
  .catch((error) => {
    console.error("Error updating profile:", error);
  });
```

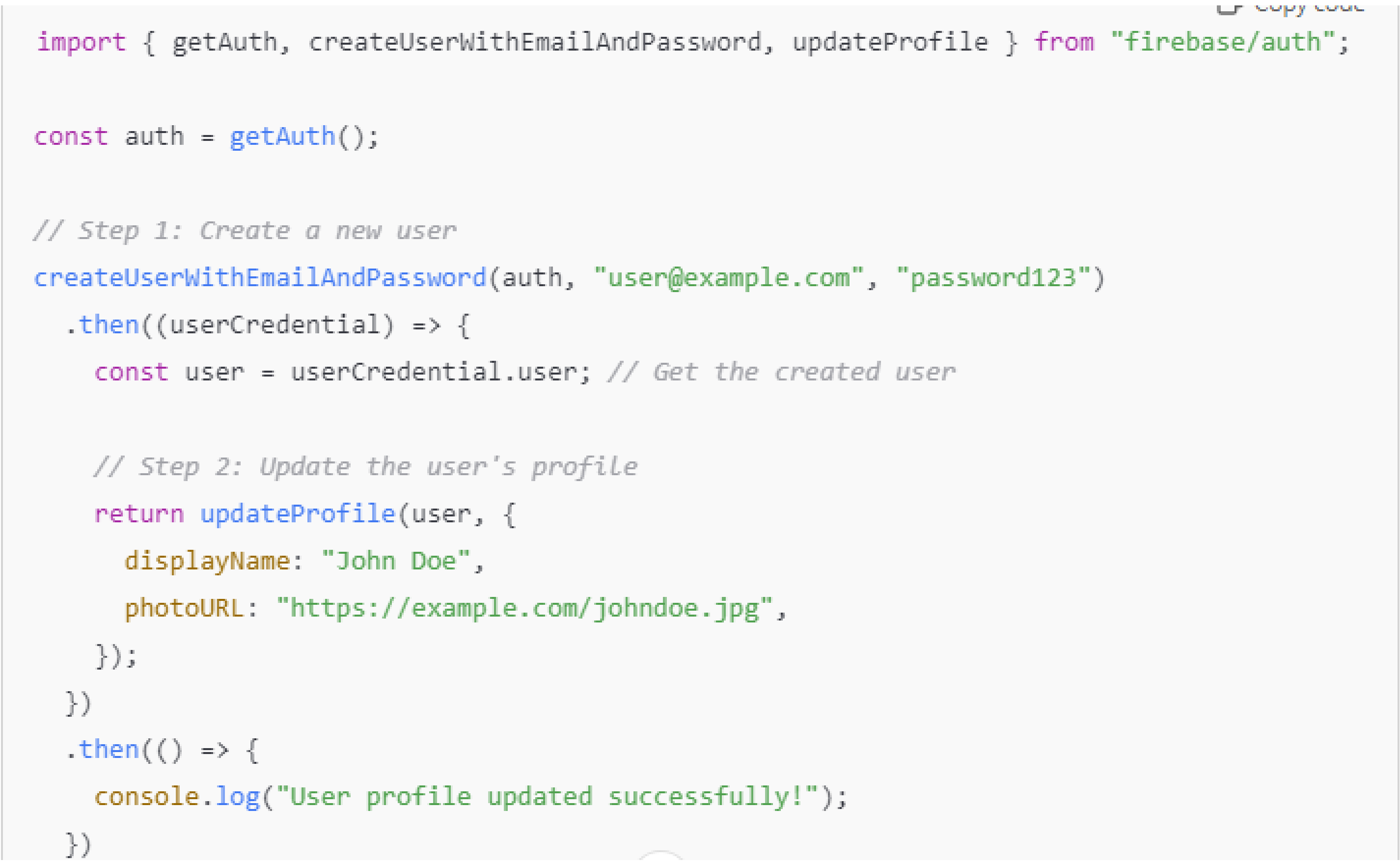
Parameters:

1. **user**: Firebase User Object jo aapko auth.currentUser se milta hai.
2. **Object**:
   - **displayName**: Naya naam jo aap set karna chahte hain.
   - **photoURL**: Nayi profile photo ka URL.

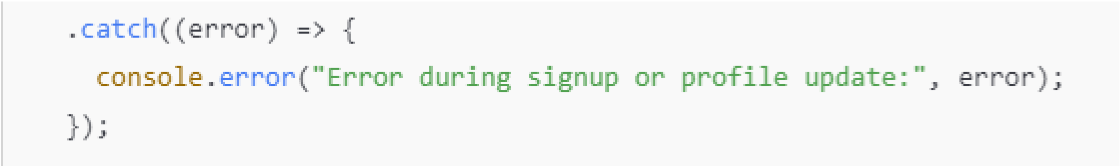
Use Cases:

1. **Updating Name and Profile Photo**:
   - Jab user signup kare aur aapko uska naam aur profile photo set karni ho.
2. **Real-Time Updates**:
   - Profile update karne ke baad agar aap real-time changes dekhna chahte ho, to aap auth.currentUser ko dobara fetch kar sakte ho.

Example Flow for Signup and Profile Update:



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

const auth = getAuth();

// Step 1: Create a new user
createUserWithEmailAndPassword(auth, "user@example.com", "password123")
  .then((userCredential) => {
    const user = userCredential.user; // Get the created user

    // Step 2: Update the user's profile
    return updateProfile(user, {
      displayName: "John Doe",
      photoURL: "https://example.com/johndoe.jpg",
    });
  })
  .then(() => {
    console.log("User profile updated successfully!");
  })
```



```
  .catch((error) => {
    console.error("Error during signup or profile update:", error);
  });
```

Key Points Summary:

1. **getAuth**:
   - Firebase Authentication ko initialize karta hai.
2. **createUserWithEmailAndPassword**:
   - Email aur password ke saath ek naye user ka account banata hai.
   - userCredential return karta hai, jo naye user ki details store karta hai.
3. **userCredential**:
   - Naye user ki authentication-related details provide karta hai, jaise uid, email, etc.
4. **updateProfile**:
   - User ka naam (displayName) aur profile photo (photoURL) update karne ke liye use hota hai.

Firebase mein authentication-related operations ke liye ye methods bohot commonly use hote hain aur har ek ka apna specific role hai.

****