خليني أشرحلك فكرة الكود ده بالعربي بطريقة بسيطة وواضحة، وأوضح إزاي بيشتغل خطوة بخطوة:

### \*\*الفكرة العامة:\*\*

الكود ده بيستخدم في تطبيقات \*\*React\*\* عشان يدير عملية \*\*المصادقة (Authentication)\*\* باستخدام \*\*JWT (JSON Web Tokens)\*\*، وبالأخص بيستخدم \*\*Access Token\*\* و\*\*Refresh Token\*\*. الهدف هو:

1. إدارة التوكن (Access Token) اللي بيستخدم لتأمين الطلبات (Requests) للسيرفر.

2. تحديث التوكن تلقائيًا لو انتهت صلاحيته (باستخدام Refresh Token).

3. توفير طريقة سهلة لكل المكونات (Components) في التطبيق عشان تستخدم التوكن ده.

الكود بيستخدم \*\*React Context\*\* عشان يشارك التوكن بين المكونات، ويستخدم \*\*Fetch API\*\* (بدل Axios) للتعامل مع طلبات الـ API، مع إضافة ميزة اعتراض (Interceptors) لإدارة الطلبات والردود.

---

### \*\*الدائرة بتمشي إزاي؟\*\*

هنقسم الشرح لجزئين: \*\*المكونات الرئيسية\*\* و\*\*إزاي بيشتغلوا مع بعض\*\*.

#### \*\*1. المكونات الرئيسية:\*\*

الكود مقسم لثلاثة أجزاء رئيسية:

- \*\*AuthContext\*\*: ده السياق (Context) اللي بيخزن التوكن ويشاركه مع باقي المكونات.

- \*\*useAuth\*\*: هوك مخصص (Custom Hook) بيسمح لأي مكون يستخدم التوكن بسهولة.

- \*\*AuthProvider\*\*: مكون رئيسي بيحيط بالتطبيق كله (عادةً في الملف الرئيسي `App.js`) وبيدير التوكن ويوفره لكل المكونات.

- \*\*TokenRefreshHandler\*\*: مكون مسؤول عن إدارة تحديث التوكن تلقائيًا وإضافة رأس الـ Authorization لكل طلب.

#### \*\*2. خطوات العمل:\*\*

1. \*\*إنشاء السياق (AuthContext)\*\*:

- الكود بيستخدم `createContext` عشان يعمل سياق (مكان مركزي يخزن فيه التوكن).

- السياق ده بيحتوي على التوكن (`token`) ودالة لتحديثه (`setToken`).

2. \*\*جلب التوكن عند بداية التطبيق\*\*:

- في `AuthProvider`، لما التطبيق يشتغل، بيستخدم `useEffect` عشان يعمل طلب (Fetch) لـ `/api/auth/me`.

- الطلب ده بيجيب التوكن (Access Token) من السيرفر لو المستخدم مسجل دخوله.

- لو نجح الطلب، بيخزن التوكن في الـ `state` باستخدام `setToken`.

- لو فشل الطلب (مثلاً المستخدم مش مسجل دخوله)، بيحط `token` بـ `null`.

3. \*\*إضافة التوكن لكل طلب (Request Interceptor)\*\*:

- في `TokenRefreshHandler`، الكود بيستخدم `useLayoutEffect` عشان يعمل محاكاة لـ "Request Interceptor" (زي اللي في Axios).

- كل طلب بيطلع بيتمرر من خلال دالة `requestInterceptorRef.current` اللي بتضيف رأس `Authorization: Bearer <token>` لو التوكن موجود ومش طلب إعادة محاولة (`\_retry`).

- ده بيضمن إن كل طلبات الـ API بتاعتك (اللي بتحتاج مصادقة) بتترح مع التوكن.

4. \*\*تحديث التوكن لو انتهت صلاحيته (Response Interceptor)\*\*:

- الكود بيستخدم `useLayoutEffect` تاني في `TokenRefreshHandler` عشان يراقب ردود السيرفر (Responses).

- لو السيرفر رد بـ `403 Unauthorized` مع رسالة "Unauthorized"، يعني التوكن منتهي الصلاحية.

- الكود هنا بيعمل طلب جديد لـ `/api/auth/refreshToken` عشان يجيب توكن جديد (باستخدام Refresh Token اللي مخزن عند السيرفر).

- لو نجح الطلب، بيحدث التوكن في الـ `state` باستخدام `setToken`، وبيعيد الطلب الأصلي مع التوكن الجديد.

- لو فشل طلب التحديث، بيحط `token` بـ `null` (يعني المستخدم لازم يسجل دخوله من جديد).

5. \*\*توفير التوكن للمكونات\*\*:

- `AuthProvider` بيوفر التوكن ودالة `setToken` لكل المكونات من خلال `AuthContext.Provider`.

- أي مكون في التطبيق يقدر يستخدم `useAuth()` عشان يجيب التوكن ويستخدمه في طلبات الـ API أو للتحقق من حالة المصادقة.

6. \*\*التنظيف (Cleanup)\*\*:

- الكود بيستخدم دوال التنظيف في `useLayoutEffect` عشان يوقف الاعتراض (Interceptors) لو المكون تم إلغاء تحميله (Unmount).

- ده بيمنع أي مشاكل زي تسريب الذاكرة (Memory Leaks).

---

### \*\*إزاي تستخدم الكود؟\*\*

1. \*\*لف التطبيق بـ AuthProvider\*\*:

في الملف الرئيسي (`App.js` أو `index.js`)، حط `AuthProvider` حوالين التطبيق:

```jsx

import AuthProvider from './Auth';

function App() {

return (

<AuthProvider>

{/\* باقي التطبيق \*/}

</AuthProvider>

);

}

```

2. \*\*استخدام useAuth في المكونات\*\*:

لو عايز تستخدم التوكن في أي مكون، استخدم `useAuth`:

```jsx

import { useAuth } from './Auth';

function MyComponent() {

const { token } = useAuth();

console.log(token); // هيطبع التوكن لو موجود

return <div>مرحبا!</div>;

}

```

3. \*\*طلبات API\*\*:

لما تعمل طلبات Fetch في المكونات، الكود في `TokenRefreshHandler` هيضيف رأس الـ Authorization تلقائيًا ويتعامل مع تحديث التوكن لو انتهت صلاحيته.

---

### \*\*ليه الكود مقسم؟\*\*

- \*\*AuthProvider\*\*: مسؤول عن إدارة الحالة (State) وتوفير السياق.

- \*\*TokenRefreshHandler\*\*: بيفصل منطق الاعتراض (Interceptors) عشان الكود يكون أوضح وأسهل في الصيانة.

- \*\*useAuth\*\*: بيسهل استخدام التوكن في أي مكون بدون تعقيد.

### \*\*مميزات الكود:\*\*

- \*\*تلقائي\*\*: بيحدث التوكن تلقائيًا لو انتهت صلاحيته.

- \*\*منظم\*\*: الكود مقسم لمكونات صغيرة، كل مكون عنده مهمة واضحة.

- \*\*آمن\*\*: بيحط `token` بـ `null` لو حصلت أي مشكلة في المصادقة.

- \*\*مرن\*\*: بيسمح لكل المكونات تستخدم التوكن بسهولة.

### \*\*ملاحظات:\*\*

- الكود بيفترض إن عندك سيرفر بيوفر الـ APIs (`/api/auth/me` و `/api/auth/refreshToken`) بنفس الصيغة.

- لو عايز تضيف ميزات زي تسجيل الخروج (Logout)، ممكن تضيف دالة في `AuthProvider` تخلي `setToken(null)` وتعمل طلب للسيرفر عشان تبطل الـ Refresh Token.

لو عندك أي سؤال إضافي أو عايز توضيح لجزء معين، قولي!