# **. FINAL OAUTH PATCH (FRONTEND + BACKEND)**

**This patch:**

* Makes sure only Supabase handles Google OAuth (no manual auth URL logic).
* Fixes session/cookie handling so login persists.
* Adds recommended Google query params for incremental/granular permissions.
* Ensures correct redirect and YouTube API token usage.

## **FRONTEND (React/Next.js/Vanilla)**

### **A. Update your Google Login button logic:**

js

CopyEdit

// login.tsx or wherever your "Sign in with Google" button is

import { supabase } from './supabaseClient'; // adjust path as needed

const handleGoogleLogin = () => {

supabase.auth.signInWithOAuth({

provider: 'google',

options: {

redirectTo: `${window.location.origin}/auth/callback`, // or your callback route

queryParams: {

access\_type: 'offline',

prompt: 'consent',

include\_granted\_scopes: 'true',

scope: [

'openid',

'email',

'profile',

'https://www.googleapis.com/auth/youtube',

'https://www.googleapis.com/auth/youtube.force-ssl',

'https://www.googleapis.com/auth/yt-analytics.readonly'

].join(' ')

}

}

});

};

**Button Example:**

jsx

CopyEdit

<Button onClick={handleGoogleLogin}>

Sign in with Google

</Button>

### **B. Callback/Session Bridge Logic (auth/callback page/component):**

js

CopyEdit

import { useEffect } from 'react';

import { supabase } from './supabaseClient';

export default function AuthCallback() {

useEffect(() => {

const hash = window.location.hash.substring(1);

const params = new URLSearchParams(hash);

const access\_token = params.get('access\_token');

const refresh\_token = params.get('refresh\_token');

if (access\_token && refresh\_token) {

supabase.auth.setSession({

access\_token,

refresh\_token,

}).then(async () => {

// POST tokens to your backend to set secure cookies (for SSR/API calls)

await fetch('/api/auth/session', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

credentials: 'include',

body: JSON.stringify({ access\_token, refresh\_token }),

});

window.location.href = '/dashboard'; // or your preferred dashboard route

});

} else {

// Handle error - show "Login failed, please try again"

window.location.href = '/login?error=missing\_tokens';

}

}, []);

return <div>Signing you in…</div>;

}

## **BACKEND**

### **A. Remove All Manual Google OAuth Routes**

* **Delete any /api/auth/google or /api/auth/callback/google endpoints.**
* **No more “no\_code” or manual OAuth error handling.**
* Your backend should only care about the Supabase sb-access-token cookie for authenticated API requests.

### **B. Auth Session Cookie Bridge Endpoint**

Add this to your Express/Node backend if not already present:

js

CopyEdit

app.post('/api/auth/session', async (req, res) => {

const { access\_token, refresh\_token } = req.body;

if (!access\_token || !refresh\_token) {

return res.status(400).json({ error: 'Missing tokens' });

}

// Set HttpOnly cookies for backend-authenticated API requests

res.cookie('sb-access-token', access\_token, {

httpOnly: true,

secure: true, // set to true in prod

sameSite: 'lax',

path: '/',

domain: process.env.NODE\_ENV === 'production' ? 'titletesterpro.com' : undefined

});

res.cookie('sb-refresh-token', refresh\_token, {

httpOnly: true,

secure: true,

sameSite: 'lax',

path: '/',

domain: process.env.NODE\_ENV === 'production' ? 'titletesterpro.com' : undefined

});

res.json({ success: true });

});

**All other routes that need a logged-in user should check for the sb-access-token cookie.**

### **C. Backend: Remove Any Old Neon/Non-Supabase DB Connections**

* Remove any DATABASE\_URL or code referencing Neon database.
* Only use Supabase’s client/SDK for database calls.