**You are an expert code agent for TitleTesterPRO. Your job is to guarantee a flawless, secure, and fully working login/session flow using ONLY Supabase Auth with Google as provider.**

### **🔒 Security, Stability, and Accuracy Mandates**

* **Absolutely NO manual Google OAuth logic** anywhere (no authUrl construction, no custom code exchanges, no legacy handlers, no direct use of Google Client Secret/ID on the front-end).
* **Supabase must be the ONLY place handling Google OAuth**. No Neon/Postgres or other DB references anywhere.
* **All environment variables referencing NEON, alternative databases, or manual OAuth flows must be deleted.**

## **TASKS**

### **1. Remove Legacy/Manual OAuth Code**

* Delete all files/functions that:  
  + Construct Google OAuth URLs
  + Handle code exchanges, token retrieval, or “no\_code” errors manually
  + Import/use Google Client Secret/ID directly in frontend or backend for login

**Only use:** js  
CopyEdit  
// Supabase v2+ method for Google login

supabase.auth.signInWithOAuth({ provider: 'google' });

* + Or, if v1:

js  
CopyEdit  
supabase.auth.signIn({ provider: 'google' });

* Remove any custom /auth/callback, /auth/google, /api/auth/\* endpoints except those required by Supabase.

### **2. Front-End Login Example (React/Next/Vite/Remix)**

#### **Replace your login button code with:**

jsx

CopyEdit

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

const handleGoogleLogin = () => {

supabase.auth.signInWithOAuth({

provider: 'google',

options: {

redirectTo: window.location.origin // or your production URL

}

});

};

<button onClick={handleGoogleLogin}>

Sign in with Google

</button>

#### **Session Handling Example:**

js

CopyEdit

import { useEffect, useState } from "react";

import { supabase } from './supabaseClient';

export default function AuthWrapper({ children }) {

const [session, setSession] = useState(null);

useEffect(() => {

supabase.auth.getSession().then(({ data: { session } }) => {

setSession(session);

});

const { data: { subscription } } = supabase.auth.onAuthStateChange(

(\_event, session) => setSession(session)

);

return () => subscription.unsubscribe();

}, []);

if (!session) {

return <LoginComponent />;

}

return children;

}

### **3. Supabase/Google Cloud Config Matching**

* Confirm the following matches in **both**:  
  + **Supabase Dashboard > Auth > Providers > Google**
    - Client ID: 618794070994-n4n3b75oktui4efj7671il3jvef23peu.apps.googleusercontent.com
    - Client Secret: GOCSPX-DGR6-7QEK1bvdd-frKlCo\_F-30TR
    - Callback URL: https://xyehwoacgpsxakhjwglq.supabase.co/auth/v1/callback
  + **Google Cloud > Credentials > OAuth 2.0 Client**
    - Authorized redirect URI: https://xyehwoacgpsxakhjwglq.supabase.co/auth/v1/callback
    - **AND** add local/dev/prod URLs if you test locally (e.g. http://localhost:5173/auth/callback)
    - Authorized JavaScript origins: https://titletesterpro.com, https://TTPro3.replit.app (as shown)
* **Authorized domains**: Ensure all domains used are in both Google and Supabase settings.

### **4. Remove Neon & Unused DB References**

* **Delete any NEON/old DB env vars** (e.g. DATABASE\_URL, NEON\_URL, etc).
* Ensure .env, Supabase config, deployment scripts reference ONLY Supabase Postgres.
* **Do not import/use NEON client, or any Postgres driver, except what’s used for Supabase integration.**

### **5. Session Handling and Cookie Fixes**

* The only cookie/session tokens used should be the ones set by Supabase:  
  + sb-access-token
  + sb-refresh-token
* **Backend endpoints** (e.g. /api/auth/user) should get the user from the Supabase session/cookie only.
* Never read access tokens from the URL after login, always let Supabase manage all session state.

### **6. Frontend/Backend Integration Checklist**

* **No manual token reads from URL hash, code, or custom storage.**
* **After login:** All API requests should send the Supabase session access token.

**No need for a custom logout unless you want to add:** js  
CopyEdit  
supabase.auth.signOut();

### **7. Resolve Google Console Issues**

* **Domain Verification**: titletesterpro.com must be verified in Search Console and appear in Google Cloud Console under “authorized domains”.
* **Incremental Authorization / Granular Permissions**: Accept the warning, or update your app to use granular scopes if needed. These do **not** break login, but can be addressed later.
* **Add Your Domains in Both Google Cloud & Supabase**
* **Fix yellow warnings about JavaScript origins, redirect URIs, etc.**

### **8. Accessibility Audit**

* For all login forms/inputs, add:  
  + <input id="google-signin" name="google-signin" ... />
  + Or, wrap inputs in <label for="google-signin"> ... </label>
  + Resolve all warnings in your devtools about missing labels, ids, or names.

### **9. README Update**

Document that Google login is now handled **100% through Supabase** with callback URL:  
  
 bash  
CopyEdit  
https://xyehwoacgpsxakhjwglq.supabase.co/auth/v1/callback

* **No other login methods or database references exist.**

### **10. Test**

* Login from scratch with Google, ensure no more redirects, “no active session,” or instant logouts.
* Ensure /api/auth/user or equivalent returns the current user when logged in.