# **🚨 FINAL SUPER-PROMPT: FIX TITLE TESTER PRO YOUTUBE ANALYTICS FAILURE**

## **🎯 OBJECTIVE**

Fix the failure of **Title Tester Pro** to retrieve YouTube Analytics (v2) data while YouTube Data API (v3) still works.  
 Your credibility and this project’s budget are on the line. No guesses. No hand-waving. No skipping steps.  
 ✅ Diagnose all plausible root causes.  
 ✅ Fix them in code, config, or DB.  
 ✅ Prove the fix with live data & logs.

# **🔍 EXECUTION PLAN**

You will resolve this systematically in **6 phases**, addressing API, OAuth, DB, frontend, and runtime context.

You MUST:  
 ✅ Check every plausible cause (OAuth, GCP, DB, channel mismatch, frontend).  
 ✅ Run all commands/scripts, validate assumptions, and show proof.  
 ✅ Log outputs at each step.  
 ✅ Deliver working analytics data flowing into dashboard with active titles, intervals, and logs showing correctly.

## **🔷 PHASE 1 — CODE & CONFIG AUDIT**

🎯 Inspect the repo for incorrect or missing scopes, API params, and DB config.

📁 Extract and audit:

* .env, .env.\* files
* OAuth-related files:  
   server/googleAuth.ts, server/services/auth.js, server/routes/auth.js, server/middleware/auth.js
* YouTube API files:  
   server/services/youtube.js, server/services/youtubeAnalytics.js, server/routes/youtube.js, server/lib/youtube.js
* DB query files:  
   server/services/database.js, server/models/\*.js
* Dashboard fetch logic:  
   client/src/hooks/useDashboardData.ts, useTests.ts, useYouTube.ts, client/src/services/api.ts

Search for scopes & config:

bash

CopyEdit

grep -r "googleapis.com/auth" ..

grep -r "SCOPES" ..

grep -r "scope" ..

cat .env

✅ Verify scopes include BOTH:

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

✅ Confirm DB points to correct instance (Supabase or Neon — not both).

## **🔷 PHASE 2 — RUNTIME TOKEN & REQUEST VALIDATION**

🎯 Verify the token and API request *at runtime*.

Check the token:

bash

CopyEdit

curl -H "Authorization: Bearer <ACCESS\_TOKEN>" https://www.googleapis.com/oauth2/v1/tokeninfo

✅ Confirm:

* Both scopes are present.
* Token is tied to correct Google account/channel.

Check the API request:

* ids: must be channel==MINE
* filters: must be video==<VIDEO\_ID>
* startDate and endDate: valid, in YYYY-MM-DD, within video’s lifetime.
* metrics: e.g., impressions.
* dimensions: optional, e.g., day.

Sample request:

js

CopyEdit

const analytics = await youtubeAnalytics.reports.query({

ids: 'channel==MINE',

startDate: '2024-01-01',

endDate: '2024-01-07',

metrics: 'impressions',

filters: `video==${videoId}`,

dimensions: 'day'

});

🚨 Additionally:  
 Pay close attention to the **Google account context**. Confirm the authenticated account owns the videoId being queried. If the account is tied to a **Brand Account**, ensure the user selects the correct Brand during OAuth and that channel==MINE resolves to the right channel ID. Use YouTube Data API mine=true to confirm the channel ID matches videoId.

## **🔷 PHASE 3 — ISOLATED PIPELINE TEST**

🎯 Run a single script to validate OAuth → API → DB.

Write & run debug-pipeline.js:

js

CopyEdit

const { google } = require('googleapis');

require('dotenv').config();

const oauth2Client = new google.auth.OAuth2(

process.env.GOOGLE\_CLIENT\_ID,

process.env.GOOGLE\_CLIENT\_SECRET,

process.env.OAUTH\_REDIRECT\_URI

);

oauth2Client.setCredentials({

access\_token: '<ACCESS\_TOKEN>',

refresh\_token: '<REFRESH\_TOKEN>'

});

(async () => {

try {

const youtube = google.youtube({ version: 'v3', auth: oauth2Client });

const channel = await youtube.channels.list({ part: 'snippet', mine: true });

console.log('Channel:', channel.data);

const youtubeAnalytics = google.youtubeAnalytics({ version: 'v2', auth: oauth2Client });

const analytics = await youtubeAnalytics.reports.query({

ids: 'channel==MINE',

startDate: '2024-01-01',

endDate: '2024-01-07',

metrics: 'impressions',

filters: `video==<VIDEO\_ID>`,

dimensions: 'day'

});

console.log('Analytics:', analytics.data);

// Insert into DB here if desired

} catch (err) {

console.error('ERROR:', err.message, err.response?.data || '');

}

})();

✅ Ensure it logs rows with impressions.

## **🔷 PHASE 4 — DATABASE (SUPABASE/NEON) AUDIT**

🎯 Validate schema, RLS, and data integrity.

Run:

sql

CopyEdit

SELECT table\_name, column\_name, data\_type

FROM information\_schema.columns

WHERE table\_schema = 'public'

ORDER BY table\_name, ordinal\_position;

SELECT 'tests' AS table\_name, COUNT(\*) AS row\_count FROM tests

UNION ALL

SELECT 'videos', COUNT(\*) FROM videos

UNION ALL

SELECT 'titles', COUNT(\*) FROM titles

UNION ALL

SELECT 'analytics\_polls', COUNT(\*) FROM analytics\_polls

UNION ALL

SELECT 'test\_rotation\_logs', COUNT(\*) FROM test\_rotation\_logs;

✅ Check that:

* Tables exist & have data.
* Schema matches API response structure.
* No write failures in logs.
* Row-Level Security (RLS) is not blocking reads/writes. Disable RLS temporarily if needed to test.

## **🔷 PHASE 5 — FRONTEND & DASHBOARD**

🎯 Ensure dashboard displays:  
 ✅ Active title, timestamp, interval.  
 ✅ History of rotated titles, intervals & timestamps.

Verify backend exposes endpoint:

bash

CopyEdit

GET /api/title-rotations?videoId=XXX

should return:

json

CopyEdit

{

"activeTitle": "Current Title",

"activeSince": "2025-07-06T14:00:00Z",

"interval": 3600,

"history": [

{

"title": "Title 1",

"timestamp": "2025-07-06T12:00:00Z",

"interval": 3600

},

…

]

}

✅ Confirm React/Vue components call this endpoint, and state updates properly.  
 ✅ Fix queries or UI rendering if blank.

## **🔷 PHASE 6 — FINAL VERIFICATION**

🎯 Prove everything works:  
 ✅ Run end-to-end: OAuth → API → DB → Dashboard.  
 ✅ Check Google Cloud Metrics: success rate > 95% on /TargetedQueriesService.Query.  
 ✅ Dashboard shows current + historical titles & intervals.  
 ✅ No 400/403/401 errors in logs.

# **🚨 RULES**

⚠️ Do NOT assume — VERIFY.  
 ⚠️ Do NOT skip steps.  
 ⚠️ DO execute every diagnostic.  
 ⚠️ DO log all findings & fixes.  
 ⚠️ DO show API response, DB rows, dashboard screenshot as proof.

If you fail to resolve this, you either:  
 — Skipped steps.  
 — Failed to validate assumptions.  
 — Or the channel/account lacks proper ownership — escalate in that case.

# **📄 DELIVERABLES**

✅ List of root cause(s).  
 ✅ Fixes applied.  
 ✅ Updated code snippets.  
 ✅ API response sample.  
 ✅ DB query results.  
 ✅ Screenshot of working dashboard.

### **👊 Now execute. Fix it. Log every step. Show proof.**

If you want, I can also package this into:

* 📄 A .md file for your repo.
* 📝 A .pdf to hand to devs.
* 🚀 A ready-to-paste Replit command block.