Developer Instructions: Rebuild Diagnostics System from Scratch

Overview

The previous diagnostics implementation consisted of three files:

1. server/actions/gmb-sync-diagnostics.ts

2. components/gmb/sync-diagnostics.tsx

3. app/[locale]/(dashboard)/sync-diagnostics/page.tsx

These files no longer function properly due to:

• Server Actions being invoked inside Client Components (not supported)

• Route conflicts with /sync-diagnostics page

• Server Action returning HTML instead of JSON

• Inconsistent data reads (adminClient / RLS issues)

• Architecture not scalable for multi■test diagnostics

Action Required

Delete the three files listed above entirely and rebuild a clean Diagnostics System.

New Architecture Requirements

1. Create a new page:

app/[locale]/(dashboard)/owner-diagnostics/page.tsx

2. Implement a diagnostics UI with tabs:

• OAuth Connection Test

• Admin Access Test

• Database Health Test

• Sync Queue Test

• Data Sync Counts

• Worker / Cron Status

• Sync Logs

• API Health

• Data Integrity Check

3. Each tab must contain:

• A title describing the test

• A "Run Test" button

• A call to a dedicated API route

• A JSON result rendered in a card

4. Create dedicated API routes:

app/api/diagnostics/[testName]/route.ts

Examples:

• /api/diagnostics/oauth

• /api/diagnostics/admin

• /api/diagnostics/db

• /api/diagnostics/sync-queue

• /api/diagnostics/data-counts

• /api/diagnostics/logs

• /api/diagnostics/api-health

• /api/diagnostics/integrity

5. Each API route must return JSON in the format:

{

"success": true,

"details": { ... }

}

Goal

Provide a fully reliable, modular, and expandable diagnostics system that does not depend on Server Actions inside Client Components, avoids routing conflicts, and gives clear insights into system health for each user.