🔍 Prisma Database Diagnostic - Complete Batch Analysis

Purpose: Systematic investigation of Prisma Client generation issue where `auditLog` model is not accessible despite valid schema.

Current Symptoms:

- ✅ Prisma schema validates successfully
- ✅ Prisma Client generates without errors
- ❌ Generated client does NOT include `auditLog` model accessor
- ❌ TypeScript compilation fails: "Property 'auditLog' does not exist on type 'PrismaClient'"

📋 PHASE 1: IMMEDIATE VERIFICATION (Run First)

Critical Questions to Answer:

1. Does the auditLog model actually exist in the schema?
2. Is auditLog present in the generated Prisma Client?
3. Are we importing PrismaClient correctly?

Commands to Run:

powershell

```
# Q1.1: Verify auditLog model exists in schema
Write-Host "`n=== Q1.1: SCHEMA MODEL DEFINITION ===" -ForegroundColor Cyan
Get-Content prisma\schema.prisma | Select-String -Pattern "model AuditLog" -Context 0,30

# Q1.2: Check datasource configuration
Write-Host "`n=== Q1.2: DATASOURCE CONFIGURATION ===" -ForegroundColor Cyan
Get-Content prisma\schema.prisma | Select-String -Pattern "datasource|provider"

# Q1.3: Check generator configuration
Write-Host "`n=== Q1.3: GENERATOR CONFIGURATION ===" -ForegroundColor Cyan
Get-Content prisma\schema.prisma | Select-String -Pattern "generator" -Context 0,5

# Q1.4: Validate schema syntax
Write-Host "`n=== Q1.4: SCHEMA VALIDATION ===" -ForegroundColor Cyan
npx prisma validate --schema=prisma\schema.prisma

# Q2.2: List ALL model accessors in generated client
Write-Host "`n=== Q2.2: GENERATED MODEL ACCESSORS ===" -ForegroundColor Cyan
Get-Content node_modules\.prisma\client\index.d.ts | Select-String "^\s+get\s+\w+\(\):\s+Prisma\.\w+Delegate" | ForEach-

# Q2.3: Check AuditLog exports from client
Write-Host "`n=== Q2.3: AUDITLOG EXPORTS ===" -ForegroundColor Cyan
Get-Content node_modules\@prisma\client\index.d.ts | Select-String "export.*AuditLog" | Select-Object -First 20

# Q3.1: Verify database service import
Write-Host "`n=== Q3.1: DATABASE SERVICE IMPORT ===" -ForegroundColor Cyan
Get-Content src\services\database.ts | Select-String -Pattern "import.*PrismaClient"

# Q3.4: Verify node_modules integrity
Write-Host "`n=== Q3.4: NODE MODULES INTEGRITY ===" -ForegroundColor Cyan
Write-Host "Checking @prisma/client/index.d.ts exists:" -ForegroundColor Yellow
Test-Path node_modules\@prisma\client\index.d.ts
Write-Host "Checking .prisma/client/index.d.ts exists:" -ForegroundColor Yellow
Test-Path node_modules\.prisma\client\index.d.ts
Write-Host "File count in @prisma/client:" -ForegroundColor Yellow
(Get-ChildItem node_modules\@prisma\client | Measure-Object).Count
```

Phase 1 Decision Point:

STOP HERE and report results before continuing.

Expected outcomes:

- ✅ If auditLog model is missing from schema → Schema corruption issue
- ✅ If auditLog is in schema but NOT in generated client → Continue to Phase 2
- ✅ If auditLog is in generated client but import fails → Continue to Phase 3

📋 PHASE 2: SCHEMA & GENERATION DEEP DIVE

Run this phase if: auditLog exists in schema but is missing from generated client.

Commands to Run:

powershell

```
# Q2.1: Verify generation output location
Write-Host "`n=== Q2.1: CLIENT GENERATION OUTPUT LOCATION ===" -ForegroundColor Cyan
Write-Host "Checking for custom output path in schema:" -ForegroundColor Yellow
Get-Content prisma\schema.prisma | Select-String -Pattern "output"
Write-Host "`nChecking for prisma config in package.json:" -ForegroundColor Yellow
Get-Content package.json | Select-String -Pattern '"prisma"' -Context 0,10

# Q2.4: Verify Prisma version consistency
Write-Host "`n=== Q2.4: PRISMA VERSION CONSISTENCY ===" -ForegroundColor Cyan
Write-Host "Installed Prisma packages:" -ForegroundColor Yellow
npm list prisma @prisma/client
Write-Host "`nPrisma CLI version:" -ForegroundColor Yellow
npx prisma --version

# Q2.5: Check generation timestamp
Write-Host "`n=== Q2.5: GENERATION TIMESTAMP ===" -ForegroundColor Cyan
Get-ChildItem node_modules\.prisma\client -Recurse | Select-Object FullName, LastWriteTime | Sort-Object LastWriteTi

# Q4.1: Check model name case sensitivity
Write-Host "`n=== Q4.1: MODEL NAME CASE SENSITIVITY ===" -ForegroundColor Cyan
Write-Host "Schema definition (looking for *Log models):" -ForegroundColor Yellow
Get-Content prisma\schema.prisma | Select-String "model\s+\w+Log"
Write-Host "`nCode usage in auditLog service:" -ForegroundColor Yellow
Get-Content src\services\auditLog.ts | Select-String "db\.\w+Log\." | Select-Object -First 5

# Q4.2: Check for problematic relations
Write-Host "`n=== Q4.2: AUDITLOG RELATIONS ===" -ForegroundColor Cyan
Get-Content prisma\schema.prisma | Select-String -Pattern "model AuditLog" -Context 0,50 | Select-String "@relation"

# Q4.3: Test database connection during generation
Write-Host "`n=== Q4.3: DATABASE CONNECTION TEST ===" -ForegroundColor Cyan
Write-Host "Checking DATABASE_URL configuration:" -ForegroundColor Yellow
$dbUrl = (Get-Content .env -ErrorAction SilentlyContinue | Select-String "DATABASE_URL").ToString()
if ($dbUrl) {
  Write-Host "DATABASE_URL is configured: True" -ForegroundColor Green
} else {
  Write-Host "DATABASE_URL is configured: False" -ForegroundColor Red
}
Write-Host "`nAttempting database introspection:" -ForegroundColor Yellow
npx prisma db pull --force --schema=prisma\schema.prisma
```

```powershell
# Q4.4: Check schema file encoding
Write-Host "`n=== Q4.4: SCHEMA FILE ENCODING ===" -ForegroundColor Cyan
Write-Host "Schema file size:" -ForegroundColor Yellow
(Get-Content prisma\schema.prisma -Raw).Length
Write-Host "Checking for non-ASCII characters:" -ForegroundColor Yellow
$nonAscii = Get-Content prisma\schema.prisma -Raw | Select-String "[^\x00-\x7F]"
if ($nonAscii) {
   Write-Host "Non-ASCII characters found (potential encoding issue)" -ForegroundColor Red
   $nonAscii
} else {
   Write-Host "No encoding issues detected" -ForegroundColor Green
}
```

Phase 2 Decision Point:

STOP HERE and analyze results.

Key things to look for:

- Version mismatches between prisma and @prisma/client
- Old generation timestamps (stale cache)
- Case sensitivity issues (AuditLog vs auditLog vs auditlog)
- Database connection failures preventing proper generation
- Schema encoding issues

📋 PHASE 3: TYPE SYSTEM & RESOLUTION ANALYSIS

Run this phase if: Generated client seems complete but TypeScript can't find the types.

Commands to Run:

powershell

```
# Q3.2: Find conflicting type declarations
Write-Host "`n=== Q3.2: CUSTOM TYPE DECLARATIONS ===" -ForegroundColor Cyan
Get-ChildItem -Recurse -Filter "*.d.ts" | Where-Object { $_.FullName -notlike "*node_modules*" } | ForEach-Object {
   $content = Get-Content $_.FullName -Raw
   if ($content -match "PrismaClient|@prisma/client") {
      Write-Host "`nFound in: $($_.FullName)" -ForegroundColor Yellow
      Get-Content $_.FullName | Select-String -Pattern "PrismaClient|@prisma/client" -Context 2,2
   }
}

# Q3.3: Trace TypeScript module resolution
Write-Host "`n=== Q3.3: TYPESCRIPT MODULE RESOLUTION ===" -ForegroundColor Cyan
Write-Host "Tracing @prisma/client resolution (first 20 lines):" -ForegroundColor Yellow
npx tsc --traceResolution 2>&1 | Select-String "@prisma/client" | Select-Object -First 20

# Q5.1: Check TypeScript configuration
Write-Host "`n=== Q5.1: TYPESCRIPT CONFIGURATION ===" -ForegroundColor Cyan
Get-Content tsconfig.json | Select-String "moduleResolution|baseUrl|paths|types" -Context 1,1

# Q5.2: Inspect build output (if exists)
Write-Host "`n=== Q5.2: BUILD OUTPUT INSPECTION ===" -ForegroundColor Cyan
if (Test-Path dist\services\database.js) {
   Write-Host "Checking compiled database.js for auditLog references:" -ForegroundColor Yellow
   Get-Content dist\services\database.js | Select-String "auditLog" | Select-Object -First 5
} else {
   Write-Host "Build output not found (expected - build is failing)" -ForegroundColor Yellow
}
Get-ChildItem dist\services -ErrorAction SilentlyContinue | Select-Object Name, Length

# Q5.3: Check cache and lock file consistency
Write-Host "`n=== Q5.3: CACHE & LOCK FILE STATUS ===" -ForegroundColor Cyan
Write-Host "Cache directory exists:" -ForegroundColor Yellow
Test-Path node_modules\.cache
Write-Host "`nPackage lock file hash:" -ForegroundColor Yellow
Get-FileHash package-lock.json | Select-Object Hash
Write-Host "`nChecking for package changes:" -ForegroundColor Yellow
npm install --dry-run 2>&1 | Select-String "up to date|added|removed|changed"

# Q5.4: Verify Prisma binary engines
Write-Host "`n=== Q5.4: PRISMA ENGINE BINARIES ===" -ForegroundColor Cyan
Write-Host "Query engine binaries in .prisma/client:" -ForegroundColor Yellow
```

```
Get-ChildItem node_modules\.prisma\client -Recurse -Filter "*.node" -ErrorAction SilentlyContinue | Select-Object Name
Write-Host "`nEngine files in @prisma/engines:" -ForegroundColor Yellow
Get-ChildItem node_modules\@prisma\engines -ErrorAction SilentlyContinue | Select-Object Name

# Q6.1: Environment variable verification
Write-Host "`n=== Q6.1: ENVIRONMENT VARIABLES ===" -ForegroundColor Cyan
Write-Host "DATABASE_URL in .env:" -ForegroundColor Yellow
Get-Content .env -ErrorAction SilentlyContinue | Select-String "DATABASE_URL"
Write-Host "`nDATABASE_URL in .env.example:" -ForegroundColor Yellow
Get-Content .env.example -ErrorAction SilentlyContinue | Select-String "DATABASE_URL"

# Q6.2: Package.json Prisma configuration (deprecated)
Write-Host "`n=== Q6.2: PACKAGE.JSON PRISMA CONFIG ===" -ForegroundColor Cyan
Get-Content package.json | Select-String -Pattern '"prisma"' -Context 0,10

# Q6.3: Runtime environment versions
Write-Host "`n=== Q6.3: RUNTIME ENVIRONMENT VERSIONS ===" -ForegroundColor Cyan
Write-Host "Node version:" -ForegroundColor Yellow
node --version
Write-Host "npm version:" -ForegroundColor Yellow
npm --version
Write-Host "Prisma version:" -ForegroundColor Yellow
npx prisma --version

# Q6.4: Working directory verification
Write-Host "`n=== Q6.4: WORKING DIRECTORY VERIFICATION ===" -ForegroundColor Cyan
Write-Host "Current directory:" -ForegroundColor Yellow
Get-Location
Write-Host "`nKey files present:" -ForegroundColor Yellow
Write-Host "  prisma\schema.prisma: $(Test-Path prisma\schema.prisma)"
Write-Host "  src\services\auditLog.ts: $(Test-Path src\services\auditLog.ts)"
Write-Host "  package.json: $(Test-Path package.json)"
```

Phase 3 Decision Point:

STOP HERE and analyze results.

Key things to look for:

- Custom type declarations shadowing @prisma/client
- TypeScript resolving to wrong location
- Conflicting path mappings in tsconfig.json

- Version incompatibilities (Node < 16, old Prisma versions)
- Deprecated package.json config interfering with generation

📋 PHASE 4: NUCLEAR OPTION - COMPLETE RESET

Run this phase if: All previous phases show valid configuration but issue persists.

⚠️ WARNING: This will delete node_modules and reinstall everything

Commands to Run:

powershell

```
# Q7.1: Complete clean slate test
Write-Host "`n=== Q7.1: NUCLEAR CLEAN & REGENERATION ===" -ForegroundColor Cyan

# Backup current schema
Write-Host "Backing up current schema..." -ForegroundColor Yellow
npx prisma db pull --schema=prisma\schema.prisma --print > schema_backup_$(Get-Date -Format 'yyyyMMdd_HHmms

# Document current package versions
Write-Host "`nDocumenting current state..." -ForegroundColor Yellow
npm list > package_state_before_clean.txt

# Nuclear clean
Write-Host "`nCleaning node_modules..." -ForegroundColor Yellow
Remove-Item -Recurse -Force node_modules -ErrorAction SilentlyContinue
Write-Host "Cleaning package-lock.json..." -ForegroundColor Yellow
Remove-Item package-lock.json -ErrorAction SilentlyContinue
Write-Host "Cleaning dist..." -ForegroundColor Yellow
Remove-Item -Recurse -Force dist -ErrorAction SilentlyContinue

# Fresh install
Write-Host "`nFresh install starting..." -ForegroundColor Yellow
npm install

# Generate Prisma Client
Write-Host "`nGenerating Prisma Client..." -ForegroundColor Yellow
npx prisma generate

# Verify auditLog is now present
Write-Host "`nVerification - Looking for auditLog in generated client..." -ForegroundColor Yellow
$auditLogFound = Get-Content node_modules\.prisma\client\index.d.ts | Select-String "get auditLog"
if ($auditLogFound) {
  Write-Host "SUCCESS: auditLog found in generated client!" -ForegroundColor Green
  $auditLogFound
} else {
  Write-Host "FAILURE: auditLog still missing from generated client" -ForegroundColor Red
}

# Q7.2: Minimal reproduction test
Write-Host "`n=== Q7.2: MINIMAL REPRODUCTION TEST ===" -ForegroundColor Cyan

# Create test file
```

```
$testContent = @"
import { PrismaClient } from '@prisma/client';

const prisma = new PrismaClient();

console.log('Testing Prisma Client model availability...');
console.log('AuditLog model available:', 'auditLog' in prisma);
console.log('All available models:', Object.keys(prisma).filter(k => !k.startsWith('_') && !k.startsWith('$')).sort());

// Try to access auditLog directly
try {
  console.log('auditLog accessor type:', typeof prisma.auditLog);
} catch (error) {
  console.error('Error accessing auditLog:', error.message);
}
"@

Write-Host "Creating test file..." -ForegroundColor Yellow
$testContent | Out-File -FilePath test-prisma-client.ts -Encoding UTF8

Write-Host "Running minimal test..." -ForegroundColor Yellow
npx ts-node test-prisma-client.ts

# Q7.3: Direct schema comparison
Write-Host "`n=== Q7.3: SCHEMA COMPARISON ===" -ForegroundColor Cyan

Write-Host "User model (WORKING - for comparison):" -ForegroundColor Yellow
Get-Content prisma\schema.prisma | Select-String -Pattern "model User" -Context 0,25

Write-Host "`nAuditLog model (NOT WORKING):" -ForegroundColor Yellow
Get-Content prisma\schema.prisma | Select-String -Pattern "model AuditLog" -Context 0,25

Write-Host "`nAll model definitions in schema:" -ForegroundColor Yellow
Get-Content prisma\schema.prisma | Select-String "^model\s+\w+" | ForEach-Object { $_.Line.Trim() }
```

Phase 4 Decision Point:

This is the final diagnostic phase.

Possible outcomes:

1. **auditLog appears after clean reinstall** → Was a cache/state issue, now resolved

2. **auditLog still missing** → Schema has a structural problem that prevents generation
3. **Test file shows auditLog exists at runtime but TypeScript fails** → Type declaration issue

---

📊 DIAGNOSTIC REPORT TEMPLATE

After running all phases, fill out this report:

markdown

## Prisma Diagnostic Results

Date: [Current Date]
Issue: auditLog model not accessible in Prisma Client

### PHASE 1 RESULTS: Immediate Verification

#### Q1.1 - Schema Model Definition
Status: [ ] PASS [ ] FAIL
Details: [Does auditLog model exist? Show first 10 lines of model]

#### Q1.2 - Datasource Configuration
Status: [ ] PASS [ ] FAIL
Details: [Provider type, connection details]

#### Q1.3 - Generator Configuration
Status: [ ] PASS [ ] FAIL
Details: [Generator provider, output path if custom]

#### Q1.4 - Schema Validation
Status: [ ] PASS [ ] FAIL
Details: [Validation result, any warnings?]

#### Q2.2 - Generated Model Accessors
Status: [ ] PASS [ ] FAIL
Details: [List all model accessors found, is auditLog present?]

#### Q2.3 - AuditLog Exports
Status: [ ] PASS [ ] FAIL
Details: [Any AuditLog-related exports found?]

#### Q3.1 - Database Service Import
Status: [ ] PASS [ ] FAIL
Details: [Import statement used]

#### Q3.4 - Node Modules Integrity
Status: [ ] PASS [ ] FAIL
Details: [Do both type definition files exist? File count]

Phase 1 Conclusion: [Continue to Phase 2/3/4 or issue identified?]

### PHASE 2 RESULTS: Schema & Generation Deep Dive
[Only if Phase 1 indicated this was needed]

#### Q2.1 - Generation Output Location
Status: [ ] PASS [ ] FAIL
Details: [Custom output path or default? Package.json config?]

#### Q2.4 - Version Consistency
Status: [ ] PASS [ ] FAIL
Details: [All Prisma packages same version?]

#### Q2.5 - Generation Timestamp
Status: [ ] PASS [ ] FAIL
Details: [Recent generation or stale?]

#### Q4.1 - Case Sensitivity
Status: [ ] PASS [ ] FAIL
Details: [Exact model name in schema vs code usage]

#### Q4.2 - Relations Check
Status: [ ] PASS [ ] FAIL
Details: [Any problematic relations?]

#### Q4.3 - Database Connection
Status: [ ] PASS [ ] FAIL
Details: [Can connect to database? Introspection works?]

#### Q4.4 - File Encoding
Status: [ ] PASS [ ] FAIL
Details: [Schema file size, encoding issues?]

Phase 2 Conclusion: [Issue identified or continue to Phase 3?]

### PHASE 3 RESULTS: Type System Analysis
[Only if needed]

#### Q3.2 - Custom Type Declarations
Status: [ ] PASS [ ] FAIL
Details: [Any custom .d.ts files affecting PrismaClient?]

#### Q3.3 - Module Resolution
Status: [ ] PASS [ ] FAIL
Details: [Where is TypeScript resolving @prisma/client to?]

#### Q5.1 - TypeScript Config
Status: [ ] PASS [ ] FAIL
Details: [Any path mappings or custom resolution?]

#### Q6.2 - Package.json Config
Status: [ ] PASS [ ] FAIL
Details: [Deprecated prisma config present?]

#### Q6.3 - Runtime Versions
Status: [ ] PASS [ ] FAIL
Details: [Node version, npm version, Prisma version]

Phase 3 Conclusion: [Issue identified or nuclear option needed?]

### PHASE 4 RESULTS: Nuclear Option
[Only if all else failed]

#### Q7.1 - Clean Reinstall Result
Status: [ ] PASS [ ] FAIL
Details: [Did auditLog appear after clean reinstall?]

#### Q7.2 - Minimal Test Result
Status: [ ] PASS [ ] FAIL
Details: [Can runtime access auditLog? What models are available?]

#### Q7.3 - Schema Comparison
Status: [ ] PASS [ ] FAIL
Details: [How does AuditLog differ from working User model?]

Phase 4 Conclusion: [Final diagnosis]

### ROOT CAUSE IDENTIFIED

Issue: [Specific problem found]

Evidence: [Commands and output that confirmed the issue]

Hypothesis: [Why this is causing the problem]

Recommended Fix: [Specific steps to resolve]

### POST-FIX VALIDATION CHECKLIST

Once fix is applied, verify:

- [ ] Schema validates: `npx prisma validate`
- [ ] Client generates: `npx prisma generate`
- [ ] auditLog in generated client: `Get-Content node_modules\.prisma\client\index.d.ts | Select-String "get auditLog"`
- [ ] TypeScript compiles: `npm run build`
- [ ] Application runs: `npm run dev`
- [ ] Feature works: [Test auditLog functionality]

### LESSONS LEARNED

What went wrong: [Brief summary]

Why it happened: [Root cause explanation]

How to prevent: [Future safeguards]

Documentation needs: [What should be documented]

🎯 MOST LIKELY ROOT CAUSES (Ranked by Probability)

1. Schema Syntax Issue (HIGH PROBABILITY - 40%)

Symptoms: Schema validates but model doesn't generate **Root Cause**: AuditLog model has some structural difference from other models that Prisma's generator skips **How Phase 7.3 will catch it**: Direct comparison with working User model will show the difference **Typical Fixes**:

- Remove problematic field types
- Fix relation syntax
- Correct enum references
- Remove unsupported features

2. Deprecated package.json Config (MEDIUM-HIGH PROBABILITY - 25%)

Symptoms: Warning about deprecated config, selective model generation **Root Cause**: Old Prisma 2.x style config in package.json filtering which models generate **How Q6.2 will catch it**: Shows if deprecated config exists **Typical Fix**: Remove `"prisma": {...}` block from package.json

3. Case Sensitivity Mismatch (MEDIUM PROBABILITY - 15%)

Symptoms: Model exists but accessor name is wrong **Root Cause**: Schema defines "AuditLog" but code uses "auditLog" (or vice versa) **How Q4.1 will catch it**: Shows exact model name vs code usage **Typical Fix**: Match case exactly (Prisma uses camelCase for accessors)

4. Stale Cache/Generation (MEDIUM PROBABILITY - 10%)

Symptoms: Multiple reinstalls don't fix it **Root Cause**: Prisma's internal cache or build artifacts interfering **How Phase 4 will catch it**: Clean slate proves if it's a cache issue **Typical Fix**: Nuclear clean and regenerate

5. TypeScript Type Declaration Shadowing (LOW-MEDIUM PROBABILITY - 5%)

Symptoms: Runtime works but TypeScript fails **Root Cause**: Custom .d.ts file overriding generated types **How Q3.2 will catch it**: Finds conflicting type declarations **Typical Fix**: Remove or rename conflicting .d.ts files

6. Database Connection Issue During Generation (LOW PROBABILITY - 3%)

Symptoms: Some models generate, others don't **Root Cause**: Prisma can't connect to verify schema during generation **How Q4.3 will catch it**: Tests database connectivity **Typical Fix**: Verify DATABASE_URL and database accessibility

7. Encoding/Unicode Issue in Schema File (VERY LOW PROBABILITY - 2%)

Symptoms: Random generation failures **Root Cause**: Non-ASCII characters breaking parser **How Q4.4 will catch it**: Detects encoding problems **Typical Fix**: Resave schema file with UTF-8 encoding

---

⚡ QUICK REFERENCE - Command Summary

Copy-paste this entire section to run all critical checks at once:

powershell

```
Write-Host "`n==================================================" -ForegroundColor Magenta
Write-Host "PRISMA DIAGNOSTIC - QUICK REFERENCE" -ForegroundColor Magenta
Write-Host "==================================================" -ForegroundColor Magenta

# 1. Does auditLog exist in schema?
Write-Host "`n[1/8] Checking schema for auditLog model..." -ForegroundColor Cyan
Get-Content prisma\schema.prisma | Select-String "model AuditLog" -Context 0,5

# 2. Is auditLog in generated client?
Write-Host "`n[2/8] Checking generated client for auditLog accessor..." -ForegroundColor Cyan
$result = Get-Content node_modules\.prisma\client\index.d.ts -ErrorAction SilentlyContinue | Select-String "get auditLog'
if ($result) { Write-Host "FOUND" -ForegroundColor Green } else { Write-Host "NOT FOUND" -ForegroundColor Red }

# 3. Version consistency
Write-Host "`n[3/8] Checking Prisma version consistency..." -ForegroundColor Cyan
npm list prisma @prisma/client | Select-String "prisma"

# 4. Schema validation
Write-Host "`n[4/8] Validating Prisma schema..." -ForegroundColor Cyan
npx prisma validate

# 5. Import statement
Write-Host "`n[5/8] Checking database service import..." -ForegroundColor Cyan
Get-Content src\services\database.ts | Select-String "import.*PrismaClient"

# 6. Case sensitivity
Write-Host "`n[6/8] Checking model name case sensitivity..." -ForegroundColor Cyan
Get-Content prisma\schema.prisma | Select-String "model\s+\w*Log" | ForEach-Object { $_.Line.Trim() }

# 7. TypeScript compilation
Write-Host "`n[7/8] Attempting TypeScript compilation..." -ForegroundColor Cyan
npm run build 2>&1 | Select-String "error TS" | Select-Object -First 5

# 8. Package.json config (deprecated)
Write-Host "`n[8/8] Checking for deprecated package.json config..." -ForegroundColor Cyan
Get-Content package.json | Select-String '"prisma"' -Context 0,3

Write-Host "`n==================================================" -ForegroundColor Magenta
```

```
Write-Host "QUICK CHECK COMPLETE" -ForegroundColor Magenta
Write-Host "================================================" -ForegroundColor Magenta
```

📖 INSTRUCTIONS FOR CLAUDE CODE

Please execute this diagnostic in the following order:

1. **Start with Quick Reference** (above) - Run all 8 checks first to get overview
2. **Report Quick Reference results** - What did you find?
3. **Based on Quick Reference**, determine which phase to run:
   - If auditLog missing from schema → Schema corruption, investigate manually
   - If auditLog in schema but NOT in generated client → Run Phase 2
   - If auditLog in generated client but TypeScript fails → Run Phase 3
   - If everything seems right but still broken → Run Phase 4
4. **Execute the appropriate phase** systematically, reporting each result
5. **Fill out the Diagnostic Report Template** with findings
6. **Identify the root cause** from the ranked list
7. **Propose specific fix** with exact commands to run

Do not skip phases or jump ahead. Each phase builds on the previous one.

Report results in this format:

```
## DIAGNOSTIC RESULTS

### Quick Reference Results
[Show all 8 check results]

### Decision: Running Phase [X] because [reason]

### Phase [X] Results
[Show each question result with PASS/FAIL/WARNING]

### Root Cause Analysis
Issue: [Specific problem]
Evidence: [Commands that proved it]
Fix: [Exact solution]
```

✅ SUCCESS CRITERIA

You'll know the issue is resolved when all of these pass:

powershell

```
# 1. Schema validates
npx prisma validate
# Expected: "The schema is valid 🚀"

# 2. Client generates successfully
npx prisma generate
# Expected: "Generated Prisma Client to .\node_modules\@prisma\client"

# 3. auditLog is present in generated client
Get-Content node_modules\.prisma\client\index.d.ts | Select-String "get auditLog"
# Expected: "get auditLog(): Prisma.AuditLogDelegate"

# 4. TypeScript compiles without errors
npm run build
# Expected: No "error TS" output

# 5. Application runs
npm run dev
# Expected: "✓ Database connected successfully"

# 6. Can actually use auditLog in code
# Test by running: npm run dev and checking console output
```

END OF DIAGNOSTIC BATCH

Ready to execute. Please run systematically and report findings.