Sentry Error Investigation Report

Error Summary

Field	Value
Error	`Engine is not yet connected.`
Endpoint	`GET /api/favorites/all`
Environment	Production
Platform	Node.js v20.11.0
First Seen	January 30, 2026, 10:47 PM UTC
Priority	High
Status	✅ Fixed

Root Cause Analysis

The Problem

The error `"Engine is not yet connected"` is a **Prisma database connection error** that occurs when the Prisma Client attempts to execute a query before establishing a connection to the database engine.

Technical Details

The issue in `/src/app/api/favorites/all/route.ts` was caused by:

1. **Creating a new PrismaClient instance on every request:**
   ```
   typescript
   // ❌ BAD - Creates new connection pool per request
   const prisma = new PrismaClient();
   ```

2. **Calling `$disconnect()` in the finally block:**
   ```
   typescript
   // ❌ BAD - Disconnects while other requests may be pending
   } finally {
     await prisma.$disconnect();
   }
   ```

Why This Fails

In a serverless/edge environment like Render:
- Creating a new `PrismaClient` for each request is expensive and slow

- The connection pool may not be ready before queries execute
- Calling `$disconnect()` can terminate connections while concurrent requests are still using them
- Race conditions occur when multiple requests hit the endpoint simultaneously

---

The Fix

Changes Made to `/src/app/api/favorites/all/route.ts`

Before:

```
import { PrismaClient, UserRole } from '@prisma/client';
const prisma = new PrismaClient();

// ... handler code ...

} finally {
  await prisma.$disconnect();
}
```

After:

```
import { UserRole } from '@prisma/client';
import { prisma } from '@/lib/prisma';

// ... handler code ...
// Removed $disconnect() - singleton manages connection lifecycle
}
```

Why This Works

The project already has a proper Prisma singleton at `/src/lib/prisma.ts`:
- Uses `globalThis` to maintain a single instance across hot reloads
- Proper connection pooling for concurrent requests
- Automatic connection management
- No manual disconnect needed

Additional Findings

⚠️ 59 Files With Same Anti-Pattern

The following API routes have the same problematic pattern and should be fixed:

```
src/app/api/homes/search/route.ts
src/app/api/family/profile/route.ts
src/app/api/search/route.ts
src/app/api/operator/homes/[id]/generate-profile/route.ts
src/app/api/operator/homes/[id]/analytics/route.ts
src/app/api/operator/homes/[id]/photos/[photoId]/route.ts
src/app/api/operator/homes/[id]/photos/reorder/route.ts
src/app/api/operator/homes/[id]/photos/route.ts
... and 51 more files
```

AWS SDK v2 Deprecation Warning

From the breadcrumbs, there's also a warning:

```
(node:486) NOTE: The AWS SDK for JavaScript (v2) has reached end-of-support.
Please migrate your code to use AWS SDK for JavaScript (v3).
```

This should be addressed separately to ensure future compatibility.

Deployment Instructions

1. **Commit the fix:**
   ```
   bash
     git add src/app/api/favorites/all/route.ts
     git commit -m "fix: use singleton Prisma client to prevent 'Engine not connected' error"
     git push origin main
   ```
2. **Verify deployment** on Render dashboard
3. **Monitor Sentry** to confirm the error stops occurring

Prevention Recommendations

1. **Add ESLint rule** to prevent direct PrismaClient instantiation:
   ```
   json
     {
       "rules": {
         "no-restricted-imports": ["error", {
           "patterns": [{
             "group": ["@prisma/client"],
             "importNames": ["PrismaClient"],
             "message": "Import prisma from '@/lib/prisma' instead of creating new
   PrismaClient instances."
           }]
         }]
       }
     }
   ```
2. **Fix remaining 59 files** using the same pattern
3. **Update AWS SDK** from v2 to v3

Report Generated

- **Date:** January 30, 2026
- **Issue ID:** 7231042457
- **Event ID:** 252aa4dfd26644ce80d90b6a1859df75