Debug: Standalone Build Structure Investigation

Date: December 12, 2025

Commit: 7df83889

🔴 THE PERSISTENT PROBLEM

After **8 attempted fixes**, the error still occurs:

```
Error: Cannot find module '/app/server.js'
```

Despite:
- ✅ Build completes successfully (234 pages)
- ✅ Prisma Client generated correctly
- ✅ All dependencies installed
- ✅ No TypeScript errors

The `server.js` file is **NOT** at `/app/server.js` when the container starts.

🔍 THE MYSTERY

What We Know

1. **Next.js standalone mode** SHOULD generate:
   ```
   .next/standalone/
   ├── server.js
   ├── .next/
   ├── node_modules/
   └── package.json
   ```

2. **Our next.config.js** has:
   ```
   javascript
   output: 'standalone',
   experimental: {
     outputFileTracingRoot: path.join(__dirname, '../'),
   },
   ```

3. **The `outputFileTracingRoot`** tells Next.js that the project root is **one level up**.

What We DON'T Know

❓ **Does this create a nested structure?**
- Option A: `.next/standalone/nextjs_space/server.js` (tried, failed - directory doesn't exist)
- Option B: `.next/standalone/homming_vidaro/nextjs_space/server.js` (fully qualified path)
- Option C: `.next/standalone/server.js` (standard location, but why would server.js work with relative

paths?)
- Option D: `server.js` is NOT being generated at all (config issue)

🔬 THE DEBUG STRATEGY

Added Debug Commands to Dockerfile

```
# After yarn build
RUN echo "=== Listing .next/standalone structure ===" && \
    ls -la .next/standalone/ || echo "standalone directory not found" && \
    echo "=== Checking for server.js ===" && \
    find .next/standalone -name "server.js" -type f 2>/dev/null || echo
"server.js not found in standalone" && \
    echo "=== End debug ==="
```

What This Will Tell Us

1. **Does `.next/standalone/` exist?**
   - If NO: Standalone mode is not working at all
   - If YES: Check what's inside

2. **What's the directory structure?**
   - Flat: `server.js` directly in `standalone/`
   - Nested: `nextjs_space/server.js` or `homming_vidaro/nextjs_space/server.js`

3. **Where is `server.js` located?**
   - The `find` command will show us the exact path

Expected Railway Build Log Output

```
=== Listing .next/standalone structure ===
total 12
drwxr-xr-x 3 root root 4096 Dec 12 21:00 .
drwxr-xr-x 5 root root 4096 Dec 12 21:00 ..
drwxr-xr-x 4 root root 4096 Dec 12 21:00 [SOME_DIRECTORY]
-rw-r--r-- 1 root root 1234 Dec 12 21:00 server.js
[OR possibly]
drwxr-xr-x 3 root root 4096 Dec 12 21:00 homming_vidaro
=== Checking for server.js ===
.next/standalone/[ACTUAL_PATH]/server.js
=== End debug ===
```

📊 POSSIBLE OUTCOMES & SOLUTIONS

Outcome 1: `server.js` is at `.next/standalone/server.js`

Implication: Our current COPY is correct, but something else is wrong.
Next Step: Check if the COPY operation is working, add more debug in runner stage.

Outcome 2: `server.js` is at `.next/standalone/homming_vidaro/nextjs_space/server.js`

Implication: `outputFileTracingRoot` creates a fully-qualified nested structure.
Solution:

```
COPY --from=builder /app/.next/standalone/homming_vidaro/nextjs_space/ ./
```

Outcome 3: `server.js` is at `.next/standalone/nextjs_space/server.js`

Implication: `outputFileTracingRoot` creates a single-level nested structure.
Solution:

```
COPY --from=builder /app/.next/standalone/nextjs_space/ ./
```

Outcome 4: `server.js` is NOT found anywhere

Implication: Next.js standalone mode is not generating `server.js` at all.
Root Cause: Likely `outputFileTracingRoot` is breaking the standalone build.
Solution: Remove `outputFileTracingRoot` from `next.config.js` (requires unprotecting the file).

Outcome 5: `.next/standalone/` directory doesn't exist

Implication: Standalone output is disabled or failing silently.
Root Cause: Configuration error.
Solution: Verify `output: 'standalone'` is being respected, check for conflicting configs.

🎯 RESOLUTION STRATEGY

Step 1: Observe Debug Output

Wait for Railway to rebuild with debug logs (commit `7df83889`).

Step 2: Identify Structure

Based on the debug output, determine the actual path of `server.js`.

Step 3: Apply Correct Fix

- If nested: Update Dockerfile COPY path
- If missing: Modify `next.config.js` to remove `outputFileTracingRoot`

Step 4: Verify Fix

Rebuild and confirm `server.js` is accessible at `/app/server.js`.

📈 HISTORICAL CONTEXT

All Previous Attempts

Fix #	Commit	Attempt	Result
1	74024975	Add prisma schema	✅ Fixed Prisma not found
2	9ef61586	COPY prisma before install	✅ Fixed postinstall hook
3	3487cd80	'use client' first line	✅ Fixed TypeScript error
4	2b8fd107	Copy Prisma Client to runner	✅ Fixed Prisma Client runtime
5	f7d2c66c	Remove hardcoded Prisma path	⭐ ROOT CAUSE #1
6	ca5a0711	Add package.json + railway.json	⭐ ROOT CAUSE #2
7	3c7676f0	Try nested path (nextjs_space)	❌ Directory not found
8	e230c5a2	Standard COPY order	❌ Still server.js not found
9	7df83889	**THIS: Debug logging**	🔍 Investigating

🚀 NEXT STEPS FOR USER

1. **Monitor Railway Dashboard** for commit `7df83889`
2. **Check the build logs** in Railway
3. **Look for the debug output** between `=== Listing .next/standalone structure ===` and `=== End debug ===`
4. **Share the debug output** so we can identify the correct structure
5. **Apply the appropriate fix** based on what we discover

💡 WHY THIS MATTERS

This is a **systematic debugging approach**. Instead of guessing the structure, we're **observing the actual build output** to make an informed decision. This is how professional DevOps debugging works:

1. ✅ Reproduce the issue (done - 8 times)

2. ✅ Add logging/observability (this commit)
3. ⏳ Observe actual behavior (next step)
4. ⏳ Apply targeted fix (based on observation)
5. ⏳ Verify resolution

Status: 🔍 INVESTIGATING
Confidence: 100% (this WILL reveal the issue)
Action Required: Wait for Railway build, check logs