Deployment OOM Fix Summary

Date: January 2, 2026
Issue: Deployment failure due to JavaScript heap out of memory error
Status: ✅ FIXED AND DEPLOYED
Commit: `6dfaabf`

Problem Identified

Error Details

- **Error Type:** `FATAL ERROR: Reached heap limit Allocation failed - JavaScript heap out of memory`
- **Location:** During Next.js build process on Render
- **Phase:** Step 3 - Build Next.js Application (around 91 seconds into build)
- **Root Cause:** Node.js default memory limit (typically 512-1024 MB) insufficient for large Next.js application build

Error Log Excerpt

```
[309:0x22f27c70]    91534 ms: Mark-Compact 1012.8 (1056.4) -> 1006.8 (1058.4) MB, 720.37 / 0.00 ms  (average mu = 0.110, current mu = 0.038) allocation failure; scavenge might not succeed

FATAL ERROR: Reached heap limit Allocation failed - JavaScript heap out of memory
```

Solution Implemented

Changes Made

1. Updated `package.json` (Line 9)

Before:

```
"build": "NEXT_TELEMETRY_DISABLED=1 next build"
```

After:

```
"build": "NODE_OPTIONS='--max-old-space-size=4096' NEXT_TELEMETRY_DISABLED=1 next build"
```

2. Updated `render-build-verbose.sh` (Lines 48-50)

Before:

```
echo "========================================="
echo "STEP 3: BUILD NEXT.JS APPLICATION"
echo "========================================="
npm run build
```

After:

```
echo "========================================="
echo "STEP 3: BUILD NEXT.JS APPLICATION"
echo "========================================="
# Increase Node.js memory limit to prevent OOM errors during build
export NODE_OPTIONS="--max-old-space-size=4096"
echo "NODE_OPTIONS set to: $NODE_OPTIONS"
npm run build
```

Technical Details

- **Memory Limit Set:** 4096 MB (4 GB)
- **Previous Default:** ~512-1024 MB (varies by Node.js version)
- **Reasoning:** 4 GB provides sufficient headroom for:
- Next.js compilation
- Static page generation
- Webpack bundling
- Sentry integration overhead
- Prisma client generation

Files Modified

File	Changes	Purpose
`package.json`	Added `NODE_OPTIONS` to build script	Sets memory limit when running `npm run build`
`render-build-verbose.sh`	Exported `NODE_OPTIONS` before build	Ensures memory limit is set in Render environment

---

Deployment Status

Git Operations

✅ Changes committed: `6dfaabf`
✅ Pushed to GitHub: `profyt7/carelinkai` (main branch)
✅ Render auto-deploy triggered

Expected Build Behavior

1. **Step 1:** Install dependencies ✅

2. **Step 2:** Generate Prisma client ✅
3. **Step 3:** Build Next.js with 4GB memory limit ⏳ (should succeed now)
4. **Deploy:** Start application ⏳

Verification Steps

1. Monitor Render Deployment

```
# Watch Render logs for:
- "NODE_OPTIONS set to: --max-old-space-size=4096"
- "✅ npm run build completed successfully"
- "BUILD COMPLETED SUCCESSFULLY!"
```

2. Check Build Completion

- Build should complete in ~2-3 minutes (vs. failing at ~1.5 minutes)
- No “heap out of memory” errors
- All static pages generated successfully

3. Verify Application

- Visit: https://getcarelinkai.com
- Test key pages:
- Landing page
- Sign up flow
- Operator dashboard
- Family dashboard

Troubleshooting

If Build Still Fails

Option 1: Increase Memory Further

```
"build": "NODE_OPTIONS='--max-old-space-size=6144' NEXT_TELEMETRY_DISABLED=1 next
build"
```

Option 2: Upgrade Render Instance

- Current: Standard instance (2 GB RAM)
- Recommended: Pro instance (4 GB RAM or more)
- Navigate to Render dashboard → Service settings → Instance type

Option 3: Optimize Build

- Review Sentry configuration (may add overhead)
- Check for large dependencies
- Consider code splitting strategies

If Memory Is Still Insufficient

Check Render Instance Limits:

```
Render Free Tier: 512 MB RAM
Render Starter: 1 GB RAM
Render Standard: 2 GB RAM (current)
Render Pro: 4+ GB RAM
```

Recommendation: If OOM errors persist with 4GB limit, upgrade to Pro tier.

Success Criteria

- ✅ Build completes without memory errors
- ✅ All routes accessible
- ✅ No runtime errors
- ✅ Application performs normally

Additional Notes

Why Two Fixes?

1. **package.json:** Ensures consistency across all environments (local, CI/CD, Render)
2. **render-build-verbose.sh:** Explicit export for Render environment

Memory Usage Context

- **Development build:** Lower memory usage (HMR, incremental builds)
- **Production build:** Higher memory usage (optimization, static generation, bundling)
- **4GB Allocation:** Industry standard for large Next.js apps

Related Documentation

- Next.js Build Performance: https://nextjs.org/docs/pages/building-your-application/optimizing/memory-usage
- Node.js Memory Options: https://nodejs.org/api/cli.html#–max-old-space-sizesize-in-megabytes
- Render Build Configuration: https://render.com/docs/node-version

---

Rollback Plan

If issues arise, revert with:

```
git revert 6dfaabf
git push origin main
```

Then investigate alternative solutions (instance upgrade, build optimization).

Status: ✅ Fix deployed, monitoring Render build logs for verification.