OpenAI Build-Time Error Fix Guide

Issue: Build fails during “Collecting page data” phase due to OpenAI initialization
Error: `Missing credentials. Please pass an apiKey, or set the OPENAI_API_KEY environment variable.`
Date: December 19, 2025

🔍 Problem Analysis

Root Cause

Next.js tries to pre-render API routes during build time (in the “Collecting page data” phase). When it encounters the `generate-response` route:

1. It imports the module
2. The `inquiry-response-generator` module is imported
3. Even though we have lazy initialization, Next.js still needs to load the module
4. During module loading, OpenAI SDK constructor gets called
5. Constructor checks for `OPENAI_API_KEY`
6. Since key is not set at build time, build fails

Why This Happens

- Next.js App Router has a “Collecting page data” phase
- This phase attempts to execute dynamic API routes
- Our API route imports `inquiryResponseGenerator`
- Module imports happen before lazy initialization

🛠️ SOLUTION OPTIONS

OPTION A: Add Route Segment Config (Recommended)

Best for: Quick fix, no environment variables needed at build time

What it does: Tells Next.js this route is fully dynamic and should not be pre-rendered

Implementation:

Edit `src/app/api/inquiries/[id]/generate-response/route.ts`:

```
import { NextRequest, NextResponse } from 'next/server';
import { getServerSession } from 'next-auth';
import { authOptions } from '@/lib/auth';
import { inquiryResponseGenerator } from '@/lib/ai/inquiry-response-generator';
import { inquiryEmailService } from '@/lib/email/inquiry-email-service';
import { prisma } from '@/lib/prisma';

// ADD THIS LINE - Tells Next.js this route is fully dynamic
export const dynamic = 'force-dynamic';
export const runtime = 'nodejs';

export async function POST(
  request: NextRequest,
  { params }: { params: { id: string } }
) {
  // ... rest of the code
}
```

Benefits:
- ✅ No environment variables needed at build time
- ✅ Clean solution
- ✅ Follows Next.js best practices
- ✅ No code changes to AI module

Trade-offs:
- Route cannot be pre-rendered (but that’s fine for POST endpoints)

OPTION B: Add OpenAI API Key to Render Environment

Best for: If you have a valid OpenAI API key ready

Steps:

1. **Get your OpenAI API key:**
   - Go to https://platform.openai.com/api-keys
   - Create a new API key
   - Copy the key (starts with `sk-proj-...`)

2. **Add to Render:**
   - Go to Render Dashboard
   - Select your service
   - Click “Environment” in left sidebar
   - Click “Add Environment Variable”
   - Key: `OPENAI_API_KEY`
   - Value: `sk-proj-...` (your actual key)
   - Click “Save Changes”

3. **Render will auto-deploy:**
   - Wait for build to complete
   - Build should succeed now

Benefits:
- ✅ AI responses will work immediately
- ✅ No code changes needed

Trade-offs:
- Requires valid OpenAI API key (costs money)
- Key is exposed during build (but that's fine in Render)

OPTION C: Conditional Module Import

Best for: Advanced use case, complex setup

Implementation:

Create a new file `src/lib/ai/inquiry-response-generator-wrapper.ts`:

```
// Lazy import to avoid build-time issues
export async function getInquiryResponseGenerator() {
  if (typeof window === 'undefined') {
    // Server-side only
    const { inquiryResponseGenerator } = await import('./inquiry-response-generator');
    return inquiryResponseGenerator;
  }
  throw new Error('This function should only be called on the server');
}
```

Then update the route:

```
export async function POST(request: NextRequest, { params }: { params: { id:
string } }) {
  try {
    // ... auth checks ...

    // Lazy import the generator
    const generator = await getInquiryResponseGenerator();
    const content = await generator.generateResponseForInquiry(params.id, {...});

    // ... rest of the code
  } catch (error) {
    // ...
  }
}
```

Benefits:
- ✅ Module only loaded at runtime
- ✅ No build-time dependency

Trade-offs:
- More complex code
- Dynamic imports can be tricky
- May not solve the issue completely

✅ RECOMMENDED SOLUTION: Option A + Option B

Step 1: Add Route Segment Config (5 minutes)

```
cd /home/ubuntu/carelinkai-project

# Edit the file
cat > src/app/api/inquiries/[id]/generate-response/route.ts << 'EOF'
import { NextRequest, NextResponse } from 'next/server';
import { getServerSession } from 'next-auth';
import { authOptions } from '@/lib/auth';
import { inquiryResponseGenerator } from '@/lib/ai/inquiry-response-generator';
import { inquiryEmailService } from '@/lib/email/inquiry-email-service';
import { prisma } from '@/lib/prisma';

// Tell Next.js this route is fully dynamic
export const dynamic = 'force-dynamic';
export const runtime = 'nodejs';

export async function POST(
  request: NextRequest,
  { params }: { params: { id: string } }
) {
  try {
    const session = await getServerSession(authOptions);

    // Check authorization
    if (!session?.user || !['ADMIN', 'OPERATOR'].includes(session.user.role)) {
      return NextResponse.json(
        { success: false, error: 'Unauthorized' },
        { status: 403 }
      );
    }

    const body = await request.json();
    const {
      type = 'INITIAL',
      tone,
      includeNextSteps = true,
      includeHomeDetails = true,
      sendEmail = false,
    } = body;

    // Generate AI response
    const content = await inquiryResponseGenerator.generateResponseForInquiry(
      params.id,
      {
        type,
        tone,
        includeNextSteps,
        includeHomeDetails,
      }
    );

    // Save response to database
    const response = await prisma.inquiryResponse.create({
      data: {
        inquiryId: params.id,
        content,
        type: 'AI_GENERATED',
        channel: 'EMAIL',
        sentBy: session.user.id,
        status: sendEmail ? 'SENT' : 'DRAFT',
        sentAt: sendEmail ? new Date() : null,
        metadata: {
          aiModel: 'gpt-4',
```

```
          responseType: type,
          tone,
        },
      },
    });

    // Send email if requested
    if (sendEmail) {
      const inquiry = await prisma.inquiry.findUnique({
        where: { id: params.id },
      });

      if (inquiry) {
        const emailSent = await inquiryEmailService.sendInquiryResponse(
          inquiry.contactEmail!,
          inquiry.contactName!,
          content,
          inquiry.id
        );

        if (emailSent) {
          // Update inquiry status
          await prisma.inquiry.update({
            where: { id: params.id },
            data: { status: 'CONTACTED' },
          });

          // Update response status
          await prisma.inquiryResponse.update({
            where: { id: response.id },
            data: { status: 'DELIVERED' },
          });
        } else {
          await prisma.inquiryResponse.update({
            where: { id: response.id },
            data: { status: 'FAILED' },
          });
        }
      }
    }

    return NextResponse.json({
      success: true,
      response: {
        id: response.id,
        content,
        status: response.status,
      },
    });
  } catch (error) {
    console.error('Error generating response:', error);
    return NextResponse.json(
      { success: false, error: 'Failed to generate response' },
      { status: 500 }
    );
  }
}
EOF

# Commit the fix
git add src/app/api/inquiries/[id]/generate-response/route.ts
git commit -m "fix: Add dynamic route config to prevent OpenAI build-time errors"
```

```
# Push to trigger deployment
git push origin main
```

Step 2: Add OpenAI API Key (Optional, for AI functionality)

If you have an OpenAI API key:
1. Go to Render Dashboard → Environment
2. Add `OPENAI_API_KEY=sk-proj-...`
3. Save (will trigger auto-deploy)

If you DON'T have an OpenAI API key yet:
- That's OK! The build will succeed now
- AI response generation will fail at runtime with a clear error
- You can add the key later when ready

🔬 TESTING THE FIX

Test 1: Verify Build Succeeds

```
cd /home/ubuntu/carelinkai-project
npm run build
```

Expected:

```
✅ Collecting page data ...
✅ Generating static pages ...
✅ Build completed successfully
```

Test 2: Verify Route Works (with API key)

```
# After deployment
curl -X POST https://carelinkai.onrender.com/api/inquiries/[some-id]/generate-re-
sponse \
  -H "Content-Type: application/json" \
  -H "Cookie: next-auth.session-token=..." \
  -d '{"type": "INITIAL"}'
```

Expected (with key):

```
{
  "success": true,
  "response": {
    "id": "...",
    "content": "Dear [Name], ...",
    "status": "DRAFT"
  }
}
```

Expected (without key):

```
{
  "success": false,
  "error": "Failed to generate response"
}
```

(Check logs for: “WARNING: OPENAI_API_KEY is missing”)

🐛 TROUBLESHOOTING

Issue: Build still fails with OpenAI error

Solution:
1. Verify `export const dynamic = 'force-dynamic';` is at the top of the route file
2. Clear Next.js cache: `rm -rf .next`
3. Try building again: `npm run build`
4. If still failing, use Option C (conditional import)

Issue: Runtime error “API key not found”

Solution:
- Add `OPENAI_API_KEY` to environment variables
- Or accept that AI responses won’t work until key is added
- The rest of the app should work fine

Issue: “Cannot find module” error

Solution:
- Verify `inquiry-response-generator.ts` exists
- Check import path is correct
- Run `npm install` to ensure dependencies are installed

📋 IMPLEMENTATION CHECKLIST

Before deploying:
- [ ] Added `export const dynamic = 'force-dynamic';` to route file
- [ ] Tested build locally (`npm run build`)
- [ ] Committed changes
- [ ] Pushed to GitHub
- [ ] Decided if adding OpenAI API key now or later
- [ ] If adding key: Added to Render environment
- [ ] Monitoring deployment logs
- [ ] Verified service starts successfully

After deploying:
- [ ] Build completed without errors
- [ ] Service is running
- [ ] Pipeline Dashboard loads
- [ ] Other features work
- [ ] AI response generator either works (with key) or fails gracefully (without key)

🎯 SUCCESS CRITERIA

The fix is successful when:

- [ ] `npm run build` completes without OpenAI errors
- [ ] Render deployment succeeds
- [ ] Service starts and is accessible
- [ ] Pipeline Dashboard loads
- [ ] AI response generator:
- ✅ Works if OPENAI_API_KEY is set
- ✅ Fails gracefully with clear error if key is missing

📊 LONG-TERM SOLUTION

After initial deployment, consider:

1. **Environment-specific builds:**
   - Set `OPENAI_API_KEY` in Render even if it's a dummy key
   - Use feature flags to enable/disable AI features

2. **Graceful degradation:**
   - Check for API key before showing "Generate AI Response" button
   - Show clear message: "AI responses require API key configuration"

3. **Testing strategy:**
   - Add unit tests that mock OpenAI
   - Add integration tests with real API key
   - Test build process in CI/CD

Ready to fix? Let's deploy! 🚀