Claude Code Debug Prompt V2: HEIC Upload STILL Broken After All Fixes

What We Already Tried (All Failed to Fully Fix)

Attempt 1 - Initial Fixes

- ✅ Removed `Content-Disposition: "attachment"` from S3 presigned URLs
- ✅ Changed `/api/file-url` to use presigned URLs
- ✅ Added URL resolution in admin dashboard for S3 images
- **Result**: JPEG/PNG/WebP now work, HEIC still broken

Attempt 2 - Based on Your First Analysis

- ✅ Updated `convertHeicToJpeg` to return `{ blob, preview, success }`
- ✅ Modified `handleFiles` to create new File from converted blob
- ✅ Installed heic2any (but it stayed at 0.0.4 despite `yarn add heic2any@latest`)
- **Result**: Still broken

Attempt 3 - Based on Your Second Analysis

- ✅ Added server-side HEIC conversion API at `/api/convert-heic` using `sharp`
- ✅ Updated `convertHeicToJpeg` to try client-side first, then server-side fallback
- ✅ Installed `sharp@0.34.5`
- ✅ Updated `getContentType` to handle empty `file.type`
- ✅ Added detailed `[HEIC]` console logging
- **Result**: STILL BROKEN - previews don't show, images broken in admin

Current State

Package versions (from package.json):

```
"heic2any": "^0.0.4",  // Note: Still 0.0.4 despite trying to update!
"sharp": "^0.34.5",
"@types/sharp": "^0.32.0"
```

What's happening:
1. User selects HEIC file
2. `isHeicFile()` correctly detects it
3. `convertHeicToJpeg()` is called
4. Client-side heic2any fails (probably due to old version 0.0.4)
5. Server-side sharp fallback is attempted via `/api/convert-heic`
6. **Unknown if server-side works** - sharp might not work in production because it needs native binaries

Possible root causes:
1. heic2any 0.0.4 is broken/old (yarn didn't update it)

2. sharp requires native libvips binaries which may not be available in production
3. The File object created from the converted blob might not be properly passed through state

Current Code

photo-uploader.tsx - convertHeicToJpeg function

```
const convertHeicToJpeg = async (file: File): Promise<{ blob: Blob; preview: string;
success: boolean }> => {
    // Attempt 1: Client-side conversion with heic2any
    try {
      console.log('[HEIC] Trying client-side conversion for:', file.name, 'size:', fil
e.size, 'type:', file.type);

      if (!file.size || file.size === 0) {
        console.error('[HEIC] File is empty');
        throw new Error('File is empty');
      }

      const heic2anyModule = await import('heic2any');
      console.log('[HEIC] heic2any module loaded:', !!heic2anyModule);

      const heic2any = heic2anyModule.default;

      const convertedBlob = await heic2any({
        blob: file,
        toType: 'image/jpeg',
        quality: 0.85
      });

      const blob = Array.isArray(convertedBlob) ? convertedBlob[0] : convertedBlob;
      console.log('[HEIC] Client-side conversion successful, blob size:', blob.size, '
type:', blob.type);

      if (blob.size > 0) {
        const preview = URL.createObjectURL(blob);
        return { blob, preview, success: true };
      }
      throw new Error('Converted blob is empty');
    } catch (clientError) {
      console.warn('[HEIC] Client-side conversion failed, trying server-side:', clien-
tError);
    }

    // Attempt 2: Server-side conversion with sharp
    try {
      console.log('[HEIC] Trying server-side conversion');
      const formData = new FormData();
      formData.append('file', file);

      const response = await fetch('/api/convert-heic', {
        method: 'POST',
        body: formData
      });

      if (response.ok) {
        const jpegBlob = await response.blob();
        console.log('[HEIC] Server response blob size:', jpegBlob.size);
        if (jpegBlob.size > 0) {
          console.log('[HEIC] Server-side conversion succeeded');
          const preview = URL.createObjectURL(jpegBlob);
          return { blob: jpegBlob, preview, success: true };
        }
      } else {
        const errorText = await response.text();
        console.error('[HEIC] Server-side conversion failed:', response.status, error-
Text);
      }
    } catch (serverError) {
      console.error('[HEIC] Server-side conversion also failed:', serverError);
    }

    // Both failed - return empty preview
    console.error('[HEIC] All conversion methods failed for:', file.name);
    return { blob: file, preview: '', success: false };
  };
```

photo-uploader.tsx - handleFiles (HEIC handling part)

```
if (isHeicFile(file)) {
  // Convert HEIC to JPEG for preview AND upload
  const converted = await convertHeicToJpeg(file);
  preview = converted.preview;
  setConvertingCount(prev => Math.max(0, prev - 1));

  if (converted.success) {
    // Create a new File from the converted blob for upload
    const convertedFile = new File(
      [converted.blob],
      file.name.replace(/\.heic$/i, '.jpg').replace(/\.heif$/i, '.jpg'),
      { type: 'image/jpeg' }
    );
    fileToUpload = convertedFile;
    orientation = await checkOrientation(converted.blob);
  } else {
    // Conversion failed - still add the photo but with empty preview
    orientation = 'landscape';
  }
}

newPhotos.push({
  id: `${Date.now()}-${i}`,
  file: fileToUpload, // Upload converted JPEG for HEIC, original for others
  preview,
  caption: '',
  subCategory: subcategories?.[0] || undefined,
  orientation,
  fileName: file.name
});
```

/api/convert-heic/route.ts

```
import { NextResponse } from "next/server";
import sharp from "sharp";

export const dynamic = "force-dynamic";
export const maxDuration = 30;

export async function POST(request: Request) {
  try {
    const formData = await request.formData();
    const file = formData.get('file') as File;

    if (!file) {
      return NextResponse.json({ error: "No file provided" }, { status: 400 });
    }

    console.log('[HEIC Server] Converting file:', file.name, 'size:', file.size);

    const buffer = Buffer.from(await file.arrayBuffer());

    const jpegBuffer = await sharp(buffer)
      .jpeg({ quality: 90 })
      .toBuffer();

    console.log('[HEIC Server] Conversion successful, output size:', jpegBuf-
fer.length);

    return new NextResponse(jpegBuffer, {
      headers: {
        'Content-Type': 'image/jpeg',
        'Content-Length': jpegBuffer.length.toString()
      }
    });
  } catch (error) {
    console.error("[HEIC Server] Conversion error:", error);
    return NextResponse.json({
      error: "Conversion failed",
      details: (error as Error).message
    }, { status: 500 });
  }
}
```

submission-form.tsx - uploadPhotos (relevant part)

```
const uploadPhotos = async () => {
  for (const [room, photos] of Object.entries(photosByRoom ?? {})) {
    for (const photo of (photos ?? [])) {
      if (!photo?.file) continue;

      const contentType = getContentType(photo.file);

      const presignedRes = await fetch('/api/upload/presigned', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({
          fileName: photo.file.name,
          contentType: contentType,
          isPublic: true
        })
      });
      // ... rest of upload
    }
  }
};

const getContentType = (file: File): string => {
  if (file.type && file.type !== '' && file.type !== 'application/octet-stream') {
    return file.type;
  }
  const ext = file.name?.split('.').pop()?.toLowerCase();
  const mimeTypes: Record<string, string> = {
    'jpg': 'image/jpeg',
    'jpeg': 'image/jpeg',
    'png': 'image/png',
    'webp': 'image/webp',
    'heic': 'image/heic',
    'heif': 'image/heif'
  };
  return mimeTypes[ext || ''] || 'application/octet-stream';
};
```

Critical Questions

1. **Why didn't heic2any update?** It's still 0.0.4. How do we force update to latest?

2. **Does sharp work in production?** Sharp requires native binaries (libvips). Does the production environment have these? If not, what alternatives exist?

3. **Is the converted File actually making it to upload?** The flow is:
   - `convertHeicToJpeg` returns blob
   - `handleFiles` creates new File from blob
   - `handleFiles` puts File in `newPhotos` array
   - `onPhotosChange` is called with newPhotos (this updates React state)
   - Later, `uploadPhotos` reads from `photosByRoom` state

Is there a possibility that the state update loses the converted File reference? Or that the File object doesn't serialize properly through React state?

1. **Alternative approaches?**
   - What if we skip client-side entirely and ALWAYS use server-side conversion for HEIC?

- What if we use a different library entirely?
- What if we use a cloud-based conversion service?

What I Need From You

1. **Diagnose why both conversion methods are failing** - analyze the code flow
2. **Fix heic2any version** - how to actually update it
3. **Verify sharp will work** - or provide alternative if it won't
4. **Check for React state issues** - is the File being lost somewhere?
5. **Provide bulletproof solution** - what will DEFINITELY work for iPhone HEIC photos?

Environment

- Next.js 14 with App Router
- Deployed on Abacus.AI platform (not standard Vercel/AWS)
- Production may have limited native binary support
- Users are on iPhones uploading HEIC photos via mobile Safari