Cloudinary Setup for Render Deployment

Problem

Upload routes (`/api/family/gallery/upload` and `/api/family/documents`) are returning **503 Service Unavailable** errors because Cloudinary environment variables are not configured in Render.

Cloudinary Credentials

Based on your screenshots, here are your Cloudinary credentials:

- **Cloud Name**: `dygtsnu8z`
- **API Key**: `328392542172231`
- **API Secret**: `KhpohAEFGsjVKuXRENaBhCoIYFQ`

Step-by-Step Fix

1. Set Environment Variables in Render

1. Go to Render Dashboard (https://dashboard.render.com)
2. Select your **carelinkai** service
3. Go to **Environment** tab
4. Add the following environment variables:

```
CLOUDINARY_CLOUD_NAME=dygtsnu8z
   CLOUDINARY_API_KEY=328392542172231
   CLOUDINARY_API_SECRET=KhpohAEFGsjVKuXRENaBhCoIYFQ
```

1. Click **Save Changes**

2. Trigger Redeploy

After adding the environment variables:

1. Go to **Manual Deploy** section
2. Click **Clear build cache & deploy**

This will:
- Clear any cached build artifacts
- Regenerate Prisma client
- Deploy with new environment variables

3. Verify Configuration

Once deployed, test the diagnostic endpoint:

```
curl https://carelinkai.onrender.com/api/diagnostic/cloudinary
```

Expected response (you must be logged in):

```json
{
  "timestamp": "2025-12-13T...",
  "isConfigured": true,
  "environmentVariables": {
    "CLOUDINARY_CLOUD_NAME": {
      "exists": true,
      "value": "***SET***"
    },
    "CLOUDINARY_API_KEY": {
      "exists": true,
      "value": "***SET***"
    },
    "CLOUDINARY_API_SECRET": {
      "exists": true,
      "value": "***SET***"
    }
  }
}
```

4. Test Uploads

After deployment:
1. Log in to https://carelinkai.onrender.com
2. Navigate to Family Portal → Gallery tab
3. Try uploading a photo
4. Navigate to Family Portal → Documents tab
5. Try uploading a document

Both should work without 503 errors.

Why This Fixes the Issue

The recent code changes (commit 54cbc40) added logging to help diagnose why uploads were failing. The upload routes check for Cloudinary configuration using `isCloudinaryConfigured()`:

```javascript
const useCloudinary = isCloudinaryConfigured();

if (!useCloudinary) {
  return NextResponse.json(
    {
      error: "File upload service not configured",
      code: "UPLOAD_SERVICE_NOT_CONFIGURED"
    },
    { status: 503 }
  );
}
```

Without the environment variables set in Render, `isCloudinaryConfigured()` returns `false`, causing 503 errors.

Rollback Plan

If uploads still fail after setting environment variables:

1. Check Render logs for errors:
   - Go to **Logs** tab in Render dashboard
   - Look for “Cloudinary” or “upload” related errors

2. Verify environment variables are visible:
   - In Render dashboard, check that variables show as “***” (hidden)
   - They should NOT be empty

3. Check Cloudinary dashboard:
   - Verify the credentials are correct
   - Ensure the API key is enabled
   - Check if there are any usage limits or restrictions

Additional Notes

- The diagnostic endpoint `/api/diagnostic/cloudinary` requires authentication
- Old S3-based uploads have been migrated to Cloudinary
- All uploads now use Cloudinary’s secure streaming API
- Files are organized in folders: `carelinkai/family/`, `carelinkai/residents/`, etc.

GitHub Push Issue

The latest code changes (commit 54cbc40) are not yet on GitHub due to authentication issues. After setting up Cloudinary in Render, you may need to manually trigger a deploy from the local commit or fix GitHub authentication first.

To fix GitHub auth:
1. Generate a new Personal Access Token (PAT) at https://github.com/settings/tokens
2. Ensure it has `repo` scope
3. Update git remote:
`bash`
```
   cd /home/ubuntu/carelinkai-project
   git remote set-url origin https://YOUR_TOKEN@github.com/profyt7/carelinkai.git
   git push origin main
```

Success Criteria

✅ Environment variables set in Render
✅ Deployment successful
✅ Diagnostic endpoint shows `isConfigured: true`
✅ Gallery photo uploads work
✅ Document uploads work
✅ No 503 errors in console

Support

If issues persist:
1. Check Render logs for specific error messages
2. Test the diagnostic endpoint
3. Verify Cloudinary dashboard shows API usage
4. Check that the deployed code matches the latest commit