ALKAMIND.COM

Website Rebuild Project

IMPLEMENTATION GUIDE

Version 6.1 — Decap CMS (Git-Based)

Zero external dependencies • Content lives in Git • Free forever

January 2026

v6.1 Update: Includes Implementation Log with real-world challenges and resolutions

What Changed in Version 6.1

This version adds Addendum A: Implementation Log documenting the actual challenges encountered during the first implementation on January 4, 2026. Key corrections from field experience:

v6.0 (Original)	v6.1 (Corrected)
npm install @nuxt/content	npm install @nuxt/content@^2
Phase 4 steps only	Phase 4 + Identity workarounds
4 CMS collections	6 CMS collections (all nav items)
No typography plugin	Includes @tailwindcss/typography

Why This Change?

Contentful's free tier restricts commercial use. Sanity was evaluated but adds unnecessary complexity for a solo-edited site. Decap CMS provides a visual editor while keeping content in Git — the simplest solution for commercial use at zero cost.

Phase 1: Prerequisites

Estimated Time: 30 minutes (if starting fresh) | 0 minutes (if continuing from v4)

1.1 Option A: Continue from v4 Build

If you completed the v4 Implementation Guide, you already have the foundation. Verify your setup:

```
git checkout nuxt-rebuild
git log --oneline -1
```

You should see your latest commit. Skip to Phase 2.

1.2 Option B: Clone from Repository

If starting fresh, clone the repository at the Contentful integration point:

1. Clone the repository:

```
git clone https://github.com/alf2rock/alkamind.git
cd alkamind
```

2. Switch to the development branch:

```
git checkout nuxt-rebuild
```

3. Install dependencies:

```
npm install
```

4. Verify it runs:

```
npm run dev
```

Open http://localhost:3000 — you should see the site (content may be missing without Contentful keys).

1.3 Required Tools

Tool	Version / Notes
Node.js	v20+ required (Nuxt 4 requirement)
Git	Any recent version
VS Code	Recommended editor
GitHub account	For repository and Decap authentication
Netlify account	Free tier — for deployment

✓ Checkpoint: Starting point confirmed — Nuxt site with navigation and page structure ready

Phase 2: Remove Contentful

Estimated Time: 15-20 minutes

In this phase, we strip out the Contentful integration to prepare for Decap CMS.

2.1 Remove Contentful Dependencies

1. Uninstall Contentful packages:

```
npm uninstall contentful @contentful/rich-text-html-renderer
```

2.2 Remove Contentful Composable

2. Delete the Contentful composable file:

```
rm app/composables/useContentful.ts
```

2.3 Clean nuxt.config.ts

Open nuxt.config.ts and remove the Contentful runtime config:

Remove this block:

```
runtimeConfig: {
  public: {
    contentfulSpaceId: process.env.CONTENTFUL_SPACE_ID,
    contentfulAccessToken: process.env.CONTENTFUL_ACCESS_TOKEN
  }
}
```

2.4 Remove Environment Variables

3. Delete .env file (if present):

```
rm .env
```

Note: You will NOT need Contentful environment variables in Netlify either.

2.5 Update Page Components

Each page component currently fetches from Contentful. We'll replace these with static content temporarily, then connect to Decap in Phase 3.

For each page in app/pages/, replace Contentful fetch calls with placeholder content. Example for index.vue:

```
<script setup lang="ts">
// Removed: const { getHomePage } = useContentful()
// Removed: const { data: homepage } = await useAsyncData(...)

const homepage = {
  heroTitle: 'Alkamind Consulting',
  heroSubtitle: 'Your placeholder text here',
  callToActionText: 'Book a Conversation',
  callToActionLink: 'https://calendly.com/your-link'
}
</script>
```

Repeat for: our-story.vue, about.vue, blog/index.vue, blog/[slug].vue, ai-portals.vue, use-cases.vue

2.6 Verify Clean Build

```
npm run dev
```

The site should run with placeholder content. No errors should reference Contentful.

✓ Checkpoint: Contentful removed — site runs with static placeholder content

Phase 3: Add Decap CMS

Estimated Time: 45-60 minutes

Decap CMS provides a visual editor that commits content directly to your Git repository.

3.1 Install Nuxt Content Module

Decap CMS works with markdown files. Nuxt Content renders them.

> **⚠ CRITICAL: Use version 2.x — version 3.x has breaking API changes**

```
npm install @nuxt/content@^2
```

Update nuxt.config.ts to include the module:

```
modules: ['@nuxtjs/tailwindcss', '@nuxt/content'],
```

3.2 Install Tailwind Typography Plugin

Required for proper markdown heading styling:

```
npm install @tailwindcss/typography
```

Create or update tailwind.config.ts:

```
import type { Config } from 'tailwindcss'

export default {
  content: [],
  theme: { extend: {} },
  plugins: [require('@tailwindcss/typography')],
} satisfies Config
```

3.3 Create Content Directory Structure

```
mkdir -p content/blog
mkdir -p content/pages
```

Your content will live in these directories as markdown files.

3.4 Create Admin Directory

Decap CMS is a single-page app that runs at /admin:

```
mkdir -p public/admin
```

3.5 Create Admin Index File

Create public/admin/index.html:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Alkamind CMS</title>
  <script src="https://identity.netlify.com/v1/netlify-identity-
widget.js"></script>
</head>
<body>
  <script src="https://unpkg.com/decap-cms@^3.0.0/dist/decap-cms.js"></script>
</body>
</html>
```

3.6 Create CMS Configuration

Create public/admin/config.yml — this defines your content structure:

```
backend:
  name: git-gateway
  branch: nuxt-rebuild

media_folder: public/images
public_folder: /images

collections:
  - name: pages
    label: Pages
    files:
      - name: home
        label: Home Page
        file: content/pages/home.md
        fields:
          - { name: title, label: Title, widget: string }
          - { name: subtitle, label: Subtitle, widget: text }
```

```
        - { name: ctaText, label: CTA Button Text, widget: string }
        - { name: ctaLink, label: CTA Button Link, widget: string }

    - name: our-story
      label: Our Story
      file: content/pages/our-story.md
      fields:
        - { name: title, label: Title, widget: string }
        - { name: body, label: Body, widget: markdown }

    - name: about
      label: About
      file: content/pages/about.md
      fields:
        - { name: title, label: Title, widget: string }
        - { name: body, label: Body, widget: markdown }

    - name: ai-portals
      label: AI Portals
      file: content/pages/ai-portals.md
      fields:
        - { name: title, label: Title, widget: string }
        - { name: body, label: Body, widget: markdown }

    - name: use-cases
      label: Use Cases
      file: content/pages/use-cases.md
      fields:
        - { name: title, label: Title, widget: string }
        - { name: body, label: Body, widget: markdown }

- name: blog
  label: Blog Posts
  folder: content/blog
  create: true
  slug: "{{slug}}"
  fields:
    - { name: title, label: Title, widget: string }
```

```
      - { name: slug, label: Slug, widget: string }
      - { name: date, label: Date, widget: datetime }
      - { name: author, label: Author, widget: string }
      - { name: excerpt, label: Excerpt, widget: text }
      - { name: body, label: Body, widget: markdown }
```

⚠ Ensure ALL navigation items have CMS collections — see Addendum A, Challenge 3

3.7 Create Initial Content Files

Create content/pages/home.md:

```
---
title: Alkamind Consulting
subtitle: Change Management & Solutions Development
ctaText: Book a Conversation
ctaLink: https://calendly.com/alf-alkamind/coaching-session
---
```

Create content/pages/our-story.md:

```
---
title: Our Story
---

Your story content here in markdown...
```

Create similar files for: about.md, ai-portals.md, use-cases.md

3.8 Update Page Components to Use Content

Update app/pages/index.vue to fetch from Nuxt Content:

```
<script setup lang="ts">
const { data: home } = await useAsyncData('home', () =>
  queryContent('/pages/home').findOne()
)
</script>

<template>
  <div>
    <h1>{{ home?.title }}</h1>
    <p>{{ home?.subtitle }}</p>
    <a :href="home?.ctaLink">{{ home?.ctaText }}</a>
  </div>
</template>
```

For pages with markdown body content, use ContentRenderer:

```
<template>
  <div class="prose">
    <h1>{{ page?.title }}</h1>
    <ContentRenderer :value="page" />
```

```
  </div>
</template>
```

3.9 Test Locally

```
npm run generate
```

This command tests the full build. Watch for errors like:

```
ERROR [unhandledRejection] queryContent is not defined
```

If you see this error, ensure @nuxt/content is version 2.x, not 3.x.

```
npm run dev
```

Test the site at http://localhost:3000

✓ Checkpoint: Decap CMS installed — content loads from markdown files

Phase 4: Deploy to Netlify

Estimated Time: 20-30 minutes

Netlify provides native integration with Decap CMS including authentication.

4.1 Create netlify.toml

Create netlify.toml in project root:

```
[build]
  command = "npm run generate"
  publish = "dist"

[build.environment]
  NODE_VERSION = "20"
```

4.2 Connect to Netlify

1. Go to https://app.netlify.com
2. Click 'Add new site' → 'Import an existing project'
3. Connect to GitHub and select 'alkamind' repository
4. Set branch to deploy: nuxt-rebuild
5. Build settings should auto-detect from netlify.toml
6. Click 'Deploy site'

4.3 Enable Netlify Identity

Netlify Identity provides authentication for the CMS:

7. Go to Site settings → Identity
8. Click 'Enable Identity'
9. Under Registration, select 'Invite only' (recommended)
10. Under Services → Git Gateway, click 'Enable Git Gateway'

4.4 Invite Yourself

11. In Netlify: Identity → Invite users
12. Enter your email address
13. Check email and accept invitation

⚠ Branch deploy URL workaround — see Addendum A, Challenge 1

If the email link redirects to your production domain instead of the branch deploy:

- Copy the recovery_token from the email link
- Replace the domain with: https://[branch]--[site-name].netlify.app
- Example: https://nuxt-rebuild--gifted-pare-29ba8f.netlify.app/admin/#recovery_token=xxxxx

4.5 Commit and Deploy

```
git add .
git commit -m "Add Decap CMS with Netlify Identity"
git push origin nuxt-rebuild
```

Netlify will automatically rebuild. Once complete:

- Visit https://your-site.netlify.app to see the site
- Visit https://your-site.netlify.app/admin to access the CMS

✓ Checkpoint: Decap CMS deployed to Netlify with authentication

Phase 5: Content Migration

Estimated Time: 30-60 minutes (depends on content volume)

Migrate content manually from Contentful to Decap CMS via copy/paste.

5.1 Export from Contentful

1. Log into Contentful (your nuxt-contentful-demo still has access)
2. For each content entry, copy the text content
3. Note: Rich text will need to be reformatted as markdown

5.2 Content Mapping

Contentful Entry	Decap Collection	File
Home Page	pages → home	content/pages/home.md
Our Story	pages → our-story	content/pages/our-story.md
About	pages → about	content/pages/about.md
AI Portals	pages → ai-portals	content/pages/ai-portals.md
Use Cases	pages → use-cases	content/pages/use-cases.md
Blog Posts	blog	content/blog/*.md

5.3 Enter Content via CMS

4. Go to https://your-site.netlify.app/admin
5. Log in with your Netlify Identity credentials
6. Navigate to each collection
7. Paste content from Contentful
8. Click 'Publish' — this commits to Git automatically

5.4 Pull Content Locally

After publishing via CMS, the content is in your repo:

```
git pull origin nuxt-rebuild --rebase
```

⚠ Use --rebase to avoid merge conflicts with CMS commits

You'll see new/updated markdown files in content/.

✓ Checkpoint: Content migrated from Contentful to Decap CMS

Phase 6: Go-Live

Estimated Time: 15-30 minutes

When ready, merge to main and connect your custom domain.

6.1 Pre-Flight Checklist

- All content migrated and verified
- Site renders correctly on Netlify preview URL
- CMS login works and you can edit/publish
- Images display correctly
- Navigation works on all pages
- Mobile responsive design verified

6.2 Archive Contentful Branch

Before merging, preserve the Contentful version:

```
git checkout nuxt-rebuild
git checkout -b nuxt-contentful-demo
git push origin nuxt-contentful-demo
```

This branch remains on Cloudflare Pages as a demo/archive.

6.3 Merge to Main

```
git checkout main
git merge nuxt-rebuild
git push origin main
```

6.4 Update Netlify Production Branch

1. In Netlify: Site settings → Build & deploy → Branches
2. Change production branch from 'nuxt-rebuild' to 'main'
3. Trigger a new deploy

6.5 Connect Custom Domain

4. In Netlify: Domain settings → Add custom domain
5. Enter: alkamind.com

6. Update DNS records as instructed (or use Netlify DNS)
7. Wait for DNS propagation and HTTPS certificate

✓ Checkpoint: Site live on Netlify with Decap CMS — zero monthly cost

Summary

Branch Strategy (Final State)

Branch	Deployment	Purpose
main	Netlify (production)	Live site with Decap CMS
nuxt-rebuild	Netlify (preview)	Development branch
nuxt-contentful-demo	Cloudflare Pages	Archived Contentful version
v1-react-backup	None	Original React site archive

Key Files Reference

File	Purpose
public/admin/index.html	Decap CMS entry point
public/admin/config.yml	CMS configuration and content models
content/pages/*.md	Page content as markdown
content/blog/*.md	Blog posts as markdown
netlify.toml	Netlify build configuration
tailwind.config.ts	Typography plugin configuration

Addendum A: Implementation Log

Date: January 4, 2026 | **Branch:** nuxt-rebuild | **Status:** Phases 1-5 Complete

This log documents the actual challenges encountered during the first implementation of this guide, and how they were resolved through human-AI collaboration.

Challenge 1: Netlify Identity Email Redirects

Problem

When enabling Netlify Identity for CMS authentication, invitation and password reset emails redirected users to alkamind.com (the production domain) instead of the branch deploy URL.

Impact

Users could not complete the authentication flow to access the CMS admin panel.

Resolution Attempts

1. Tried modifying the email link URL manually (partial success)
2. Attempted to use Netlify dashboard to set passwords directly
3. Changed registration from 'Invite only' to 'Open' temporarily

Final Solution

Discovered the branch-specific deploy URL format:

```
https://[branch]--[site-name].netlify.app/admin/#recovery_token=xxxxx
```

Lesson Learned

Netlify Identity uses the primary domain for all email links. For branch deploys, manually construct the URL using the pattern above.

Challenge 2: CMS Content Not Appearing on Site

Problem

After configuring Decap CMS, publishing content, and verifying successful Netlify deploys, content changes were not appearing on the preview site.

Symptoms

- CMS showed 'Published' status
- Netlify showed successful deploys
- Git commits from CMS visible in repository
- Content files contained correct updated content

- Preview site showed no changes

Human Operator's Critical Insight

> *"My money is that the code is where the problem is. What do you think?"*

While the AI assistant was focused on caching, CDN, and browser issues, the human operator recognized that the symptoms pointed to a code-level problem.

Root Cause Discovery

Running npm run generate locally revealed:

```
ERROR [unhandledRejection] queryContent is not defined
```

The @nuxt/content module was installed at version 3.x, which uses a different API than version 2.x. The queryContent() composable was not available in v3.

Resolution

```
npm uninstall @nuxt/content
npm install @nuxt/content@^2
```

Lesson Learned

Always verify module API compatibility when installing latest versions. The @nuxt/content v3 migration introduced breaking changes not apparent during installation.

Challenge 3: Navigation/CMS Collection Mismatch

Problem

The site navigation included 6 pages, but the CMS only had 4 collections configured.

Human Operator's Observation

> *"The CMS has Pages 'home page', 'Our Story', 'About' and a 'Blog post' in the collections. The site also has AI Portals and Use Cases - which are not in the CMS. This leads me to suspect that we still have a mismatch between code and CMS!"*

Analysis

NavBar Links	CMS Collections	Status
Home	Home Page	✓
Our Story	Our Story	✓
About Us	About	✓
Blog	Blog Posts	✓
AI Portals	—	✗ Missing
Use Cases	—	✗ Missing

Resolution

1. Added AI Portals and Use Cases to public/admin/config.yml
2. Created content files: content/pages/ai-portals.md and use-cases.md
3. Updated Vue pages to use queryContent() and ContentRenderer

Challenge 4: Markdown Heading Styling

Problem

Markdown headings (H1, H2, etc.) were rendering as plain text without proper styling.

Cause

The @tailwindcss/typography plugin was not installed, so the prose CSS classes had no effect.

Resolution

```
npm install @tailwindcss/typography
```

Then created tailwind.config.ts with the typography plugin.

Challenge 5: Git Push Conflicts from CMS Commits

Problem

Multiple push attempts failed because the CMS was simultaneously committing content changes to the remote branch.

Error

```
! [rejected] nuxt-rebuild -> nuxt-rebuild (fetch first)
error: failed to push some refs
```

Resolution

```
git pull origin nuxt-rebuild --rebase
git push origin nuxt-rebuild
```

Lesson Learned

When working with a Git-based CMS, expect concurrent commits. Always pull with rebase before pushing code changes.

Addendum B: Lessons Learned

Extracted from the implementation log for quick reference:

1. **Pin module versions:** Specify exact versions in package.json to avoid unexpected API changes. Use @nuxt/content@^2, not @nuxt/content.
2. **Test locally first:** Run npm run generate locally before pushing to catch build errors that won't appear in npm run dev.
3. **Branch deploy URLs:** Netlify Identity uses the primary domain. For branch deploys, manually construct: https://[branch]--[site-name].netlify.app
4. **CMS/Code sync check:** Before deployment, verify ALL navigation items have corresponding CMS collections.
5. **Git rebase with CMS:** When working with Git-based CMS, expect concurrent commits. Use git pull --rebase before pushing.
6. **Typography plugin:** Install @tailwindcss/typography for proper markdown heading styling with Tailwind's prose classes.

Key Commits from Implementation

Commit	Description
be7380d	Migrate from Contentful to Decap CMS (Git-based)
abd3e8e	Fix: Downgrade @nuxt/content to v2 for queryContent support
90fcb56	Add AI Portals and Use Cases to CMS, fix markdown styling

Working URLs (as of January 4, 2026)

- Preview Site: https://nuxt-rebuild--gifted-pare-29ba8f.netlify.app/
- CMS Admin: https://nuxt-rebuild--gifted-pare-29ba8f.netlify.app/admin/

Recognition

The successful resolution of Challenge 2 (the critical content rendering blocker) was directly attributable to the human operator's pattern recognition. While the AI assistant pursued infrastructure-level causes, the human operator's intuition identified the code-level root cause.

> *"My money is that the code is where the problem is."*

This insight redirected troubleshooting and led directly to discovering the queryContent is not defined error.

— End of Document —

Alkamind Implementation Guide v6.1

January 2026

Developed through AI-human collaboration

This artifact is sacred. Preserve it.