FREERESOURCE

Claude Code CMS Dashboard Playbook

How to let clients edit AI-built websites without breaking the design

01 What This CMS Is All About

A CMS is the control panel that lets you or your client edit a live website - text, pages, images, SEO, and publishing - without opening the code or touching your private developer account. Claude Code can build the site, but the CMS gives the owner a safe editing layer after launch.

QUICK TIPS

- Keep GitHub, Vercel, and database access private.
- Let clients edit content and SEO, not raw code.
- Use one master CMS pattern across many client websites.

02 The Problem It Solves

Without a CMS, every small update comes back to you: pricing, headlines, testimonials, photos, or page copy. That turns a website project into endless manual support unless you create a safe client handoff layer.

QUICK TIPS

- Do not give clients your developer login.
- Use the CMS as a higher-value offer: build + editor + support.
- Charge setup plus optional hosting, maintenance, or analytics.

03 Core Dashboard Features

Version one should stay simple: page editor, SEO panel, multi-page manager, preview modes, AI edit chat, publish button, version history, client login, and optional form inbox.

QUICK TIPS

- Start with editable text, images, buttons, pages, and SEO fields.
- Add mobile preview before publishing.
- Save drafts until the client intentionally clicks Publish.

04 Simple Architecture

Think of it as two connected systems: the public website and the private editing dashboard. The public site stays fast on Vercel. The dashboard stores structured content in MongoDB and publishes approved changes to the live site.

QUICK TIPS

- Next.js or React for the dashboard.
- MongoDB Atlas for content, pages, settings, and versions.
- OpenRouter or another AI API for assisted copy editing.

05 Implementation Plan

Build the website first, identify editable content, create the dashboard, connect MongoDB, add AI editing, add safety validation, add version history, deploy the CMS, and test the full edit-to-publish flow.

QUICK TIPS

- Do not expose layout controls in version one.
- Every publish should create a snapshot for rollback.
- Test login permissions before handoff.

06

MongoDB Data Model

Use three simple collections to start: Site, Page, and Version. Site stores the client and live URL. Page stores slug, title, SEO fields, sections, and status. Version stores publish snapshots for rollback.

QUICK TIPS

- Keep sections as structured JSON, not raw HTML.
- Store draft and published status separately.
- Save publishedBy and publishedAt on every version.

07

Claude Code Prompt

Ask Claude Code to build the CMS in phases: data model, dashboard UI, page editor, preview and publish flow, MongoDB connection, AI rewrite tool, login/access control, and deployment instructions.

QUICK TIPS

- Paste the full prompt from the appendix into Claude Code.
- Ask Claude to validate fields before publishing.
- Keep client permissions scoped to one website.

08

Client Handoff Checklist

Before you give the client access, confirm the live site works, CMS URL works, database is connected, AI editing is budget-limited, login works, publish updates the live site, rollback works, SEO saves, and mobile preview is checked.

QUICK TIPS

- Create a one-page client usage guide.
- Keep master infrastructure access under your control.
- Offer support as a recurring monthly service.

Ready to go deeper?

This playbook is just the start. In Vibe Coding Mastery, you learn how to turn Claude Code workflows into deployed products, client systems, and repeatable AI builds.

Join the Vibe Coding Build ->

https://builds.digicuratoragency.com/

Claude Code CMS Dashboard Prompt

Paste this into Claude Code after your website is deployed.

```
Build a CMS dashboard for my deployed website.

Goal:
I want one admin dashboard where I can manage multiple client websites. Each website should have editable
pages, SEO settings, preview, save draft, publish, and version history.

Tech stack:
- Next.js or React for the dashboard
- MongoDB Atlas for content and version storage
- Vercel for deployment
- GitHub repo is already connected
- OpenRouter API for AI-assisted copy editing

Rules:
- Clients can edit copy, images, buttons, SEO fields, and add simple pages.
- Clients cannot edit raw code or break the core design.
- Every change must be validated before publishing.
- Every publish creates a version snapshot for rollback.
- Each client only sees their own website after login/password.

Build this in phases:
1. Data model
2. Dashboard UI
3. Page editor
4. Preview and publish flow
5. MongoDB connection
6. AI rewrite tool
7. Login/access control
8. Deployment instructions
```

Data Model Starter

```
Site: siteId, clientName, liveUrl, vercelProjectId, editorPasswordHash
Page: pageId, siteId, slug, title, seoTitle, metaDescription, sections, status
Version: versionId, pageId, publishedAt, publishedBy, snapshot, notes
```