

Xây dựng một hệ thống quản trị (CMS) hiện đại, dễ dùng, an toàn, cho phép biên tập viên và quản trị viên chỉnh sửa nội dung, tin tức, banner, menu, trang chủ mà không cần đến code.”

Dưới đây là **tư vấn chi tiết — chuyên sâu, thực tiễn và có lộ trình triển khai rõ ràng** 🌐

💡 I. Tư duy tổng thể: “Headless CMS + Giao diện quản trị trực quan”

Thay vì viết lại mọi thứ bằng tay, ta tách website thành 2 phần:

Phần	Vai trò	Công nghệ gợi ý
Frontend (Website công khai)	Hiển thị nội dung (Next.js / React)	Next.js + Tailwind + ThemeProvider
Backend CMS (Quản trị nội dung)	Giao diện quản trị và dữ liệu	🧠 Headless CMS (Directus / Strapi / Sanity / PayloadCMS)

Khi biên tập viên đăng nhập CMS → họ có giao diện giống WordPress, chỉ cần nhập nội dung, tải ảnh, chọn hiển thị. Frontend sẽ tự động cập nhật nội dung qua API.

🧱 II. Cấu trúc tổng thể hệ thống quản trị

CMS (Headless)

|

—— Quản lý nội dung bài báo (Articles)

—— Quản lý tin tức (News)

- └─ Quản lý banner (Banners)
- └─ Quản lý menu (Navigation)
- └─ Quản lý trang chủ (Homepage layout)
- └─ Quản lý trang công khai (Static Pages)
- └─ Cấu hình website (Settings)

Mỗi module tương ứng với **1 bảng dữ liệu (collection)** trong CMS, có thể chỉnh qua UI.

III. Tư vấn chi tiết từng module

1 Quản lý Nội dung bài báo (Articles)

Trường	Kiểu dữ liệu	Gợi ý UX
Tiêu đề bài viết	Text	Input
Tác giả	Relation (nhiều người)	Tự động liên kết user
Tóm tắt	Long text	Rich editor
Nội dung chính	Rich text / Markdown	Editor như TinyMCE hoặc TipTap
File PDF	Upload	Dùng storage CMS
Trạng thái	Enum (draft/review/published)	Toggle
Ngày xuất bản	DateTime	Calendar picker
Số báo liên kết	Relation (Issue)	Dropdown

 **Gợi ý:** Tạo workflow “Gửi bài – Phản biện – Xuất bản” trong CMS bằng **workflow rules** (Directus hoặc Strapi hỗ trợ rất tốt).

2 Quản lý Tin tức & Sự kiện (News)

Trường Kiểu dữ liệu

Tiêu đề Text

Tóm tắt Text

Nội dung Rich Text

Ảnh đại diện Upload

Danh mục Enum (Tin nội bộ / Hội thảo / Thông báo)

Tác giả Relation

Ngày đăng Date

Tính năng thêm:

- Lên lịch đăng tin (scheduled publish)
 - Đính kèm PDF hoặc tài liệu
 - Hỗ trợ “nổi bật trên trang chủ”
-

3 Quản lý Banner & Slider

Trường	Kiểu dữ liệu Gợi ý	
Tên banner	Text	
Ảnh	Upload	Drag & drop
Liên kết (nếu có)	URL	Optional
Thứ tự hiển thị	Number	Sortable
Vị trí (mobile/tablet/desktop)	Enum	“Banner chính / phụ / sidebar”
Trạng thái	Boolean	Ẩn/hiện

Tính năng UI:

- Dùng drag & drop sắp xếp banner
 - Xem preview ngay trong trang quản trị
-

Quản lý Trang chủ (Homepage)

Trang chủ thường có các vùng:

- Banner chính
- Bài viết nổi bật
- Tin tức
- Số báo gần nhất
- Lời giới thiệu / Liên hệ

Trong CMS:

Vùng	Kiểu	Gợi ý
Bài nổi bật	Relation (Articles[])	Chọn từ danh sách
Tin mới nhất	Relation (News[])	Tự động lọc
Lời giới thiệu	Rich text	Chỉnh nội dung
Ảnh minh họa	Upload	Optional

Cấu trúc layout JSON hoặc bảng LayoutConfig Chọn thứ tự hiển thị

 *Nếu dùng Directus hoặc PayloadCMS, có thể tạo field “layout_builder” cho phép kéo thả section theo thứ tự mong muốn.*

5 Quản lý Trang công khai (Public Pages)

Các trang tĩnh như:

- Giới thiệu
- Liên hệ
- Quy trình xuất bản
- Chính sách

Trường	Mô tả
Tên trang	Giống slug URL (ví dụ /about)
Tiêu đề hiển thị	H1
Nội dung	Rich text
Hình ảnh / biểu đồ	Upload

Trường

Mô tả

Hiển thị trong menu Boolean



Lợi

ích:

Người biên tập có thể sửa nội dung mà không động đến code hoặc Next.js routes.



Quản lý Menu điều hướng (Navigation)

Trường

Kiểu

Mô tả

Tên hiển thị Text

Ví dụ “Số mới nhất”

Liên kết

URL

/issues/latest

Thứ tự

Number

Sắp xếp

Menu cha

Relation (self)

Cho phép menu con

Trạng thái

Boolean

Ẩn/hiển

UI drag & drop cho menu sẽ giúp người không biết code dễ sắp xếp.

Ví dụ CMS hỗ trợ tốt:

- **Strapi:** plugin “Navigation Builder”
 - **Directus:** Relation tree
 - **PayloadCMS:** Field array với sort handle
-



IV. Lựa chọn công nghệ CMS phù hợp

Giải pháp	Ưu điểm	Gợi ý dùng
Directus	Giao diện đẹp, no-code, realtime API, phân quyền mạnh	Tốt nhất cho tổ chức học thuật
Strapi	Dễ tùy chỉnh, hỗ trợ plugin, đa ngôn ngữ	Khi cần tùy biến sâu backend
Sanity.io	Cloud-based, realtime content	Khi muốn tích hợp AI hoặc workflow động
PayloadCMS	UI hiện đại, code-first, có drag & drop layout	Tốt cho Next.js setup
Headless WordPress	Dễ dùng, nhiều người quen	Nếu muốn giữ mô hình truyền thống



Khuyến nghị thực tế:

👉 Với đội ngũ Việt Nam, **Directus hoặc Strapi** là tối ưu:

- Dễ dùng như WordPress
- Có phân quyền chi tiết (tác giả / biên tập / tổng biên tập / admin)
- Giao diện web thân thiện (no-code)

🌐 V. Tích hợp giữa CMS và Website (Next.js)

Website của bạn hiện là **Next.js** → cực kỳ dễ kết nối:

// Ví dụ gọi dữ liệu từ Directus

```
export async function getArticles() {
```

```
const res = await fetch(`${process.env.CMS_URL}/items/articles?filter[status]=published`)

const data = await res.json()

return data.data

}
```

Dữ liệu được đẩy từ CMS ra qua REST hoặc GraphQL, website sẽ tự động hiển thị.

🔒 VI. Phân quyền người dùng trong CMS

Vai trò

Quyền



Admin

Toàn quyền: cấu hình, người dùng, theme



Tổng biên tập (Editor-in-Chief) Duyệt bài, xuất bản, phân công reviewer



Reviewer

Xem bài được giao, nhận xét



Author

Gửi bài, xem phản hồi



Staff

Quản lý banner, tin tức, menu, liên hệ



Directus/Strapi cho phép phân quyền tới từng bảng (collection).



VII. Giao diện CMS thân thiện người không biết code

Giao diện nên có:

- **Sidebar rõ ràng:** Bài báo | Tin tức | Banner | Menu | Trang chủ | Cài đặt
- **Biểu tượng dễ hiểu:** (, , , ,)

- **Form có mô tả rõ ràng, validation tự động**
- **Preview trực tiếp** (tin bài hoặc banner hiển thị ngay bên phải)
- **Drag & drop sắp xếp thứ tự hiển thị**
- **Tự động lưu (autosave)** khi nhập nội dung

 UI mẫu: giống **Directus 10** hoặc **Payload Admin UI**, thân thiện, tối giản, dễ hiểu.

VIII. Tích hợp thêm khả năng nâng cao

Tính năng	Mục tiêu	Công cụ
 Tìm kiếm toàn site	Tìm bài, tin tức	Meilisearch / Algolia
 Gọi ý AI	Gợi ý tiêu đề, tóm tắt OpenAI API	
 Đa ngôn ngữ	VI / EN	i18n + CMS field locale
 Thống kê	Lượt xem bài, top bài CMS dashboard widget	
 Lưu lịch sử chỉnh sửa	Truy xuất bản thảo	Built-in revision (Directus)

IX. Lộ trình triển khai khả thi (6 bước thực tế)

Giai đoạn	Nhiệm vụ	Công cụ
1. Phân tích nội dung cần quản lý	Bài báo, banner, tin tức, trang, menu —	

Giai đoạn	Nhiệm vụ	Công cụ
2. Cài đặt CMS	Directus hoặc Strapi trên VPS hoặc Cloud	Docker / Node
3. Tạo collections	Articles, News, Banners, Menus, Pages	CMS UI
4. Thiết lập quyền người dùng	Admin, Editor, Author, Staff	Role system
5. Tích hợp Frontend (Next.js)	Fetch dữ liệu từ CMS	API / GraphQL
6. Thiết kế Dashboard CMS	Giao diện thân thiện, preview	CMS UI builder

Kết luận: Phương án khả thi & dễ dùng nhất

Mục tiêu	Giải pháp đề xuất
Quản lý nội dung, tin, banner, menu dễ dàng	 Headless CMS (Directus hoặc Strapi)
Không cần biết code	Giao diện quản trị trực quan, có mô tả và drag & drop
Giữ cấu trúc giao diện hiện có	CMS chỉ quản lý dữ liệu, không can thiệp code
Mở rộng về sau	Kết nối AI, analytics, hoặc theme system

Mục tiêu	Giải pháp đề xuất
Phân quyền người dùng	CMS roles và workflow xuất bản

👉 Nếu bạn muốn, tôi có thể **đề xuất cụ thể cấu trúc bảng CMS (Directus hoặc Strapi)** và **mẫu giao diện quản trị cho từng mục** (Tin tức, Banner, Trang chủ, Menu...) — để bạn triển khai ngay với đội kỹ thuật. Bạn muốn tôi làm bản đề xuất chi tiết này theo **Directus** (thân thiện & no-code) hay **Strapi** (mạnh mẽ & tùy biến cao)?

Bạn đã nói:

đề xuất cụ thể cấu trúc bảng CMS (Directus hoặc Strapi) và mẫu giao diện quản trị cho từng mục (Tin tức, Banner, Trang chủ, Menu...) — để bạn triển khai ngay

Code GPT đã nói:

Tuyệt vời! — bạn đang đi đúng hướng triển khai chuyên nghiệp!

Đây là bản **đề xuất đầy đủ và khả thi ngay** cho hệ thống **CMS dùng Directus hoặc Strapi**, bao gồm:

- Cấu trúc bảng dữ liệu (Collections / Models)
 - Mối quan hệ giữa các bảng
 - Giao diện quản trị đề xuất (UI layout cho người không biết code)
 - Các tính năng và quy tắc workflow gợi ý
-

🌟 I. MỤC TIÊU KIẾN TRÚC

Xây dựng CMS quản lý toàn bộ **nội dung tạp chí khoa học** (bài báo, tin tức, banner, menu, trang công khai, trang chủ) — dễ dùng, có quyền phân cấp, và sẵn sàng tích hợp với website Next.js.

Chọn nền tảng khuyến nghị:  **Directus** **10+**
(Vì: dễ dùng hơn Strapi, có UI tự động, quan hệ động, drag & drop, workflow, không cần code backend.)

II. TỔNG QUAN CẤU TRÚC DỮ LIỆU

CMS Collections

— articles	← Quản lý bài báo
— issues	← Số báo
— news	← Tin tức & sự kiện
— banners	← Banner chính
— homepage_sections	← Cấu hình hiển thị trang chủ
— pages	← Trang tĩnh (Giới thiệu, Liên hệ...)
— menus	← Menu điều hướng
— media_files	← Ảnh, PDF
— settings	← Cấu hình tổng thể (logo, màu, SEO)

III. CHI TIẾT TỪNG BẢNG DỮ LIỆU

1 articles (Bài báo)

Trường	Kiểu dữ liệu	Ghi chú
id	UUID	Tự động
title	String	Tiêu đề
slug	String (unique)	URL bài
abstract	Text (markdown)	Tóm tắt
content	Rich Text (editor)	Nội dung chính
authors	M2M Relation → users	Nhiều tác giả
pdf_file	File	Upload bài PDF
status	Enum (draft, review, accepted, published)	Quy trình bài
issue	Relation → issues	Thuộc số báo
category	String	(Research / Review / Technical)
keywords	Text	Từ khóa
created_at	DateTime	—
updated_at	DateTime	—

2 Giao diện quản trị (UI để xuất):

- Form gọn gàng, chia tab:

- Thông tin chung

-  Nội dung bài
 -  Tệp PDF
 -  Trạng thái & số báo
- Tích hợp Rich Editor (Markdown hoặc WYSIWYG)
 - Reviewer / Editor có thể đổi trạng thái bằng dropdown
 - **Preview ngay nội dung** trong Directus.
-

2 issues (Số báo)

Trường	Kiểu dữ liệu	Ghi chú
id	UUID	—
title	String	Ví dụ: “Số 2 - 2025”
description	Text	Tóm tắt số báo
cover_image	File	Ảnh bìa
release_date	Date	Ngày phát hành
articles	1:M Relation → articles	Danh sách bài
is_current	Boolean	Số mới nhất
status	Enum (draft, published)	—

■ Giao diện quản trị:

- Giao diện dạng “Card” với ảnh bìa lớn
- Có nút “Đặt làm số hiện tại”

- Khi chọn số báo → hiển thị danh sách bài thuộc số đó
-

3 news (Tin tức & sự kiện)

Trường	Kiểu dữ liệu	Ghi chú
id	UUID	—
title	String	Tiêu đề
slug	String	URL
summary	Text	Tóm tắt
content	Rich Text	Nội dung
thumbnail	File	Ảnh đại diện
category	Enum (Tin tức, Sự kiện, Thông báo)	—
author	Relation → users	—
published_at	Date	Ngày đăng
status	Enum (draft, published)	—

4 Giao diện:

- Hiển thị ảnh thumbnail preview
 - Có tab “Đăng lên trang chủ” (boolean toggle)
 - Sắp xếp theo ngày đăng
-

4 banners (Banner / Slider chính)

Trường	Kiểu dữ liệu	Ghi chú
id	UUID	—
title	String	Tên banner
image	File	Ảnh banner
link	String (optional)	Liên kết đến trang
order	Integer	Thứ tự hiển thị
device	Enum (desktop, tablet, mobile)	—
is_active	Boolean	Ẩn/hiện

Giao diện:

- Dạng **Drag & Drop sortable list**
 - Preview ảnh trực tiếp
 - Có tab chọn thiết bị hiển thị (PC/Mobile)
-

5 homepage_sections (Cấu hình trang chủ)

Trường	Kiểu dữ liệu	Ghi chú
id	UUID	—
section_name	Enum (hero, featured, news, archive, contact)	—
is_visible	Boolean	Ẩn/hiện
order	Number	Thứ tự

Trường	Kiểu dữ liệu	Ghi chú
layout_type	Enum (grid, carousel, list)	—
bg_color	String (HEX)	Màu nền
highlight_color	String (HEX)	Màu tiêu đề

■ Giao diện:

- Dạng bảng có drag & drop để đổi thứ tự section
- Toggle hiển thị section
- Chọn màu nền bằng color picker
- Không cần biết code → có thể tự thay đổi bố cục trang chủ

6 pages (Trang công khai tĩnh)

Trường	Kiểu dữ liệu	Ghi chú
id	UUID	—
title	String	Tiêu đề
slug	String	URL (ví dụ /about)
content	Rich Text	Nội dung
cover_image	File	Ảnh minh họa
show_in_menu	Boolean	Hiển thị trong menu
status	Enum (draft, published)	—

■ Giao diện:

- Rich text editor (dễ nhập nội dung)
 - Xem preview ngay trong CMS
 - Có nút bật/tắt “Hiển thị trên menu”
-

7 menus (Điều hướng / Navigation)

Trường Kiểu dữ liệu Ghi chú

id	UUID	—
label	String	Tên hiển thị
link	String	URL hoặc chọn page
parent_id	Relation (self)	Menu cha
order	Number	Thứ tự
is_active	Boolean	Hiển thị/ẩn

Giao diện:

- Drag & drop tạo cấu trúc menu con
 - Xem trước cây menu trực tiếp
 - Dễ thao tác như trong WordPress
-

8 settings (Cấu hình chung)

Trường	Kiểu dữ liệu	Ghi chú
site_name	String	Tên website
logo	File	Logo
favicon	File	Icon
primary_color	Color	Màu chủ đạo
secondary_color	Color	Màu phụ
footer_text	Text	Thông tin bản quyền
contact_email	Email	—
analytics_id	String	Google Analytics ID
social_links	JSON	Facebook / YouTube / ORCID

■ Giao diện:

- Tab “Giao diện”, “Liên hệ”, “Mạng xã hội”
- Color picker
- Preview thay đổi ngay trong CMS

🧠 IV. PHÂN QUYỀN NGƯỜI DÙNG (Directus Roles)

Vai trò	Quyền
 Admin	Toàn quyền, cấu hình CMS
 Tổng biên tập	Duyệt bài, xuất bản, chỉnh sửa issue

Vai trò	Quyền
 Tác giả	Tạo/sửa bài của mình
 Phản biện viên	Xem bài được phân công
 Nhân viên biên tập	Quản lý banner, tin tức, menu
 Khách / Công chúng	Chỉ xem nội dung công khai

V. GIAO DIỆN QUẢN TRỊ TỔNG QUAN (Dễ dùng, không cần code)

◆ **Sidebar menu CMS:**

 Trang chủ

 Bài báo

 Tin tức

 Số báo

 Banner

 Trang tĩnh

 Menu

 Cài đặt

◆ **Trang chính “Dashboard”:**

- Biểu đồ tổng bài theo trạng thái
- Tin gần đây

- Số báo hiện tại
 - Nút “+ Tạo bài mới”, “+ Tạo tin tức”
- ◆ **Form nhập liệu:**
- Chia thành **Tab** (Thông tin – Nội dung – Hình ảnh – Xuất bản)
 - Có **Tooltip hướng dẫn** từng trường
 - Có **Preview** cho RichText
-

VI. KẾT NỐI FRONTEND (Next.js)

Directus hoặc Strapi đều sinh API tự động:

Ví dụ Directus REST:

```
const res = await fetch(`${
  process.env.DIRECTUS_URL
}/items/news?filter[status][_eq]=published`)
```

```
const data = await res.json()
```

hoặc GraphQL:

```
query {  
  news(filter: { status: { _eq: "published" } }) {  
    title  
    thumbnail { id }  
    published_at  
  }  
}
```

}

VII. LỢI THẾ HỆ THỐNG NÀY

Tiêu chí	Đạt được
 Dễ dùng	Drag & drop, form có hướng dẫn
 Không cần code	Tất cả quản trị nội dung trong giao diện
 Xem trước nội dung	Preview bài & banner ngay trong CMS
 Kết nối linh hoạt	API cho frontend Next.js
 Chuẩn hóa quy trình xuất bản Từ gửi bài → phản biện → duyệt	
 Tùy chỉnh dễ dàng	Thêm trường mới không ảnh hưởng frontend

VIII. LỘ TRÌNH TRIỂN KHAI THỰC TẾ

Giai đoạn	Thời gian Kết quả
1. Cài đặt Directus	1 ngày CMS trống chạy tại /admin
2. Tạo cấu trúc bảng	2–3 ngày Có đủ 8 bảng quản lý
3. Phân quyền người dùng	1 ngày 5 nhóm quyền hoạt động
4. Kết nối với Next.js frontend	3–5 ngày Website hiển thị dữ liệu CMS
5. Thiết kế giao diện CMS	2–3 ngày Dashboard + form đẹp, dễ hiểu

Giai đoạn	Thời gian Kết quả
6. Kiểm thử & triển khai	1–2 ngày CMS hoàn chỉnh cho biên tập viên

❖ KẾT LUẬN

Phương án tốt nhất cho bạn là:

- 🧠 Dùng **Directus CMS** (hoặc Strapi nếu đội có lập trình viên backend)
 - Xây **8 bảng dữ liệu chuẩn hóa** như trên
 - Giao diện quản trị có tab, tooltip, drag & drop
 - Biên tập viên không cần code, chỉ cần thao tác web
 - Frontend Next.js tự động cập nhật nội dung qua API
-

👉 Tôi có thể giúp bạn **tạo file cấu hình Directus hoàn chỉnh (.yaml)** hoặc **mẫu Strapi model JSON** cho các bảng này — để bạn import vào CMS trực tiếp (chạy ngay).

Bạn muốn tôi chuẩn bị mẫu cho **Directus** hay **Strapi**?

Bạn đã nói:

Nếu làm theo phương án này có phải viết code lại các chức năng khác của phần mềm không? Hay chỉ phát triển thêm

Code GPT đã nói:

Câu hỏi này rất chính xác 🙌 — vì đây là **điểm mấu chốt khi chuyển từ hệ thống tự code sang mô hình CMS hiện đại (Directus / Strapi)**.

Câu trả lời ngắn gọn là:

 Không cần viết lại toàn bộ phần mềm

 Chỉ cần phát triển thêm lớp kết nối (API integration) và điều chỉnh logic lấy dữ liệu

 Cấu trúc, chức năng, và giao diện cũ vẫn giữ nguyên (Header, Footer, Menu, Body layout...)

I. Tư duy tổng thể

Hiện tại phần mềm của bạn (Next.js) đang có:

-  **Frontend (giao diện hiển thị)** — viết bằng Next.js, Tailwind, React
-  **Chức năng xử lý (menu, bài viết, tìm kiếm, dashboard, v.v.)**
-  **Không có CMS trực quan cho biên tập viên** (mọi dữ liệu thường viết cứng, hoặc nhập bằng JSON/DB thủ công)

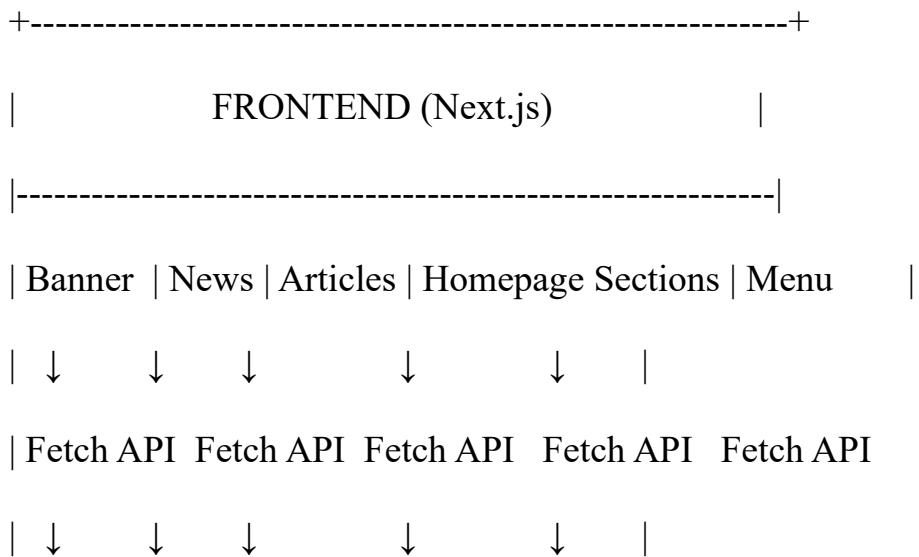
Khi tích hợp Directus hoặc Strapi, ta chỉ **thay nguồn dữ liệu (data source)**, chứ **không thay giao diện hoặc kiến trúc hiện tại**.

II. So sánh trước và sau tích hợp CMS

Thành phần	Hiện tại	Sau khi thêm CMS
Banner / Menu	Viết trong code hoặc Lưu trong bảng banners, menus tệp JSON	
Tin tức / Bài báo	Lưu trong DB hoặc file	Quản lý bằng CMS UI, lấy qua API

Thành phần	Hiện tại	Sau khi thêm CMS
Trang chủ / Section layout	Cố định trong code	Cấu hình từ bảng homepage_sections
Footer / Header	Giữ nguyên	Không thay đổi (vẫn cố định)
Dashboard (Admin / Author)	Đang có code	Giữ nguyên, có thể gọi API CMS để lấy dữ liệu thật
Phân quyền người dùng	Dựa vào logic trong Kết hợp CMS roles hoặc đồng bộ tài khoản	
Search / Filter / Analytics	Giữ nguyên	Có thể kết hợp CMS query cho dữ liệu động
	Tóm tắt:	
	Bạn không cần viết lại — chỉ cần thay vì fetch() dữ liệu từ DB cũ, giờ fetch() từ CMS API.	

✳️ III. Mô hình mới sau khi tích hợp



+----- CMS (Directus/Strapi) -----+

| Articles | News | Banners | Pages | Menus | Settings |

| Giao diện quản trị trực quan (no-code) |

+-----+

Khi biên tập viên thêm nội dung trong CMS → website tự động cập nhật thông qua API.

🧠 IV. Cụ thể hóa: bạn chỉ cần phát triển thêm các phần sau

Mục tiêu	Cách thực hiện	Code lại?
🎯 Lấy dữ liệu từ CMS	Thay import data from ... bằng fetch('/api/items/...')	✗ Không
📰 Hiển thị bài báo, tin tức	Map dữ liệu CMS vào component hiện tại	✗ Không
📋 Quản lý banner / menu động	Gọi API banners, menus	✗ Không
⚙️ Cấu hình trang chủ (section)	Đọc từ homepage_sections	✗ Không
👤 Dashboard hiển thị dữ liệu CMS	Thêm route /dashboard/admin → gọi API CMS	✓ Phát triển thêm
🔒 Đồng bộ tài khoản CMS và hệ thống hiện có	Tùy chọn: sử dụng JWT hoặc API Key	✓ Phát triển thêm nếu cần

⚡ V. Đánh giá tác động kỹ thuật

Hạng mục	Ảnh hưởng	Ghi chú
Frontend layout	Không thay đổi	Banner/Footer cố định
Giao diện hiển thị	Không thay đổi	Vẫn dùng các component hiện tại
CSDL cũ (nếu có)	Có thể ngừng sử dụng	CMS thay thế
Logic chức năng	Giữ nguyên	Chỉ đổi nguồn dữ liệu
Dashboard quản lý	Có thể kết nối CMS	Không cần viết lại
Phân quyền user	Có thể tách biệt (CMS vs Web)	Có thể đồng bộ về sau

🧱 VI. Cách triển khai thực tế từng giai đoạn

Giai đoạn	Hành động	Mục tiêu
GĐ 1 – Cài CMS	Cài Directus/Strapi, tạo bảng dữ liệu theo CMS mẫu tôi đã đề xuất	hoạt động
GĐ 2 – Kết nối API	Thay dữ liệu tĩnh trong Next.js bằng fetch CMS	Hiển thị động
GĐ 3 – Bổ sung config file	cmsConfig.ts chứa URL, token	Dễ bảo trì

Giai đoạn	Hành động	Mục tiêu
GĐ 4 – Kiểm thử	So sánh giao diện trước/sau	Không đổi giao diện
GĐ 5 – Mở rộng dashboard	Cho phép admin duyệt bài, xem thống kê Tích hợp CMS	2 chiều

🧠 VII. Ví dụ thực tế

Trước:

```
const news = [
  { title: "Tin 1", summary: "..." },
  { title: "Tin 2", summary: "..." }
]
```

Sau:

```
async function getNews() {
  const res = await fetch(`${process.env.CMS_URL}/items/news?filter[status][_eq]=published`)
  const data = await res.json()
  return data.data
}
```

→ UI hiển thị nguyên, chỉ đổi nguồn dữ liệu.

VIII. Khi nào cần viết thêm code mới

Chỉ khi bạn muốn **thêm các tính năng nâng cao**:

Tính năng	Có cần viết thêm?	Ghi chú
 Trình soạn thảo bài báo online	<input checked="" type="checkbox"/> Có (trình editor riêng)	
 Dashboard thống kê	<input checked="" type="checkbox"/> Có (nhưng chỉ frontend)	
 Hệ thống phản biện (Peer review)	<input checked="" type="checkbox"/> Có, dùng CMS + API	
 AI hỗ trợ biên tập	<input checked="" type="checkbox"/> Có	
 Ứng dụng di động quản trị	<input checked="" type="checkbox"/> Có	

Tất cả phần giao diện hiển thị web công khai vẫn **tái sử dụng 100%**.

IX. Kết luận

Câu hỏi	Trả lời ngắn gọn
Có phải viết lại phần mềm không?	 Không
Cần làm gì thêm?	<input checked="" type="checkbox"/> Phát triển thêm kết nối và cấu hình API
Giao diện có thay đổi không?	 Giữ nguyên
Có thể triển khai song song không?	<input checked="" type="checkbox"/> Có (CMS hoạt động riêng, web chỉ lấy API)
Phức tạp không?	 Trung bình (2–3 tuần cho tích hợp hoàn thiện)

X. Gợi ý triển khai thực tế

Thành phần	Giải pháp
CMS	Directus (no-code, UI mạnh)
Frontend	Next.js + Tailwind
Kết nối	REST API hoặc GraphQL
Auth đồng bộ (tùy chọn)	JWT giữa web và CMS
Triển khai	CMS: VPS riêng / Docker container; Website: Vercel / VPS
Quản trị viên không biết code	Sử dụng UI CMS hoàn toàn

Tóm lại:

Bạn không cần viết lại hệ thống. Chỉ cần **thêm lớp kết nối CMS → Frontend** là có thể quản lý nội dung dễ dàng, linh hoạt, không thay đổi giao diện, và vẫn tuân thủ quy định banner/footer.