

Trang 5.4: Nâng cấp và cải tiến mã nguồn hiện có với /enhance.

1. Mục đích và triết lý

`/enhance` là công cụ của vòng lặp phát triển hàng ngày — dùng để **thêm tính năng mới vào hệ thống đang chạy** mà không làm gián đoạn hoặc phá vỡ những gì đang hoạt động.

Triết lý cốt lõi của `/enhance` là "**phẫu thuật chính xác**" (**surgical precision**): chỉ thay đổi đúng những gì cần thiết, không viết lại, không refactor không liên quan.

“ Khi nào `/enhance` vs `/create` ?

- `/enhance`: "Dự án đang chạy, tôi muốn thêm tính năng X" → ENHANCE
- `/create`: "Tôi chưa có gì, tôi muốn xây dựng tính năng X từ đầu" → CREATE

2. Điều đặc biệt: Phân tích code trước khi thay đổi

Khác với `/create` (bắt đầu từ trang trắng), `/enhance` **bắt buộc phải đọc và hiểu codebase hiện tại** trước khi đề xuất bất kỳ thay đổi nào.

```
/enhance [yêu cầu]
```



☐ AI đọc cấu trúc dự án hiện tại

|

▼

☐ Phân tích các file liên quan đến yêu cầu

|

▼

☐ Lập kế hoạch thay đổi tối thiểu cần thiết

|

▼

☐ Đề xuất plan → Bạn phê duyệt → AI triển khai

|

▼

☐ Chạy kiểm tra để xác nhận không có regression

3. Phương pháp "Vertical Slice" (Lát cắt dọc)

Khi thêm tính năng mới, `/enhance` áp dụng phương pháp **Vertical Slice** — triển khai đầy đủ một tính năng theo chiều dọc từ database đến UI, thay vì làm từng lớp riêng lẻ.

Ví dụ: Thêm tính năng "Bình luận bài viết"

Vertical Slice (☐ cách `/enhance` làm):

├ Database: Bảng comments + migration

├ Backend: API `/comments` (GET, POST, DELETE)

├ Frontend: Component `CommentSection` + `CommentItem`

└ Testing: Unit test + Integration test

Thay vì (☐ cách dễ gây lỗi):

├ Tuần 1: Xong hết phần database

├ Tuần 2: Xong hết phần backend

└ Tuần 3: Xong hết phần frontend

Ưu điểm: Bạn có thể test và dùng tính năng ngay khi một Vertical Slice hoàn thành, không cần đợi toàn bộ.

4. Quy trình thực hiện

Bước 1: Mô tả tính năng cần thêm

```
bash
/enhance [mô tả tính năng + ràng buộc nếu có]
```

Bước 2: AI phân tích codebase

AI sẽ đọc:

- package.json để biết tech stack và thư viện.
- Cấu trúc thư mục `src/` để hiểu tổ chức code.
- Các file liên quan đến yêu cầu của bạn.

Bước 3: AI đề xuất kế hoạch thay đổi

☐ Kế hoạch thay đổi:

Tìm thấy:

- Framework: Next.js 14 với App Router
- Đang dùng Tailwind CSS
- Chưa có cấu hình dark mode

Thay đổi dự kiến:

1. Cập nhật tailwind.config.ts (thêm darkMode: 'class')
2. Tạo component ThemeProvider (dùng next-themes)
3. Tạo component ThemeToggle
4. Cập nhật app/layout.tsx để bọc trong ThemeProvider
5. Thêm class dark: vào các component chính

Thư viện cần cài: next-themes

Không ảnh hưởng đến: logic nghiệp vụ, database, API

Tiến hành? (Y/N)

Bước 4: Phê duyệt và triển khai

Sau khi bạn xác nhận, AI triển khai từng thay đổi và chạy kiểm tra.

5. Ví dụ phiên làm việc thực tế

Ví dụ 1: Thêm dark mode

Bạn: /enhance thêm dark mode toggle vào ứng dụng Next.js của tôi, đảm bảo lưu lại tùy chọn của người dùng

AI: Đang phân tích codebase...

Kết quả phân tích:

- Next.js 14 + App Router
- Tailwind CSS (chưa có darkMode config)
- localStorage có thể dùng (không có SSR conflict nếu dùng next-themes)

Kế hoạch (5 thay đổi):

1. npm install next-themes
2. Cập nhật tailwind.config.ts
3. Tạo ThemeProvider component
4. Tạo ThemeToggle button (sun/moon icon)
5. Cập nhật layout.tsx

Tiến hành? (Y/N)

Bạn: Y

AI:

- Đã cài next-themes
- Đã cập nhật tailwind.config.ts
- Đã tạo components/ThemeProvider.tsx
- Đã tạo components/ThemeToggle.tsx
- Đã cập nhật app/layout.tsx

Chạy `/preview start` để xem kết quả.

Tùy chọn dark/light mode được lưu vào localStorage tự động.

Ví dụ 2: Thêm tính năng dashboard analytics

Bạn: /enhance thêm dashboard analytics cho admin.

Cần: biểu đồ doanh thu 30 ngày, top 5 sản phẩm bán chạy, số lượng đơn hàng theo trạng thái

Requirements:

- Dùng Recharts cho biểu đồ
- API endpoint riêng để aggregate data
- Chỉ admin mới xem được

AI: Đang phân tích codebase...

Kết quả phân tích:

- Tìm thấy role-based auth trong middleware.ts
- Đang dùng Prisma + PostgreSQL
- Chưa có Recharts

☐ Kế hoạch (Vertical Slice):

Database: Không cần migration mới (aggregate từ bảng orders)

Backend: GET /api/admin/analytics (3 queries tổng hợp)

Frontend: Trang /admin/analytics với 3 chart components

Testing: Test API endpoint + mock data

Tiến hành? (Y/N)

6. Bảo vệ code hiện có — Các nguyên tắc an toàn

`/enhance` theo các nguyên tắc sau để không làm hỏng hệ thống:

Nguyên tắc	Diễn giải
Thay đổi tối thiểu	Chỉ tạo/sửa file thực sự cần thiết
Cô lập tính năng	Tính năng mới trong thư mục/module riêng
Không refactor ngoài phạm vi	Không sửa những gì không liên quan đến yêu cầu
Regression test	Chạy test hiện có sau mỗi thay đổi
Phê duyệt trước khi làm	Luôn trình bày kế hoạch để bạn xem xét trước

7. Mẹo để `/enhance` hoạt động hiệu quả

Kiểm tra dependencies trước

"Trước khi thêm thư viện XYZ, hãy kiểm tra xem nó có tương thích với phiên bản Node.js/React hiện tại không"

Yêu cầu xem kế hoạch trước khi làm

Luôn đọc kỹ kế hoạch thay đổi trước khi gõ Y. Đặc biệt chú ý:

- Số lượng file bị thay đổi có hợp lý không?
- Có file nào "không liên quan" cũng bị sửa không?

Yêu cầu test cụ thể

```
/enhance thêm tính năng export CSV,  
và viết test để đảm bảo CSV output đúng định dạng
```

Dùng với `/test` sau mỗi lần enhance

```
bash  
# Sau khi enhance xong  
/test # Chạy toàn bộ test suite để kiểm tra regression
```

Phiên bản #2

Được tạo 2026-03-04 06:12:05 UTC bởi Nam Đặng

Được cập nhật 2026-03-04 09:31:54 UTC bởi Nam Đặng