



CLAUDE CODE

15 Prompt

Vibe Coding

prompt chính xác cho từng giai đoạn của dự án



01

Viết một bản PRD đầy đủ

Viết một bản PRD hoàn chỉnh cho tính năng dưới đây.

Tên năng: [TÊN NĂNG]
Mục đích: [MỤC ĐÍCH]
Stack: [STACK CỦA BẠN]

Bao gồm:

- Mô tả cần giải quyết - success metrics
- User story và acceptance criteria
- Scope: cái gì làm vào, cái gì thì không
- Các thay đổi về data model
- Edge case - trạng thái lỗi
- Câu hỏi mở cần tôi trả lời

ĐIỀU 2: Trạng thái, không làm nạn.

Lưu ý: `checklist(features)` để mọi prompt sau đều tham chiếu được.



02

Tạo file CLAUDE.md của bạn

Đọc toàn bộ codebase này, rồi tạo một file CLAUDE.md.

Bao gồm:

- Mô tả dự án là gì, giải quyết 2 dòng
- Tên stack - những version quan trọng
- Commands: [DEV / BUILD / TEST / LINT]
- Kiểm tra: thứ gì nên ở đâu và vì sao
- Code convention theo sự được trong code
- Luật công: [LIỆU KHÔNG ĐƯỢC THA] - thứ không bao giờ được dùng nếu chưa hỏi
- Các lỗi mà một engineer mới sẽ dính ngay tuần đầu

Viết luật viết đúng các mẫu liên quan, không chung chung - chỉ những gì đúng với team này. Nếu không hiểu về một luật, hãy hỏi tôi thay vì tự bịa ra.



03

Chế độ Ultra Plan

Vào plan mode, không viết code với.

Mô tả dự án: [MÔ TẢ DỰ ÁN]
Mục đích: [MỤC ĐÍCH] / [STACK] / [VÙNG CẬP]

Định dạng spec:

- Đầu tiên tôi sẽ mô tả nhiệm vụ này phân tích, liệt kê nên 3 dòng mô tả chung chung
- Về ra hình và liên hệ nó với hình vẽ trong mind
- Mô tả 3-5 hướng làm việc trong mind: mô tả ra, phân tích, phân tích
- Chọn một hướng và giải thích trong 1 dòng
- Các mô tả trong các bước để mô tả các bước trong từng bước một, mỗi bước có cách check riêng
- Liệt kê ra 1-2 - các rủi ro chính xác cho từng cái
- Đánh dấu một thứ được tôi [LƯU Ý / REMARKS / GHI CHÚ] để tôi duyệt riêng

Nếu đúng lại, cho tôi xem kế hoạch và đợi tôi duyệt trước khi dùng vào một file nào.



04

Phát triển theo Spec

Chúng ta viết spec trước, viết spec cho: [TÊN NĂNG]

Mô tả: [MÔ TẢ DỰ ÁN] / [STACK] / [VÙNG CẬP]

Định dạng spec:

- Mô tả: given / when / then, một trường hợp - happy path, edge case, trạng thái lỗi
- API contract: input, output, bản dạng lỗi, status code
- 00 Lưu: thay đổi schema - migration cần thiết
- Trạng thái lỗi: loading, error, success
- Non-pool: những gì cần làm để làm lại
- Acceptance checklist tôi viết được từng dòng

Sau khi tôi duyệt xong, hãy làm đúng như spec. Nếu thực tế bước phải làm, hãy dùng, cập nhật spec, xin tôi đồng ý, rồi mới đi tiếp. Spec mới là nguồn chân lý, không phải code.



05

Design Brief UI & UX đầy đủ

Tạo một design brief UI/UX đầy đủ cho: [TÊN NĂNG/HƯỚNG ĐIỂM]

Mô tả: [MÔ TẢ DỰ ÁN]
Thương hiệu: [THƯƠNG HIỆU] / [TÊN] / [MÀU SẮC]

Bao gồm:

- Mô tả trình người dùng đi qua flow, từng bước một
- Layout từng màn: phân cấp, spacing, alignment
- Danh sách component của một trạng thái - hover, empty, error, loading
- Typography - color system
- Interaction: các gì animate, thời lượng, easing
- UI cho accessibility

Học pattern từ [TÊN SẢN PHẨM KHÁC] để lấy hướng đi. Dùng sau giờ copy chúng.



06

Kế hoạch triển khai

Tạo kế hoạch triển khai cho: [MỤC ĐÍCH] / [MÔ TẢ DỰ ÁN]

Định dạng:

- Mô tả dự án và các bước sau cho app vẫn build và chạy được sau mỗi bước
- Mô tả: file nào sẽ thay, thay đổi những gì, tôi cần chúng bằng cách nào
- Danh sách bước nào cần migration hoặc dependency mới
- Đặt phần rủi ro và nó sẽ mất bao lâu
- Ước lượng mỗi bước: [H / M / S]

Viết ra một chuỗi bước được đánh số để tôi chạy từng bước một. Đợi tôi nói "go" giữa các bước.



07

Kết nối một MCP Server

Dùng một MCP server cho: [MỤC ĐÍCH] / [API]

Tôi cần nó làm được: [MÔ TẢ CÔNG VIỆC CẦN LÀM]

- Kiểm tra xem đã có server chính thức hoặc được duy trì tốt chưa - nếu có nguồn
- Mô tả: cấu hình của chính xác - config, env, scope trong project này
- Mô tả: dùng một cái bằng MCP SDK - tools, auth, xử lý lỗi, response có kiểu
- Chỉ thêm những thứ tôi chưa biết: [MÔ TẢ CÔNG VIỆC]
- Trước secret của [TÊN API], không bao giờ hardcode hay trong file config
- Kiểm tra xem nó có gì với một thứ đầu cuối, cho tôi xem output
- Chỉ cho tôi một trong 1 dòng để các session sau biết khi nào cần dùng tôi



08

Kết nối Database của bạn

Kết nối app này với: [DATABASE] / [MÔ TẢ DỰ ÁN]

- Chọn client phù hợp với stack này, giải thích trong 1 dòng
- Ưu tiên: kết nối, tên vào .env.example, không bao giờ commit giá trị thật
- Schema: bằng cho [CÁC THỰC THỂ], xem kiểu dữ liệu, quan hệ, và index cho [TÊN SẢN PHẨM]
- Tạo - copy migration, và cho tôi xem cách rollback từng cái
- Mô tả bằng một query hoặc 1-2 câu - những cái nào cần, không component
- Access rule / row-level security nếu đây là [MÔ TẢ DỰ ÁN]
- Connection pooling nếu chúng ta chạy serverless

Nếu đúng như: seed một dòng, đọc lại, cho tôi xem output.



09

Tìm lỗi hỏng bảo mật

Đọc toàn bộ codebase này để tìm lỗi hỏng bảo mật.

Mô tả dự án: [MÔ TẢ DỰ ÁN] / [STACK] / [VÙNG CẬP]

Vùng tập trung: [AUTH / PAYMENTS / MÔ TẢ DỰ ÁN]

Hiện tại:

- Secret nằm trong code, config hay git history
- Injection: SQL, XSS, command, path traversal
- Auth: secure session, session, session, session, session, session
- 200: user A có thể được quyền của user B không?
- File upload - validate input ở mọi form
- CVE của dependencies - cập nhật và đọc về
- Header lỗi: [MÔ TẢ DỰ ÁN] / [MÔ TẢ DỰ ÁN]
- Thông tin rò rỉ qua error message và log

Hãy phân phát hiện theo các nguyên nhân của file: tên chính xác, sửa ngay các lỗi critical, phân cấp lỗi liệt kê thành ticket vào một bảng công việc.



10

Debug lỗi thật nhanh

Debug lỗi này, không đoán mò.

Mô tả dự án: [MÔ TẢ DỰ ÁN] / [STACK] / [VÙNG CẬP]

Mô tả lỗi: [MÔ TẢ LỖI]

- Đọc stack trace, nó dùng những file liên quan
- Mô tả: mô tả về những gì nó xảy ra trong 1 dòng
- Liệt kê 3 giả thuyết, mỗi giả thuyết có 1 câu
- Chúng ta nên làm gì để kiểm tra từng giả thuyết một - bằng chứng, những câu hỏi
- Mô tả: những nguyên nhân gốc, những dấu hiệu chứng
- Tên công cụ debug trong repo - nếu không được ở đây thì công cụ khác
- Thêm regression test sau nếu nó báo lỗi đi thì test phải fail
- Mô tả: cho tôi trong 2 dòng về sau nó hỏng và vì sao nó hỏng bao giờ hỏng

Viết này ra.



11

Viết E2E Test cho ứng dụng

Viết Playwright E2E test cho: [TÊN DỰ ÁN]

Stack: [STACK CỦA BẠN] / [MÔ TẢ DỰ ÁN] / [MÔ TẢ DỰ ÁN]

Bao gồm:

- Mô tả: mô tả về ứng dụng, [MÔ TẢ DỰ ÁN] / [MÔ TẢ DỰ ÁN] / [MÔ TẢ DỰ ÁN]
- Test case người dùng nhìn thấy, không test chỉ viết code
- Locator: dùng rule và locator, dùng bao giờ dùng locator CSS để vẽ
- Mô tả: mô tả về ứng dụng, [MÔ TẢ DỰ ÁN] / [MÔ TẢ DỰ ÁN] / [MÔ TẢ DỰ ÁN]
- Test phải đơn giản - chạy thử tự nào cũng được, không dùng chung state, mỗi test tự seed dữ liệu
- Headless trên CI, headless ở đây local để debug
- Có setup screenshot - trace khi fail

Chạy cả test, cho tôi xem kết quả, sửa những gì fail, và nói rõ bộ test vẫn được ghi được gì.



12

Dọn sạch Dead Code

Tin vào xác dead code trong repo này.

Mô tả dự án: [MÔ TẢ DỰ ÁN] / [MÔ TẢ DỰ ÁN] / [MÔ TẢ DỰ ÁN]

Bao gồm:

- Export, component, hook, utility không ai dùng
- Mô tả: mô tả về ứng dụng, [MÔ TẢ DỰ ÁN] / [MÔ TẢ DỰ ÁN] / [MÔ TẢ DỰ ÁN]
- Dependency trong package.json không ai import
- Feature flag có bị bỏ quên hoặc luôn tắt
- Logic trong file nào là nên giữ lại
- Class CSS một vài asset không dùng

Tên kiểm tra xem trước khi xóa - dynamic import và tham chiếu bằng chuỗi cộng tính. Xóa theo từng commit nhỏ, chạy [BUILD] / [TEST] sau mỗi commit, rồi báo tổng số dòng đã xóa cùng những chỗ nào chưa được 100%.



13

Viết Git Commit gọn gàng

Commit cho thay đổi ở stage một cách rõ ràng.

Mô tả dự án: [MÔ TẢ DỰ ÁN] / [MÔ TẢ DỰ ÁN] / [MÔ TẢ DỰ ÁN]

Định dạng:

- Tên các thay đổi chỉ những liên quan tham commit trong
- Tên động: [MÔ TẢ DỰ ÁN] / [MÔ TẢ DỰ ÁN] / [MÔ TẢ DỰ ÁN]
- Feat / fix / refactor / chore / docs / test
- Subject: mô tả rõ ràng, dùng câu mệnh lệnh
- Phân loại: mô tả rõ ràng, mô tả rõ ràng
- Thêm commit: [MÔ TẢ DỰ ÁN]
- Dùng bao giờ refactor với thay đổi hàm vì trong cùng một commit
- Mô tả: mô tả về ứng dụng, [MÔ TẢ DỰ ÁN] / [MÔ TẢ DỰ ÁN] / [MÔ TẢ DỰ ÁN]

Cho tôi xem kế hoạch - file nào vào commit nào - nội dung message - trước khi commit bắt cứ thứ gì. Mỗi commit từng cái một để tôi có thể check commit.



14

Dùng Hooks làm Guardrails

Thiết lập Claude Code hooks làm guardrail ở đây.

Stack: [STACK CỦA BẠN] / [MÔ TẢ DỰ ÁN] / [MÔ TẢ DỰ ÁN]

Bao gồm:

- PostCommit: chạy [LINT] / [CHECK] sau mỗi lần sửa file, đẩy lên ngay
- PreCommit: chặn nếu [MÔ TẢ DỰ ÁN] / [MÔ TẢ DỰ ÁN] / [MÔ TẢ DỰ ÁN]
- PrePush: chạy [MÔ TẢ DỰ ÁN] / [MÔ TẢ DỰ ÁN] / [MÔ TẢ DỰ ÁN]
- Notification: báo tôi bằng [MÔ TẢ DỰ ÁN] / [MÔ TẢ DỰ ÁN]

Viết các script hook - phân loại báo trong settings, tên, có mỗi script dưới 20 dòng và exit như ở bên trong báo rõ ràng để agent biết chính xác phải sửa gì. Mỗi hook hoạt động theo mô tả chính xác và cho tôi xem chúng chạy.



15

Biến một tác vụ thành Skill

Biến tác vụ lặp đi lặp lại này thành một Claude Code skill.

Tác vụ tôi làm hoài: [MÔ TẢ TÁC VỤ] / [MÔ TẢ TÁC VỤ]

Bao gồm:

- Tên: [MÔ TẢ TÁC VỤ] / [MÔ TẢ TÁC VỤ]
- Precondition: name - description về những câu hỏi hay hỏi để kích hoạt nó
- Body: quy trình định sẵn, các quy tắc của tôi, edge case
- Các gì nên hỏi, các gì tự họ ra được
- "trong" thì trong như thế nào

Mô tả: mô tả về ứng dụng, [MÔ TẢ DỰ ÁN] / [MÔ TẢ DỰ ÁN] / [MÔ TẢ DỰ ÁN]



01

Viết một bản PRD đầy đủ

Viết một bản PRD hoàn chỉnh cho tính năng dưới đây.

Tính năng: [MÔ TẢ TÍNH NĂNG]

Người dùng: [AI SẼ DÙNG]

Stack: [STACK CỦA BẠN]

Bao gồm:

- * Vấn đề cần giải quyết + success metrics
- * User story kèm acceptance criteria
- * Scope: cái gì lên v1, cái gì thì không
- * Các thay đổi về data model
- * Edge case + trạng thái lỗi
- * Câu hỏi mở cần tôi trả lời

Giữ dưới 2 trang. Cụ thể, không lan man.

Lưu thành docs/prd-[feature].md để mọi prompt sau đều tham chiếu được.



02

Tạo file **CLAUDE.md** của bạn

Quét toàn bộ codebase này, rồi tạo một file **CLAUDE.md**.

Bao gồm:

- * Dự án này là gì, gói trong 2 dòng
- * Tech stack + những version quan trọng
- * Commands: **[DEV / BUILD / TEST / LINT]**
- * Kiến trúc: thứ gì nằm ở đâu và vì sao
- * Code convention thực sự đọc được trong code
- * Luật cứng: **[ĐIỀU KHÔNG ĐƯỢC PHÁ]** - thứ không bao giờ được đụng nếu chưa hỏi
- * Các bẫy mà một engineer mới sẽ dính ngay tuần đầu

Viết luật dưới dạng câu mệnh lệnh ngắn. Không chung chung - chỉ những gì đúng với repo NÀY. Nếu không chắc về một luật, hãy hỏi tôi thay vì tự bịa ra.



03

Chế độ Ultra Plan

Vào plan mode. KHÔNG viết code vội.

Nhiệm vụ: [DÁN NHIỆM VỤ]

Ràng buộc: [DEADLINE / STACK / VÙNG CẤM]

1. Đọc mọi file mà nhiệm vụ này chạm tới, liệt kê kèm 1 dòng mô tả chúng đang làm gì
2. Vẽ ra hành vi hiện tại so với hành vi mong muốn
3. Đề xuất 2-3 hướng làm kèm tradeoff thật: độ phức tạp, rủi ro, phạm vi ảnh hưởng
4. Chọn một hướng và giải thích trong 3 dòng
5. Chia nhỏ thành các bước đủ nhỏ để kiểm chứng từng bước một, mỗi bước có cách check riêng
6. Liệt kê rủi ro + cách rollback chính xác cho từng cái
7. Đánh dấu mọi thứ động tới [AUTH / PAYMENTS / PROD DATA] để tôi duyệt riêng

Rồi dừng lại. Cho tôi xem kế hoạch và đợi tôi duyệt trước khi đụng vào bất kỳ file nào.



04

Phát triển theo Spec

Chúng ta viết spec trước. Viết spec cho: [TÍNH NĂNG]

Bối cảnh: [AI DỪNG + VÌ SAO LÀ LÚC NÀY]

Định dạng spec:

- * Hành vi: given / when / then, mọi trường hợp - happy path, edge case, trạng thái lỗi
- * API contract: input, output, hình dạng lỗi, status code
- * Dữ liệu: thay đổi schema + migration cần thiết
- * Trạng thái UI: loading, empty, error, success
- * Non-goal: những gì spec này cố tình bỏ qua
- * Acceptance checklist tôi soát được từng dòng

Sau khi tôi duyệt spec, hãy làm ĐÚNG như spec. Nếu thực tế buộc phải lệch, hãy dừng, cập nhật spec, xin tôi đồng ý, rồi mới đi tiếp. Spec mới là nguồn chân lý, không phải code.



05

Design Brief UI & UX đầy đủ

Tạo một design brief UI/UX đầy đủ cho: [MÀN HÌNH HOẶC FLOW]

Đối tượng: [NGƯỜI DÙNG]

Thương hiệu: [MÀU / FONT / CẢM GIÁC]

Bàn giao:

- * Hành trình người dùng đi qua flow, từng bước một
- * Layout từng màn: phân cấp, spacing, breakpoint
- * Danh sách component kèm mọi trạng thái - hover, empty, error, loading
- * Typography + color token
- * Motion: cái gì animate, thời lượng, easing
- * Ghi chú accessibility

Học pattern từ [2-3 SẢN PHẨM BẠN THÍCH] để lấy hướng đi. Đừng bao giờ copy chúng.



06

Kế hoạch triển khai

Tạo kế hoạch triển khai cho: **[SPEC / PRD ĐÃ DUYỆT]**

Quy tắc:

- * Sắp thứ tự các bước sao cho app vẫn build và chạy được sau MỖI bước
- * Mỗi bước: file nào bị chạm, thay đổi những gì, tôi kiểm chứng bằng cách nào
- * Đánh dấu bước nào cần migration hoặc dependency mới
- * Đặt phần rủi ro và mơ hồ nhất lên trước
- * Ước lượng mỗi bước: **[S / M / L]**

Xuất ra một chuỗi build được đánh số để tôi chạy từng bước một. Đợi tôi nói "go" giữa các bước.



07

Kết nối một MCP Server

Dựng một MCP server cho: **[DỊCH VỤ / API]**

Tôi cần nó làm được: **[CÔNG VIỆC CẦN LÀM]**

1. Kiểm tra xem đã có server chính thức hoặc được duy trì tốt chưa - nêu rõ nguồn
2. Nếu có: câu lệnh cài chính xác + config `.mcp.json`, scope trong project này
3. Nếu chưa: dựng một cái bằng MCP SDK - tools, auth, xử lý lỗi, response có kiểu
4. CHỈ thêm những tool tôi thực sự dùng: **[DANH SÁCH]**
5. Truyền secret qua **[ENV VARS]**, không bao giờ hardcode key trong file config
6. Kiểm tra kết nối và gọi thử một tool đầu-cuối, cho tôi xem output
7. Ghi chú mỗi tool trong 1 dòng để các session sau biết khi nào cần dùng tới



08

Kết nối Database của bạn

Kết nối app này với **[POSTGRES / SUPABASE / DB CỦA BẠN]**.

- * Chọn client hợp với stack này, giải thích trong 1 dòng
- * Env var: đặt tên, thêm vào `.env.example`, không bao giờ commit giá trị thật
- * Schema: bảng cho **[CÁC THỰC THỂ]** kèm kiểu dữ liệu, quan hệ, và index cho **[TRUY VẤN NÓNG]**
- * Tạo + chạy migration, và cho tôi xem cách rollback từng cái
- * Mỗi bảng một query helper có kiểu - không rải raw SQL khắp component
- * Access rule / row-level security nếu đây là **[MULTI-TENANT]**
- * Connection pooling nếu chúng ta chạy serverless

Rồi chúng mình: seed một dòng, đọc lại, cho tôi xem output.



09

Tìm lỗ hổng bảo mật

Audit codebase này để tìm lỗ hổng bảo mật.
Soi nó như thể bạn đang tìm đường chui vào.

Vùng tập trung: **[AUTH / PAYMENTS / DỮ LIỆU NGƯỜI DÙNG]**

Kiểm tra:

- * Secret nằm trong code, config hay git history
- * Injection: SQL, XSS, command, path traversal
- * Auth: route thiếu kiểm tra, session yếu, redirect hỏng
- * IDOR: user A có đọc được dữ liệu của user B không?
- * File upload + validate input ở mọi form
- * CVE của dependency - chạy audit và đọc kỹ
- * Rate limit trên **[ENDPOINT TỐN TÀI NGUYÊN]**
- * Thông tin rò rỉ qua error message và log

Xếp hạng phát hiện theo mức nghiêm trọng kèm file:line chính xác, sửa ngay các lỗi critical, phần còn lại liệt kê thành ticket kèm ước lượng công sức.



10

Debug lỗi thật nhanh

Debug lỗi này. KHÔNG đoán mò.

Lỗi: [DÁN TOÀN BỘ LỖI + STACK TRACE]

Khi nào xảy ra: [CÁC BƯỚC TÁI HIỆN]

1. Đọc stack trace, mở đúng những file liên quan
2. Nêu hành vi mong đợi so với thực tế trong 1 dòng
3. Liệt kê 3 giả thuyết, xếp theo khả năng xảy ra
4. Chứng minh hoặc loại từng cái bằng log hay một test nhỏ - bằng chứng, không cảm tính
5. Sửa nguyên nhân gốc, không sửa triệu chứng
6. Tìm cùng pattern đó trong repo - nếu hỏng được ở đây thì cũng hỏng ở chỗ khác
7. Thêm regression test mà nếu bỏ bản fix đi thì test phải fail
8. Nói cho tôi trong 2 dòng vì sao nó hỏng và vì sao nó không bao giờ hỏng kiểu này nữa



11

Viết E2E Test cho ứng dụng

Viết Playwright E2E test cho: **[FLOW]**

Stack: **[STACK CỦA BẠN]** CI: **[GITHUB ACTIONS / KHÁC]**

- * Ưu tiên luồng ra tiền trước: **[SIGNUP / CHECKOUT / HÀNH ĐỘNG CỐT LỖI]**
- * Test cái người dùng nhìn thấy, không test chi tiết cài đặt
- * Selector: dùng role và label, đừng bao giờ dùng chuỗi CSS dễ vỡ
- * Mỗi flow một unhappy path: input sai, mất mạng, session hết hạn
- * Test phải độc lập - chạy thứ tự nào cũng được, không dùng chung state, mỗi test tự seed dữ liệu
- * Headless trên CI, headed ở máy local để debug
- * Chỉ chụp screenshot + trace khi fail

Chạy cả bộ test, cho tôi xem kết quả, sửa những gì fail, và nói rõ bộ test vẫn CHƯA phủ được gì.



12

Dọn sạch Dead Code

Tìm và xóa dead code trong repo này.

Phạm vi: [TOÀN BỘ REPO / THƯ MỤC CỤ THỂ]

- * Export, component, hook, util không ai dùng
- * Nhánh không bao giờ chạy tới + block code bị comment
- * Dependency trong package.json không ai import
- * Feature flag cũ bị kẹt luôn bật hoặc luôn tắt
- * Logic trùng lặp đáng lẽ nên gộp làm một
- * Class CSS chết và asset không dùng

Tìm kiếm xác minh trước MỖI lần xóa - dynamic import và tham chiếu dạng chuỗi cũng tính. Xóa theo từng commit nhỏ, chạy [BUILD + TESTS] sau mỗi commit, rồi báo tổng số dòng đã xóa cùng những chỗ bạn chưa chắc 100%.



13

Viết Git Commit gọn gàng

Commit các thay đổi đã stage một cách tử tế.

Quy ước: **[CONVENTIONAL COMMITS / ĐỊNH DẠNG CỦA BẠN]**

- * Tách các thay đổi không liên quan thành commit riêng
- * Định dạng: type(scope): đã đổi gì và vì sao
feat / fix / refactor / chore / docs / test
- * Subject dưới 50 ký tự, dùng câu mệnh lệnh
- * Phần body giải thích VÌ SAO, wrap ở cột 72
- * Tham chiếu ticket: **[TICKET-ID]**
- * Đừng bao giờ trộn refactor với thay đổi hành vi trong cùng một commit
- * Đừng bao giờ commit **[SECRETS / .ENV / FILE SINH TỰ ĐỘNG]**

Cho tôi xem kế hoạch - file nào vào commit nào + nội dung message - trước khi commit bất cứ thứ gì. Rồi commit từng cái một để tôi có thể chặn giữa chừng.



14

Dùng Hooks làm Guardrails

Thiết lập Claude Code hooks làm guardrail ở đây.

Stack: `[STACK + PACKAGE MANAGER CỦA BẠN]`

- * PostToolUse: chạy `[LINT + TYPECHECK]` sau mỗi lần sửa file, đẩy lỗi ngược lại ngay
- * PreToolUse: chặn sửa `[ĐƯỜNG DẪN ĐƯỢC BẢO VỆ: MIGRATIONS, .ENV, PROD CONFIG]`
- * Stop: chạy `[TEST SUITE]` trước khi kết thúc session
- * Notification: báo tôi bằng `[ÂM THANH / SLACK]` khi cần tôi vào cuộc

Viết các script hook + phần khai báo trong settings.json. Giữ mỗi script dưới 20 dòng và exit khác 0 kèm thông báo rõ ràng để agent biết chính xác phải sửa gì. Rồi kích hoạt từng hook có chủ đích và cho tôi xem chúng chạy.



15

Biến một tác vụ thành Skill

Biến tác vụ lặp đi lặp lại này thành một Claude Code skill.

Tác vụ tôi làm hoài: [MÔ TẢ TÁC VỤ + CÁC BƯỚC]

- * Tạo `.claude/skills/[NAME]/SKILL.md`
- * Frontmatter: name + description kèm đúng những câu tôi hay nói để kích hoạt nó
- * Body: quy trình đánh số, các quy ước của tôi, edge case
- * Cái gì nên hỏi tôi, cái gì tự suy ra được
- * "Xong" thì trông như thế nào

Rồi chạy thử skill trên một ví dụ thật và tinh chỉnh cho tới khi output khớp với cách tôi vẫn làm tay.

