TEDUCO

Công ty TNHH Công nghệ TEDU

BỘ PROMPT KHỞI TẠO DỰ ÁN CHO CLAUDE CODE

Tài liệu nội bộ — quy chuẩn khởi tạo tài liệu quản trị dự án (PRD, Architecture, UX, Deploy, Conventions, Status, Decisions, CLAUDE.md) khi làm việc với Claude Code

Người soạn: Bạch Ngọc Toàn — Founder & CEO, TEDU

3 tháng 8, 2026

Mục lục

Bộ Prompt Khởi Tạo Dự Án Cho Claude Code

Tài liệu này chứa 9 prompt để đưa cho Claude Code, mỗi prompt sinh ra 1 file quản trị dự án. 9 file này liên kết chặt với nhau: PRD là nguồn sự thật về "làm gì", Domain Glossary là "gọi tên mọi khái niệm nghiệp vụ thống nhất ra sao" (để PRD/Architecture/UX/code không mỗi nơi gọi 1 tên khác nhau cho cùng 1 khái niệm), Architecture là "làm bằng gì", UX là "giao diện trông và hoạt động ra sao", Deploy là "đưa lên môi trường thật như thế nào", Conventions là "làm như thế nào" (code), Status là "đang ở đâu", Decisions là "tại sao lại chọn thế" (lịch sử quyết định để tra cứu, tránh hỏi lại/làm lại), và CLAUDE.md là "luật chơi" ràng buộc agent phải đọc và tuân theo các file kia trong mọi task.

Bước 0 — Trước khi chạy bất kỳ prompt nào

0.1 Cấu trúc thư mục: vì sao 8 file nằm trong `docs/` còn CLAUDE.md thì không

8 file (`prd.md`, `domain-glossary.md`, `decisions.md`, `architecture.md`, `ux.md`, `deploy.md`, `conventions.md`, `status.md`) đặt trong thư mục `docs/` ở gốc repo cho gọn, tránh rác gốc repo khi dự án có nhiều tài liệu khác. Riêng `CLAUDE.md` **bắt buộc nằm ở gốc repo**, không được đặt trong `docs/` — vì Claude Code chỉ tự động nạp `CLAUDE.md` khi nó nằm ở gốc repo (hoặc các thư mục cha khi làm việc trong thư mục con), đặt trong `docs/` thì Claude Code sẽ không tự đọc, mọi ràng buộc trong đó vô hiệu mà không có cảnh báo gì.

Tạo cấu trúc trước khi chạy Prompt 1, bằng lệnh trực tiếp:

PROMPT — copy toàn bộ khối bên dưới và dán cho Claude Code

```
mkdir -p docs
```

Hoặc yêu cầu Claude Code tự tạo bằng câu: *"Tạo thư mục `docs/` ở gốc repo nếu chưa có, dùng để chứa các file quản trị dự án."*

0.2 Model nào cho việc nào

Claude Code hiện có các model alias sau (`/model <alias>` để chọn, hoặc gõ `/model` không kèm gì để mở picker):

Alias	Model hiện tại phân giải tới	Dùng khi nào
`opus`	Opus 5	Việc cần suy luận sâu, nhiều bước, ảnh hưởng lâu dài — kiến trúc, quyết định khó, debug root-cause chưa rõ nguyên nhân
`sonnet`	Sonnet 5	Việc code hàng ngày: implement theo pattern đã có sẵn trong conventions/ux/architecture, khối lượng lớn, cần cân bằng tốc độ/chi phí
`haiku`	Haiku 4.5	Việc lặp lại đơn giản, thao tác file hàng loạt, boilerplate, không cần suy luận sâu
`fable`	Claude Fable 5	Việc lớn, mơ hồ, chạy tự động dài hơi (root-cause khó, quyết định kiến trúc lớn) — không phải model mặc định, phải chọn tay
`opusplan`	Opus khi ở plan mode, tự chuyển sang Sonnet khi thực thi	Chế độ lai hợp lý cho phần lớn công việc: để Opus lên kế hoạch, Sonnet viết code — tận dụng được cả hai mà không phải tự tay chuyển model liên tục

Mặc định theo gói: Max/Team Premium/Enterprise trả theo dùng và API mặc định `Opus 5`; Pro/Team Standard mặc định `Sonnet 5`. Xem chi tiết và các thay đổi mới nhất tại https://code.claude.com/docs/en/model-config.

Áp dụng cho bộ prompt trong tài liệu này:

Việc	Model đề xuất	Vì sao
Chạy Prompt 1–9 (sinh 9 file quản trị)	`opus` hoặc `opusplan`	Đây là quyết định nền tảng, sai ở bước này kéo sai toàn bộ pipeline phía sau — đáng đầu tư model mạnh nhất dù chỉ chạy 1 lần cho mỗi file
Implement 1 hạng mục trong checklist của `status.md` (code theo pattern đã định sẵn trong conventions/ux/architecture)	`sonnet` (hoặc để nguyên `opusplan`, nó tự chuyển)	Không cần quyết định kiến trúc mới, chỉ cần thực thi đúng chuẩn đã có — sonnet đủ và rẻ hơn cho khối lượng lớn
Viết test boilerplate, sinh dữ liệu mẫu, thao tác file hàng loạt	`haiku`	Việc cơ học, không cần suy luận
Debug lỗi khó/root-cause chưa rõ, hoặc phát sinh quyết định kiến trúc giữa chừng	`opus` (hoặc `fable` nếu là 1 phiên điều tra dài, nhiều bước)	Cần điều tra sâu trước khi hành động
Deploy/release thật lên production	`opus`	Sai sót ở đây có hậu quả thật ngoài code, ưu tiên cẩn trọng hơn tốc độ

0.3 Effort level nào cho việc nào

Effort level điều khiển mức độ Claude "suy nghĩ" trước khi trả lời — effort cao hơn thì sâu hơn nhưng tốn token/thời gian hơn. Đặt bằng `/effort <level>`, xem effort hiện tại ở status line.

Level	Khi dùng
`low`	Việc ngắn, phạm vi rõ, cần tốc độ, không nhạy cảm về độ chính xác
`medium`	Việc cần tiết kiệm chi phí, chấp nhận đánh đổi 1 phần chất lượng suy luận
`high`	Mặc định, cân bằng tốt cho hầu hết việc code hàng ngày
`xhigh`	Suy luận sâu hơn, tốn token hơn — dùng cho việc khó
`max`	Sâu nhất, dễ "nghĩ quá" (overthinking) ở việc đơn giản — chỉ dùng cho việc thực sự khó, và nên thử trước khi áp dụng rộng
`ultracode`	Không phải effort level thuần túy mà là chế độ của Claude Code: tự lên kế hoạch (dynamic workflow) cho cả 1 phase lớn, chạy effort `xhigh` cho từng bước — hợp khi giao 1 khối việc lớn và để agent tự chia nhỏ thay vì bạn chia tay

Áp dụng cho bộ prompt trong tài liệu này:

- Chạy Prompt 1–9 (sinh file quản trị): `high` hoặc `xhigh` — đây là việc làm 1 lần nhưng ảnh hưởng lâu dài, đáng trả thêm chi phí để suy luận kỹ.
- Implement checklist thường ngày theo `status.md`: giữ `high` (mặc định) là đủ; hạ xuống `medium` nếu là việc lặp lại thuần túy để tiết kiệm.

- Giao nguyên 1 phase lớn cho agent tự chạy dài (nhiều hạng mục liên tiếp trong `status.md`): cân nhắc `ultracode`.
- Debug khó hoặc quyết định kiến trúc phát sinh giữa chừng: `xhigh`, thử `max` nếu vẫn bí.

Không phải model nào cũng hỗ trợ đủ các level — Haiku không có effort; hầu hết Opus/Sonnet/Fable đời mới có đủ `low/medium/high/xhigh/max`, riêng Opus 4.6/Sonnet 4.6 không có `max`. Xem bảng đầy đủ tại link ở mục 0.2.

Thứ tự chạy khuyến nghị

Không chạy tùy ý theo số thứ tự liệt kê ở đề bài, vì hầu hết các file sau cần dữ liệu từ `docs/prd.md`, `docs/domain-glossary.md`, và `docs/architecture.md`. Thứ tự đúng:

1. **docs/prd.md** — cần input là yêu cầu thô của bạn (điền vào mục "Input Requirement" dưới đây)
2. **docs/domain-glossary.md** — đọc docs/prd.md để trích xuất mọi thuật ngữ nghiệp vụ, tạo NGAY sau PRD vì mọi file sau (architecture, ux, conventions, status) đều phải dùng đúng tên gọi đã thống nhất ở đây, không tự bịa từ đồng nghĩa
3. **docs/decisions.md** — tạo khung rỗng trước, để các bước sau có chỗ ghi quyết định ngay khi phát sinh
4. **docs/architecture.md** — đọc docs/prd.md và docs/domain-glossary.md (đặt tên entity/module đúng theo thuật ngữ đã thống nhất) để suy ra tech stack, bảo mật, performance; mọi lựa chọn quan trọng phải ghi vào docs/decisions.md
5. **docs/ux.md** — đọc docs/prd.md (persona, FR), docs/domain-glossary.md (copy UI phải dùng đúng thuật ngữ), và docs/architecture.md (frontend stack) để ra nguyên tắc thiết kế, information architecture, component pattern làm cơ sở sinh UI khi code
6. **docs/deploy.md** — đọc docs/prd.md (NFR uptime/bảo mật) và docs/architecture.md (hosting, tech stack) để ra quy trình deploy, môi trường, CI/CD
7. **docs/conventions.md** — đọc docs/prd.md (NFR chất lượng), docs/domain-glossary.md (tên class/entity phải truy được về 1 thuật ngữ trong glossary), và docs/architecture.md để ra convention khớp với stack đã chọn
8. **docs/status.md** — đọc cả 7 file trên (prd, domain-glossary, decisions, architecture, ux, deploy, conventions) để chia phase và checklist bao trùm đủ mọi mảng việc, không chỉ backend
9. **CLAUDE.md** — sinh cuối, vì nó tham chiếu tên và vị trí của các file trên

Mỗi prompt dưới đây là độc lập, copy nguyên khối và dán cho Claude Code trong repo của dự án đó. Chạy xong prompt nào, commit file đó trước khi qua prompt kế tiếp — vì prompt sau sẽ yêu cầu Claude Code đọc file trước.

Input Requirement (điền trước khi chạy Prompt 1)

Đây không phải prompt, đây là input bạn cần chuẩn bị. Copy khối này, điền vào, rồi dán kèm vào đầu Prompt 1.

PROMPT — copy toàn bộ khối bên dưới và dán cho Claude Code

```
## Yêu cầu dự án (điền tay)

- Tên dự án:
- Bài toán đang giải quyết là gì (mô tả tình huống thực tế, ai đang gặp khó khăn gì):
- Đối tượng sử dụng (persona cụ thể, không phải "mọi người"):
- Phạm vi MVP (những gì PHẢI có ở bản đầu tiên):
- Ngoài phạm vi (những gì KHÔNG làm ở giai đoạn này, để tránh scope creep):
- Ràng buộc kỹ thuật đã biết trước (nếu có: phải dùng .NET, phải multi-tenant, phải tích hợp hệ thống X...):
- Ràng buộc phi kỹ thuật (deadline, ngân sách, team size...):
```

Prompt 1 — Sinh `docs/prd.md`

PROMPT — copy toàn bộ khối bên dưới và dán cho Claude Code

```
Bạn là Product Owner kỹ thuật. Nhiệm vụ: viết file `docs/prd.md` (tạo thư mục `docs/` ở gốc
repo nếu chưa có) dựa trên input yêu cầu dự án dưới đây. Không tự suy diễn thêm tính năng
ngoài input, nhưng được quyền đặt câu hỏi làm rõ nếu input thiếu thông tin cần cho FR/NFR —
nếu tôi không trả lời được ngay, hãy đánh dấu [CẦN LÀM RÕ] ngay trong file thay vì tự bịa ra
giả định.

=== INPUT YÊU CẦU DỰ ÁN ===
[dán nội dung đã điền ở mục Input Requirement vào đây]
=== HẾT INPUT ===

Cấu trúc `docs/prd.md` bắt buộc gồm các phần sau, theo đúng thứ tự:

1. **Context & Problem Statement** — mô tả bài toán, tại sao nó tồn tại, hậu quả nếu không giải
quyết. Viết như đang giải thích cho một kỹ sư mới join dự án, không viết như slide bán hàng.
2. **Đối tượng người dùng** — persona cụ thể, hành vi, mức độ am hiểu kỹ thuật của họ (ảnh
hưởng đến UX sau này).
3. **Phạm vi (Scope)** — trong phạm vi / ngoài phạm vi, tách rõ 2 cột.
4. **Functional Requirements (FR)** — đánh mã FR-01, FR-02... mỗi FR là 1 hành vi quan sát
được (user làm gì → hệ thống phản hồi gì), không viết chung như "hệ thống phải linh hoạt".
5. **Non-Functional Requirements (NFR)** — đánh mã NFR-01, NFR-02... bắt buộc có ít nhất: hiệu
năng (số liệu cụ thể, không viết "nhanh"), bảo mật (mức tối thiểu chấp nhận được), khả năng mở
rộng, độ tin cậy/uptime nếu liên quan.
6. **Success Metrics** — làm sao biết dự án này thành công, số liệu đo được.
7. **Open Questions** — liệt kê mọi điểm [CẦN LÀM RÕ] đã đánh dấu ở trên, gom lại một chỗ.

Ràng buộc: mỗi FR/NFR phải có thể trace được — sau này docs/architecture.md và docs/status.md
sẽ tham chiếu ngược lại mã FR/NFR này, nên đừng đổi cách đánh mã giữa đường.
```

Prompt 2 — Sinh `docs/domain-glossary.md`

PROMPT — copy toàn bộ khối bên dưới và dán cho Claude Code

```
Đọc kỹ `docs/prd.md` trong repo trước khi làm gì khác — rà toàn bộ Context, Đối tượng người dùng, FR, NFR để trích ra MỌI danh từ nghiệp vụ được lặp lại từ 2 lần trở lên (tên thực thể, tên vai trò người dùng, tên trạng thái/quy trình, tên khái niệm riêng của dự án). Nhiệm vụ: viết file `docs/domain-glossary.md` — đây là "ngôn ngữ chung" (ubiquitous language) của dự án, mọi file sinh sau (architecture, ux, conventions, status) và mọi đoạn code/UI copy sau này đều PHẢI dùng đúng tên gọi đã chốt ở đây, không được tự ý dùng từ đồng nghĩa khác.

Cấu trúc `docs/domain-glossary.md` bắt buộc:

1. **Mục đích file** — ghi ngay đầu file: đây là nơi chốt DUY NHẤT 1 tên gọi cho mỗi khái niệm nghiệp vụ, không phải nơi giải thích kiến trúc (đó là việc của architecture.md) và không phải nơi mô tả UI (đó là việc của ux.md).
2. **Bảng thuật ngữ** — mỗi dòng gồm các cột:
   - **Thuật ngữ chuẩn** — tên gọi DUY NHẤT được phép dùng (ghi rõ cả bản tiếng Việt hiển thị cho người dùng lẫn tên tiếng Anh dùng trong code nếu 2 tên khác nhau, ví dụ: "Học viên" (hiển thị) / `Student` (code))
   - **Định nghĩa** — 1-2 câu, định nghĩa theo đúng ngữ cảnh dự án này, không phải định nghĩa từ điển chung chung
   - **KHÔNG được gọi là** — liệt kê các từ đồng nghĩa dễ bị dùng nhầm lẫn (ví dụ nếu thuật ngữ chuẩn là "Học viên" thì liệt kê "người dùng, user, client, khách hàng" là các từ CẤM dùng thay thế trong ngữ cảnh này)
   - **Thuật ngữ liên quan** — các thuật ngữ khác trong bảng có quan hệ trực tiếp (để agent tra cứu chéo)
3. **Nhóm theo domain** — nếu dự án có nhiều domain con (ví dụ Billing, Identity, Content), nhóm thuật ngữ theo domain đó, vì cùng 1 từ có thể mang nghĩa khác nhau ở domain khác nhau — nếu phát hiện trường hợp này (đồng âm khác nghĩa giữa 2 domain), phải nêu rõ và tách biệt, không gộp chung 1 dòng.
4. **Quy tắc bổ sung thuật ngữ mới** — ghi rõ ngay trong file: "Khi implement phát sinh khái niệm nghiệp vụ mới chưa có trong bảng trên, PHẢI thêm vào đây (đúng format 4 cột) TRƯỚC KHI dùng tên đó trong code hoặc UI, không được tự đặt tên tạm rồi dùng luôn. Nếu có nhiều cách gọi hợp lý và cần chọn 1, ghi quyết định chọn tên vào `docs/decisions.md`."
5. **Ghi chú về đa ngôn ngữ** — nếu UI hiển thị tiếng Việt nhưng code viết tiếng Anh (trường hợp phổ biến), quy định rõ: thuật ngữ chuẩn trong bảng luôn đi kèm cả 2 dạng, và đây là bảng map DUY NHẤT giữa 2 ngôn ngữ — không được tự dịch khác đi ở bất kỳ đâu khác trong dự án.

Nếu `docs/prd.md` có thuật ngữ mơ hồ hoặc dùng lẫn lộn (ví dụ đôi chỗ gọi "Học viên" đôi chỗ gọi "Người học" cho cùng 1 khái niệm), phải hỏi tôi để chốt 1 tên duy nhất, không tự chọn 1 trong 2 rồi im lặng.
```

Prompt 3 — Sinh `docs/decisions.md`

PROMPT — copy toàn bộ khối bên dưới và dán cho Claude Code

```
Nhiệm vụ: viết file `docs/decisions.md`. Đây là nhật ký quyết định (decision log) của dự án — nơi tra cứu lại "tại sao lúc đó lại chọn thế" khi có người (kể cả tôi) quay lại dự án sau vài tháng và không nhớ lý do.

Ở bước này file chỉ cần dựng khung, chưa có quyết định nào cả vì docs/architecture.md/docs/conventions.md chưa được tạo. Cấu trúc bắt buộc:

1. **Mục đích file** — ghi ngay đầu file 1 đoạn ngắn: file này dùng để tra cứu lý do ra quyết định, KHÔNG dùng để mô tả trạng thái hiện tại (đó là việc của docs/status.md) và KHÔNG dùng để mô tả kiến trúc (đó là việc của docs/architecture.md).
2. **Quy tắc khi thêm 1 quyết định mới** — nêu rõ format bắt buộc cho mỗi entry:
   - Ngày (hoặc số thứ tự tăng dần nếu không cần ngày)
   - Bối cảnh (Context) — vấn đề gì cần quyết định, tại sao phải quyết định
   - Các lựa chọn đã xem xét (Options considered) — kể cả lựa chọn không chọn, và vì sao loại
   - Quyết định (Decision) — chọn cái gì, 1 câu rõ ràng
   - Lý do (Rationale) — vì sao chọn cái này, gắn với NFR/ràng buộc cụ thể trong docs/prd.md hoặc docs/architecture.md nếu có
   - Đánh đổi/hệ quả (Trade-off) — cái gì bị mất/rủi ro khi chọn hướng này
   - Trạng thái (Status): Active / Superseded (nếu sau này có quyết định khác thay thế, KHÔNG xóa entry cũ, chỉ đổi status và trỏ link tới entry mới)
3. **Ngưỡng khi nào cần ghi vào đây** — liệt kê rõ loại quyết định PHẢI ghi (chọn công nghệ/thư viện chính, đổi kiến trúc, đổi convention đã thống nhất, chọn giải pháp cho 1 vấn đề có nhiều cách làm hợp lý, bỏ qua 1 best practice vì lý do thực tế) và loại KHÔNG cần ghi (tên biến, format code, việc chi tiết không ảnh hưởng người khác) — để tránh file này bị spam bởi việc nhỏ.
4. **Danh sách quyết định** — để trống, chờ điền dần trong quá trình implement (mục "## Log" hoặc tương đương, mỗi entry là 1 heading con để dễ tra cứu/link tới).

File này không tạo 1 lần rồi xong — nó được ghi tiếp trong suốt đời dự án, nên CLAUDE.md (tạo ở bước cuối) sẽ ràng buộc việc ghi vào đây thành quy tắc bắt buộc.
```

Prompt 4 — Sinh `docs/architecture.md`

PROMPT — copy toàn bộ khối bên dưới và dán cho Claude Code

```
Đọc kỹ file `docs/prd.md` và `docs/domain-glossary.md` trong repo trước khi làm gì khác. Nhiệm vụ: viết file `docs/architecture.md`, kiến trúc phải giải quyết đúng FR/NFR đã liệt kê trong docs/prd.md — không đề xuất kiến trúc phù hợp với "dự án lý tưởng" mà phù hợp với ràng buộc thực tế đã nêu trong docs/prd.md (team size, deadline, ràng buộc kỹ thuật có sẵn). Mọi tên module, entity, bảng dữ liệu PHẢI dùng đúng thuật ngữ đã chốt trong docs/domain-glossary.md (dùng cột tên tiếng Anh/code nếu có) — không tự đặt tên khác hoặc viết tắt khác đi so với glossary.

Cấu trúc `docs/architecture.md` bắt buộc:

1. **Kiến trúc tổng quan** — mô tả bằng văn xuôi + 1 diagram dạng text/ASCII hoặc mermaid, thể hiện các thành phần chính và luồng dữ liệu.
2. **Tech Stack** — với mỗi lựa chọn (backend, frontend, database, hosting, message queue nếu có...), nêu rõ lý do chọn gắn với NFR cụ thể (ví dụ: "chọn PostgreSQL vì NFR-03 yêu cầu ACID transaction cho billing"). Không liệt kê tech stack như một wishlist không lý do.
3. **Module/Component breakdown** — chia theo domain, mỗi module map với FR nào và với thuật ngữ nào trong docs/domain-glossary.md.
4. **Bảo mật** — authentication/authorization strategy, xử lý secret, threat mặc định phải chống (injection, IDOR, rate limit...), map với NFR bảo mật đã nêu trong docs/prd.md.
5. **Performance** — target cụ thể (latency, throughput, concurrent users) lấy từ NFR, và chiến lược đạt được nó (caching, indexing, async...).
6. **Data model tổng quan** — entity chính và quan hệ, không cần chi tiết từng field. Tên entity phải trùng khớp thuật ngữ trong docs/domain-glossary.md; nếu cần đặt tên 1 khái niệm kỹ thuật thuần túy chưa có trong glossary (không phải khái niệm nghiệp vụ), ghi rõ đây là tên kỹ thuật nội bộ, không phải thuật ngữ nghiệp vụ.
7. **Rủi ro kỹ thuật đã biết** — điểm nào trong kiến trúc này là đánh đổi (trade-off), rủi ro gì có thể phát sinh khi scale.

Nếu có FR hoặc NFR trong docs/prd.md mà kiến trúc đề xuất không đáp ứng được, phải nêu rõ trong mục Rủi ro, không được im lặng bỏ qua. Nếu phát hiện docs/domain-glossary.md thiếu thuật ngữ cần dùng để đặt tên 1 entity/module, phải bổ sung vào docs/domain-glossary.md trước, không tự đặt tên rồi bỏ qua glossary.
```

Prompt 5 — Sinh `docs/ux.md`

PROMPT — copy toàn bộ khối bên dưới và dán cho Claude Code

```
Đọc kỹ `docs/prd.md` (đặc biệt phần Đối tượng người dùng và FR), `docs/domain-glossary.md` (bảng thuật ngữ chuẩn), và `docs/architecture.md` (đặc biệt phần frontend tech stack) trong repo trước khi làm gì khác. Nhiệm vụ: viết file `docs/ux.md` — đây là cơ sở duy nhất để bạn (Claude Code) sinh UI/UX nhất quán khi implement, không phải tài liệu mô tả cho người đọc rồi bỏ đó. Mọi label, tên màn hình, tên trạng thái xuất hiện trong file này PHẢI dùng đúng thuật ngữ hiển thị đã chốt trong docs/domain-glossary.md, không tự dịch hoặc diễn đạt khác đi. Không tự bịa phong cách thiết kế nếu tôi chưa có brand guideline — nếu chưa có, đề xuất 1 baseline hợp lý dựa trên đối tượng người dùng trong docs/prd.md và đánh dấu [ĐỀ XUẤT - CẦN DUYỆT] để tôi xác nhận lại.

Cấu trúc `docs/ux.md` bắt buộc:

1. **Nguyên tắc thiết kế** — 3-5 nguyên tắc cụ thể gắn với đối tượng người dùng đã mô tả trong docs/prd.md (ví dụ: người dùng không rành kỹ thuật → ưu tiên giảm số bước, tránh thuật ngữ chuyên môn trong UI copy). Không viết nguyên tắc chung chung kiểu "giao diện phải đẹp và dễ dùng".
2. **Information Architecture** — sơ đồ cây điều hướng (navigation structure), map mỗi màn hình chính với FR nào trong docs/prd.md giải quyết, đặt tên màn hình theo đúng thuật ngữ trong docs/domain-glossary.md.
3. **Danh sách màn hình/luồng chính** — với mỗi luồng quan trọng (ví dụ: đăng ký, thanh toán...), mô tả các bước, trạng thái loading/empty/error/success bắt buộc phải có — không được thiếu trạng thái lỗi hay rỗng khi implement.
4. **Design tokens cơ bản** — bảng màu (primary/secondary/semantic: success/warning/error), typography scale (heading/body/caption), spacing scale, breakpoint cho responsive. Nếu chưa có brand, đề xuất baseline và đánh dấu [ĐỀ XUẤT - CẦN DUYỆT].
5. **Component pattern chung** — quy tắc cho các thành phần lặp lại (form, table, modal, toast/notification, empty state, loading skeleton): hành vi mặc định, khi nào dùng loại nào, để agent không tự sáng tạo pattern mới mỗi lần code 1 màn hình khác.
6. **Accessibility tối thiểu** — yêu cầu bắt buộc (contrast tối thiểu, keyboard navigation, label cho input, alt text cho ảnh) — mức tối thiểu chấp nhận được, không viết như checklist WCAG đầy đủ nếu dự án không cần mức đó.
7. **Content/tone of voice trong UI copy** — cách xưng hô, mức độ trang trọng, ngôn ngữ (theo docs/prd.md và persona), áp dụng cho label, error message, thông báo hệ thống — mọi danh từ nghiệp vụ trong copy phải khớp nguyên văn với docs/domain-glossary.md.
8. **Responsive rule** — hành vi tối thiểu ở mobile/tablet/desktop nếu dự án cần đa nền tảng (dựa vào docs/architecture.md/frontend stack).

Nếu cần dùng 1 thuật ngữ hiển thị chưa có trong docs/domain-glossary.md, bổ sung vào đó trước, không tự đặt tên riêng cho UI. File này sẽ được docs/conventions.md tham chiếu khi định nghĩa naming/cấu trúc cho component, và được CLAUDE.md bắt buộc đọc trước khi code bất kỳ màn hình nào.
```

Prompt 6 — Sinh `docs/deploy.md`

PROMPT — copy toàn bộ khối bên dưới và dán cho Claude Code

```
Đọc kỹ `docs/prd.md` (đặc biệt các NFR về uptime, bảo mật, tuân thủ/compliance nếu có) và `docs/architecture.md` (đặc biệt phần tech stack, hosting nếu đã đề cập, bảo mật) trong repo trước khi làm gì khác — docs/deploy.md phải đáp ứng đúng NFR đã cam kết trong docs/prd.md, không chỉ đúng kỹ thuật theo docs/architecture.md (ví dụ NFR yêu cầu uptime 99.9% thì phải có health check/rollback tương ứng, không thể chỉ ghi "deploy lên 1 server"). Nếu docs/architecture.md chưa nói rõ hosting/infra, hỏi tôi trước khi viết thay vì tự chọn — vì đây là quyết định ảnh hưởng chi phí thực tế, không phải chi tiết kỹ thuật nhỏ. Nhiệm vụ: viết file `docs/deploy.md` — đây là hướng dẫn vận hành để agent tự thực hiện hoặc hỗ trợ deploy đúng quy trình, không phải tài liệu tham khảo chung chung.

Cấu trúc `docs/deploy.md` bắt buộc:

1. **Môi trường (Environments)** — liệt kê rõ các môi trường tồn tại (local/dev, staging, production), khác nhau ở điểm gì (config, database, domain), và mục đích từng môi trường.
2. **Hạ tầng** — nơi host (server/cloud provider/container), map với lựa chọn đã ghi trong docs/architecture.md, kèm cấu trúc thư mục/service nếu có nhiều thành phần (API, worker, frontend...).
3. **Quản lý biến môi trường/secret** — nơi lưu (không bao giờ commit vào repo), cách agent lấy giá trị khi cần chạy local, danh sách biến bắt buộc phải có để chạy được dự án.
4. **Quy trình build & CI/CD** — các bước build, thứ tự chạy test (unit → integration → e2e) như một gate bắt buộc trước khi cho phép deploy — nếu bất kỳ tầng test nào fail, pipeline phải dừng, không được deploy tiếp.
5. **Database migration** — cách chạy migration ở từng môi trường, quy tắc migration phải reversible hoặc có kế hoạch rollback rõ ràng trước khi áp dụng lên production.
6. **Quy trình deploy từng bước** — checklist tuần tự agent phải làm theo đúng thứ tự khi được yêu cầu deploy (pre-check → build → migrate → deploy → smoke test sau deploy).
7. **Rollback** — điều kiện khi nào phải rollback, các bước rollback cụ thể cho từng môi trường.
8. **Monitoring & Logging sau deploy** — cần kiểm tra gì ngay sau khi deploy để xác nhận hệ thống ổn (log lỗi, health check endpoint, alert nếu có).
9. **Quy tắc bắt buộc dành cho agent** — ghi rõ ngay trong file: "KHÔNG được tự ý deploy lên production khi chưa có xác nhận rõ ràng từ người dùng trong cùng phiên làm việc, kể cả khi mọi test đã pass. Deploy lên staging có thể tự thực hiện nếu pipeline cho phép, nhưng production luôn cần xác nhận thủ công."

File này được CLAUDE.md bắt buộc đọc trước khi thực hiện bất kỳ hành động liên quan đến deploy/release.
```

Prompt 7 — Sinh `docs/conventions.md`

PROMPT — copy toàn bộ khối bên dưới và dán cho Claude Code

```
Đọc kỹ `docs/prd.md` (đặc biệt NFR về độ tin cậy/chất lượng), `docs/domain-glossary.md`, và
`docs/architecture.md` trong repo trước khi làm gì khác. Nhiệm vụ: viết file
`docs/conventions.md` — đây là chuẩn code dùng để bạn (Claude Code) tự review lại code mình
viết ra, nên phải cụ thể đến mức có thể áp dụng máy móc, không viết nguyên tắc mơ hồ.

Cấu trúc `docs/conventions.md` bắt buộc, khớp với tech stack đã chọn trong
docs/architecture.md:

1. **Naming convention** — cho từng loại: file, class, function/method, biến, database
table/column, API endpoint, branch git, commit message. Cho ví dụ đúng và ví dụ sai cho mỗi
loại. Với class/entity/table đại diện cho 1 khái niệm nghiệp vụ, tên PHẢI truy ngược được về
đúng 1 thuật ngữ trong docs/domain-glossary.md (chỉ khác biệt ở quy tắc viết hoa/viết thường
theo convention, không khác nghĩa) — nêu rõ quy tắc chuyển đổi từ tên tiếng Việt/tiếng Anh
trong glossary sang tên biến/class theo convention đã chọn (camelCase, PascalCase,
snake_case...).
2. **Cấu trúc thư mục** — cây thư mục chuẩn, giải thích lớp nào chứa gì (map với module
breakdown trong docs/architecture.md nếu là kiến trúc phân lớp/Clean Architecture).
3. **Coding style theo ngôn ngữ đã chọn** — quy tắc riêng cho backend và frontend theo đúng
stack trong docs/architecture.md (không viết convention chung cho ngôn ngữ không dùng).
4. **Error handling convention** — cách throw/catch, cách trả lỗi ra API, không được silent
fail.
5. **Testing convention** — naming test file/test case, cấu trúc Arrange-Act-Assert hoặc tương
đương, mock strategy, coverage tối thiểu chấp nhận được cho unit/integration/e2e — mức
coverage và độ nghiêm ngặt phải tương xứng với NFR về độ tin cậy trong docs/prd.md (dự án yêu
cầu uptime/độ chính xác cao thì coverage tối thiểu phải cao hơn).
6. **Quy tắc review** — checklist cụ thể để tự chấm code trước khi báo "done": ví dụ không
hardcode secret, không có TODO chưa giải quyết, không có console.log/print debug còn sót, tên
biến không viết tắt tùy tiện, tên class/entity nghiệp vụ khớp với docs/domain-glossary.md.
7. **Điều cấm tuyệt đối** — liệt kê pattern bị cấm rõ ràng trong dự án này (ví dụ: cấm business
logic trong controller, cấm gọi DB trực tiếp từ frontend, cấm đặt tên entity/class nghiệp vụ
khác với docs/domain-glossary.md...).

File này sẽ được dùng làm checklist review sau mỗi task, nên mỗi mục phải là điều kiện có thể
trả lời đúng/sai, không phải khuyến nghị chung.
```

Prompt 8 — Sinh `docs/status.md`

PROMPT — copy toàn bộ khối bên dưới và dán cho Claude Code

```
Đọc kỹ `docs/prd.md`, `docs/domain-glossary.md`, `docs/architecture.md`, `docs/ux.md`, `docs/deploy.md`, `docs/conventions.md`, và `docs/decisions.md` trong repo trước khi làm gì khác — docs/status.md là bản kế hoạch triển khai tổng hợp từ TẤT CẢ các file trên, không chỉ từ docs/prd.md và docs/architecture.md, vì phase phải bao gồm cả việc dựng UI (theo docs/ux.md) và dựng hạ tầng/CI-CD (theo docs/deploy.md), không chỉ code backend logic. Tên phase và tên hạng mục PHẢI dùng đúng thuật ngữ trong docs/domain-glossary.md, không tự diễn đạt khác đi. Nhiệm vụ: viết file `docs/status.md` — file này là nguồn sự thật duy nhất về tiến độ dự án, phải được cập nhật sau MỖI prompt implement, không phải cuối ngày hay cuối phase.

Cấu trúc `docs/status.md` bắt buộc:

1. **Tổng quan phase** — chia dự án thành các phase theo thứ tự triển khai hợp lý (không nhất thiết theo thứ tự FR trong docs/prd.md — ưu tiên phase nào tạo nền tảng cho phase sau, ví dụ Auth/Identity trước Billing). Phải có ít nhất 1 phase riêng cho việc dựng hạ tầng/CI-CD ban đầu (theo docs/deploy.md) trước khi có phase nào deploy thật, và mỗi phase có tính năng user-facing phải có hạng mục dựng UI tương ứng (theo docs/ux.md), không gộp UI vào chung 1 dòng mơ hồ.
2. **Với mỗi phase**: tên phase, mục tiêu, danh sách FR/NFR liên quan (tham chiếu mã đã đánh trong docs/prd.md), và checklist hạng mục dạng markdown checkbox `- [ ]`. Mỗi checkbox là 1 đơn vị việc đủ nhỏ để hoàn thành trong 1 prompt, và chỉ được tick `- [x]` khi đã có unit test + integration test + e2e test liên quan đều pass (theo convention coverage đã định trong docs/conventions.md) — không tick khi code "chạy được" nhưng chưa có test.
3. **Trạng thái hiện tại** — phase nào đang làm, hạng mục nào đang làm, để agent biết tiếp tục từ đâu khi mở lại conversation mới.
4. **Deferred Work** — danh sách việc đã CỐ Ý hoãn lại (không phải quên), mỗi mục ghi rõ: việc gì bị hoãn, hoãn từ hạng mục/phase nào, lý do hoãn (thiếu thời gian, chờ quyết định khác, không phải ưu tiên MVP...), và phase/mốc nào sẽ quay lại làm — nếu chưa xác định mốc thì ghi rõ "chưa xác định, cần review lại trước khi release". Mục này để tránh việc trì hoãn bị quên luôn, đặc biệt là nợ kỹ thuật hoặc test bị bỏ qua tạm thời.
5. **Quy tắc cập nhật file này** — ghi rõ ngay trong file: "Sau khi hoàn thành bất kỳ hạng mục nào, phải tick checkbox tương ứng trong file này TRƯỚC KHI báo hoàn thành task cho người dùng. Nếu hạng mục chưa đủ test 3 tầng (unit/integration/e2e) thì không được tick, dù code đã chạy đúng. Nếu trong lúc làm phát sinh việc cần hoãn lại, phải thêm ngay vào Deferred Work, không được bỏ qua âm thầm. Nếu trong lúc làm phát sinh quyết định đáng ghi, phải thêm vào docs/decisions.md trước, rồi mới tick checklist ở đây."
6. **Blocker/Risk log** — chỗ để agent tự ghi lại nếu gặp vướng mắc cần người dùng quyết định, không được tự quyết rồi im lặng.
```

Prompt 9 — Sinh `CLAUDE.md`

PROMPT — copy toàn bộ khối bên dưới và dán cho Claude Code

```
Đọc `docs/prd.md`, `docs/domain-glossary.md`, `docs/decisions.md`, `docs/architecture.md`, `docs/ux.md`, `docs/deploy.md`, `docs/conventions.md`, `docs/status.md` đã có trong repo để biết tên và vai trò từng file trước khi viết. Nhiệm vụ: viết file `CLAUDE.md` tại GỐC repo (KHÔNG đặt trong docs/, vì Claude Code chỉ tự động đọc CLAUDE.md ở gốc repo, không quét vào thư mục con) — đây là file cấu hình hành vi của bạn (Claude Code) cho toàn bộ dự án này, mọi session sau đều phải tuân theo, không phải hướng dẫn một lần.

Nội dung `CLAUDE.md` bắt buộc gồm:

1. **Quy tắc đọc file trước khi làm task** — nêu rõ: trước KHI bắt đầu bất kỳ task nào (viết code mới, sửa bug, refactor), phải đọc lại `docs/prd.md` (phần FR/NFR liên quan đến task), `docs/domain-glossary.md` (toàn bộ — vì bất kỳ task nào chạm vào khái niệm nghiệp vụ đều cần đúng thuật ngữ, không chỉ task UI), `docs/architecture.md` (phần module liên quan), `docs/conventions.md` (toàn bộ), `docs/status.md` (phase hiện tại, kể cả mục Deferred Work để không lặp lại việc đã cố ý hoãn), và `docs/decisions.md` (rà xem có quyết định cũ nào liên quan đến task này không, để không đi ngược lại quyết định đã chốt mà không biết) — không dựa vào ký ức từ session trước, vì context có thể đã đổi. Ngoài ra: nếu task có liên quan đến giao diện (tạo/sửa màn hình, component, form...) BẮT BUỘC đọc thêm `docs/ux.md` trước khi code, không tự sáng tạo pattern UI khác với đã định nghĩa. Nếu task có liên quan đến build/deploy/release/CI-CD BẮT BUỘC đọc thêm `docs/deploy.md` trước khi thực hiện. Nếu bất kỳ file nào trong 9 file bắt buộc (`docs/prd.md`, `docs/domain-glossary.md`, `docs/decisions.md`, `docs/architecture.md`, `docs/ux.md`, `docs/deploy.md`, `docs/conventions.md`, `docs/status.md`, `CLAUDE.md`) không tồn tại hoặc trống khi lẽ ra phải có, DỪNG LẠI và báo cho người dùng, không tự suy diễn nội dung thay thế.
2. **Định nghĩa "Done" (Definition of Done)** — nêu rõ: một task CHỈ được coi là hoàn thành (xanh) khi đồng thời có (a) unit test cho logic mới/thay đổi, (b) integration test cho luồng liên quan, (c) e2e test cho hành vi người dùng cuối nếu task ảnh hưởng đến luồng user-facing, và cả 3 tầng test đều pass. Thiếu 1 trong 3 tầng (khi tầng đó áp dụng được cho loại thay đổi này) thì task coi như CHƯA xong, phải báo rõ cho người dùng là còn thiếu tầng test nào, không được báo "hoàn thành" khi chưa đủ.
3. **Quy trình sau khi hoàn thành 1 task** — bắt buộc: (a) chạy đủ 3 tầng test và xác nhận pass, (b) tick checklist tương ứng trong `docs/status.md`, đồng thời thêm vào Deferred Work nếu có việc bị hoãn lại trong quá trình làm, (c) review code vừa viết theo checklist trong `docs/conventions.md` trước khi báo kết quả.
4. **Quy tắc ghi quyết định vào `docs/decisions.md`** — bắt buộc: mỗi khi phải chọn giữa nhiều cách làm hợp lý (chọn thư viện, đổi cách tiếp cận so với docs/architecture.md ban đầu, chọn giải pháp cho vấn đề chưa có tiền lệ trong dự án, hoặc quyết định bỏ qua 1 best practice vì lý do thực tế), PHẢI ghi ngay 1 entry vào `docs/decisions.md` theo đúng format đã định nghĩa trong file đó (Context/Options/Decision/Rationale/Trade-off/Status) — ghi TRƯỚC KHI báo hoàn thành task, không gộp lại ghi sau vì sẽ quên lý do thật. Việc nhỏ không ảnh hưởng người khác (tên biến, format code...) thì không cần ghi, theo đúng ngưỡng đã định nghĩa trong `docs/decisions.md`.
5. **Quy tắc kỷ luật thuật ngữ nghiệp vụ (`docs/domain-glossary.md`)** — bắt buộc: mọi tên biến/class/entity/bảng dữ liệu/label UI đại diện cho 1 khái niệm nghiệp vụ PHẢI truy ngược được về đúng 1 thuật ngữ trong `docs/domain-glossary.md`. Nếu task cần dùng 1 khái niệm nghiệp vụ chưa có trong glossary, PHẢI bổ sung vào `docs/domain-glossary.md` (đúng format: thuật ngữ chuẩn / định nghĩa / không được gọi là / thuật ngữ liên quan) TRƯỚC KHI viết code hoặc UI copy dùng tên đó — không tự đặt tên tạm rồi dùng luôn, không tự dịch hoặc diễn đạt khác đi so với những gì đã có trong glossary dù chỉ là viết tắt hay đồng nghĩa. Nếu phát hiện code/UI hiện tại đang dùng tên khác với glossary (do task trước tạo ra hoặc do phát hiện lỗi), phải báo cho người dùng và hỏi có nên đổi lại cho khớp hay cập nhật glossary theo tên hiện có, không tự ý sửa hàng loạt.
6. **Quy tắc khi implement giao diện** — mọi màn hình/component mới phải khớp với nguyên tắc thiết kế, design token, và component pattern trong `docs/ux.md`; nếu task cần 1 pattern UI chưa có trong `docs/ux.md`, phải bổ sung vào `docs/ux.md` trước (và ghi lý do vào `docs/decisions.md` nếu đó là 1 lựa chọn đáng kể), không tự tạo pattern rời rạc riêng cho từng màn hình.
```

```
7. **Quy tắc khi thực hiện deploy/release** — phải làm đúng tuần tự trong `docs/deploy.md`; tuyệt đối KHÔNG tự ý deploy lên production khi chưa có xác nhận rõ ràng từ người dùng trong phiên làm việc hiện tại, kể cả khi mọi test đã pass và pipeline cho phép chạy tự động.
8. **Quy tắc khi có mâu thuẫn giữa các file** — áp dụng cho MỌI cặp file, không riêng docs/architecture.md: nếu lúc implement phát hiện 2 file bắt buộc (ví dụ docs/conventions.md và docs/ux.md, hoặc docs/architecture.md và docs/deploy.md, hoặc tên gọi trong docs/domain-glossary.md khác với tên đang dùng trong docs/architecture.md) mô tả 2 cách làm khác nhau cho cùng 1 vấn đề, hoặc không thể đáp ứng đúng những gì 1 file đã mô tả do lý do kỹ thuật phát sinh, PHẢI dừng lại và hỏi người dùng, không tự chọn 1 bên rồi tiếp tục, và không tự sửa file gốc để hợp thức hóa lựa chọn của mình. Thứ tự ưu tiên tạm thời khi cần quyết định nhanh nhưng chưa hỏi kịp người dùng: `docs/prd.md` và `docs/domain-glossary.md` (yêu cầu nghiệp vụ và tên gọi thống nhất) > `docs/architecture.md`/`docs/ux.md`/`docs/deploy.md` (thiết kế kỹ thuật) > `docs/conventions.md` (chi tiết code) — nhưng đây chỉ là thứ tự tạm để không bị kẹt cứng, quyết định cuối vẫn phải xác nhận lại với người dùng và ghi vào `docs/decisions.md`. Nếu người dùng đồng ý đổi hướng, phải ghi lại vào `docs/decisions.md` với Status: Active và đánh dấu quyết định cũ liên quan là Superseded.
9. **Quy tắc khi docs/prd.md có mục [CẦN LÀM RÕ] chưa giải quyết liên quan đến task đang làm** — phải hỏi người dùng trước khi code, không tự giả định.
10. **Giới hạn phạm vi task** — không tự ý implement thêm FR chưa được yêu cầu trong task hiện tại, dù thấy "tiện làm luôn".

Viết ngắn gọn, dạng chỉ dẫn trực tiếp cho AI agent (không viết như văn bản giải thích cho người), vì file này được Claude Code đọc lại ở đầu mỗi session.
```