Tài liệu: Hướng Dẫn Vibe Coding cho phát triển sản phẩm nhanh

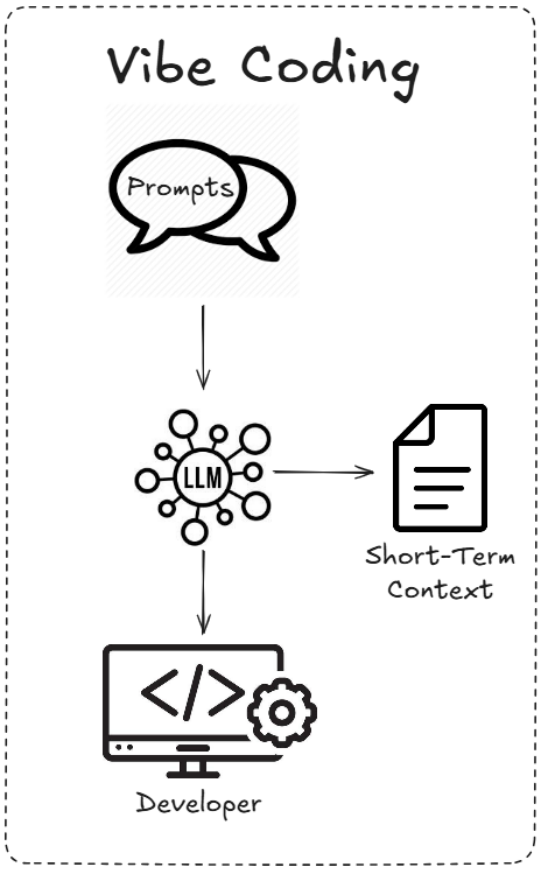
Abstract: Trong thế giới phát triển phần mềm ngày nay, việc áp dụng trí tuệ nhân tạo (AI) vào quy trình làm việc không còn là điều mới mẻ. Vibe coding, một khái niệm mới nổi, đề cập đến việc lập trình viên diễn đạt bằng ngôn ngữ tự nhiên để AI hỗ trợ sinh mã và lặp lại qua các vòng phản hồi. Tuy nhiên thường gặp các vấn đề như sai lệch so với yêu cầu, khả năng bảo trì, các UI và logic bị trộn lẫn dẫn đến lặp code và chưa khai thác hiệu quả các component có sẵn khi "code theo cảm hứng". Tài liệu này sẽ đi sâu vào cách tiếp cận và đề xuất một quy trình vibe coding có kiểm soát để có thể tận dụng tối đa sức mạnh của AI trong việc tạo ra sản phẩm nhanh chóng. Quy trình gồm các bước: (1) Business Idea → (2) Low-code (UI/Flow/Rule) → (3) PRD (BA mode) → (4) POC (Dev mode) → (5) Decoupling → (6) GitHub Delivery.

Keywords: Vibe coding, PRD, POC, tách lớp kiến trúc; tái sử dụng component

I/ Introduction

a) Context

Sự phát triển nhanh chóng của các mô hình ngôn ngữ lớn (LLM) đã thúc đẩy việc tích hợp AI vào nhiều công đoạn của vòng đời phát triển phần mềm, từ phân tích yêu cầu, thiết kế, sinh mã, đến hỗ trợ kiểm thử và tài liệu hóa. Trong bối cảnh đó, vibe coding nổi lên như một cách tiếp cận tập trung vào tương tác human-in-the-loop, nơi lập trình viên diễn đạt ý định ở mức cao bằng ngôn ngữ tự nhiên và AI hỗ trợ sinh mã thông qua các vòng phản hồi lặp (prompt–evaluate–refine)[1] như mô tả Hình 1 là cơ chế tương tác cốt lõi của vibe coding, trong đó LLM đóng vai trò sinh mã từ prompt và ngữ cảnh ngắn hạn. Còn vai trò của lập trình viên chủ yếu là hướng dẫn hoặc kiểm tra. Cách tiếp cận này đặc biệt phù hợp với giai đoạn prototype/POC, khi mục tiêu chính là tạo ra sản phẩm chạy được nhanh dựa trên ý tưởng và luồng nghiệp vụ.



Hình 1: Quy trình Vibe Coding dựa trên prompt và ngữ cảnh ngắn hạn

Tuy nhiên, các bằng chứng thực nghiệm gần đây cho thấy lợi ích về tốc độ của vibe coding đi kèm những rủi ro đáng kể về chất lượng khi mở rộng (scale). Một tổng quan hệ thống (Systematic Literature Review) đăng trên Tạp chí Khoa học Trường Đại học Mở Hà Nội, tuân thủ PRISMA 2020 và tổng hợp 17 nghiên cứu giai đoạn 2022–2025, ghi nhận vibe coding có thể cải thiện tốc độ phát triển trung bình 19% đến 23%. Tuy vậy, độ chính xác của mã sinh ra chỉ khoảng 48%, và tỷ lệ lỗi trong lần sinh đầu tiên khoảng 31%, phản ánh nguy cơ lệch yêu cầu, lỗi chất lượng và rủi ro bảo mật nếu thiếu cơ chế kiểm soát [2].

Vì vậy, động lực của tài liệu này là đề xuất một quy trình vibe coding có kiểm soát nhằm tận dụng ưu thế tăng tốc của AI trong giai đoạn phát triển nhanh (prototype/POC) để đảm bảo mã nguồn có thể mở rộng và duy trì chất lượng khi dự án tiến tới mức độ hoàn thiện cao hơn.

b) Problem statement

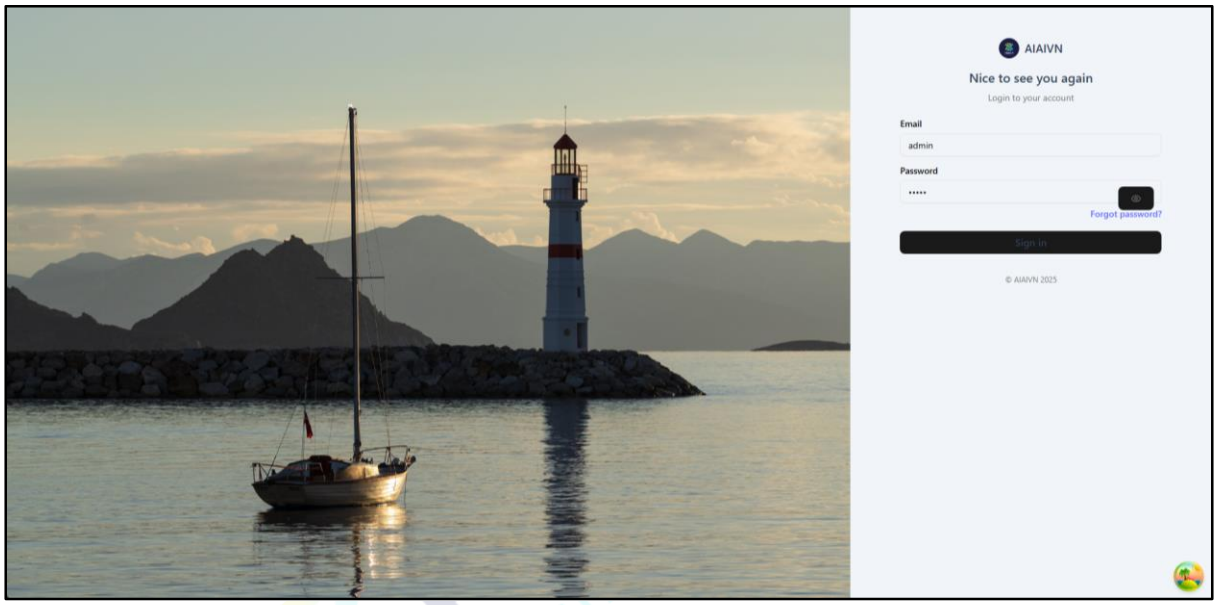
Mặc dù vibe coding giúp tăng tốc quá trình xây dựng prototype/POC, việc áp dụng theo hướng "code theo cảm hứng" thường dẫn đến sai lệch so với yêu cầu, giảm khả năng bảo trì. Do đó, cần một quy trình có kiểm soát để đảm bảo đầu ra từ AI có thể chuyển hóa thành mã nguồn có cấu trúc, tái sử dụng tốt và phù hợp với yêu cầu nghiệp vụ.

- **Các vấn đề cụ thể trong dự án (GPS Tours)**

Trong dự án GPS Tours, phần backend đã được hoàn thiện với các API phục vụ nghiệp vụ chính. Tuy nhiên, ở phía frontend, mặc dù đã có các component bản đồ (map components)

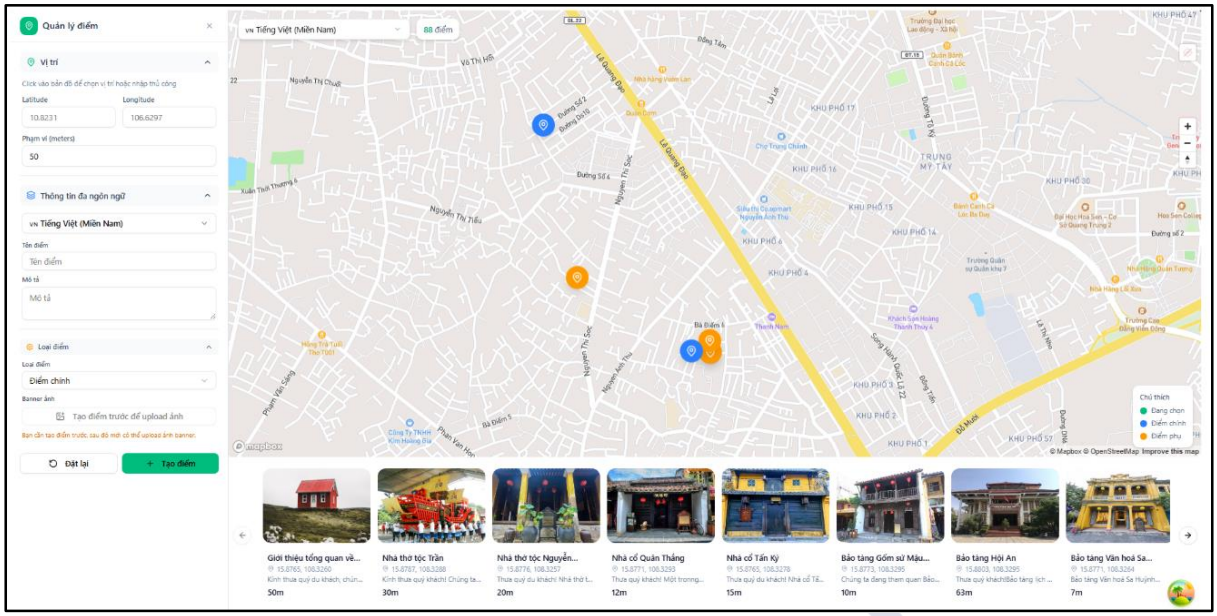
dùng chung, hệ thống vẫn còn một số điểm chưa hoàn thiện về giao diện và tổ chức màn hình nghiệp vụ. Cụ thể, các vấn đề trọng tâm hiện tại tập trung vào ba khu vực sau:

- **(1) Trang đăng nhập (Login page):** Giao diện đăng nhập hiện chưa đạt về mặt trải nghiệm người dùng (UI/UX), bao gồm bố cục, các button bị ghi đè lên. Bên cạnh đó, một số thành phần chức năng liên quan đến xác thực vẫn chưa hoàn thiện, bao gồm JWT, AuthGuard và các xử lý sau khi đăng nhập. Việc chỉnh sửa theo hướng vibe coding, nếu thiếu kiểm soát, có thể tạo ra các thay đổi thiên về "cải thiện bề ngoài" nhưng không đồng bộ với design system, đồng thời bỏ sót các trạng thái quan trọng như xác thực và tích hợp luồng gọi API tới backend.



Hình 2: Giao diện của trang đăng nhập hiện tại

- **(2) Trang quản lý POIs (Points of Interest):** Giao diện quản lý điểm (POIs) hiện tương đối ổn định về mặt hiển thị và đã có thành phần bản đồ hỗ trợ thao tác vị trí. Tuy nhiên, các chức năng nghiệp vụ cốt lõi vẫn chưa hoàn thiện, đặc biệt là **quy trình CRUD chưa rõ ràng** và còn thiếu một số thao tác quan trọng như **xóa điểm**. Bên cạnh đó, trang POIs **chưa được tích hợp đầy đủ với backend** (CRUD), dẫn đến nguy cơ lệch dữ liệu giữa giao diện và hệ thống thực tế. Trong bối cảnh triển khai nhanh theo vibe coding, nếu thiếu kiểm soát, các phần xử lý nghiệp vụ có thể bị phân tán trong UI, gây khó kiểm thử và hạn chế tái sử dụng các component/hook hiện có.



Hình 3: Giao diện quản lý POIs

- **(3) Trang quản lý Tour.** Trang quản lý Tour: Hiện tại hệ thống chưa triển khai giao diện (UI) cho module quản lý tour. Tuy nhiên, do yêu cầu nghiệp vụ "một tour bao gồm nhiều điểm (POIs)", màn hình này dự kiến phải hỗ trợ các chức năng như CRUD POIs vào tour, và quản lý thứ tự điểm trong tour. Khi triển khai nhanh theo vibe coding, nếu thiếu kiểm soát, các quyết định thiết kế và nghiệp vụ có thể bị "nhúng" trực tiếp vào UI, dẫn đến coupling cao, trùng lặp logic, và làm giảm khả năng mở rộng/kiểm thử trong các giai đoạn phát triển tiếp theo.

Tóm lại, trong bối cảnh backend đã hoàn thiện, vấn đề của dự án hiện tại chuyển trọng tâm sang việc xây dựng các màn hình frontend. Do đó, cần một quy trình vibe coding có kiểm soát để vừa triển khai nhanh các màn hình còn thiếu, vừa đảm bảo tính đúng đắn nghiệp vụ, khả năng tái sử dụng components và khả năng bảo trì khi hệ thống mở rộng.

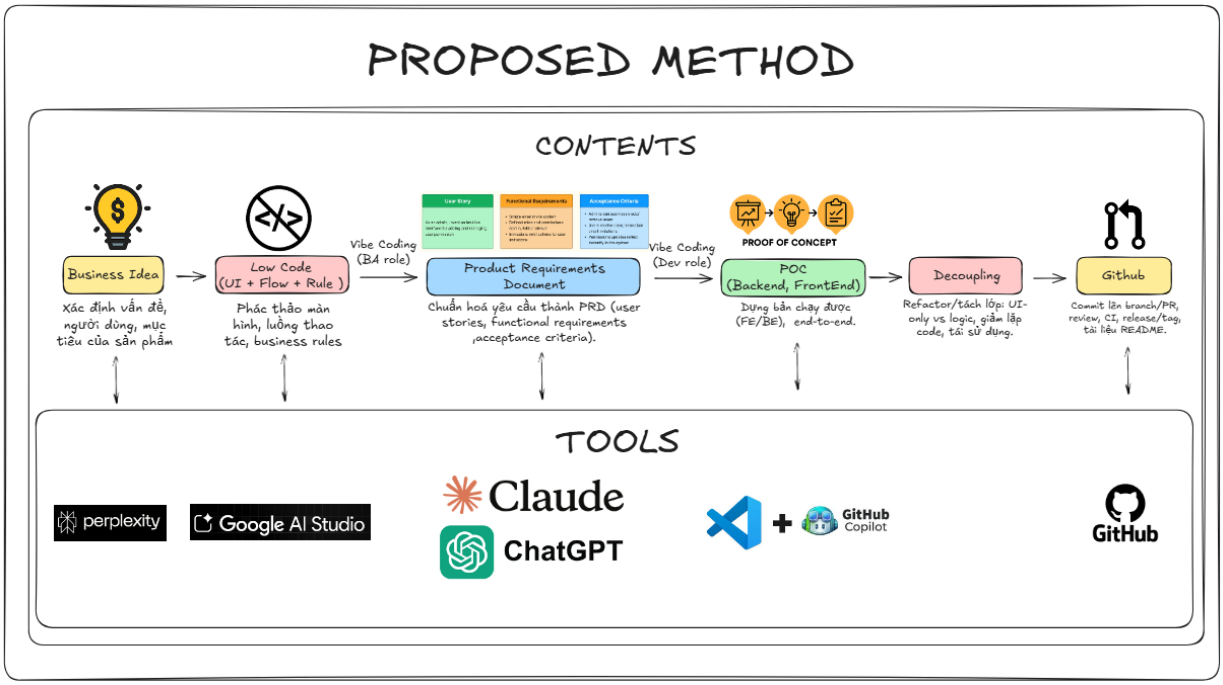
c) Objectives

Mục tiêu của tài liệu này là xây dựng một quy trình mang tính hệ thống về vibe coding trong phát triển sản phẩm nhanh, đồng thời đề xuất một quy trình có kiểm soát nhằm khai thác hiệu quả lợi thế tăng tốc của AI nhưng vẫn đảm bảo chất lượng kỹ thuật khi mở rộng. Cụ thể: Thiết kế quy trình từ ý tưởng đến sản phẩm: Bussiness Idea → Low-code/Flow → PRD → POC → Decoupling → GitHub Delivery. Thể hiện tính khả thi của quy trình bằng việc áp dụng vào một dự án thực tế (trình bày ở phần kết quả/ứng dụng), từ đó rút ra bài học và khuyến nghị.

II/ Proposed method

a) Method Overview

Phương pháp được đề xuất là một quy trình Vibe Coding có kiểm soát nhằm tăng tốc phát triển sản phẩm thông qua sự hỗ trợ của LLM nhưng vẫn duy trì chất lượng khi mở rộng. Điểm cốt lõi của phương pháp là tổ chức toàn bộ theo tiến trình, thay vì triển khai tự phát theo cảm hứng. Hình 4 minh hoạ kiến trúc tổng quan của phương pháp, bao gồm lớp Contents (các bước/đầu ra cần tạo) và lớp Tools (các công cụ AI và công cụ kỹ thuật hỗ trợ thực thi).



Hình 4: Khung phương pháp đề xuất (Proposed Method Framework)

Quy trình gồm 6 giai đoạn tuần tự từ ý tưởng đến bàn giao:

1. **Business Idea:** xác định vấn đề, người dùng mục tiêu và tiêu chí thành công ở mức sản phẩm.
2. **Low-code (UI + Flow + Rule):** phác thảo màn hình, luồng thao tác, và business rules để "chốt" phạm vi và hành vi hệ thống trước khi viết code.
3. **PRD (BA role):** chuẩn hoá yêu cầu thành tài liệu PRD (user stories, functional requirements, acceptance criteria), đóng vai trò "nguồn sự thật" (single source of truth) cho các bước triển khai.
4. **POC (Dev role):** triển khai bản chạy được (frontend/backend) để kiểm chứng luồng nghiệp vụ chính end-to-end.
5. **Decoupling:** tái cấu trúc từ POC sang mã nguồn có khả năng mở rộng: tách UI khỏi logic, giảm trùng lặp, tăng tái sử dụng component, và chuẩn hoá kiến trúc.
6. **GitHub Delivery:** bàn giao theo chuẩn kỹ thuật: commit/branch/PR, review, CI, và cập nhật tài liệu.

b) Tooling support

Để triển khai quy trình một cách nhất quán, phương pháp đề xuất sử dụng kết hợp các công cụ AI và công cụ kỹ thuật, tương ứng với từng giai đoạn trong Hình 4. Các công cụ này không thay thế hoàn toàn vai trò của lập trình viên mà đóng vai trò tăng tốc tạo artifact, hỗ trợ sinh mã, và giảm thao tác lặp, trong khi việc tích hợp, kiểm thử và kiểm soát chất lượng vẫn do con người đảm nhiệm.

1. **Perplexity – hỗ trợ tra cứu:** Công cụ này được sử dụng chủ yếu ở giai đoạn Business Idea, nhằm tổng hợp thông tin, lên ý tưởng cho sản phẩm
2. **Google AI Studio – phác thảo yêu cầu:** Công cụ này được dùng ở giai đoạn Low-code và hỗ trợ PRD để tạo nhanh danh sách màn hình, user flows, business rules và bản nháp user stories/acceptance criteria theo cấu trúc.
3. **ChatGPT / Claude – hỗ trợ chuẩn hoá đặc tả và tạo nội dung có cấu trúc:** Ở giai đoạn (BA role), ChatGPT/Claude được dùng để chuyển đổi flow/rules thành user stories, functional requirements và acceptance criteria để sinh ra tài liệu PRD.
4. **VS Code + GitHub Copilot – hỗ trợ triển khai code trong môi trường IDE:** VS Code và GitHub Copilot được sử dụng ở giai đoạn POC và Decoupling để tăng tốc thao tác coding thực tế: gợi ý code theo ngữ cảnh file, hỗ trợ tạo boilerplate, refactor nhanh.
5. **GitHub – quản lý phiên bản và kiểm soát chất lượng bàn giao:** GitHub là nền tảng trung tâm cho giai đoạn GitHub Delivery, bao gồm quản lý branch/PR, review, và tích hợp CI nhằm đảm bảo các thay đổi chỉ được hợp nhất khi đạt tiêu chí chất lượng (lint/test/build).

Tóm lại, việc kết hợp công cụ theo vai trò giúp tối ưu hiệu quả: các công cụ AI hỗ trợ tạo và chuẩn hoá artifact, trong khi công cụ kỹ thuật (IDE, GitHub/CI) đảm nhiệm việc tích hợp, kiểm thử và kiểm soát chất lượng theo checkpoints của quy trình.

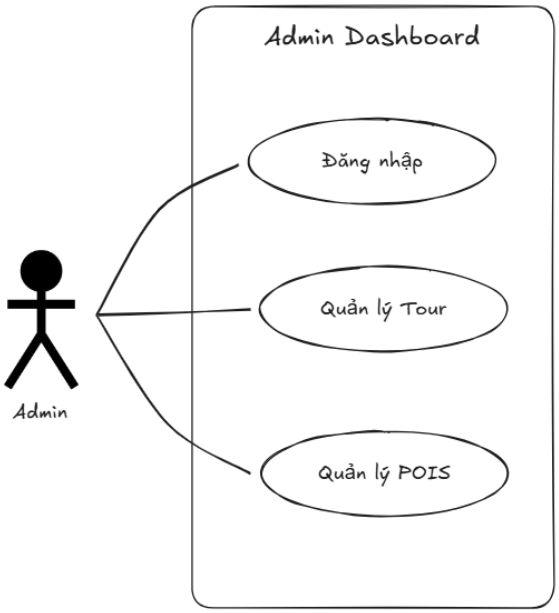
III/ Application results

Quy trình đề xuất như mô tả Hình 4 được áp dụng vào một dự án web admin panel phục vụ quản lý tour và các điểm tham quan (POIs). Ở thời điểm áp dụng, backend đã hoàn thiện các API nghiệp vụ chính, trong khi frontend đã có một số thành phần nền tảng (đặc biệt là map components) nhưng còn thiếu các màn hình quan trọng và việc tích hợp xác thực/API chưa hoàn chỉnh. Quy trình được áp dụng vào dự án như sau:

- **Step 1 – Business Idea / Thu thập yêu cầu (Requirements intake)**

Trong trường hợp này, yêu cầu hệ thống đã được xác định sẵn, do đó giai đoạn requirements intake không cần sử dụng công cụ tra cứu như Perplexity. Đầu vào của bước này là mô tả

nghiệp vụ tổng quan của sản phẩm và phạm vi MVP. Hệ thống được xây dựng dưới dạng admin dashboard phục vụ quản lý POIs và Tours như hình 5, với các yêu cầu cốt lõi như sau:



Hình 5: Use Case Diagram mức hệ thống cho Admin Dashboard

- **Trang đăng nhập (Admin Login):** admin đăng nhập để truy cập vào hệ thống. Sau khi đăng nhập thành công, hệ thống hiển thị giao diện dashboard và các phân hệ quản lý.
- **Quản lý POIs (Points of Interest):** cho phép admin CRUD các điểm trên bản đồ. Admin chọn location trực tiếp trên bản đồ để thiết lập POI và phân loại theo nhóm điểm bao gồm điểm chính và điểm phụ (WC, bán vé, gửi xe, bến thuyền).
- **Quản lý Tours:** cho phép quản trị viên tạo tour dựa trên danh sách POIs đã được tạo từ phân hệ quản lý POIs. Quản trị viên có thể xây dựng một tour chứa nhiều POIs và điều chỉnh thứ tự lộ trình (sắp xếp POIs) để hình thành tuyến tham quan.

Output của Step 1 là yêu cầu ở mức tối thiểu, bao gồm mục tiêu sản phẩm, phạm vi MVP (in/out), và các mô-đun chức năng chính (Login, POIs, Tours) làm cơ sở cho bước Low-code và PRD ở các giai đoạn tiếp theo.

- **Step 2 – Low-code (UI + Flow + Rule)**

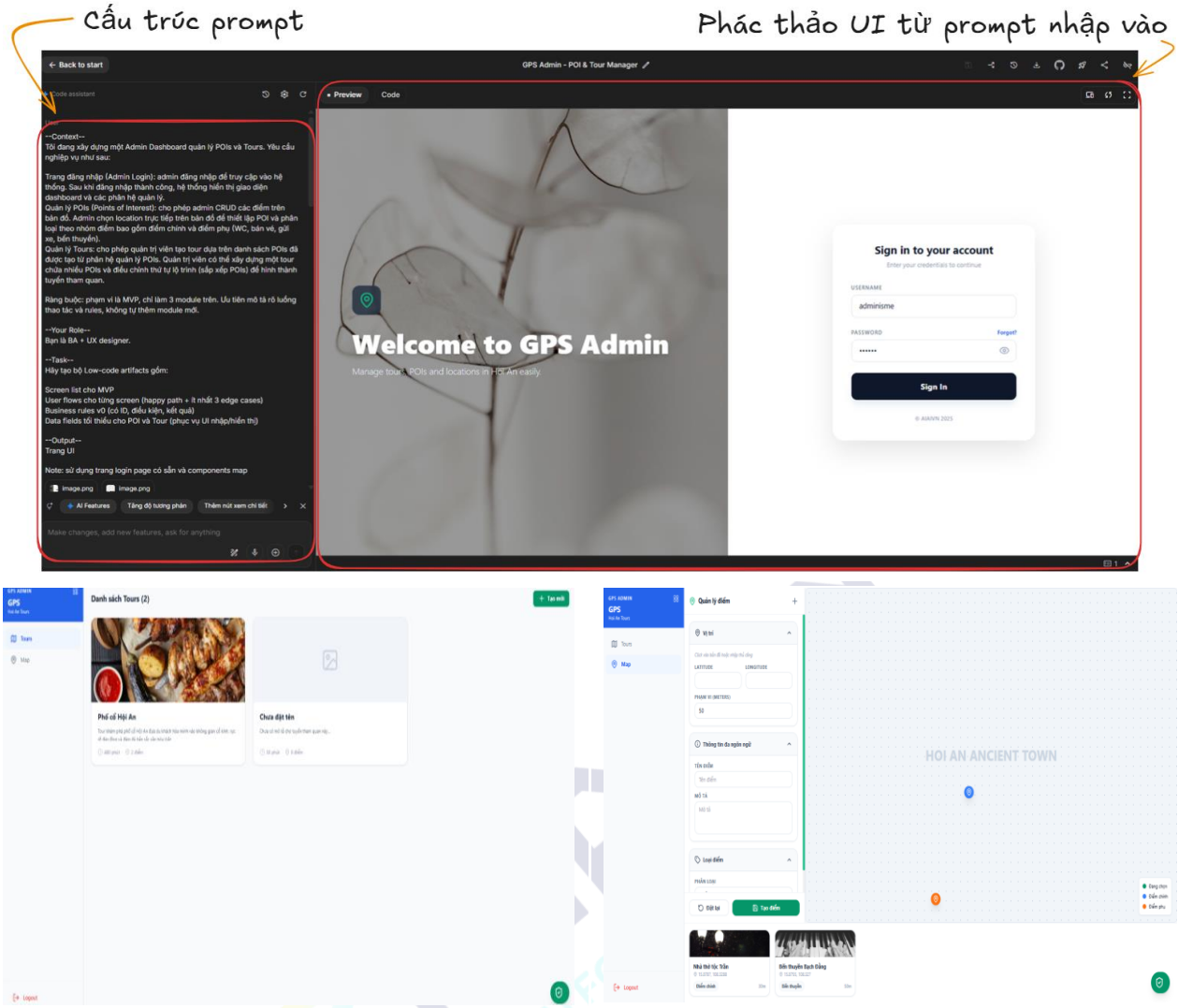
Ở bước Low-code, yêu cầu được chuyển từ mô tả tổng quan kiểm chứng hơn gồm danh sách màn hình, luồng thao tác, và business rules. Ở step 2, đóng vai trò cầu nối giữa yêu cầu (Step 1) và PRD/POC ở các bước tiếp theo.

Trong bước này, ta thực hiện phác thảo hệ thống ở mức "low-code" nhằm chốt nhanh phạm vi MVP trước khi đi vào đặc tả chi tiết và triển khai. Cụ thể, từ yêu cầu **Step 1**, ta sử dụng công cụ Google AI Studio như một môi trường thử nghiệm để chuyển mô tả nghiệp vụ thành bản phác thảo giao diện, workflow người dùng và business rules. Để chuẩn hóa cách tương

tác với LLM và đảm bảo đầu ra nhất quán, ta áp dụng prompt format có cấu trúc Context–Role–Task–Output[3], trong đó (i) Context cung cấp ngữ cảnh và ràng buộc, (ii) Role xác định vai trò, (iii) Task nêu rõ nhiệm vụ cần sinh, và (iv) Output quy định định dạng đầu ra. Cách tổ chức prompt này giúp sinh ra có cấu trúc rõ ràng và có thể dùng trực tiếp làm đầu vào cho bước PRD/POC ở các giai đoạn tiếp theo. Dưới đây là cấu trúc prompt có thể tham khảo:

--Context--
Tôi đang xây dựng [TÊN HỆ THỐNG / DỰ ÁN] thuộc loại [admin dashboard / web app / mobile app].
Mục tiêu MVP: [mục tiêu ngắn 1-2 câu].
Các module trong phạm vi:
1) [MODULE 1]: [mô tả 1-2 câu]
2) [MODULE 2]: [mô tả 1-2 câu]
UI/Components đã có sẵn (Reuse-first):
- Trang đã có: [Login page / Dashboard layout / ...]
- Component đã có: [Map component / Table / Form / Modal / ...]
- Design system: [shadcn / tailwind / custom / ...] (nếu có)
Ràng buộc:
- Scope là MVP, KHÔNG tự thêm module ngoài danh sách trên.
- Ưu tiên tái sử dụng UI/components sẵn có, tránh tạo UI trùng lặp.
- Nếu thiếu thông tin, ghi rõ giả định trong mục Notes.
--Your Role--
Bạn là BA + UX designer.
--Task--
Hãy tạo bộ Low-code artifacts gồm:
1) Screen list cho MVP
2) User flows cho từng screen (happy path + ít nhất 3 edge cases)
3) Business rules v0 (có ID, điều kiện, kết quả)
4) Data fields tối thiểu cho [ENTITY 1] và [ENTITY 2] (phục vụ UI nhập/hiển thị)
5) Reuse notes: gợi ý phần nào dùng lại UI/components có sẵn, phần nào cần tạo mới
--Output--
A) Screen list (bảng: Screen | Purpose | Main UI blocks | Actions | States)
B) User flows (theo từng screen: Happy path + Edge cases)
C) Business rules v0 (bảng: Rule ID | Condition | Expected outcome | Notes)
D) Data fields v0 (bảng: Entity | Field | Type | Required | UI usage)

Hình 6 minh hoạ cơ chế prompt-to-UI trong quy trình Vibe Coding có kiểm soát. Phần bên trái thể hiện **cấu trúc prompt theo định dạng Context–Role–Task–Output**, trong đó ngữ cảnh nghiệp vụ và ràng buộc (MVP, reuse components sẵn có) được cung cấp rõ ràng trước khi yêu cầu tác vụ. Phần bên phải là kết quả phác thảo giao diện được sinh ra từ prompt trong Google AI Studio, bao gồm bố cục trang và các thành phần giao diện cơ bản. Các hình minh hoạ phía dưới cho thấy một số màn hình chức năng được tạo theo cùng cách tiếp cận, giúp chuyển đổi nhanh từ mô tả yêu cầu sang bản phác thảo UI làm đầu vào cho bước PRD/POC tiếp theo.



Hình 6: Minh hoạ quy trình Prompt → Phác thảo UI trong Google AI Studio

Các bước thao tác, video hướng dẫn xem tại đây: Link Step 2

- **Step 3 – Product requirements document (BA role)**

Sau khi Step 2 đã có Screens + Flows + Rules, Step 3 là bước chuẩn hoá yêu cầu thành PRD để dev "build được" và để AI/Dev không tự suy diễn. Đây là một bản prompt template bạn có thể tham khảo dựa trên cấu trúc theo định dạng **Context–Role–Task–Output.**

--Context-- Tôi đã có một UI draft và codebase (đính kèm zip) cho dự án Admin Dashboard quản lý POIs và Tours. Mục tiêu: chuyển những gì đã có trong UI/code thành PRD v1.0 "buildable" để team dev triển khai/hoàn thiện. Phạm vi MVP gồm 3 module: 1) Admin Login 2) POIs Management (CRUD trên map, phân loại major/minor; minor category: WC, bán vé, gửi xe, bến thuyền)

3) Tours Management (tạo tour từ POIs, sắp xếp thứ tự POIs theo lộ trình)

Ràng buộc:
- PRD phải bám sát UI/flow hiện có trong repo, KHÔNG tự thêm module ngoài scope.
- Nếu repo thiếu chỗ nào (ví dụ API chưa rõ), hãy liệt kê thành "Open Questions/Assumptions".

--Your Role--
Bạn là Business Analyst + Product Owner.

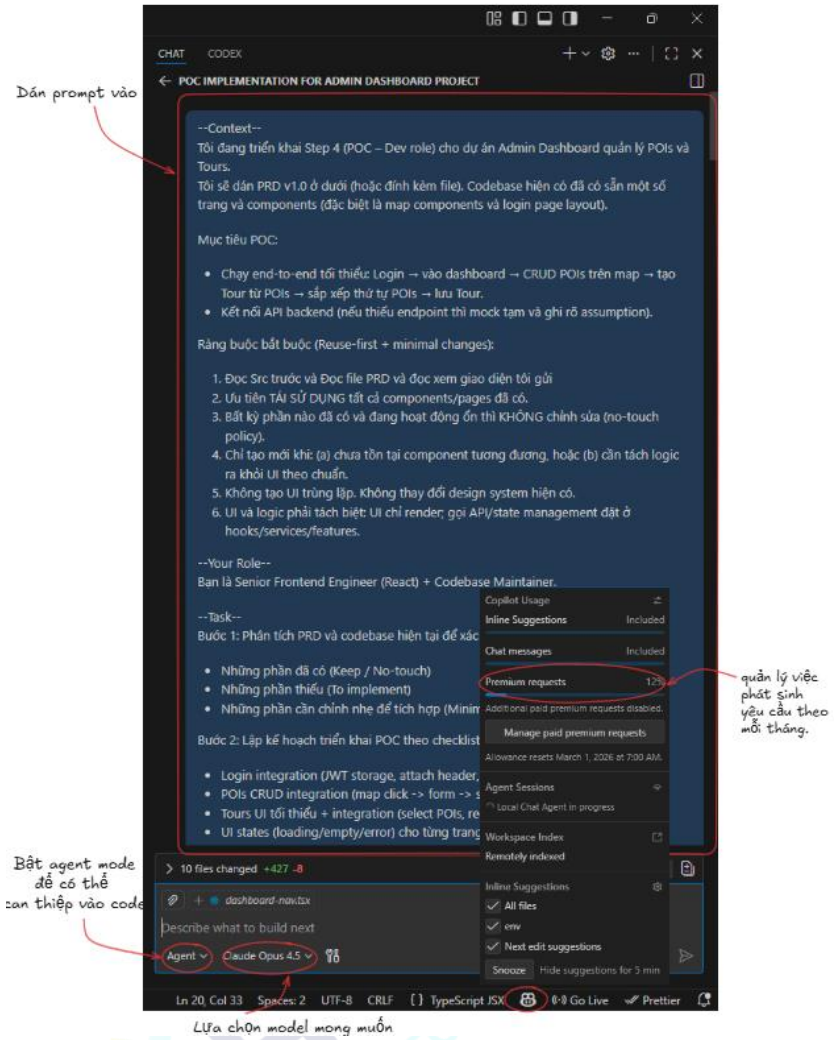
--Task--
1) Đọc cấu trúc project và các màn hình/luồng hiện có trong zip.
2) Suy ra danh sách màn hình, hành vi UI, state (loading/empty/error), và các thao tác chính.
3) Viết PRD v1.0 gồm:
  - Overview + Goals
  - In-scope / Out-of-scope (MVP)
  - Personas/Roles (Admin)
  - User Stories theo từng module
  - Functional Requirements (FR) theo nhóm tính năng
  - Acceptance Criteria (Given-When-Then) cho từng user story quan trọng
  - Non-functional Requirements tối thiểu (auth, validation, error handling, logging, performance cơ bản)
  - Data requirements: field-level cho POI & Tour (chỉ những gì UI cần)
  - API assumptions: các endpoint cần có (chỉ mô tả interface, không cần implement)
  - Dependencies / Risks
  - Open questions

--Output--
- Trả về PRD theo format heading rõ ràng (Markdown).
- Bắt buộc có: bảng User Stories + bảng Acceptance Criteria (Given-When-Then).
- Không viết lý thuyết chung về vibe coding.
- Không thêm tính năng ngoài scope; mọi đề xuất thêm đặt trong mục "Future Enhancements".

Các bước thao tác, xem video hướng dẫn tại đây: Link Step 3

- **Step 4 – Triển khai POC và tích hợp end-to-end (Dev role)**

Ở bước POC, các yêu cầu đã được chuẩn hoá trong PRD (Step 3) được chuyển thành một bản chạy được end-to-end nhằm kiểm chứng luồng nghiệp vụ chính trước khi tái cấu trúc (Decoupling). Trọng tâm của bước này là tạo ra demo tối thiểu nhưng hoạt động, ưu tiên kết nối API thật và xử lý các trạng thái cơ bản trên UI. Cách sử dụng như mô tả Hình 7:



Hình 7: Thiết lập GitHub Copilot Chat (Agent mode) và chọn Claude Sonnet 4.5 để triển khai Step 4 (POC)

Đây là một prompt template bạn có thể tham khảo dựa trên cấu trúc định dạng **Context–Role–Task–Output.**

Note:

- **Trong context có thể đưa UI mà bạn mong muốn từ Step 2**

--Context-- Tôi đang triển khai Step 4 (POC – Dev role) cho dự án Admin Dashboard quản lý POIs và Tours. Tôi sẽ dán PRD v1.0 ở dưới (hoặc đính kèm file). Codebase hiện có đã có sẵn một số trang và components (đặc biệt là map components và login page layout). Mục tiêu POC: - Chạy end-to-end tối thiểu: Login → vào dashboard → CRUD POIs trên map → tạo Tour từ POIs → sắp xếp thứ tự POIs → lưu Tour. - Kết nối API backend (nếu thiếu endpoint thì mock tạm và ghi rõ assumption). Ràng buộc bắt buộc (Reuse-first + minimal changes):

1) Đọc Src trước và Đọc file PRD và đọc xem giao diện tôi gửi
2) Ưu tiên TÁI SỬ DỤNG tất cả components/pages đã có.
3) Bất kỳ phần nào đã có và đang hoạt động ổn thì KHÔNG chỉnh sửa (no-touch policy).
4) Chỉ tạo mới khi: (a) chưa tồn tại component tương đương, hoặc (b) cần tách logic ra khỏi UI theo chuẩn.
5) Không tạo UI trùng lặp. Không thay đổi design system hiện có.
6) UI và logic phải tách biệt: UI chỉ render; gọi API/state management đặt ở hooks/services/features.

--Your Role--
Bạn là Senior Frontend Engineer (React) + Codebase Maintainer.

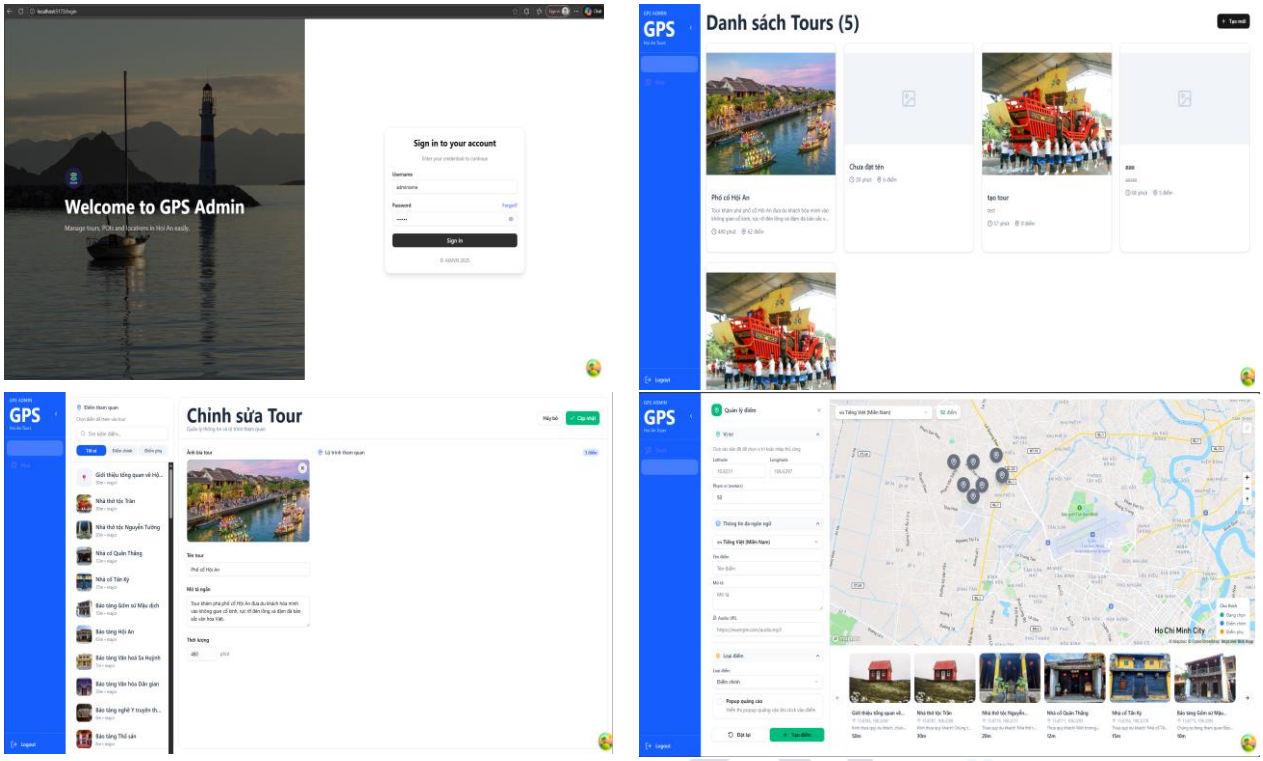
--Task--
Bước 1: Phân tích PRD và codebase hiện tại để xác định:
- Những phần đã có (Keep / No-touch)
- Những phần thiếu (To implement)
- Những phần cần chỉnh nhẹ để tích hợp (Minimal change)

Bước 2: Lập kế hoạch triển khai POC theo checklist (theo PRD) gồm:
- Login integration (JWT storage, attach header, AuthGuard, error states)
- POIs CRUD integration (map click -> form -> save, list sync, delete)
- Tours UI tối thiểu + integration (select POIs, reorder, save)
- UI states (loading/empty/error) cho từng trang

Bước 3: Thực thi bằng cách output:
- Danh sách file cần tạo/sửa (kèm lý do)
- Patch/code theo từng file (copy-paste được)
- Hướng dẫn chạy/test nhanh (5–10 dòng)
- Nếu thiếu thông tin API: đưa mock + “Open questions/Assumptions”

--Output--
Trả theo format sau (bắt buộc):
A) KEEP (No-touch): liệt kê pages/components đã có và không chỉnh
B) IMPLEMENT: liệt kê features/pages thiếu cần làm
C) FILE PLAN: bảng {File | Action(create/update) | Reason | Risk}
D) CODE: cung cấp code/pseudo-code theo từng file (ưu tiên copy-paste)
E) TEST CHECKLIST: smoke test end-to-end theo PRD
F) OPEN QUESTIONS/ASSUMPTIONS

Các bước thao tác, xem video hướng dẫn tại đây: Link Step 4



Hình 8: Kết quả sau khi sử dụng prompt kết hợp PRD và UI được phác thảo

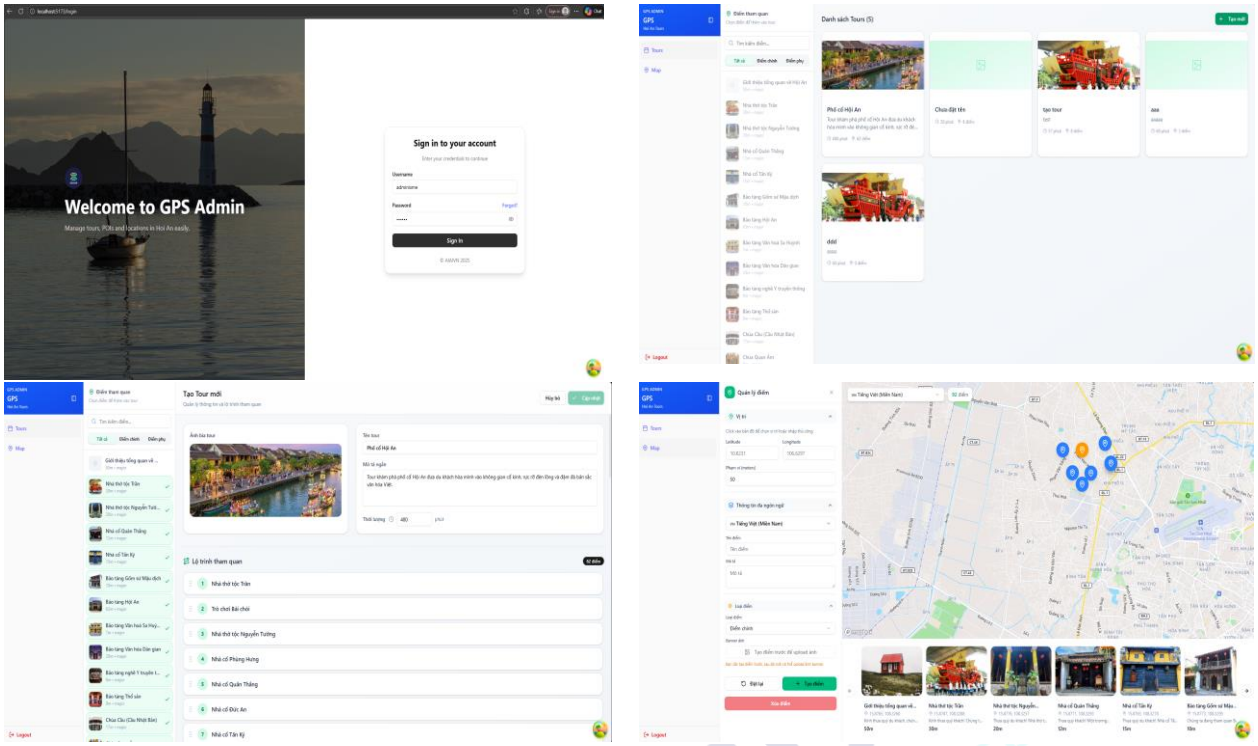
- **Step 5 – Decoupling (Refactor & Modularization)**

Step 5 tập trung vào decoupling nhằm chuyển mã nguồn từ trạng thái "chạy được" sang trạng thái dễ mở rộng và dễ bảo trì. Khác với các bước trước có thể tận dụng AI để tăng tốc tạo artifact hoặc sinh mã, bước này chủ yếu là developer tự chủ động rà soát và chỉnh sửa dựa trên hiểu biết về codebase. Mục tiêu chính là tách biệt trách nhiệm (separation of concerns), giảm trùng lặp và chuẩn hoá kiến trúc theo conventions của dự án.

Các hoạt động chính trong bước decoupling gồm:

- Tách UI và logic: UI components chỉ đảm nhiệm render; logic nghiệp vụ/side-effect (call API, validation, mapping data, state) được đưa về hooks/services/features.
- Reuse-first: loại bỏ copy-paste, gom các phần lặp (form, modal, table, schema/validation) thành component/hook dùng chung.
- Chuẩn hoá cấu trúc module: thiết lập ranh giới rõ cho từng feature (POIs/Tours/Auth), thống nhất naming và folder structure.
- Chuẩn hoá chất lượng: lint/format pass, bổ sung smoke test tối thiểu và đảm bảo build/CI ổn định.

Kết quả của Step 5 là một codebase được "làm sạch" sau POC, giảm tình trạng UI–logic trộn lẫn và hạn chế việc code phình to dẫn đến tình trạng khó bảo trì.



Hình 9: Kết quả sau khi decoupling và cải thiện UI

- **Step 6 – Github**

Sau khi POC đã chạy được (Step 4) và mã nguồn được decouple/tái cấu trúc để dễ mở rộng (Step 5), Step 6 tập trung vào bàn giao và kiểm soát chất lượng thông qua GitHub. Mục tiêu là đảm bảo các thay đổi được quản lý theo chuẩn kỹ thuật (version control), có thể review, và có thể tái hiện (reproducible) thông qua tài liệu và CI.

Các hoạt động chính

- **Quản lý phiên bản:** tổ chức branch theo tính năng (feature branches), commit theo convention và mô tả rõ nội dung thay đổi.
- **Pull Request (PR):** tạo PR kèm mô tả *what/why/how to test*, đính kèm ảnh/video demo (nếu có) và liên kết PRD/issue tương ứng.
- **Review & Quality gates:** thực hiện review và chỉ merge khi đạt tiêu chí tối thiểu (lint/test/build).
- **CI/CD:** cấu hình hoặc sử dụng pipeline hiện có để chạy tự động các bước kiểm tra chất lượng.
- **Documentation:** cập nhật README (cách chạy dự án, env variables), ghi chú thay đổi (changelog/notes), và lưu các artifact quan trọng (PRD link, API docs link).

IV/ Conclusion and Discussion

4.1 Conclusion

Tài liệu này đề xuất một quy trình Vibe Coding có kiểm soát nhằm khai thác LLM để tăng tốc phát triển sản phẩm, nhưng vẫn giữ được tính nhất quán về yêu cầu và khả năng mở rộng khi chuyển từ prototype sang triển khai thực tế. Điểm cốt lõi của phương pháp là cách tiếp cận artifact-driven, trong đó mỗi giai đoạn tạo ra các đầu ra rõ ràng (screen list, user flows, business rules, PRD, POC, cấu trúc module sau decoupling và bàn giao qua GitHub) trước khi chuyển bước.

Khi áp dụng vào bối cảnh dự án admin dashboard quản lý POIs và Tours, quy trình giúp chuyển đổi yêu cầu từ mức ý tưởng sang đặc tả có thể kiểm chứng (PRD) và triển khai POC end-to-end theo lộ trình rõ ràng. Việc chuẩn hoá prompt **Context–Role–Task–Output** và nguyên tắc reuse-first/no-touch cũng hỗ trợ hạn chế thay đổi tuỳ hứng, giảm trùng lặp giao diện và tránh việc viết lại các component đã có.

4.2 Discussion

Mặc dù quy trình đem lại tính hệ thống và khả năng lặp lại, việc áp dụng vẫn phụ thuộc vào chất lượng đầu vào (PRD, API docs, UI hiện có) và kỷ luật thực thi DoD ở từng bước. Trong thực tế, các rủi ro thường gặp là (i) thiếu rõ ràng về API contract dẫn đến sai lệch khi tích hợp, (ii) xu hướng trộn UI và logic khi tăng tốc ở POC, và (iii) khó kiểm soát phạm vi nếu không chốt rules và acceptance criteria từ sớm. Vì vậy, bước Decoupling đóng vai trò quan trọng để “hợp thức hoá” mã nguồn sau POC, đảm bảo cấu trúc module rõ ràng và tăng khả năng tái sử dụng.

TÀI LIỆU THAM KHẢO

[1] Ranjan Sapkota, Konstantinos I. Roumeliotis, and Manoj Karkee, “Vibe Coding vs. Agentic Coding: Fundamentals and Practical Implications of Agentic AI,” arXiv, arXiv:2505.19443, May 26, 2025. doi: 10.48550/arXiv.2505.19443.

[2] Q. T. Hạnh, N. T. Uyên, and V. T. Điệp, “Vibe coding trong phát triển phần mềm: Một tổng quan hệ thống về tác động đến tốc độ, chất lượng và nợ kỹ thuật,” Tạp chí Khoa học Trường Đại học Mở Hà Nội, Số đặc biệt 3A (Tháng 10), 2025, doi: 10.59266/houjs.2025.726.

[3] Đ. Trương Thành, “Các tips prompt hiệu quả cho ae lập trình,” *Viblo*, May 5, 2025. [Online]. Available: https://viblo.asia/p/cac-tips-prompt-hieu-qua-cho-ae-lap-trinh-2oKLnzqZLQO