TRUNG TÂM ĐÀO TẠO AI SAO VIỆT

AI-03

VIBE CODING

DỰNG WEB APP CÙNG AI

Từ ý tưởng đến sản phẩm chạy thật

2026

Dành cho người muốn tự tay
dựng sản phẩm số bằng AI

LỜI NÓI ĐẦU

Bạn có một ý tưởng. Một trang web bán hàng cho cửa hàng của mình, một công cụ nhỏ để quản lý khách, hay một ứng dụng đặt lịch cho phòng khám. Ý tưởng rõ trong đầu, bạn hình dung được gần như từng màn hình: khách vào thấy gì, bấm nút nào, thông tin chạy đi đâu.

Rồi bạn dừng lại ở đúng một chỗ. Bạn không biết viết code.

Từ trước tới nay, giữa một ý tưởng và một sản phẩm chạy được luôn có một bức tường. Muốn qua tường, bạn có vài lối, và lối nào cũng nặng. Thuê người viết phần mềm thì tốn kém, và mỗi lần muốn đổi một dòng chữ hay một cái nút, bạn lại phải chờ họ. Tự đi học lập trình thì mất hàng tháng, và phần lớn người ta bỏ cuộc từ lâu trước khi làm ra được thứ đầu tiên chạy được.

Có một cách khác

Cách này gọi là vibe coding.

Bạn không gõ code. Bạn mô tả bằng tiếng Việt thứ mình muốn, giống như đang nói cho một lập trình viên ngồi cạnh nghe. Một AI Agent nghe mô tả đó, tự viết code, tự tạo file, tự chạy thử. Bạn nhìn kết quả trên màn hình, thấy chỗ nào chưa ưng thì nói lại bằng lời, và nó sửa.

Công việc của bạn chuyển từ "viết code" sang "mô tả cho rõ và kiểm tra cho kỹ". Đó là hai việc bạn làm được ngay hôm nay, không cần học trước tháng nào.

Cuốn sách này đưa bạn tới đâu

Tám bài của cuốn sách bám theo một hành trình có thật: dựng một cửa hàng bán đồ thủ công trên mạng, từ con số không cho tới lúc nó chạy thật và ai cũng vào được bằng một đường link.

Đi hết tám bài, bạn tự làm được những việc sau trên máy của mình:

- Làm một game nhỏ chạy trên trình duyệt để hiểu một trang web được ghép từ gì
- Dựng đầy đủ giao diện một cửa hàng: trang chủ, danh sách sản phẩm, giỏ hàng, trang quản trị
- Hiểu vì sao có lúc phải chuyển sang một công cụ mạnh hơn, và tự tay chuyển
- Đưa dữ liệu thật vào, cho phép khách đặt hàng, cho người quản lý thêm sửa sản phẩm
- Đưa cả cửa hàng lên mạng bằng một đường link gửi được cho bất kỳ ai

Bạn cần chuẩn bị gì

Một máy tính nối được Internet, một tài khoản Google, và tiếng Việt. Công cụ chính là Antigravity, một phần mềm miễn phí cài trên máy, có sẵn AI Agent bên trong. Bài 1 hướng dẫn cài từ đầu.

Bạn sẽ không phải nhớ một câu lệnh máy tính nào. Thứ bạn viết là mô tả công việc, và thứ bạn rèn là hai thói quen: mô tả sao cho AI hiểu đúng, và kiểm tra sao cho mình không nhận nhầm một kết quả hỏng.

Cách dùng bộ tài liệu đi kèm

Mỗi bài đều có phần kiểm tra kết quả với những dấu hiệu cụ thể nhìn thấy được, để bạn tự biết mình làm đúng hay chưa. Mỗi bài cũng có một thư mục mẫu chứa sản phẩm đã hoàn chỉnh của bài đó. Nếu bạn làm theo mà bị kẹt, mở thư mục mẫu ra đối chiếu, hoặc dùng luôn nó để đi tiếp bài sau.

Bức tường "không biết code" đã đứng đó rất lâu. Cuốn sách này không dạy bạn trèo qua nó. Nó chỉ cho bạn thấy cánh cửa vẫn luôn ở đó, và cách mở.

MỤC LỤC

BÀI 1: VIBE CODING LÀ GÌ

Mục tiêu bài học:

- Hiểu vibe coding là gì và nó khác gì với việc tự học viết code
- Biết một phần mềm gồm ba lớp: giao diện, xử lý, và dữ liệu
- Phân biệt được web chạy trên máy mình và web đã đưa lên mạng
- Hiểu ngôn ngữ lập trình là gì, và vai trò của HTML, CSS, JavaScript
- Cài được Antigravity và mở một thư mục làm việc cho nó
- Viết được một yêu cầu theo bốn phần VIỆC, NGUỒN, KẾT QUẢ, LƯU Ý
- Tự tay làm xong một game nhỏ chạy được trên trình duyệt
- Biết cách nói lại khi Agent làm chưa đúng ý

1.1 Vibe coding là gì

Hãy tưởng tượng bạn cần dựng một quầy bán hàng.

Cách thứ nhất: bạn tự học nghề mộc. Mua gỗ, mua cưa, học cưa cho thẳng, học đóng đinh cho chắc. Vài tháng sau bạn mới có cái quầy đầu tiên, và nó thường xộc xệch.

Cách thứ hai: bên cạnh bạn có một người thợ mộc giỏi. Bạn không cầm cưa. Bạn mô tả cho họ: "làm cho tôi một quầy dài hai mét, mặt gỗ, có ba ngăn kéo bên dưới". Họ làm. Bạn nhìn, thấy ngăn kéo hơi nhỏ, nói "ngăn kéo to gấp đôi". Họ sửa.

Vibe coding là cách thứ hai, nhưng người thợ ở đây là một AI, và thứ nó đóng ra là phần mềm.

Bạn không viết code. Bạn mô tả bằng tiếng Việt thứ mình muốn. AI viết code, tạo file, chạy thử. Bạn nhìn kết quả, thấy chưa ưng thì nói lại, nó sửa. Chữ "vibe" ý nói bạn làm việc theo cảm nhận về kết quả: nhìn màn hình, thấy đúng hay sai, rồi điều chỉnh bằng lời, chứ không chúi đầu vào từng dòng lệnh.

Điều này nghe như phép màu, nhưng nó có một cái giá công bằng. Người thợ giỏi tới đâu cũng chỉ làm đúng thứ bạn mô tả. Mô tả mơ hồ thì ra sản phẩm mơ hồ. Vì vậy kỹ năng thật của vibe coding không nằm ở code, mà nằm ở hai chỗ: **mô tả cho rõ** và **kiểm tra cho kỹ**. Cả cuốn sách này xoay quanh đúng hai việc đó.

1.2 Một phần mềm gồm những gì

Trước khi dựng, cần biết mình đang dựng cái gì. Một trang web hay một ứng dụng, dù nhỏ, thường có ba lớp. Hãy hình dung một nhà hàng.

Lớp thứ nhất là phòng ăn. Đây là chỗ khách nhìn thấy và chạm vào: bàn ghế, thực đơn, ánh đèn. Trong phần mềm, lớp này gọi là **giao diện** (tiếng Anh viết tắt là FE,

front-end). Là mọi thứ hiện trên màn hình mà người dùng nhìn và bấm: nút, ô nhập, hình ảnh, danh sách sản phẩm.

Lớp thứ hai là nhà bếp. Khách không vào bếp, nhưng mọi món ăn đều nấu ở đó. Trong phần mềm, lớp này gọi là **xử lý** (viết tắt là BE, back-end). Là phần chạy ngầm: khi khách bấm "đặt hàng", chính lớp này nhận yêu cầu, tính tiền, ghi đơn.

Lớp thứ ba là kho. Nơi cất nguyên liệu, cất sổ sách. Trong phần mềm, lớp này gọi là **dữ liệu** (viết tắt là DB, database). Là nơi lưu danh sách sản phẩm, danh sách khách, các đơn hàng đã đặt. Tắt máy mở lại, dữ liệu vẫn còn đó.

Ba lớp của một web app

Ba lớp này không phải học thuộc. Bạn chỉ cần nhớ chúng để khi đọc các bài sau, bạn biết mình đang làm ở lớp nào. Trong khóa học này, bài 2 và 3 dựng lớp giao diện. Bài 5 và 6 mới đụng tới lớp dữ liệu và lớp xử lý. Cửa hàng đầu tiên của bạn được xây theo đúng thứ tự đó: dựng phòng ăn trước, rồi mới nối bếp và kho.

1.3 Web chạy trên máy mình và web trên mạng

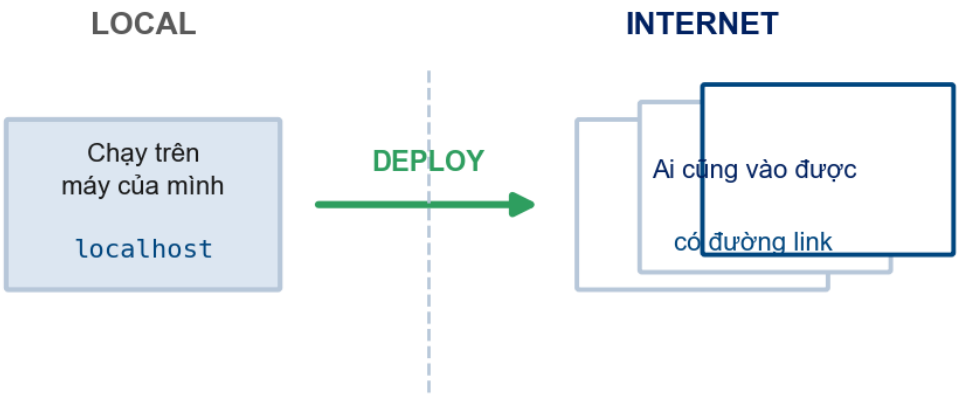
Có một chỗ người mới hay hiểu nhầm, và hiểu sớm sẽ đỡ hụt hẫng về sau.

Khi AI dựng xong trang web đầu tiên cho bạn, nó chạy ngay trên máy tính của bạn. Bạn mở trình duyệt, thấy trang hiện ra đầy đủ, bấm nút chạy tốt. Nhìn thì y như một trang web thật trên mạng.

Nhưng nó chưa ở trên mạng. Nó chỉ đang chạy trong máy bạn, giống một món ăn vừa nấu xong còn để trong bếp nhà bạn. Người khác không nếm được. Nếu bạn gửi địa chỉ trang đó cho một người bạn, họ mở sẽ không thấy gì, vì địa chỉ đó chỉ có ý nghĩa trên máy bạn.

Trạng thái này gọi là chạy **cục bộ**, hay quen hơn là chạy **local**. Địa chỉ của nó thường bắt đầu bằng những con số như `127.0.0.1` hoặc chữ `localhost`. Thấy hai dấu hiệu đó nghĩa là trang đang chạy trong máy bạn, chưa ai ngoài kia vào được.

Để người khác vào được bằng một đường link, bạn phải **đưa web lên mạng**, gọi là **deploy**. Lúc đó trang có một địa chỉ thật, ai gõ vào cũng mở được. Việc này để dành tới bài cuối, khi cửa hàng của bạn đã hoàn chỉnh và đáng để khoe.



Chạy được trên máy mình chưa nghĩa là đã lên mạng

Web chạy local và web đã lên mạng

Nhớ một câu: **chạy được trên máy mình chưa có nghĩa là đã lên mạng.** Suốt bảy bài đầu, bạn làm việc với bản local. Điều đó hoàn toàn bình thường và đúng quy trình.

1.4 Ngôn ngữ lập trình, HTML, CSS và JavaScript

Máy tính không hiểu tiếng Việt hay tiếng Anh. Muốn ra lệnh cho nó, người ta dùng một thứ tiếng riêng gọi là **ngôn ngữ lập trình**. Có nhiều ngôn ngữ lập trình khác nhau, giống như thế giới có nhiều thứ tiếng. Trong khóa học này, cái tên bạn sẽ gặp nhiều nhất là **JavaScript**.

Một trang web được ghép từ ba thứ, và chỉ một trong ba là ngôn ngữ lập trình. Hãy hình dung việc xây một căn nhà.

HTML là phần khung nhà. Nó dựng nên bộ xương: chỗ này là tiêu đề, chỗ kia là một đoạn chữ, đây là một cái nút, kia là một tấm ảnh. HTML nói cho trình duyệt biết trang có những gì. Bản thân HTML không phải ngôn ngữ lập trình, nó chỉ là cách đánh dấu "cái này là cái gì".

CSS là phần sơn và nội thất. Cùng một khung nhà, CSS quyết định nó đẹp hay xấu: màu gì, chữ to nhỏ ra sao, nút bo tròn hay vuông, mọi thứ nằm ở đâu trên trang. CSS cũng không phải ngôn ngữ lập trình, nó chỉ lo phần nhìn.

JavaScript là phần điện nước. Đây mới là ngôn ngữ lập trình thật sự. Nó làm cho căn nhà sống dậy: bấm công tắc thì đèn sáng, bấm nút thì có chuyện xảy ra. Khi bạn bấm "thêm vào giỏ" và số trên giỏ hàng tăng lên, đó là JavaScript đang làm việc.

Chỉ JavaScript là ngôn ngữ lập trình; HTML và CSS thì không

HTML dựng khung, CSS làm đẹp, JavaScript tạo tương tác

Ba thứ này đủ để dựng một trang web hoàn chỉnh, và bạn sẽ làm đúng như vậy ở bài 2. Về sau bạn sẽ nghe tới một cái tên nữa là **Next.js**. Xin nói trước để tránh nhầm: Next.js **không phải một ngôn ngữ lập trình**. Nó là một bộ công cụ dựng sẵn, gọi là **framework**, xây bên trên JavaScript, giúp làm những trang web lớn gọn gàng hơn. Bài 3 sẽ giải thích vì sao có lúc cần tới nó. Bây giờ chỉ cần nhớ: JavaScript là ngôn ngữ, Next.js là bộ đồ nghề dựng trên ngôn ngữ đó.

Tin tốt là bạn sẽ không phải tự gõ một dòng HTML, CSS hay JavaScript nào. AI viết hết. Nhưng biết ba cái tên này để làm gì thì bạn sẽ hiểu AI đang làm gì, và ra lệnh cho nó chính xác hơn nhiều.

1.5 Cài Antigravity và mở chỗ làm việc

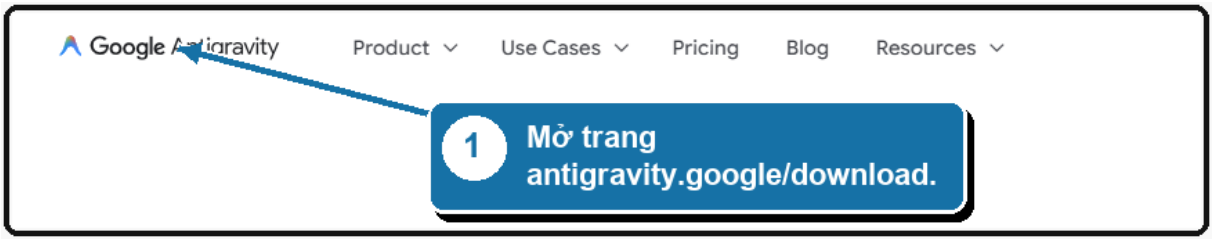
Công cụ bạn dùng suốt cuốn sách là **Antigravity**, một phần mềm miễn phí của Google, cài trên máy, bên trong có sẵn một AI Agent. Phần này cài từ đầu, từng bước một. Bạn nên dành khoảng mười lăm phút và làm liền một mạch, máy cần nối Internet suốt quá trình.

Chặng 1: Tải bộ cài về máy

Bước 1. Mở trang tải. Mở trình duyệt web bạn hay dùng, thường là Chrome hoặc Microsoft Edge. Trên thanh địa chỉ ở đầu cửa sổ, gõ dòng sau rồi nhấn phím Enter:

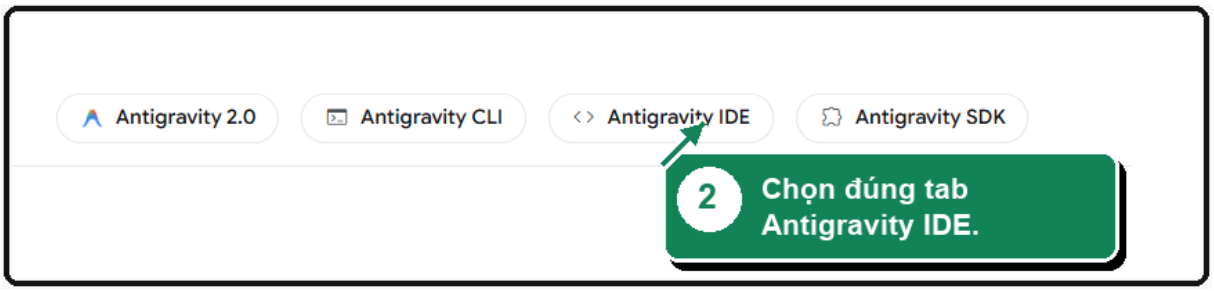
```
antigravity.google/download
```

Bạn sẽ thấy: Trang chủ Antigravity hiện ra, góc trên bên trái có logo hình chữ A nhiều màu.



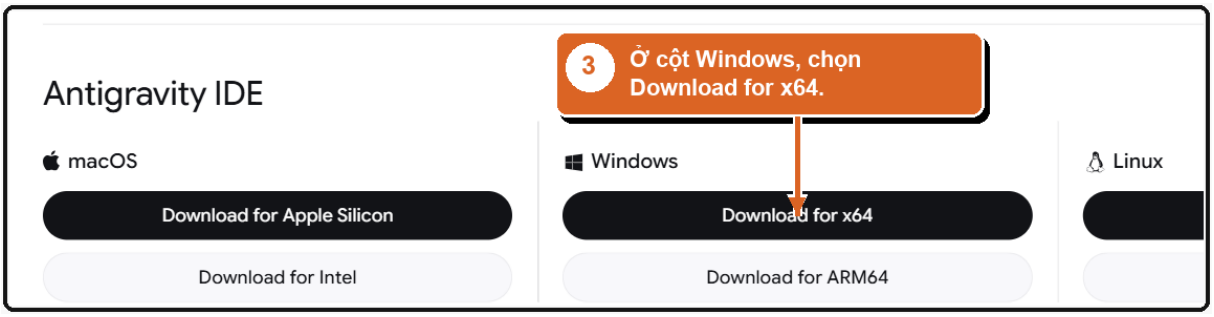
Trang tải Antigravity

Bước 2. Chọn đúng sản phẩm. Trên trang có mấy ô sản phẩm nằm cạnh nhau. Bấm vào ô **Antigravity IDE**. Các ô còn lại dành cho mục đích khác, chọn nhầm thì phần mềm cài ra sẽ không giống hướng dẫn trong sách.



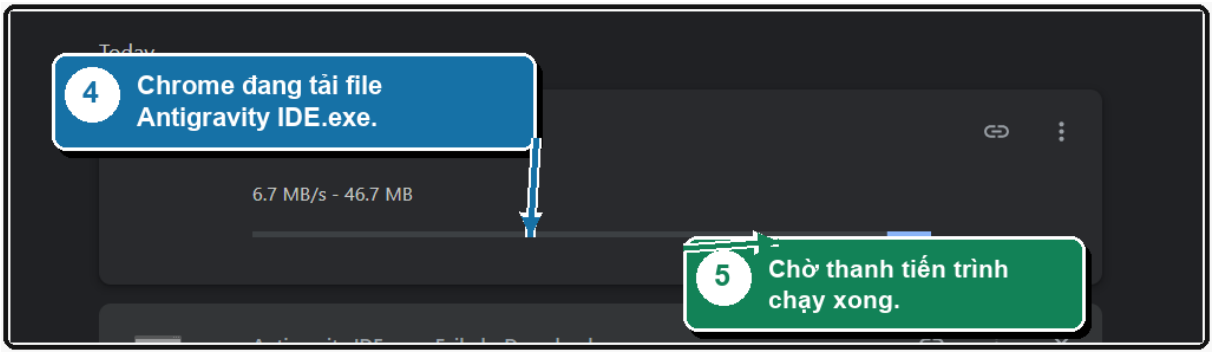
Bốn sản phẩm, chỉ chọn Antigravity IDE

Bước 3. Chọn bản Windows. Kéo trang xuống một chút, bạn thấy ba cột macOS, Windows, Linux. Trong cột **Windows** ở giữa, bấm nút **Download for x64**. Nếu bạn dùng máy Mac thì bấm ở cột macOS bên trái.



Cột Windows, nút Download for x64

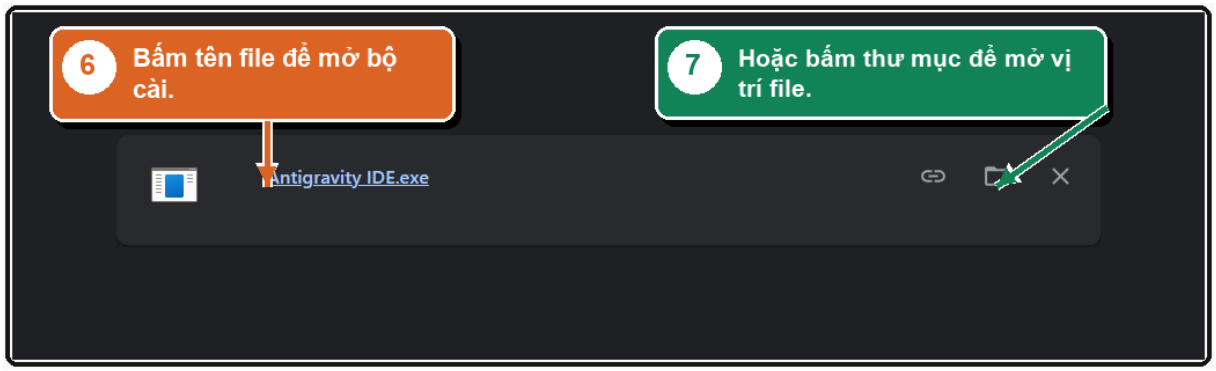
Bước 4. Chờ tải xong. Trình duyệt tải về một file tên **Antigravity IDE.exe**, nặng vài trăm megabyte, mất một tới vài phút. Đừng tắt cửa sổ trình duyệt trong lúc này.



Trình duyệt đang tải bộ cài

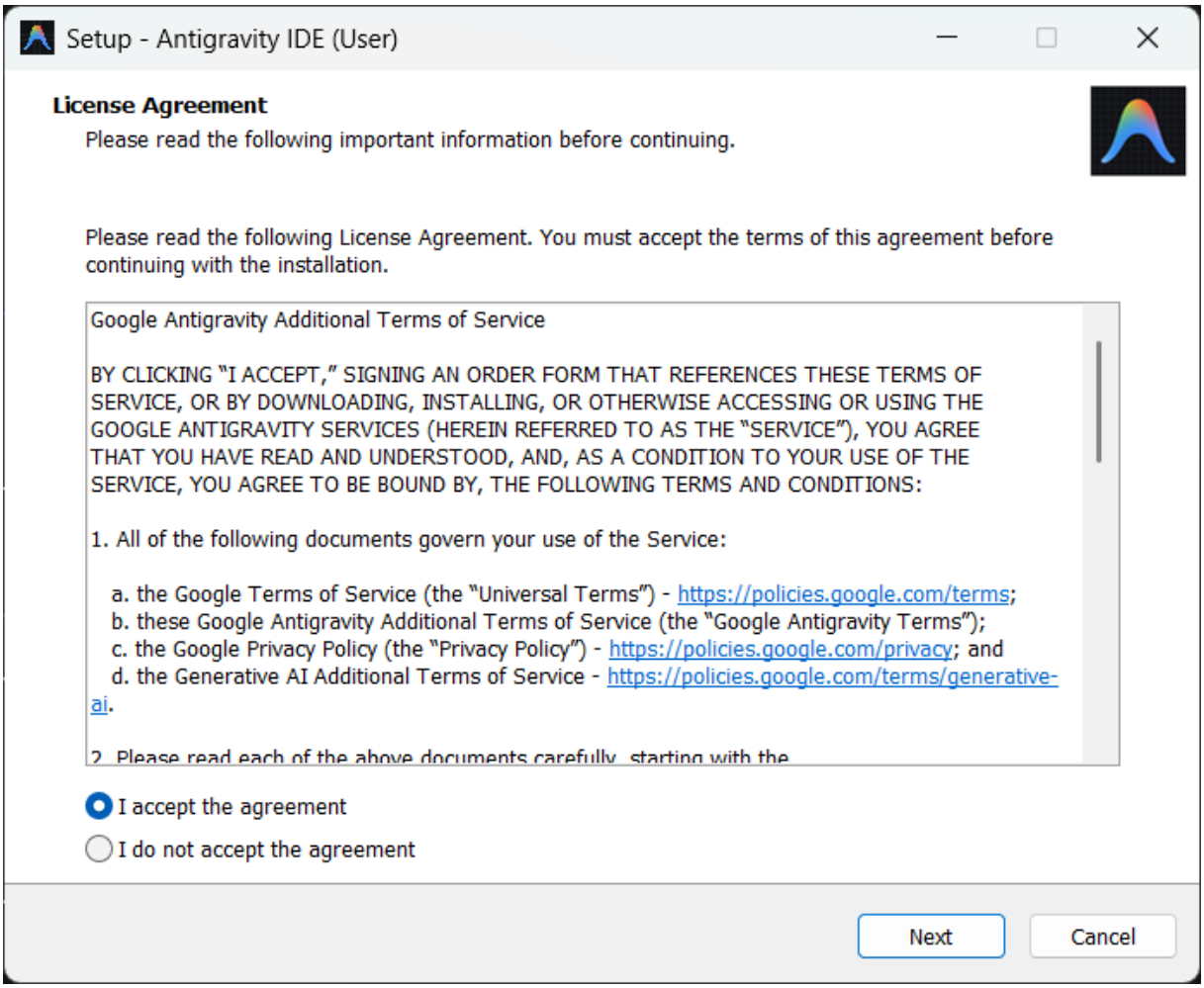
Chặng 2: Chạy bộ cài

Bước 5. Mở file vừa tải. Bấm vào tên file **Antigravity IDE.exe** trong khung thông báo tải của trình duyệt. Nếu Windows hiện hộp thoại hỏi có cho phép ứng dụng thay đổi thiết bị không, bấm **Yes**. Đây là câu hỏi bảo mật bình thường với mọi phần mềm cài mới.



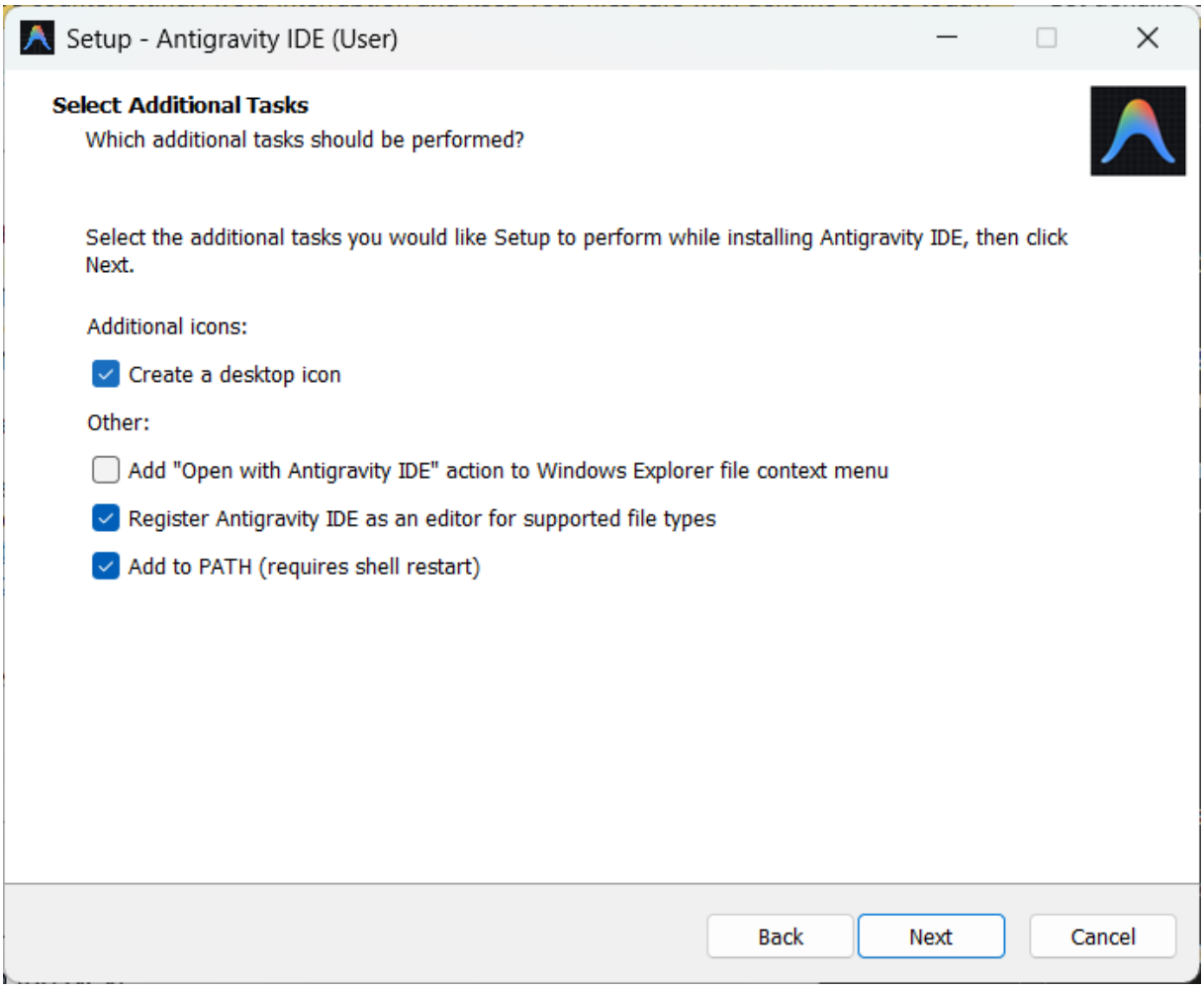
Bấm tên file để mở bộ cài

Bước 6. Đồng ý điều khoản. Màn hình đầu tiên tên **License Agreement** chứa điều khoản bằng tiếng Anh. Bấm vào vòng tròn trước dòng **I accept the agreement**, rồi bấm **Next** ở góc dưới bên phải. Chừng nào chưa chọn dòng này thì nút Next còn mờ, bấm không được.



Màn hình License Agreement

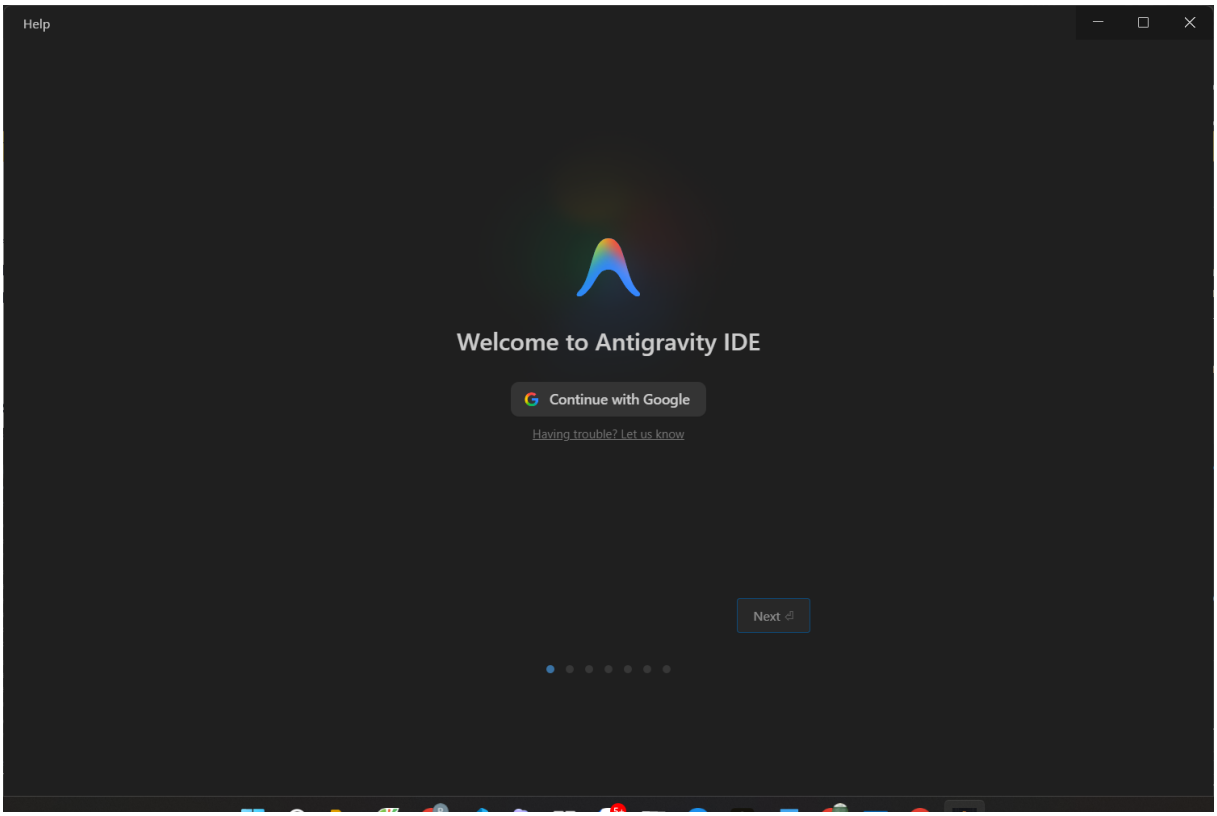
Bước 7. Đi qua các màn cài đặt. Vài màn tiếp theo hỏi nơi cài và tên thư mục, bạn cứ để nguyên lựa chọn có sẵn và bấm **Next**. Tới màn **Select Additional Tasks** có mấy ô đánh dấu, bạn để nguyên các ô đã tích sẵn rồi bấm **Next**. Màn cuối bấm **Install**, chờ thanh chạy đầy, rồi bấm **Finish**.



Màn hình Select Additional Tasks

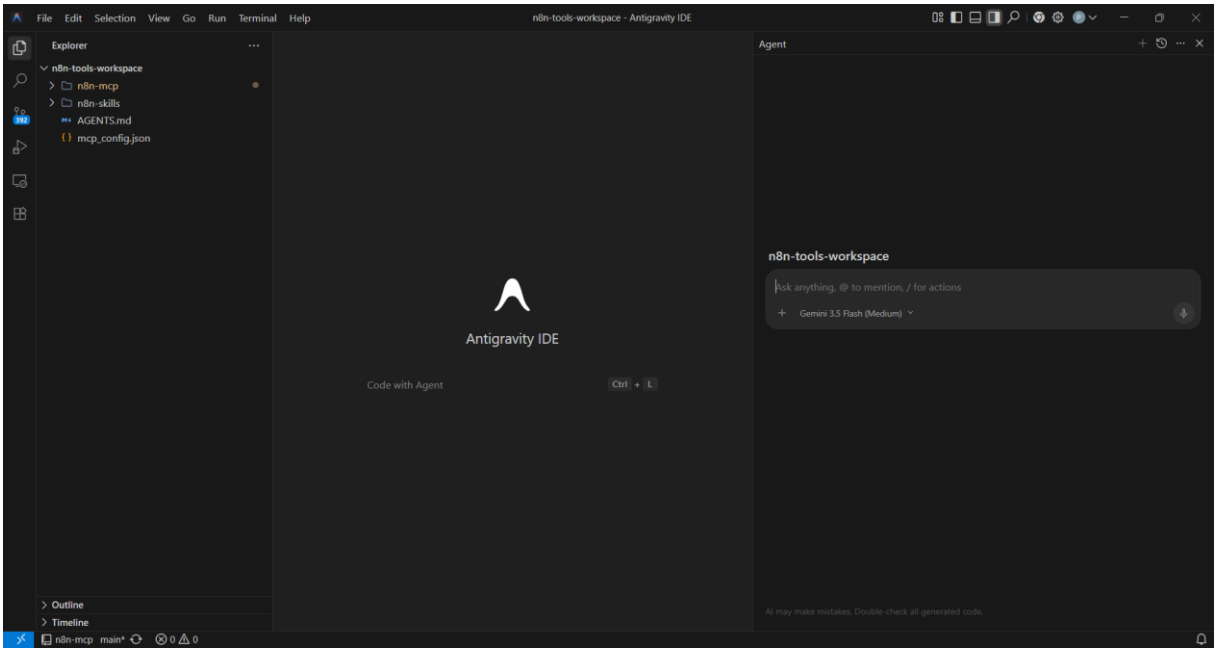
Chặng 3: Đăng nhập

Bước 8. Đăng nhập bằng Google. Antigravity mở lên với màn hình nền tối. Bấm nút **Continue with Google**, chọn tài khoản Google bạn muốn dùng, rồi bấm nút đồng ý cấp quyền. Hãy chọn tài khoản cẩn thận, vì nó gắn với toàn bộ việc về sau. Sau khi đăng nhập, Antigravity dẫn qua vài màn giới thiệu, bạn bấm **Next** cho tới khi giao diện làm việc chính hiện ra.



Màn hình đăng nhập lần đầu

Bạn sẽ thấy: Màn hình chính chia ba phần. Khung bên trái tên **Explorer** là nơi hiện danh sách file. Khung bên phải tên **Agent** là nơi bạn gõ yêu cầu, trông giống một khung chat.



Giao diện làm việc chính

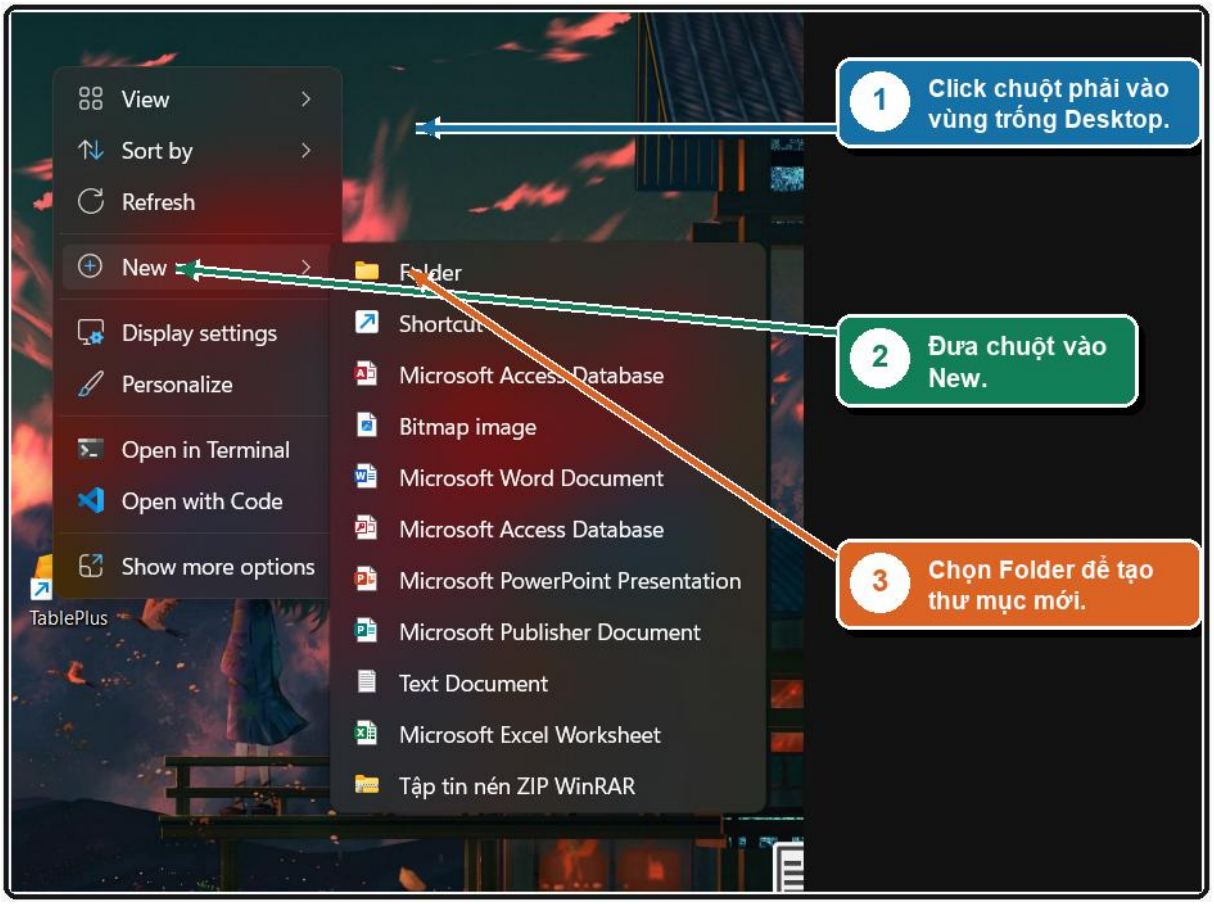
Tạo và mở thư mục làm việc

Antigravity chỉ nhìn thấy những file nằm trong thư mục bạn chỉ định. Vì vậy hãy tạo cho nó một chỗ riêng.

Bước 9. Tạo thư mục trên Desktop. Thu nhỏ các cửa sổ để nhìn thấy màn hình Desktop. Bấm chuột phải vào một khoảng trống, rê chuột vào dòng **New**, rồi bấm **Folder**. Một thư mục mới hiện ra đang chờ đặt tên. Gõ tên sau rồi nhấn Enter:

```
VIBE-CODING
```

Tên viết hoa, không dấu, dùng dấu gạch ngang thay dấu cách. Đây là thói quen tốt cho mọi thư mục làm việc với AI, vì tên có dấu tiếng Việt hoặc có dấu cách đôi khi gây rắc rối.



Tạo thư mục mới trên Desktop

Bước 10. Chỉ thư mục đó cho Antigravity. Quay lại Antigravity. Trên thanh menu sát mép trên cửa sổ, bấm chữ **File**, rồi chọn **Open Folder**. Một cửa sổ chọn thư mục hiện ra. Nhìn cột bên trái bấm **Desktop**, tìm thư mục **VIBE-CODING**, bấm chọn nó một lần, rồi bấm **Select Folder**.

Bạn sẽ thấy: Khung Explorer bên trái hiện tên **VIBE-CODING**. Từ giờ mọi thứ AI tạo ra sẽ nằm trong thư mục này.

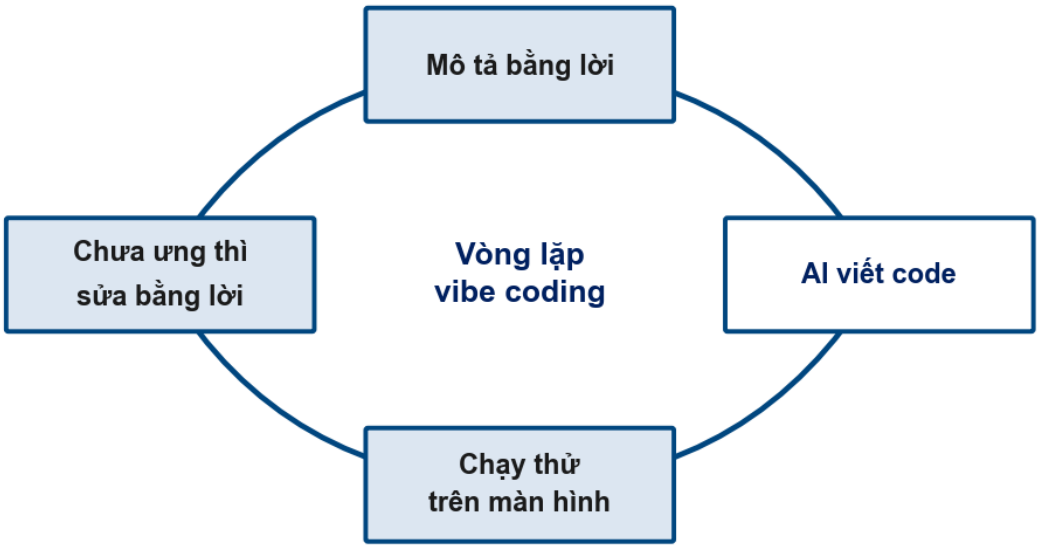
1.6 Cách viết một yêu cầu rõ ràng

Đây là kỹ năng quan trọng nhất của cả khóa học. Mọi thứ còn lại chỉ là áp dụng nó vào từng loại việc.

Người thợ giỏi tới đâu cũng chỉ làm đúng thứ bạn mô tả. Để mô tả cho đủ, hãy tự hỏi bốn câu, và viết câu trả lời thành bốn dòng có nhãn.

Nhãn	Trả lời câu hỏi	Nếu thiếu thì sao
VIỆC	Cần làm cái gì	AI làm một thứ gần giống nhưng không phải thứ bạn cần
NGUỒN	Dựa trên cái gì để làm	AI tự bịa ra dữ liệu, tự chọn công cụ theo ý nó
KẾT QUẢ	Xong thì nhìn thấy gì	AI trả lời bằng chữ, không tạo ra sản phẩm chạy được
LƯU Ý	Có ràng buộc riêng nào	AI tự quyết theo cách của nó, thường không giống ý bạn

Trong bốn nhãn, nhãn **KẾT QUẢ** đáng nói riêng. Với vibe coding, kết quả không phải một đoạn văn, mà là một thứ **nhìn thấy được và chạm được**: một trang mở lên trong trình duyệt, một cái nút bấm vào thì số tăng, một danh sách hiện đúng sản phẩm. Khi viết KẾT QUẢ, hãy tả cái dấu hiệu bạn sẽ nhìn để biết việc đã xong. Đó cũng chính là thứ bạn dùng để kiểm tra ở cuối.



Vòng lặp của vibe coding: mô tả, AI tạo, chạy thử, sửa, lưu lại

Bốn nhãn này theo bạn suốt cuốn sách. Ngay dưới đây, bạn dùng chúng lần đầu để làm một game.

1.7 Làm game đầu tay: Bắt Điểm 30 Giây

Việc đầu tiên nên nhỏ, vui, và thấy kết quả ngay. Bạn sẽ làm một game: những chấm sáng hiện lên khắp màn hình, bấm trúng thì được điểm, bấm nhầm quả bom thì mất một mạng, chơi trong ba mươi giây.

Game Bắt Điểm 30 Giây chạy trên trình duyệt

Trước khi gõ câu đúng, hãy xem chuyện gì xảy ra với một câu viết vội. Đây là cách gần như ai cũng gõ trong lần đầu.

Yêu cầu chưa tốt

```
Làm cho tôi một game bằng code.
```

Câu này có bốn chỗ hổng, và mỗi chỗ dẫn tới một kiểu hỏng:

Chỗ hổng	AI sẽ làm gì
Không nói game gì	Đoán đại một game bất kỳ, thường không phải thứ bạn hình dung
Không nói chạy ở đâu	Có thể tạo ra file không mở lên chơi được
Không nói luật chơi	Tự bịa luật, cách tính điểm không giống ý bạn
Không nói dấu hiệu xong	Bạn không có gì để đối chiếu xem nó làm đúng chưa

Bây giờ viết lại cho đủ. Việc này làm hai bước, và bước đầu chính là chỗ khác biệt của vibe coding thật.

Yêu cầu tốt

Bước 1. Đưa ảnh mẫu cho Agent. Kéo thẳng ảnh game mẫu vào khung Agent, thả vào ô nhập chữ. Ảnh sẽ hiện thành một ô nhỏ ngay trong ô nhập. Đưa ảnh giúp Agent nhìn thấy game trông thế nào, thay vì phải tự đoán.

Bước 2. Gõ yêu cầu ngay sau ảnh. Gõ tiếp đoạn dưới đây, bạn có thể chép nguyên văn:

```
VIỆC: Làm một game bắt điểm theo phong cách trong ảnh tôi vừa gửi, chạy trên
trình duyệt
NGUỒN: Chỉ dùng HTML, CSS và JavaScript, không dùng thư viện ngoài
KẾT QUẢ: Một thư mục có file index.html mở lên chơi được, có nút Bắt đầu,
đồng hồ đếm ngược 30 giây, điểm số và số mạng hiện rõ trên đầu màn hình
LƯU Ý: Chấm xanh bấm trúng thì cộng một điểm, bom bấm nhầm thì trừ một mạng;
hết 30 giây hoặc hết 3 mạng thì dừng và hiện điểm cuối. Giao diện gọn gàng, chữ
tiếng Việt.
```

Gõ xong, nhấn phím Enter. Với một ảnh lẻ như game này, kéo thẳng vào khung chat là gọn nhất. Ở bài sau, khi cần trỏ tới ảnh nằm sẵn trong dự án, bạn sẽ dùng thêm dấu **@**.

Agent sẽ làm gì

Khung bên phải bắt đầu chạy. Bạn sẽ thấy Agent lần lượt báo những việc nó làm: tạo file `index.html` cho khung, `style.css` cho giao diện, `script.js` cho phần chơi. Nó có thể tự mở trang lên để chạy thử. Toàn bộ mất vài chục giây tới một hai phút. Trong lúc đó cứ để yên, đừng gõ thêm.

Chơi thử và kiểm tra

Khi Agent báo xong, mở game lên chơi. Nếu Agent đã tự mở sẵn một cửa sổ xem trước thì bạn chơi luôn ở đó. Nếu chưa, nhìn khung Explorer bên trái, bấm phải vào file `index.html`, chọn dòng mở bằng trình duyệt (thường ghi **Open with Live Server** hoặc **Reveal in File Explorer** rồi bấm đúp để mở).

Đối chiếu năm dấu hiệu sau, đây chính là phần KẾT QUẢ và LƯU Ý bạn đã viết:

Dấu hiệu	Đúng là
Nút Bắt đầu	Bấm vào thì game khởi động
Đồng hồ	Đếm ngược từ 30 về 0
Điểm	Bấm trúng chấm xanh thì tăng
Mạng	Bấm nhầm quả bom thì giảm một
Kết thúc	Hết giờ hoặc hết mạng thì dừng, hiện điểm cuối

Nếu cả năm đều đúng, bạn vừa làm xong phần mềm đầu tiên của mình mà không viết một dòng code nào.

1.8 Khi Agent làm chưa đúng

Lần đầu, kết quả lệch ở đâu đó là chuyện thường. Điều quan trọng là biết cách nói lại. Bạn không cần bắt đầu lại từ đầu, cứ gõ tiếp vào ô nhập, nói rõ chỗ nào chưa đúng, Agent nhớ việc vừa làm và sửa tiếp trên đó.

Game không mở lên được. Nói lại cho rõ nơi chạy:

```
Trang không mở được. Hãy cho tôi một file index.html mở thẳng bằng trình duyệt là chơi được, không cần cài thêm gì.
```

Đồng hồ không chạy, hoặc điểm không tăng. Chỉ đúng chỗ hỏng:

```
Đồng hồ đứng yên không đếm. Hãy sửa để khi bấm Bắt đầu thì đồng hồ đếm ngược từ 30 về 0.
```

Agent làm thừa những thứ bạn không xin. Ví dụ nó tự thêm bảng xếp hạng, âm thanh, nhiều màn. Nói rõ giới hạn:

```
Bỏ hết phần thừa. Chỉ giữ đúng một màn chơi với nút Bắt đầu, đồng hồ, điểm và mạng.
```

Một thói quen nên tập từ bây giờ: sau mỗi lần Agent báo xong, hãy tự tay chạy thử và nhìn kết quả, đừng tin lời báo cáo mà bỏ qua bước nhìn. Ở những bài sau, khi sản phẩm là cả một cửa hàng, thói quen nhìn kỹ này là thứ giữ cho bạn không đưa lên mạng một thứ đang hỏng.

Tóm tắt bài

- Vibe coding là mô tả bằng lời cho AI viết code; kỹ năng thật nằm ở mô tả cho rõ và kiểm tra cho kỹ
- Một phần mềm có ba lớp: giao diện (FE), xử lý (BE), dữ liệu (DB)
- Web chạy trên máy mình gọi là local, địa chỉ có `localhost`; muốn người khác vào được phải deploy lên mạng
- HTML dựng khung, CSS làm đẹp, JavaScript tạo tương tác; chỉ JavaScript là ngôn ngữ lập trình, còn Next.js là framework
- Antigravity chỉ nhìn thấy file trong thư mục bạn mở qua menu File và Open Folder
- Một yêu cầu rõ gồm bốn phần VIỆC, NGUỒN, KẾT QUẢ, LƯU Ý; phần KẾT QUẢ tả dấu hiệu nhìn thấy được để sau này đối chiếu
- Kết quả chưa đúng thì gõ tiếp để sửa, không cần làm lại từ đầu; luôn tự chạy thử trước khi tin là xong

BÀI 2: DỰNG CỬA HÀNG BẰNG HTML, CSS VÀ JAVASCRIPT

Mục tiêu bài học:

- Biết một cửa hàng trực tuyến gồm những màn hình nào, dựng theo thứ tự nào
- Đưa ảnh mẫu giao diện cho AI bằng dấu @ rồi mô tả để nó dựng theo
- Viết prompt để giao diện không bị nhạt nhẽo kiểu AI (tránh AI slop)
- Dựng lần lượt: khung chung, danh sách, lọc, tìm, chi tiết, giỏ hàng, yêu thích, thanh toán
- Dựng được đăng nhập giả và khu quản trị cho người bán
- Biết bổ sung tính năng và chỉnh giao diện bằng những câu prompt ngắn
- Nhận ra chỗ cách làm này bắt đầu đuối, để hiểu vì sao bài sau đổi công cụ

2.1 Một cửa hàng gồm những màn hình nào

Ở bài trước bạn làm một game gói gọn trong một màn. Cửa hàng thì khác, nó có nhiều màn hình nối nhau. Nếu xin AI làm hết trong một câu, bạn sẽ nhận về một mớ rối và không biết chỗ nào hỏng. Vì vậy ta dựng từng màn một, theo thứ tự, xong màn này mới sang màn sau.

Thứ tự dựng có một nguyên tắc: làm phần khách nhìn thấy trước, phần quản trị sau. Khách vào cửa hàng thì xem sản phẩm, bỏ vào giỏ, đặt mua. Người bán thì đăng nhập vào khu riêng để thêm sửa sản phẩm và xem đơn. Dựng phần khách trước vì nó cho bạn thấy kết quả đẹp sớm, có động lực đi tiếp.

Đây là những màn hình bạn sẽ dựng, theo đúng thứ tự:

Nhóm	Màn hình
Khách	Khung chung và trang chủ
Khách	Danh sách sản phẩm
Khách	Lọc theo danh mục và tìm kiếm
Khách	Chi tiết một sản phẩm
Khách	Giỏ hàng
Khách	Sản phẩm yêu thích
Khách	Thanh toán, đặt hàng
Khách	Đăng nhập, đăng ký
Quản trị	Trang tổng quan
Quản trị	Quản lý sản phẩm
Quản trị	Quản lý đơn hàng

2.2 Chuẩn bị: tải ảnh và sắp xếp chỗ làm việc

Cửa hàng bạn sắp dựng tên **Mini Shop**, bán đồ thủ công và trang trí. Để dựng nó, bạn cần một bộ ảnh, tải về ở đường link sau:

Link tải ảnh: Bộ ảnh Mini Shop

Mở link, bạn thấy hai thư mục. Tải cả hai về máy: bấm chuột phải vào tên thư mục rồi chọn **Tải xuống**, Drive sẽ nén lại thành một file rồi bạn giải nén ra.

- **assets**: ảnh các mặt hàng thật (giỏ mây, bình gốm, đèn tre, sofa…) và ảnh banner. Đây là hàng hóa sẽ hiện trên web.
- **giao_dien_web**: vài ảnh chụp mẫu các trang (trang chủ, danh sách, chi tiết, khu quản trị). Đây là ảnh bạn đưa cho AI để nó dựng giao diện trông giống vậy.

Tạo một thư mục mới trên Desktop tên **MINI-SHOP**, rồi chép cả hai thư mục vừa giải nén vào trong đó. Quay lại Antigravity, vào **File** rồi **Open Folder**, chọn thư mục **MINI-SHOP**.

Vì sao ảnh phải xếp đúng chỗ

Mở thư mục **assets** ra, bạn thấy ảnh không nằm lộn xộn mà xếp theo tầng:

```
assets/
└─ images/
   ├─ banner/            (ảnh banner đầu trang)
   └─ products/          (ảnh sản phẩm, chia theo danh mục)
      ├─ do-thu-cong/
      ├─ do-my-nghe/
      └─ noi-that-gia-dung/
```

Vì sao phải xếp gọn như vậy thay vì đổ hết ảnh vào một chỗ? Ba lý do:

- **Bạn tìm nhanh.** Cần ảnh đèn thì vào thẳng nhóm đồ mỹ nghệ, không phải lục giữa hàng chục tấm.
- **AI hiểu đúng.** Khi bạn bảo "dùng ảnh trong thư mục products", AI biết chính xác lấy ở đâu.
- **Web không vỡ ảnh**, và đây là lý do quan trọng nhất, cần nói kỹ.

Ảnh được đưa vào web như thế nào

Một trang web không "chứa" ảnh bên trong. Nó chỉ **trỏ tới đường dẫn** của ảnh, giống như bạn ghi địa chỉ nhà chứ không bê cả căn nhà theo. Trong code, mỗi sản phẩm có một dòng đại ý "ảnh của tôi nằm ở `assets/images/products/do-thu-cong/gio-may-dan.jpg`". Khi mở trang, trình duyệt đi theo địa chỉ đó lấy ảnh về hiện.

Hệ quả: nếu ảnh nằm sai chỗ so với địa chỉ code ghi, ô ảnh sẽ trống hoặc hiện dấu vỡ. Vì vậy **giữ nguyên cấu trúc thư mục assets như khi tải về** là cách an toàn nhất, để code và ảnh khớp địa chỉ với nhau. Bạn không phải tự viết các địa chỉ này, AI làm. Nhưng hiểu cơ chế thì khi thấy ô ảnh trống, bạn biết ngay là "ảnh đặt sai chỗ" và bảo AI sửa.

Đưa ảnh mẫu cho AI bằng dấu @

Các ảnh trong **giao_dien_web** thì không bỏ vào code, mà bạn đưa cho AI xem để nó dựng giao diện theo. Cách đưa: trong ô nhập của khung Agent, gõ dấu **@**. Một danh sách các file trong dự án hiện ra ngay. Bạn chọn đúng ảnh của trang đang làm, ví dụ dựng trang chủ thì chọn `mini-shop-user-home-reference.png`. Tên ảnh được chèn vào ô nhập, và AI biết bạn đang trỏ tới ảnh đó. Mỗi màn dưới đây sẽ nói rõ chọn ảnh nào.

Một điều nên nhớ trước khi bắt đầu

AI hiếm khi làm đúng ý hoàn toàn ngay lần đầu, và điều đó bình thường. Cách làm việc ở đây không phải "ra lệnh một câu rồi xong", mà là **dựng thô rồi chỉnh dần bằng lời**. Bạn dựng một trang, nhìn, rồi gõ tiếp những câu ngắn để sửa: "nút to hơn", "thêm khoảng cách giữa các sản phẩm", "làm cho xem được trên điện thoại". Mỗi màn dưới đây cho bạn một câu để bắt đầu, phần chỉnh sau là của bạn.

Điểm cần kiểm tra trước khi dựng

- Thư mục **MINI-SHOP** đã mở trong Antigravity, bên trong nhìn thấy cả **assets** và **giao_dien_web**.
- Cấu trúc bên trong **assets** giữ nguyên như khi tải, không xáo trộn.
- Gõ thử dấu **@** trong khung Agent, thấy danh sách file hiện ra.

2.3 Dựng khung chung và trang chủ

Đây là màn quan trọng nhất, vì nó đặt ra bộ mặt cho toàn cửa hàng: màu sắc, kiểu chữ, kiểu nút. Dựng đúng một lần ở đây thì mọi màn sau chỉ việc noi theo.

Trước khi gõ câu đúng, hãy xem một câu viết vội sẽ hỏng thế nào.

Yêu cầu chưa tốt

```
Làm cho tôi trang chủ một shop bán hàng cho đẹp.
```

Câu này bỏ trống bốn chỗ, và AI sẽ tự lấp bằng phỏng đoán của nó:

Chỗ hổng	AI sẽ làm gì
Không đưa ảnh mẫu	Tự chế một giao diện bất kỳ, thường không giống thứ bạn hình dung
Không nói dùng ảnh sản phẩm nào	Chèn ảnh vu vơ hoặc để ô trống
"Cho đẹp" nghĩa là gì	Mỗi lần một kiểu, không lần nào giống ý bạn
Không nói cần những phần gì	Thiếu banner, thiếu chân trang, hoặc thừa thứ bạn không cần

Cách chữa là **đưa ảnh mẫu cho AI trước, rồi mới mô tả**. Làm hai bước.

Yêu cầu tốt

Bước 1. Đưa ảnh mẫu. Trong ô nhập của khung Agent, gõ dấu **@** rồi chọn ảnh `mini-shop-user-home-reference.png` (hoặc kéo ảnh vào khung Agent).

Bước 2. Gõ yêu cầu:

```
@mini-shop-user-home-reference.png
VIỆC: Dựng khung chung và trang chủ cho một shop bán đồ thủ công, trang trí,
theo bố cục trong ảnh
NGUỒN: Ảnh mẫu tôi vừa gửi; ảnh sản phẩm trong thư mục assets của dự án
KẾT QUẢ: Trang chủ mở trên trình duyệt: thanh menu trên cùng, banner lớn, lưới
vài sản phẩm nổi bật, chân trang
LƯU Ý: Bám tinh thần thiết kế trong ảnh mẫu – khoảng cách thoáng và đều, một
màu nhấn
nhất quán, bo góc và bóng đổ nhẹ; dùng ảnh sản phẩm thật trong assets, không
dùng ô xám giả;
không emoji trang trí, không nền gradient tím mặc định. Responsive xem được
trên điện thoại;
giá dạng 290.000đ; tách thanh menu và chân trang để các trang sau dùng lại
```

Kiểm tra: Mở trang chủ. Có thanh menu với tên cửa hàng, banner ở đầu, và một lưới sản phẩm hiện ảnh thật cùng giá có chữ **đ**. Chưa ưng chỗ nào thì gõ tiếp một câu, ví dụ "banner cao hơn một chút" hoặc "đổi tông màu nút sang xanh dương".

Giữ giao diện khỏi bị nhạt

Có một điều đặc biệt về phần giao diện: khi bạn để AI tự do, nó hay cho ra những trang na ná nhau và nhạt nhẽo — nền gradient tím, khoảng cách lộn xộn, ô ảnh xám để trống, emoji rải khắp nơi. Nhìn là biết máy làm. Người trong nghề gọi hiện tượng này là **AI slop**, tạm hiểu là "đồ nhạt do AI đổ ra hàng loạt".

Phần giao diện dễ dính AI slop nhất, nên đáng để chặn ngay từ prompt. Cách chặn nằm gọn trong ba việc, và việc quan trọng nhất bạn đã làm rồi:

1. **Luôn đưa ảnh mẫu đẹp và bảo AI bám theo tinh thần của nó.** Một ảnh mẫu tử tế kéo AI ra khỏi lối mòn nhạt nhẽo. Đây là lý do mỗi màn ta đều đưa ảnh trước khi mô tả.
2. **Nói rõ vài tiêu chí chất lượng**, chính là dòng LƯU Ý ở trên: khoảng cách thoáng và đều, một màu nhấn nhất quán, dùng ảnh sản phẩm thật, chữ dễ đọc.
3. **Cấm vài thứ hay gây nhạt**: không nền gradient tím mặc định, không emoji trang trí, không ô ảnh xám giả.

Dòng LƯU Ý về chất lượng này nên có ở mọi màn, không riêng trang chủ. Ở các màn sau, bạn không phải chép lại cả đoạn — chỉ cần thêm "giữ đúng phong cách và chất lượng như các trang đã dựng" là đủ, vì AI đã thấy các trang trước. Và nhớ: giao diện đẹp là kết quả của vài vòng nhìn rồi chỉnh ("khoảng cách giữa các thẻ đều hơn", "chữ tên sản phẩm to hơn chút"), không phải một câu lệnh duy nhất.

2.4 Danh sách sản phẩm

Trang chủ chỉ khoe vài sản phẩm nổi bật. Giờ dựng một trang riêng liệt kê tất cả mặt hàng.

Bước 1. Gõ **@** chọn ảnh `mini-shop-product-list-reference.png`.

Bước 2:

```
@mini-shop-product-list-reference.png
VIỆC: Dựng trang danh sách sản phẩm theo bố cục trong ảnh, hiện tất cả sản phẩm
NGUỒN: Ảnh sản phẩm trong thư mục assets; kiểu thẻ giống lưới ở trang chủ
KẾT QUẢ: Trang hiện lưới các thẻ sản phẩm, mỗi thẻ có ảnh, tên, giá, nút Xem
chi tiết
LƯU Ý: Dùng lại thanh menu và chân trang đã có; giá dạng VND như 290.000đ
```

Kiểm tra: Các sản phẩm hiện thành lưới, ảnh không méo, giá đúng dạng tiền Việt. Ví dụ Giỏ mây đan 290.000đ, Sofa phòng khách vài triệu đồng.

2.5 Lọc theo danh mục và tìm kiếm

Vài món thì còn dễ nhìn, nhưng cửa hàng thật có hàng trăm. Thêm cho khách cách lọc và tìm.

Màn này không có ảnh mẫu riêng, nên mô tả bằng lời và bảo AI làm theo phong cách các trang đã dựng.

```
VIỆC: Thêm bộ lọc theo danh mục và ô tìm kiếm vào trang danh sách sản phẩm
NGUỒN: Các danh mục sẵn có trong dữ liệu; theo phong cách các trang đã dựng
KẾT QUẢ: Bấm một danh mục thì chỉ hiện sản phẩm nhóm đó; gõ tên vào ô tìm thì
lọc dần theo chữ;
bấm Tất cả thì hiện lại đủ mọi sản phẩm
LƯU Ý: Không tải lại trang khi lọc, chỉ ẩn hiện sản phẩm cho mượt
```

Kiểm tra: Bấm một danh mục thì chỉ còn nhóm đó. Gõ "đèn" vào ô tìm thì chỉ còn các sản phẩm đèn. Bấm Tất cả thấy lại đủ.

2.6 Chi tiết một sản phẩm

Khách bấm vào một sản phẩm thì cần một trang riêng xem kỹ.

Bước 1. Gõ **@** chọn ảnh `mini-shop-product-detail-reference.png`.

Bước 2:

```
@mini-shop-product-detail-reference.png
VIỆC: Dựng trang chi tiết sản phẩm theo bố cục trong ảnh
NGUỒN: Sản phẩm được chọn từ danh sách, ảnh trong thư mục assets
KẾT QUẢ: Trang hiện ảnh lớn, tên, giá, mô tả, nút Thêm vào giỏ và nút Yêu thích
LƯU Ý: Bấm một thẻ ở trang danh sách thì mở đúng trang chi tiết của sản phẩm đó
```

Kiểm tra: Bấm một sản phẩm ở danh sách, trang mở ra đúng ảnh và giá của sản phẩm đó, không phải món khác.

2.7 Giỏ hàng

Đây là màn đầu tiên có trí nhớ: thêm món vào giỏ, sang trang khác quay lại giỏ vẫn còn. Màn này không có ảnh mẫu, mô tả bằng lời.

```
VIỆC: Làm giỏ hàng cho cửa hàng
NGUỒN: Nút Thêm vào giỏ ở trang chi tiết và các thẻ sản phẩm; theo phong cách
trang đã dựng
KẾT QUẢ: Bấm Thêm vào giỏ thì số trên biểu tượng giỏ tăng; mở trang giỏ thấy
danh sách món,
tăng giảm số lượng được, có nút xóa và tổng tiền
```

```
LƯU Ý: Giỏ hàng phải nhớ được khi chuyển trang, dùng bộ nhớ trình duyệt để lưu tạm
```

Kiểm tra: Thêm hai món, số trên giỏ thành 2. Sang trang chủ rồi quay lại giỏ, hai món vẫn còn. Tăng số lượng thì tổng tiền tăng theo.

2.8 Sản phẩm yêu thích

Giống giỏ hàng nhưng để khách lưu món thích xem sau.

```
VIỆC: Thêm chức năng yêu thích
NGUỒN: Nút hình trái tim trên mỗi thẻ và trang chi tiết; theo phong cách trang đã dựng
KẾT QUẢ: Bấm trái tim thì nó tô đậm và món được lưu; có trang Yêu thích liệt kê món đã lưu;
bấm lại trái tim thì bỏ khỏi danh sách
LƯU Ý: Danh sách yêu thích cũng nhớ khi chuyển trang như giỏ hàng
```

Kiểm tra: Thả tim ba món, trang Yêu thích hiện đúng ba. Bỏ tim một món, còn hai.

2.9 Thanh toán và đặt hàng

Khách chọn xong thì đặt hàng. Ở bài này chưa có thanh toán thật, chỉ ghi nhận đơn.

```
VIỆC: Làm trang thanh toán và đặt hàng
NGUỒN: Các món đang có trong giỏ hàng; theo phong cách trang đã dựng
KẾT QUẢ: Trang có form họ tên, số điện thoại, địa chỉ, phần tóm tắt đơn và tổng tiền;
bấm Đặt hàng thì hiện thông báo đặt thành công và làm trống giỏ
LƯU Ý: Chưa cần thanh toán tiền thật, chỉ ghi nhận đơn và báo thành công
```

Kiểm tra: Điền form, bấm Đặt hàng, hiện thông báo cảm ơn, giỏ hàng về 0.

2.10 Đăng nhập và đăng ký

Cửa hàng cần phân biệt khách với người bán. Ở bài này ta làm đăng nhập **giả**: chưa nối máy chủ thật, chỉ dựng sẵn màn hình và luồng đi. Đăng nhập thật để dành tới bài 6.

```
VIỆC: Làm trang đăng nhập và đăng ký giả
NGUỒN: Theo phong cách và màu của cửa hàng đã dựng
KẾT QUẢ: Có trang đăng nhập và đăng ký; đăng nhập tài khoản thường thì vào trang khách,
tài khoản admin thì vào khu quản trị
LƯU Ý: Đây là đăng nhập giả chưa nối máy chủ, chỉ cần luồng đi đúng
```

Kiểm tra: Đăng nhập tài khoản thường thì về trang chủ. Đăng nhập tài khoản admin thì mở khu quản trị ở màn sau.

2.11 Khu quản trị cho người bán

Phần khách đã xong. Giờ dựng khu riêng cho người bán, gồm ba màn. Có hai ảnh mẫu cho khu này, đưa lần lượt cho AI.

Bước 1. Gõ @ chọn ảnh `mini-shop-admin-dashboard-reference.png` (và có thể kèm luôn `mini-shop-admin-management-reference.png`).

Bước 2:

```
@mini-shop-admin-dashboard-reference.png @mini-shop-admin-management-
reference.png
VIỆC: Dựng khu quản trị theo bố cục hai ảnh mẫu này
NGUỒN: Ảnh sản phẩm và đơn hàng đã có
KẾT QUẢ: Màn Tổng quan có vài con số thống kê; màn Quản lý sản phẩm là bảng các
sản phẩm
có nút thêm sửa xóa; màn Quản lý đơn hàng liệt kê đơn đã đặt
LƯU Ý: Theo bố cục ảnh nhưng dùng dữ liệu đồ thủ công, giá VND; chưa cần lưu
thật,
sửa xong hiện ngay trên màn là được
```

Kiểm tra: Vào khu quản trị thấy có mấy con số thống kê và một bảng sản phẩm. Thêm thử một sản phẩm thì nó hiện thêm vào bảng.

2.12 Nhìn lại: chỗ này bắt đầu đuối

Dừng lại nhìn thứ vừa dựng. Bạn có một cửa hàng đủ màn, chạy được, đẹp. Đó là thành quả thật.

Nhưng nếu để ý, bạn sẽ thấy vài chỗ bắt đầu nặng nề.

Thanh menu và chân trang bị lặp. Mỗi trang mới, AI phải chép lại nguyên khối menu và chân trang. Muốn đổi một chữ trên menu, bạn phải sửa ở mọi trang. Cửa hàng càng nhiều trang, việc này càng khổ.

Trí nhớ chỉ là tạm bợ. Giỏ hàng và yêu thích lưu trong bộ nhớ trình duyệt, tức là trong máy của đúng người khách đó. Đổi máy khác mở lên là mất. Người bán cũng không nhìn thấy đơn của khách, vì đơn chỉ nằm trong máy khách.

Sửa một chỗ dễ vỡ chỗ khác. Khi mọi thứ nằm rải trong các file HTML lặp nhau, một thay đổi nhỏ dễ kéo theo lỗi ở nơi bạn không ngờ.

Những cái đuối này không phải do bạn làm sai. Chúng là giới hạn tự nhiên của cách dựng bằng HTML, CSS, JavaScript thuần cho một trang web lớn. Bài 3 sẽ chỉ cho bạn một công cụ dựng gọn hơn để gỡ đúng những cái đuối này, và bạn sẽ tự tay chuyển cửa hàng sang đó.

Tóm tắt bài

- Cửa hàng gồm nhiều màn hình, dựng từng màn một theo thứ tự, phần khách trước phần quản trị sau
- Cách dựng mỗi màn: đưa ảnh mẫu cho AI bằng dấu @, rồi mô tả theo bốn phần VIỆC, NGUỒN, KẾT QUẢ, LƯU Ý
- Màn nào không có ảnh mẫu thì mô tả bằng lời và bảo AI làm theo phong cách các trang đã dựng
- Không cần AI làm đúng từng chi tiết ngay; dựng thô rồi chỉnh dần bằng những câu prompt ngắn

- Giao diện dễ bị nhạt kiểu AI (AI slop); tránh bằng cách luôn đưa ảnh mẫu đẹp, nêu tiêu chí chất lượng, và cấm vài thứ gây nhạt (gradient tím, emoji trang trí, ô ảnh giả)
- Giỏ hàng và yêu thích nhớ được nhờ lưu tạm trong bộ nhớ trình duyệt, nhưng đó chỉ là cách tạm
- Cách dựng bằng HTML, CSS, JavaScript thuần bắt đầu đuối khi cửa hàng lớn dần, đó là lý do bài 3 đổi công cụ

BÀI 3: CHUYỂN CỬA HÀNG SANG NEXT.JS

Mục tiêu bài học:

- Hiểu vì sao HTML, CSS, JavaScript thuần đuối dần khi web lớn lên
- Biết framework là gì, Next.js là gì, và khái niệm **component** giải quyết điều gì
- Kiểm kê được cửa hàng bằng một prompt rà soát trước khi chuyển
- Biết nhờ Agent tự cài công cụ còn thiếu trên máy, không phải cài tay
- Tổ chức thư mục dự án gọn gàng: bản mới nằm riêng, không trộn với bản cũ
- Biết bắt AI lên kế hoạch cho việc lớn, đọc duyệt kế hoạch rồi mới cho làm
- Chuyển xong cửa hàng sang Next.js và nghiệm thu bằng đúng bảng kiểm kê

3.1 Nhắc lại ba chỗ đuối

Cuối bài trước, bạn đã có một cửa hàng chạy được nhưng bắt đầu thấy nặng nề ở ba chỗ. Nhắc lại để thấy rõ cái ta sắp gỡ:

- **Thanh menu và chân trang bị lặp** ở mọi trang. Đổi một chữ trên menu phải sửa khắp nơi.
- **Trí nhớ chỉ là tạm bợ**, giỏ hàng nằm trong máy từng khách, người bán không thấy đơn.
- **Sửa một chỗ dễ vỡ chỗ khác**, vì mọi thứ nằm rải trong các file lặp nhau.

Bài này gỡ chỗ thứ nhất và thứ ba bằng một công cụ dựng gọn hơn. Chỗ thứ hai, chuyện dữ liệu, để dành cho bài 5.

3.2 Framework là gì, và Next.js cho bạn thứ gì

Ở bài 1 bạn đã gặp cái tên Next.js một lần, kèm lời hứa sẽ giải thích sau. Giờ là lúc đó.

Hãy nhớ lại việc dựng nhà. HTML, CSS, JavaScript giống như gạch, xi măng, dây điện rời. Bạn dựng được nhà từ vật liệu thô đó, nhưng mỗi bức tường phải xây lại từ đầu. **Framework** là một bộ đồ nghề dựng sẵn: có khung nhà đúc sẵn, có những mảng tường lắp ghép, có đường điện đi sẵn. Bạn ghép nhanh hơn nhiều, và các mảng ghép với nhau cho khớp.

Next.js là một framework như vậy, xây bên trên JavaScript. Nó cho cửa hàng của bạn nhiều thứ, nhưng thứ quan trọng nhất, thứ gỡ đúng cái đuối của bạn, gọi là **component**.

Component: dựng một lần, dùng nhiều nơi

Một component là một mảng giao diện bạn dựng một lần rồi dùng lại ở mọi nơi cần. Thanh menu là một component. Chân trang là một component. Thẻ sản phẩm cũng là một component.

Khác biệt nằm ở đây. Ở bản cũ, mỗi trang chứa một bản sao của thanh menu. Ở Next.js, mọi trang cùng gọi tới **một** component thanh menu. Sửa component đó một lần, mọi trang đổi theo ngay. Cái khổ "đổi menu phải sửa khắp nơi" biến mất.

Bạn không phải học cách viết component. AI viết. Nhưng hiểu ý niệm này thì bạn ra lệnh đúng: thay vì nói "chép menu vào trang này", bạn nói "dùng lại component menu đã có".

3.3 Kiểm kê trước khi chuyển

Trước một việc lớn như dọn nhà, người cẩn thận đi một vòng đếm lại đồ đạc. Chuyển cửa hàng cũng vậy: kiểm kê trước, để không chuyển thiếu.

Có một điều cần biết ở đây, và biết rồi bạn sẽ tránh được nhiều bực mình về sau: **AI không nhớ các cuộc trò chuyện cũ.** Hôm nay mở chat mới, nó không nhớ ảnh bạn đưa tuần trước, không nhớ những gì hai bên đã bàn ở bài 2. Thứ nó luôn đọc được là những gì **đang nằm trong thư mục làm việc**: toàn bộ code và ảnh của cửa hàng. Nói cách khác, trí nhớ thật của dự án là file, không phải cuộc chat. Vì vậy ở bài này ta không cần đưa lại ảnh mẫu nào cả — ta trỏ thẳng vào code cũ, vì code chính là bản mô tả đầy đủ nhất của cửa hàng.

Mở lại thư mục **MINI-SHOP** trong Antigravity (menu **File**, chọn **Open Folder** nếu chưa mở sẵn), mở cuộc trò chuyện mới, rồi giao việc kiểm kê:

```
VIỆC: Rà soát cửa hàng trong thư mục này xem đã đủ các phần chưa, chưa cần sửa gì
NGUỒN: Toàn bộ code HTML, CSS, JavaScript đang có của dự án
KẾT QUẢ: Một bảng đối chiếu theo danh sách sau, mỗi mục ghi rõ có hay thiếu: trang chủ, danh sách sản phẩm, lọc và tìm kiếm, chi tiết sản phẩm, giỏ hàng, yêu thích, thanh toán, đăng nhập giả, admin tổng quan, admin sản phẩm, admin đơn hàng
LƯU Ý: Chỉ kiểm tra và báo cáo, không tự ý thêm hay sửa code
```

Danh sách mười một mục trong prompt chính là bảng màn hình của bài 2. Nó sẽ được dùng lại lần nữa ở cuối bài, khi nghiệm thu bản mới.

Kết quả rà soát rẽ hai nhánh:

- **Đủ cả mười một mục:** sang phần tiếp theo.
- **Thiếu mục nào đó** (chuyện bình thường nếu bạn làm bài 2 chưa trọn): bảo AI làm bù đúng theo nếp bài 2 — mô tả bốn phần, và nếu màn đó có ảnh mẫu trong thư mục `giao_dien_web` thì gõ **@** chọn ảnh. Làm xong màn thiếu, chạy lại prompt kiểm kê cho tới khi đủ.

3.4 Nhờ Agent cài Node.js

Next.js cần một công cụ nền tên **Node.js** để chạy. Bạn không cần hiểu Node.js là gì, và cũng không cần tự cài. Nguyên tắc từ giờ về sau: **việc gì Agent tự làm được trên máy, cứ để Agent làm.** Cài phần mềm nền là một việc như vậy.

Gõ vào khung Agent:

```
Kiểm tra máy tôi đã có Node.js chưa. Chưa có thì cài giúp tôi luôn, xong báo
lại phiên bản đã cài.
```

Nếu máy đã có, Agent báo phiên bản và bạn đi tiếp ngay. Nếu chưa, Agent tự tải và cài.

Có thể xuất hiện: Windows hiện hộp thoại hỏi có cho phép ứng dụng thay đổi thiết bị không. Bấm **Yes**. Đây là bước duy nhất bạn phải tự bấm, vì hộp xin quyền này Windows chỉ cho con người bấm.

Kiểm tra: Agent báo lại một số phiên bản, ví dụ `v22`. Thấy số phiên bản là xong.

3.5 Chỗ ở cho bản mới, và bắt AI lên kế hoạch

Bản mới nằm riêng một thư mục

Trước khi cho AI bắt tay, phải trả lời một câu: code Next.js sẽ nằm ở đâu?

Câu trả lời dứt khoát: **trong một thư mục con mới, tên `mini-shop-next`, nằm ngay trong MINI-SHOP.** Toàn bộ code mới gói gọn trong đó, không một file nào trộn vào chỗ cũ. Thư mục lúc này trông như sau:

```
MINI-SHOP/
├─ index.html, style.css, script.js...   (bản cũ, giữ nguyên làm mẫu đối chiếu)
├─ assets/            (ảnh của web, hai bản dùng chung)
├─ giao_dien_web/     (ảnh mẫu giao diện)
└─ mini-shop-next/    (bản Next.js mới, toàn bộ code mới nằm ở đây)
```

Tổ chức như vậy vì ba lẽ. Một, cũ và mới không trộn lẫn, nhìn tên thư mục biết ngay file thuộc bản nào. Hai, bản cũ còn nguyên bên cạnh để đối chiếu từng màn khi nghiệm thu. Ba, lỡ bản mới hỏng nặng, xóa nguyên thư mục `mini-shop-next` làm lại là xong, không sợ mất gì. Đây là thói quen tổ chức dự án nên giữ suốt về sau: **mỗi bản, mỗi thứ, một chỗ riêng có tên rõ ràng.**

Việc lớn thì bắt AI lên kế hoạch trước

Ở bài 1, một việc nhỏ cần một prompt. Ở bài 2, mỗi màn một prompt. Việc lần này lớn hơn hẳn: chuyển cả một cửa hàng mười một hạng mục. Với việc cỡ này có một cách làm mới: **bắt AI lên kế hoạch trước, mình đọc duyệt, rồi mới cho làm.** Giống như thuê thợ sửa cả căn nhà, không ai để thợ đập tường ngay, phải xem bản kế hoạch đã.

Trước khi gõ câu đúng, xem câu viết vội hỏng ra sao.

Yêu cầu chưa tốt

```
Chuyển web của tôi sang Next.js.
```

Chỗ hổng	AI sẽ làm gì
Không kiểm kê trước	Chuyển thiếu màn mà không ai phát hiện
Không nói code mới nằm đâu	File mới cũ trộn một chỗ, rối không gỡ nổi
Không bắt lên kế hoạch	Lao vào code ngay, việc quá lớn, hỏng giữa chừng khó lần lại

Chỗ hổng	AI sẽ làm gì
Không nói nguồn là code cũ	Tự chế lại theo ý nó, khác hẳn bản bạn đã ưng

Kiểm kê thì bạn làm rồi. Ba chỗ còn lại, gói vào một prompt:

Yêu cầu tốt

```
VIỆC: Lên kế hoạch chuyển cửa hàng này sang Next.js, chưa code vội
NGUỒN: Toàn bộ code HTML, CSS, JavaScript đang có trong thư mục này
KẾT QUẢ: Một bản kế hoạch liệt kê các bước làm, danh sách trang sẽ chuyển,
và những component dùng chung sẽ tách ra
LƯU Ý: Toàn bộ code mới nằm gọn trong thư mục con mini-shop-next, không đụng
code cũ;
giữ nguyên giao diện như bản cũ; dùng lại thư mục assets; lên xong kế hoạch thì
dừng chờ tôi duyệt
```

Đọc kế hoạch trước khi gật đầu

Agent trả về một bản kế hoạch nhiều bước. Bạn không cần hiểu hết chữ nghĩa kỹ thuật trong đó. Chỉ cần soi ba điều:

1. **Đủ màn chưa** — kế hoạch có nhắc đủ mười một mục trong bảng kiểm kê không.
2. **Có tách component không** — có dòng nào nói tách thanh menu, chân trang, thẻ sản phẩm thành phần dùng chung không.
3. **Có giữ đồ cũ không** — có dùng lại thư mục assets, có giữ giỏ hàng và yêu thích không.

Thiếu điều nào, gõ một câu bổ sung, ví dụ: "Bổ sung vào kế hoạch: tách thanh menu và chân trang thành component dùng chung." Kế hoạch ổn rồi mới sang bước chạy.

3.6 Thực thi và nghiệm thu

Cho AI chạy theo kế hoạch bằng một câu ngắn:

```
Kế hoạch ổn. Làm theo đúng kế hoạch, xong bước nào báo bước đó.
```

Đây là lượt chạy dài nhất từ đầu sách tới giờ. Agent sẽ tạo thư mục `mini-shop-next`, tự cài các gói cần thiết, rồi dựng dần từng trang theo kế hoạch — có thể mất mười tới hai mươi phút. Cứ để nó chạy, đừng ngắt giữa chừng. Nếu nó dừng lại hỏi, đọc câu hỏi rồi trả lời cho nó làm tiếp. Xong việc, nó mở bản xem trước ở địa chỉ dạng `localhost:3000`.

Nghiệm thu bằng đúng bảng kiểm kê

Mở bản mới lên và đi lại đúng danh sách mười một mục đã dùng ở đầu bài: trang chủ, danh sách, lọc và tìm, chi tiết, giỏ hàng, yêu thích, thanh toán, đăng nhập giả, ba màn admin. Mỗi mục bấm thử vài cái: thêm món vào giỏ xem số có tăng, thả tim xem trang yêu thích có nhận, đặt thử một đơn.

Muốn chắc tay hơn, mở bản cũ và bản mới cạnh nhau, so từng màn. Hai bản phải trông như nhau; khác biệt nằm ở bên trong code, không nằm trên màn hình.

Có màn nào hỏng hoặc thiếu, chỉ đích danh cho Agent: "Trang giỏ hàng chưa cộng tổng tiền, sửa lại." Xong hết danh sách, cuộc chuyển nhà kết thúc.

3.7 Nhìn lại: bạn được gì sau khi chuyển

Đứng lùi một bước nhìn thành quả.

Cửa hàng trông vẫn như cũ với khách, nhưng bên trong đã khác hẳn. Thanh menu giờ là **một** component. Thử bảo Agent "đổi tên cửa hàng trên menu thành Mini Shop Decor", bạn sẽ thấy nó sửa đúng một chỗ và mọi trang cùng đổi theo. Cái khổ "sửa khắp nơi" đã hết. Thử xong, bảo nó đổi lại tên cũ.

Bạn cũng vừa học được cách làm việc mới với AI, đáng giá hơn cả bản Next.js: **việc lớn thì kiểm kê trước, bắt lên kế hoạch, duyệt rồi mới cho chạy, và nghiệm thu bằng đúng danh sách đã kiểm kê.** Cách này dùng lại được cho mọi việc lớn về sau, ở bất kỳ công cụ nào.

Còn một chỗ đuối chưa gỡ: dữ liệu vẫn là tạm bợ. Giỏ hàng vẫn nằm trong máy từng khách, sản phẩm vẫn ghi cứng trong code chứ chưa nằm ở một kho chung. Bài 4 lo chuyện cất giữ và chia sẻ công việc cho an toàn, rồi bài 5 mới đưa dữ liệu thật vào.

Tóm tắt bài

- Trí nhớ thật của dự án là file trong thư mục làm việc, không phải cuộc chat; giao việc lớn thì trỏ AI vào code, không cần đưa lại ảnh
- Trước việc lớn phải kiểm kê: một prompt rà soát theo danh sách cố định, thiếu gì bổ sung trước
- Việc gì Agent tự làm được trên máy thì để Agent làm, kể cả cài Node.js; bạn chỉ bấm hộp xin quyền của Windows
- Code bản mới nằm gọn trong thư mục con riêng, không trộn với bản cũ; mỗi thứ một chỗ có tên rõ ràng
- Việc lớn thì bắt AI lên kế hoạch, đọc duyệt ba điều (đủ màn, tách component, giữ đồ cũ) rồi mới cho chạy
- Nghiệm thu bằng đúng bảng kiểm kê ban đầu, so bản mới với bản cũ từng màn
- Component giúp sửa một chỗ, mọi trang đổi theo; chuyện dữ liệu thật để dành bài 5

BÀI 4: CẤT GIỮ CÔNG SỨC VÀ TRANG BỊ THÊM NGHỀ CHO AI

Mục tiêu bài học:

- Hiểu GitHub là gì và vì sao dự án nào cũng nên nằm trên đó
- Đưa được cửa hàng lên GitHub, gần như toàn bộ bằng prompt
- Tập được thói quen lưu mốc sau mỗi buổi làm việc
- Hiểu skill là gì, vì sao nên trang bị skill cho AI
- Cài được hai bộ skill bằng prompt và thấy khác biệt khi dùng

4.1 Công sức của bạn đang mong manh

Hãy làm một phép thử tưởng tượng. Tối nay máy tính của bạn hỏng ổ cứng. Sáng mai, cửa hàng bạn dựng suốt ba bài vừa qua còn lại gì?

Câu trả lời là không còn gì. Toàn bộ code nằm trong đúng một chiếc máy. Mất máy là mất trắng, và không ai muốn dựng lại từ đầu lần thứ hai.

Còn một nỗi lo thứ hai kín đáo hơn. Càng về sau, mỗi lần bạn nhờ AI sửa lớn, luôn có một xác suất nhỏ là nó làm hỏng thứ đang chạy tốt. Nếu không có cách quay về "bản hôm qua còn ngon", một lần sửa hỏng có thể phá nhiều ngày công sức.

Cả hai nỗi lo có chung một lời giải, và đó là thứ dân làm phần mềm cả thế giới đều dùng: **Git** và **GitHub**.

Hãy hình dung một cuốn **nhật ký công trình**. Mỗi khi ngôi nhà xây thêm được một phần ra hồn, người ta chụp ảnh, ghi ngày tháng và vài dòng mô tả vào nhật ký. Muốn biết tuần trước nhà trông ra sao, giở lại trang cũ. **Git** chính là cuốn nhật ký đó cho code: mỗi lần bạn "lưu mốc", nó chụp lại toàn bộ dự án kèm dòng mô tả, và bạn có thể quay về bất kỳ mốc nào.

GitHub là nơi cất cuốn nhật ký đó **trên mạng**. Máy bạn hỏng, cuốn nhật ký vẫn còn nguyên trên GitHub, tải về là tiếp tục. Kho của bạn để chế độ riêng tư thì chỉ mình bạn thấy; muốn cho ai xem hay cùng làm thì mời họ vào.

Với cửa hàng của bạn, GitHub mang lại bốn thứ:

Lợi ích	Nghĩa là
Không mất trắng	Máy hỏng, code vẫn nằm trên mạng
Quay lại được	Sửa hỏng thì lùi về mốc gần nhất còn tốt
Chia sẻ được	Mời người khác xem hoặc cùng làm
Cửa ngõ lên mạng	Bài 8, dịch vụ đưa web lên mạng sẽ lấy code từ đây

4.2 Tạo tài khoản GitHub

Tạo tài khoản là việc tay, vì các dịch vụ chỉ cho con người tự đăng ký.

Mở trình duyệt, vào trang **github.com**, bấm nút **Sign up**. Điền email, đặt mật khẩu, chọn tên tài khoản. Tên tài khoản nên ngắn gọn, viết liền không dấu, vì nó sẽ xuất hiện trong địa chỉ kho code của bạn. GitHub gửi một mã xác nhận vào email, bạn mở email chép mã dán vào là xong.

Ghi tên tài khoản và mật khẩu vào chỗ bạn vẫn cất mật khẩu quan trọng. Tài khoản này còn dùng cho hai dịch vụ nữa ở bài 5 và bài 8.

4.3 Đưa cửa hàng lên GitHub

Giờ tới phần chính. Trước khi gõ câu đúng, xem một câu viết vội hỏng ra sao.

Yêu cầu chưa tốt

```
Lưu code của tôi lên mạng.
```

Chỗ hổng	AI sẽ làm gì
Không nói dịch vụ nào	Chọn đại một chỗ, có khi là nơi bạn chưa từng nghe tên
Không nói kho tên gì	Đặt tên vu vơ, sau này khó tìm
Không nói riêng tư hay công khai	Code có thể bị phơi công khai ngoài ý muốn
Không nói thứ gì cấm mang theo	Đẩy luôn cả file bí mật và rác tạm lên mạng

Chỗ hổng cuối đáng nói riêng. Trong dự án luôn có những thứ **không được** đưa lên mạng: file chứa khóa bí mật (bài 5 bạn sẽ có), và các thư mục rác máy tự sinh. Git có một danh sách tên là `.gitignore`, ghi vào đó thứ gì thì thứ đó ở nhà. Bạn không phải tự viết, chỉ cần nhắc AI làm.

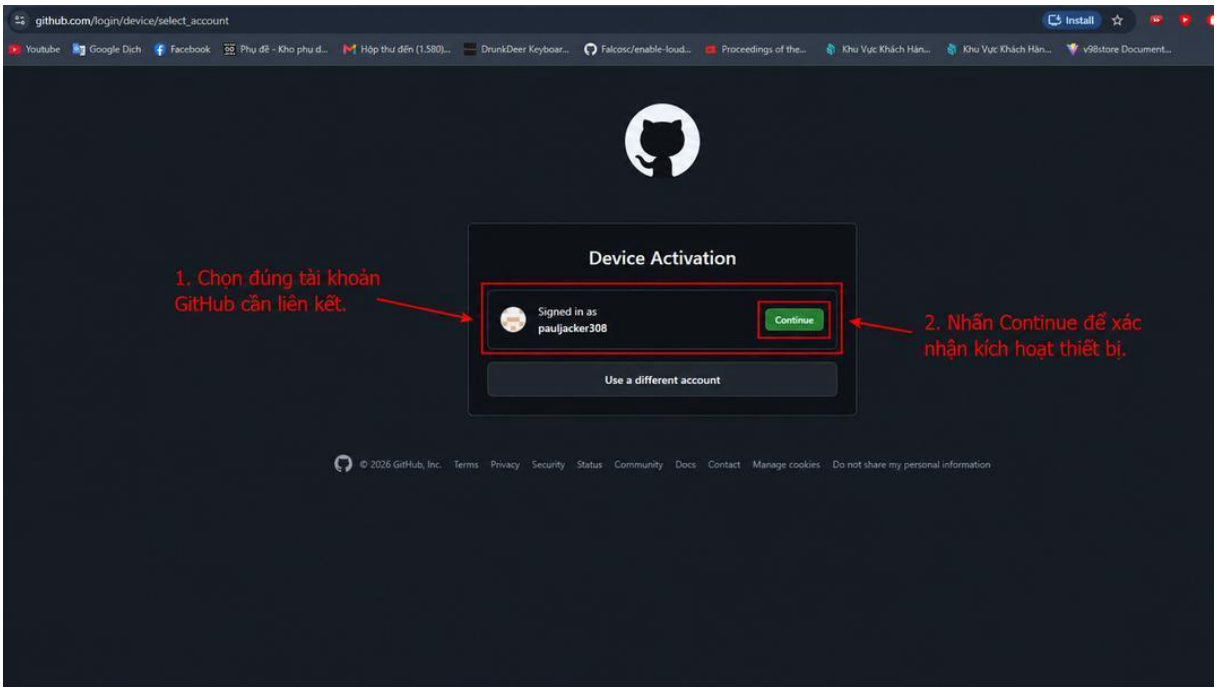
Yêu cầu tốt

```
VIỆC: Đưa dự án mini-shop-next lên GitHub để cất giữ
NGUỒN: Tài khoản GitHub tôi vừa tạo; toàn bộ code trong thư mục mini-shop-next
KẾT QUẢ: Một kho tên mini-shop trên GitHub ở chế độ riêng tư, chứa đủ code;
tôi mở github.com là nhìn thấy kho này
LƯU Ý: Máy thiếu công cụ gì thì cài luôn; cần tôi đăng nhập hay bấm gì thì dừng
lại
chỉ tôi; tạo danh sách .gitignore để file bí mật và thư mục tạm không bị đẩy
lên
```

Agent sẽ làm gì, và hai cú bấm của bạn

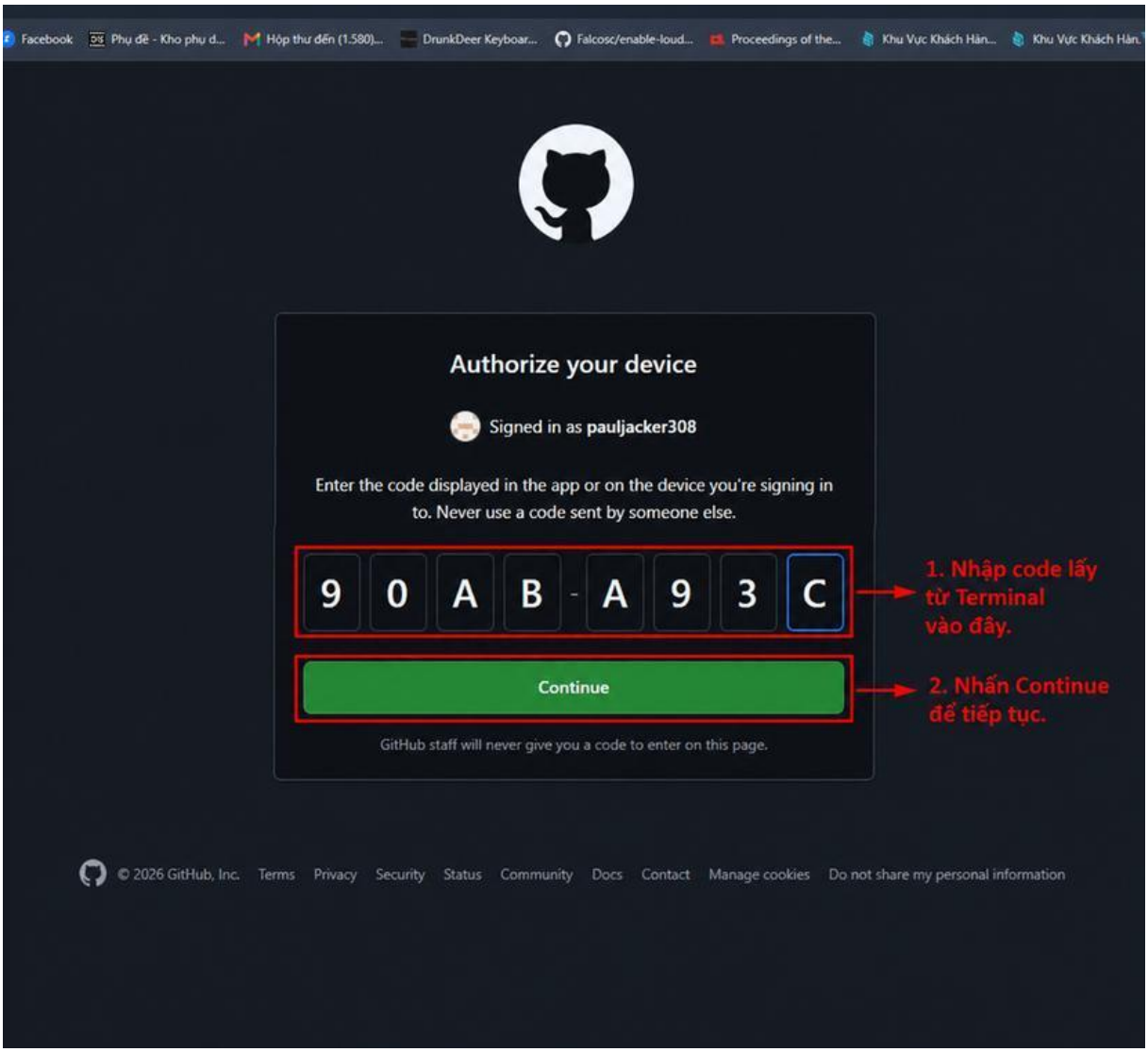
Agent kiểm tra máy, thiếu Git thì tự cài (Windows hỏi xin quyền thì bấm **Yes**, như lần cài Node.js). Tới đoạn kết nối với GitHub, nó dừng lại và đưa bạn một **mã tám ký tự**, kèm địa chỉ trang để nhập mã. Đây là bước tay thật, vì GitHub cần chính chủ xác nhận.

Trình duyệt mở trang GitHub. Nếu máy đang đăng nhập sẵn, GitHub hỏi bạn xác nhận đúng tài khoản:



Chọn đúng tài khoản GitHub cần liên kết

Sau đó tới màn nhập mã. Chép tám ký tự Agent đưa vào các ô, rồi bấm **Continue**, màn kế tiếp bấm nút cho phép:



Nhập mã tám ký tự rồi bấm Continue

Quay lại Antigravity, Agent nhận ra đã kết nối xong và tự làm nốt: tạo kho `mini-shop`, tạo `.gitignore`, đẩy toàn bộ code lên.

Kiểm tra: Mở trình duyệt vào **github.com**, bấm ảnh đại diện góc trên bên phải, chọn **Your repositories**. Thấy kho **mini-shop** kèm chữ **Private** là công sức của bạn đã nằm an toàn trên mạng.

4.4 Thói quen lưu mốc

Nhật ký chỉ quý khi được ghi đều. Từ giờ, kết thúc mỗi buổi làm việc, hoặc ngay trước một lần sửa lớn, gõ một câu:

```
Lưu mốc lên GitHub, ghi chú ngắn gọn những gì vừa làm.
```

Agent gom các thay đổi, ghi một dòng mô tả, đẩy lên. Mỗi mốc như một trang nhật ký có ảnh chụp: sau này lỡ hỏng, bạn nói "quay về mốc gần nhất còn chạy tốt" là cứu được. Cả cuốn sách còn lại sẽ nhắc bạn lưu mốc ở những chỗ quan trọng, nhưng tốt nhất là để nó thành phản xạ.

4.5 Skill: trang bị thêm nghề cho AI

Chuyển sang chuyện thứ hai của bài này.

AI trong Antigravity giống một người thợ đa năng mới tới làm. Rất giỏi, nhưng chưa thuộc nề nếp riêng của bạn. Vì vậy suốt bài 2 và 3, cứ tới phần giao diện là bạn phải dặn đi dặn lại trong LƯU Ý: khoảng cách cho thoáng, một màu nhấn nhất quán, không trang trí lòe loẹt. Quên dặn là nó lại làm nhạt.

Skill sinh ra để khỏi phải dặn lại. Một skill là một **cẩm nang nghề** soạn sẵn, cài vào dự án cho AI đọc. Cài rồi, mỗi lần làm việc liên quan, AI tự giở cẩm nang ra làm theo, như người thợ đã thuộc lòng sổ tay của xưởng. Lời dặn từ chỗ nằm trong trí nhớ của bạn chuyển sang nằm cố định trong dự án.

Cuốn sách này dùng hai bộ:

- **frontend-design**, cẩm nang thiết kế giao diện: cách chọn màu, cỡ chữ, khoảng cách, bố cục cho ra sản phẩm nhìn chuyên nghiệp. Đây là tấm khiên dài hạn chống kiểu giao diện nhạt của AI.
- **superpowers**, cẩm nang nề nếp làm việc: gặp việc lớn thì lên kế hoạch trước, làm xong thì tự soát lại. Chính là những thói quen bạn học ở bài 3, giờ AI cũng thuộc.

4.6 Cài skill và dùng thử

Cài bằng prompt, đúng nguyên tắc quen thuộc:

```
Cài hai bộ skill frontend-design và superpowers vào dự án này giúp tôi.
Cài xong liệt kê những skill đã có để tôi biết.
```

Agent tự tải và cài, xong liệt kê danh sách skill. Skill cài một lần cho dự án, các buổi sau dùng luôn.

Giờ cho người thợ mới học nghề trổ tài. Nhờ nó rà lại trang chủ:

```
VIỆC: Dùng skill frontend-design, rà lại trang chủ và cải thiện những chỗ chưa
đẹp
NGUỒN: Trang chủ hiện có của mini-shop-next
KẾT QUẢ: Trang chủ sau khi sửa nhìn chuyên nghiệp hơn: khoảng cách đều, cỡ chữ
hợp lý, màu nhất quán
LƯU Ý: Giữ nguyên bố cục và các tính năng, chỉ tinh chỉnh phần nhìn
```

Kiểm tra: Mở trang chủ trước và sau khi sửa, đặt cạnh nhau. Thường bạn sẽ thấy khác biệt ở những chỗ nhỏ: chữ dễ đọc hơn, các khối thẳng hàng hơn, khoảng thở đều hơn. Ưng thì lưu mốc; chưa ưng chỗ nào thì nói tiếp cho nó chỉnh.

Tóm tắt bài

- Code nằm trong một máy là công sức mong manh; GitHub là cuốn nhật ký công trình cất trên mạng
- Đưa dự án lên GitHub bằng một prompt; bước tay duy nhất là nhập mã tám ký tự xác nhận chính chủ

- File bí mật và rác tạm phải nằm trong danh sách .gitignore để không bị đẩy lên mạng
- Tập phản xạ lưu mốc sau mỗi buổi làm việc và trước mỗi lần sửa lớn
- Skill là cẩm nang nghề cài vào dự án cho AI, khỏi phải dặn đi dặn lại
- Bộ frontend-design giữ giao diện chuyên nghiệp, bộ superpowers giữ nề nếp lên kế hoạch và tự soát

BÀI 5: ĐƯA DỮ LIỆU THẬT VÀO CỬA HÀNG

Mục tiêu bài học:

- Hiểu vì sao cửa hàng cần một kho dữ liệu chung, và Supabase là gì
- Tự tạo được tài khoản và dự án Supabase
- Tạo bảng và đổ dữ liệu sản phẩm bằng SQL do AI viết
- Lấy được địa chỉ và chìa khóa kết nối, cất đúng chỗ an toàn
- Nối cửa hàng vào kho: đổi dữ liệu trên kho, trang web đổi theo

5.1 Vì sao cần một kho dữ liệu chung

Cửa hàng của bạn đang chạy đẹp, nhưng dữ liệu của nó có một điểm yếu chết người mà bạn đã biết từ bài 2: **mọi thứ đều tạm bợ**.

Danh sách sản phẩm đang ghi cứng trong code. Muốn đổi giá một món, phải sửa code. Giỏ hàng và đơn đặt nằm trong trình duyệt của từng khách; khách đóng máy là mất, còn người bán thì tuyệt nhiên không nhìn thấy đơn nào.

Hãy hình dung một chuỗi cửa hàng có nhiều quầy. Nếu mỗi quầy giữ một cuốn sổ riêng thì giá cả lệch nhau, đơn hàng thất lạc. Cách đúng là đặt **một cuốn sổ cái ở văn phòng trung tâm**: mọi quầy cùng đọc một bảng giá, mọi đơn hàng cùng ghi về một chỗ.

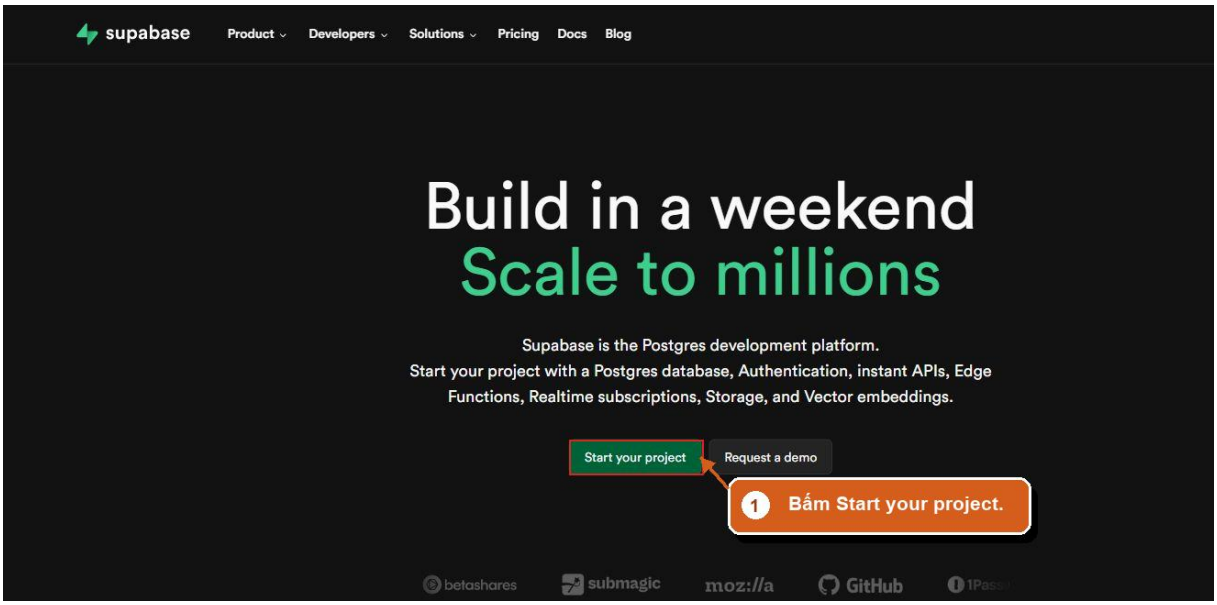
Cuốn sổ cái đó, trong phần mềm, gọi là **cơ sở dữ liệu** (database, chữ DB bạn gặp ở bài 1). Nó chính là lớp "kho" trong ba lớp của một phần mềm, và là mảnh còn thiếu cuối cùng của cửa hàng.

Supabase là dịch vụ cho thuê kho dữ liệu trên mạng. Bạn không phải mua máy chủ hay cài đặt gì, chỉ cần tạo tài khoản là có một kho riêng, mức khởi đầu miễn phí và quá đủ cho cửa hàng này. Supabase còn kèm sẵn hệ thống đăng nhập và quản lý người dùng, thứ bạn sẽ dùng ở bài 6.

5.2 Tạo tài khoản và dự án Supabase

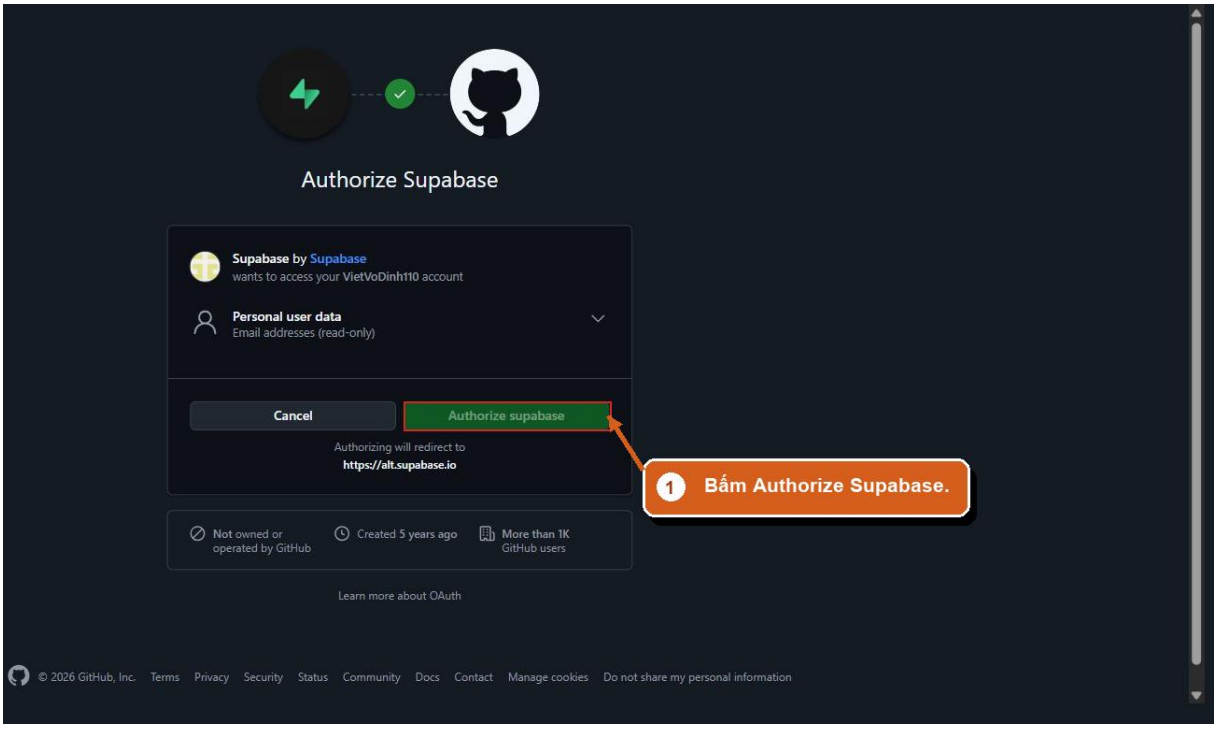
Phần này là việc tay trên trang web của Supabase, làm một lần duy nhất.

Bước 1. Vào trang Supabase. Mở trình duyệt, vào **supabase.com**, bấm nút **Start your project**.



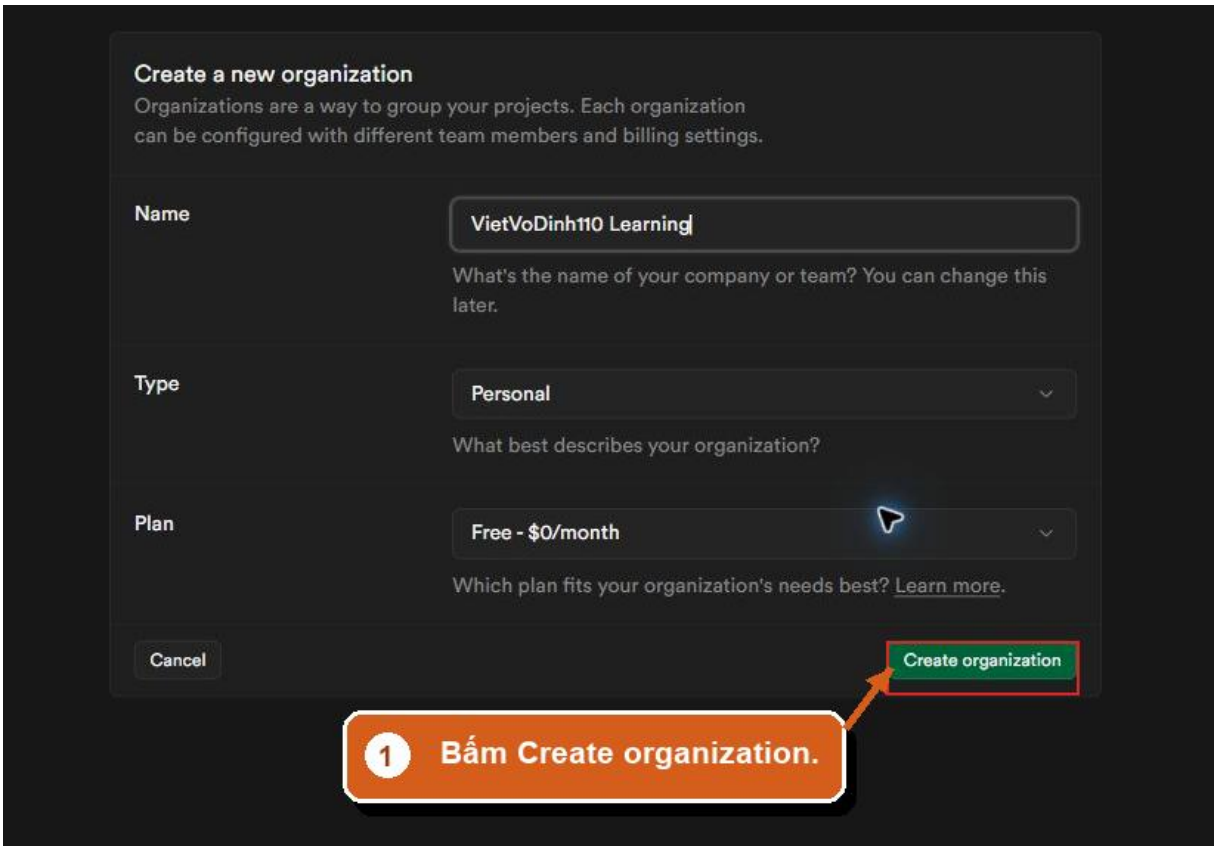
Trang chủ Supabase, bấm Start your project

Bước 2. Đăng nhập bằng GitHub. Trang đăng nhập hiện ra, chọn **Continue with GitHub**. Đây là lúc tài khoản GitHub của bài 4 phát huy: khởi tạo thêm tài khoản mới. GitHub hỏi có cho phép Supabase không, kiểm tra đúng tên tài khoản của mình rồi bấm **Authorize supabase**.



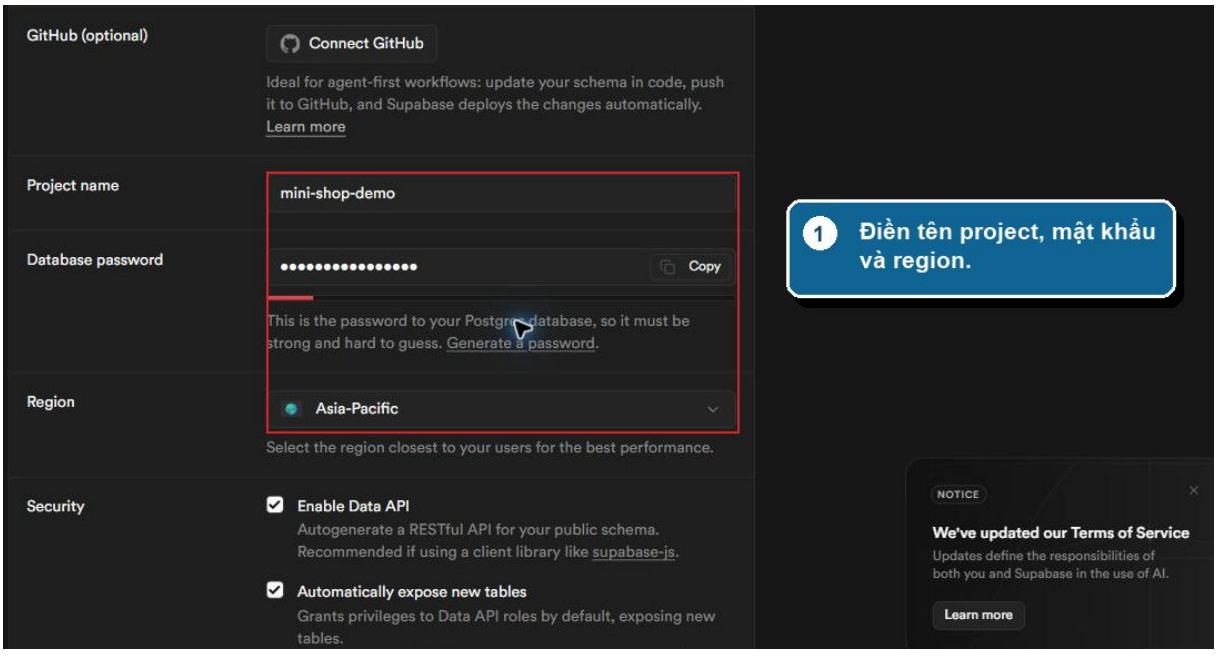
Cho phép Supabase dùng tài khoản GitHub

Bước 3. Tạo tổ chức. Lần đầu vào, Supabase bảo tạo một "organization", hiểu là cái tủ đựng các dự án của bạn. Đặt tên tùy ý, để nguyên hai lựa chọn **Personal** và **Free**, bấm **Create organization**.



Tạo organization cho tài khoản

Bước 4. Tạo dự án. Tiếp theo là màn tạo dự án, tức là tạo cái kho. Đặt tên **mini-shop**, đặt một **mật khẩu cho kho dữ liệu** rồi ghi ngay vào chỗ cất mật khẩu của bạn, phần Region chọn khu vực gần Việt Nam là **Southeast Asia (Singapore)**. Xong bấm **Create new project** và chờ khoảng một hai phút để Supabase dựng kho.



Điền tên dự án, mật khẩu và khu vực

5.3 Tạo bảng và đổ dữ liệu

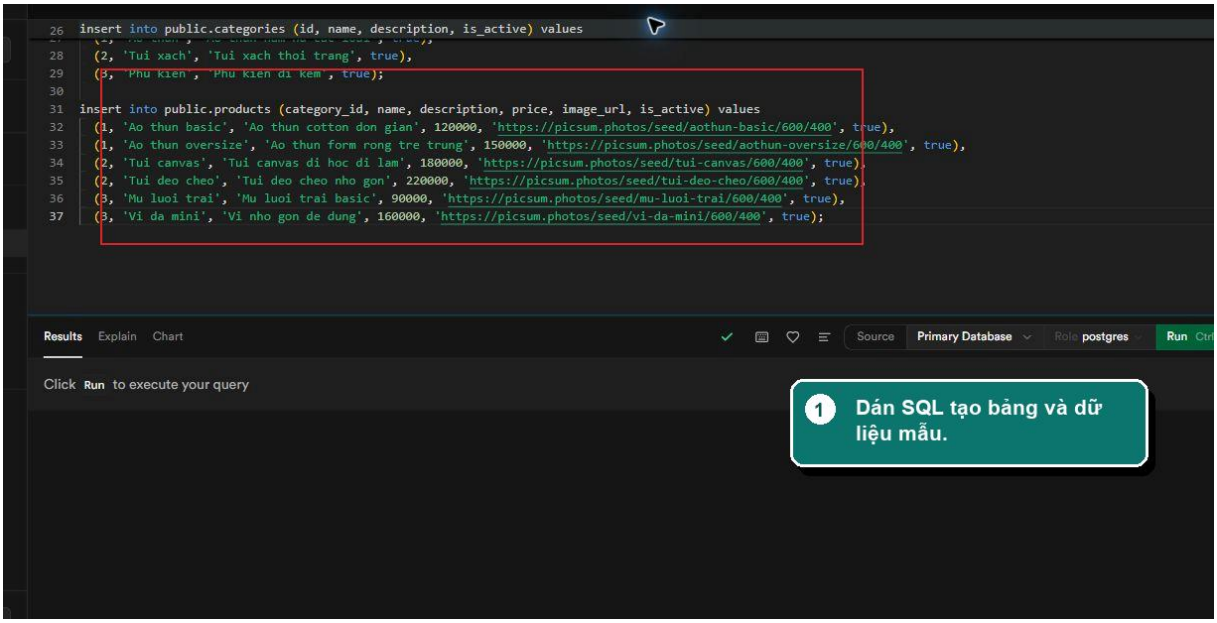
Kho đã có, nhưng đang trống trơn. Giờ cần dựng các **bảng** bên trong. Một bảng trông y như trang tính Excel: mỗi cột một loại thông tin, mỗi dòng một mục. Cửa hàng cần vài bảng: danh mục, sản phẩm, đơn hàng.

Người ta ra lệnh cho kho dữ liệu bằng một ngôn ngữ tên là **SQL**. Bạn không cần học nó, vì cách phân công như sau: **AI viết SQL, bạn mang qua dán vào Supabase.** Phải có bước dán tay này vì Agent làm việc trên máy của bạn, còn cái kho nằm trên trang của Supabase.

Về Antigravity, giao việc:

```
VIỆC: Viết một đoạn SQL dựng kho dữ liệu cho cửa hàng
NGUỒN: Dữ liệu danh mục và sản phẩm đang có trong code mini-shop-next
KẾT QUẢ: Một đoạn SQL duy nhất tạo bảng danh mục, sản phẩm, đơn hàng, và chèn sẵn
toàn bộ sản phẩm đang có, để tôi chép dán vào SQL Editor của Supabase
LƯU Ý: Kèm vài dòng chỉ dẫn ngắn cho tôi biết dán vào đâu và bấm gì
```

Agent trả về một khối SQL dài. Bấm nút chép ở góc khối code. Sang trang Supabase, nhìn thanh biểu tượng dọc bên trái, bấm biểu tượng **SQL Editor**, dán khối SQL vào khung soạn:



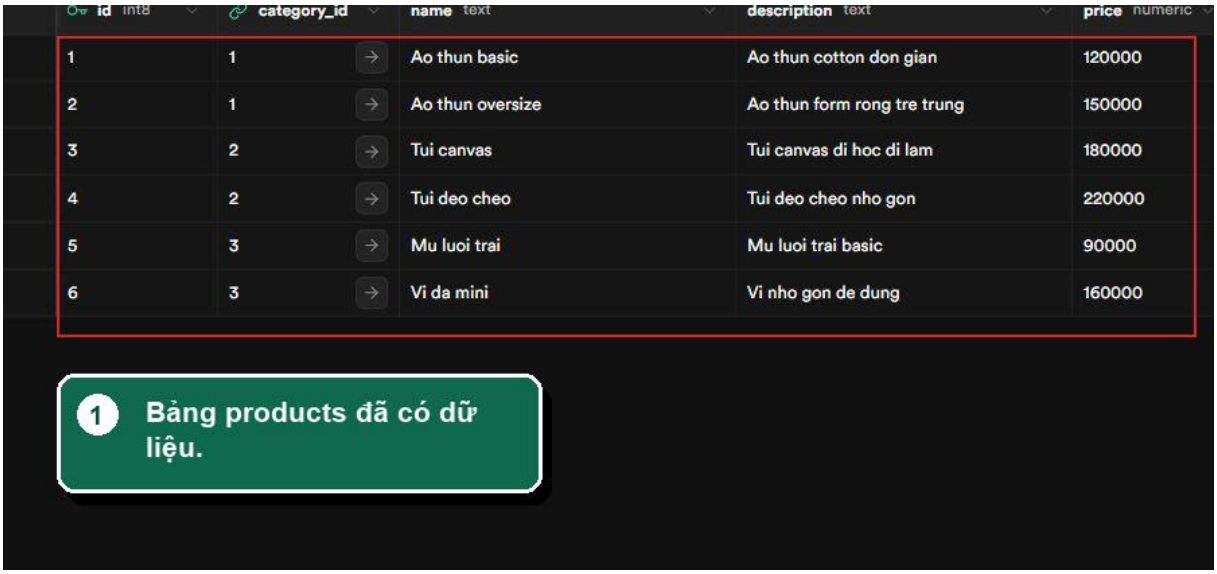
Dán khối SQL vào SQL Editor

Bấm nút **Run** để chạy. Supabase có thể hiện một hộp cảnh báo hỏi lại, trong đó có lựa chọn **Run without RLS**:

Hộp xác nhận khi chạy SQL

Chữ **RLS** đáng được giải thích tử tế, vì đây là chuyện an toàn. RLS là **khóa an toàn gắn trên từng bảng**, quy định ai được đọc, ai được ghi. Lúc này cửa hàng còn đang chạy trong máy bạn, chưa ai ngoài kia vào được, nên ta chọn **Run without RLS**, tức là tạm để bảng mở cho dễ làm. Nhưng hãy nhớ một lời hẹn: **trước khi đưa web lên mạng ở bài 8, ta sẽ quay lại bật khóa này.** Sách sẽ nhắc đúng lúc.

Kiểm tra: Chạy xong hiện chữ **Success**. Bấm biểu tượng **Table Editor** ở thanh bên trái, mở bảng sản phẩm, thấy đủ các món quen thuộc của cửa hàng nằm thành từng dòng:



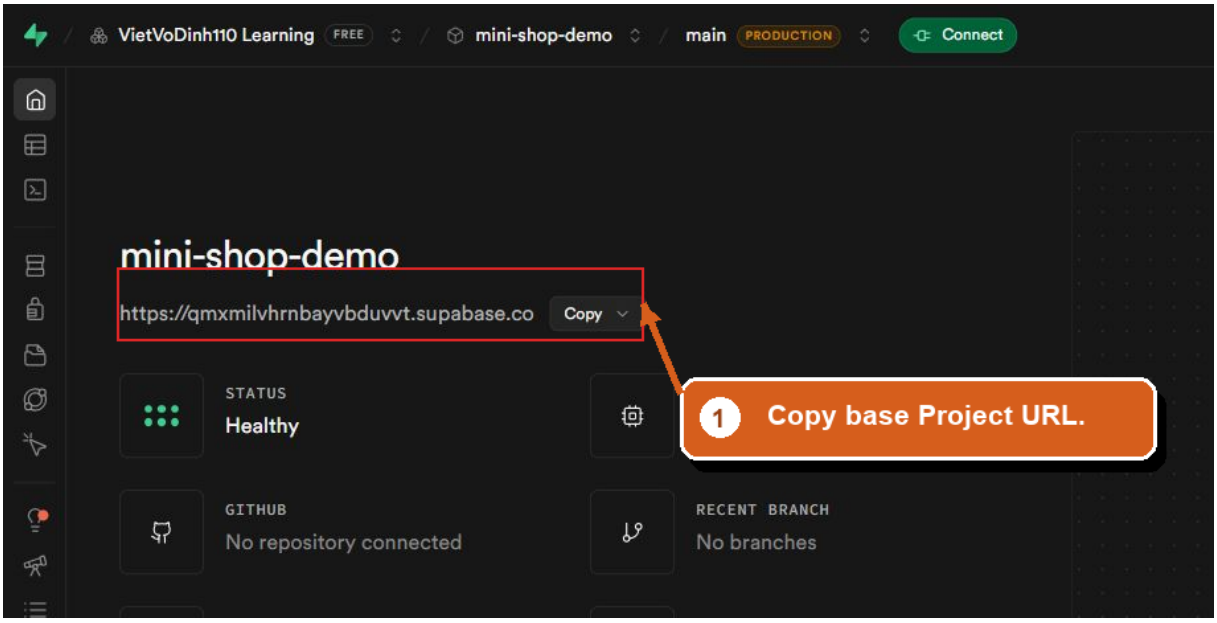
Bảng sản phẩm đã đầy dữ liệu

Khoảnh khắc này đáng dừng lại một giây: dữ liệu cửa hàng của bạn lần đầu nằm trong một cuốn sổ cái thật, trên một máy chủ thật.

5.4 Lấy địa chỉ và chìa khóa kết nối

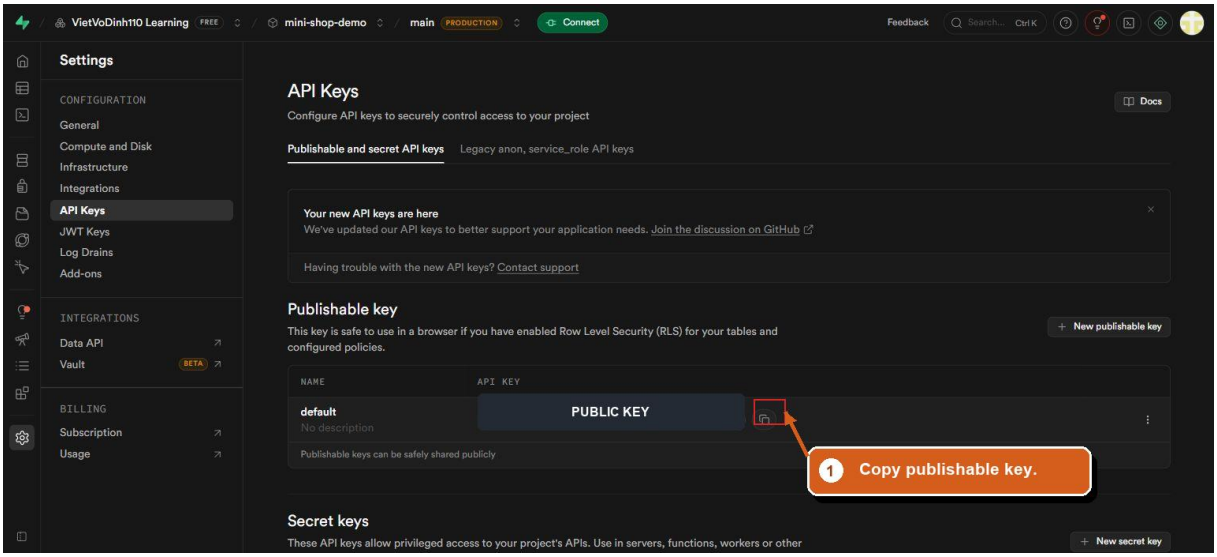
Muốn trang web đọc được kho, nó cần hai chuỗi ký tự: **địa chỉ kho** và **chìa khóa ra vào**. Cả hai nằm trong phần cài đặt của dự án Supabase.

Ở trang tổng quan dự án, tìm mục **Project URL**, bấm **Copy**. Đây là địa chỉ kho:



Chép địa chỉ Project URL

Cũng trong trang đó, tìm mục **API Keys**, chép tiếp chuỗi **publishable key**. Đây là chìa khóa loại công khai, sinh ra để đặt trong các trang web:



Chép khóa publishable

Hai chuỗi này dán tạm vào một ghi chú trên máy. Đừng gửi chúng lên mạng xã hội hay nhóm chat công khai; tuy khóa publishable thuộc loại ít nguy hiểm, thói quen giữ khóa kín đáo là thói quen đáng có từ đầu.

5.5 Nối cửa hàng vào kho

Đến bước nối. Trước khi gõ câu đúng, xem câu viết vội hỏng thế nào.

Yêu cầu chưa tốt

```
Kết nối web của tôi với Supabase đi.
```

Chỗ hổng	AI sẽ làm gì
Không đưa địa chỉ và khóa	Bịa ra chỗ điền, chạy sao được
Không nói cất khóa ở đâu	Ghi thẳng khóa vào code, rồi lần lưu mốc sau đẩy luôn lên GitHub
Không nói phần nào đọc từ kho	Nối nửa vời, chỗ đọc kho chỗ vẫn xài dữ liệu cũ trong code
Không có dấu hiệu nghiệm thu	Xong rồi bạn cũng chẳng biết lấy gì kiểm

Chỗ hổng thứ hai nguy hiểm nhất. Khóa phải nằm trong một **file môi trường** riêng, đúng loại file mà danh sách `.gitignore` của bài 4 giữ lại ở nhà. Nhắc rõ điều đó trong LƯU Ý.

Yêu cầu tốt

```
VIỆC: Nối cửa hàng mini-shop-next vào kho dữ liệu Supabase của tôi
NGUỒN: Địa chỉ dự án và khóa publishable tôi dán ở cuối tin nhắn này
KẾT QUẢ: Trang chủ và trang danh sách hiển thị sản phẩm đọc từ Supabase;
tôi đổi giá một món trên Supabase, tải lại trang là thấy giá mới
LƯU Ý: Cất địa chỉ và khóa vào file môi trường riêng, không ghi thẳng vào code,
và file đó không được đẩy lên GitHub
<dán hai chuỗi vừa chép vào đây>
```

Agent tạo file môi trường, sửa code để đọc từ kho, rồi chạy lại trang. Nhìn bề ngoài web không đổi gì, các sản phẩm vẫn hiện như cũ. Khác biệt nằm ở nguồn: giờ chúng chảy ra từ cuốn sổ cái.

5.6 Nghiệm thu: đổi sổ cái, cửa hàng đổi theo

Kiểm chứng bằng một phép thử sờ được.

Sang Supabase, vào **Table Editor**, mở bảng sản phẩm. Bấm đúp vào ô giá của món Giỏ mây đan, sửa thành một con số khác hẳn, lưu lại. Quay về trang web, tải lại trang.

Giá trên web đổi theo. Bạn vừa chứng kiến đúng cái cảnh "sổ cái ở văn phòng, mọi quầy đọc chung": dữ liệu giờ chỉ có một nguồn sự thật, sửa một chỗ, mọi nơi thấy ngay. Thử xong, sửa giá về như cũ.

Còn một việc khép bài, kiểm tra cái khóa không bị lộ:

```
Kiểm tra giúp tôi: file chứa địa chỉ và khóa Supabase có đang bị đẩy lên GitHub
không?
Nếu có thì gỡ ra và chặn lại.
```

Agent xác nhận file nằm trong danh sách giữ lại ở nhà là ổn. Lưu mốc lên GitHub rồi nghỉ; lần lưu này chỉ chứa code, chìa khóa vẫn ở nhà.

Tóm tắt bài

- Dữ liệu ghi cứng trong code và nằm rải trong máy khách là tạm bợ; cửa hàng cần một sổ cái chung gọi là cơ sở dữ liệu
- Supabase cho thuê kho dữ liệu trên mạng, đăng nhập bằng tài khoản GitHub có sẵn, mức khởi đầu miễn phí
- Ra lệnh cho kho bằng SQL: AI viết, bạn dán vào SQL Editor và bấm Run
- RLS là khóa an toàn trên từng bảng; đang làm trong nhà thì tạm để mở, hẹn bật lại trước khi lên mạng ở bài 8
- Web nối kho bằng địa chỉ và khóa; khóa cất trong file môi trường không bị đẩy lên GitHub
- Nghiệm thu kiểu sổ cái: đổi giá trên Supabase, tải lại web thấy giá mới

BÀI 6: CHO CỬA HÀNG VẬN HÀNH THẬT

Mục tiêu bài học:

- Biết bốn động tác với kho dữ liệu: thêm, đọc, sửa, xóa
- Cho khách đặt hàng và đơn được ghi thẳng vào kho, người bán nhìn thấy
- Thay đăng nhập giả bằng đăng nhập thật của Supabase
- Phân vai người mua và người bán, khu quản trị chỉ chủ cửa hàng vào được
- Cho khu quản trị thêm sửa xóa thật: admin đổi giá, khách thấy ngay

6.1 Từ trưng bày sang vận hành

Sau bài 5, cửa hàng của bạn đọc được sổ cái. Nhưng để ý kỹ, nó mới chỉ **đọc**. Khách đặt hàng, đơn vẫn nằm trong máy khách. Người bán "sửa" sản phẩm trong khu quản trị, thay đổi chỉ hiện trên màn hình rồi mất. Cửa hàng đang giống một gian trưng bày đẹp: xem được, nhưng chưa mua bán thật được.

Làm việc với một kho dữ liệu, xét cho cùng chỉ có bốn động tác: **thêm** một dòng mới, **đọc** các dòng ra, **sửa** một dòng, **xóa** một dòng. Dân trong nghề gộp bốn chữ tiếng Anh thành **CRUD**, bạn chỉ cần nhớ ý: thêm, đọc, sửa, xóa. Bài 5 làm xong động tác đọc. Bài này làm nốt ba động tác còn lại, mỗi động tác gắn vào một việc thật của cửa hàng.

6.2 Khách đặt hàng, đơn vào sổ cái

Bắt đầu từ động tác **thêm**, ở đúng chỗ đau nhất: đơn hàng.

```
VIỆC: Sửa phần thanh toán để đơn đặt hàng được ghi vào kho Supabase
NGUỒN: Trang thanh toán hiện có; bảng đơn hàng trong kho
KẾT QUẢ: Khách điền form và bấm Đặt hàng thì trong bảng đơn hàng trên Supabase
xuất hiện một dòng mới, đủ tên, số điện thoại, địa chỉ, các món và tổng tiền
LƯU Ý: Nếu kho còn thiếu bảng hay cột nào thì viết SQL đưa tôi dán vào Supabase
trước, đừng tự bịa chỗ lưu khác
```

Kiểm tra: Ra trang web, bỏ hai món vào giỏ, điền form đặt hàng với tên của chính bạn, bấm Đặt hàng. Sang Supabase mở bảng đơn hàng: một dòng mới mang tên bạn vừa nằm đó. Đóng hẳn trình duyệt, mở lại, dòng đó vẫn còn. Đơn hàng đã thoát kiếp tạm bợ.

6.3 Đăng nhập thật

Đăng nhập giả của bài 2 đã xong vai. Giờ thay bằng đăng nhập thật, và đây là chỗ phải cẩn thận nhất cuốn sách, vì nó dính tới mật khẩu của người khác.

Trước khi gõ câu đúng, xem câu viết vội nguy hiểm thế nào.

Yêu cầu chưa tốt

```
Làm chức năng đăng nhập cho web.
```

Chỗ hổng	AI sẽ làm gì
Không nói dùng hệ nào	Có thể tự chế, lưu mật khẩu người dùng vào một bảng thường, lộ là thảm họa
Không tả đủ luồng	Có đăng nhập mà thiếu đăng ký, thiếu đăng xuất, thiếu nhớ phiên
Không nói thay thế bản giả	Bản giả và bản thật chồng lên nhau, loạn
Không có dấu hiệu nghiệm thu	Không biết thế nào là xong

Nguyên tắc an toàn chỉ có một câu, và đáng thuộc lòng: **chuyện mật khẩu không bao giờ tự chế, luôn dùng hệ đăng nhập dựng sẵn của dịch vụ lớn.** Supabase có sẵn hệ đó, đã được người ta lo giùm phần khó nhất.

Yêu cầu tốt

```
VIỆC: Thay đăng nhập giả bằng đăng nhập thật dùng hệ Auth có sẵn của Supabase
NGUỒN: Các trang đăng nhập, đăng ký hiện có của mini-shop-next
KẾT QUẢ: Khách đăng ký được tài khoản mới bằng email và mật khẩu, đăng nhập rồi
thấy tên mình trên thanh menu, đăng xuất được; đóng trình duyệt mở lại vẫn còn
đăng nhập
LƯU Ý: Tuyệt đối không tự lưu mật khẩu vào bảng thường, mọi thứ đi qua hệ Auth
của Supabase; gỡ hẳn phần đăng nhập giả cũ
```

Kiểm tra: Đăng ký một tài khoản với email của bạn, đăng nhập, thấy tên trên menu. Đăng xuất, đăng nhập lại. Sang Supabase, vào mục **Authentication**, thấy tài khoản vừa tạo nằm trong danh sách người dùng. Mật khẩu thì không ai xem được, kể cả bạn, và như vậy là đúng.

6.4 Ai là chủ cửa hàng

Đăng nhập thật rồi, nhưng mọi tài khoản đang bình đẳng như nhau. Cửa hàng cần phân vai: **khách** mua hàng, **chủ cửa hàng** (admin) vào khu quản trị. Vai được ghi ngay trong kho, như cột chức danh trong sổ nhân sự.

```
VIỆC: Thêm phân vai người dùng, mặc định là khách, và chặn khu quản trị
NGUỒN: Hệ Auth và kho Supabase đang dùng
KẾT QUẢ: Tài khoản thường vào địa chỉ khu quản trị thì bị đưa về trang đăng
nhập;
tài khoản có vai admin thì vào được
LƯU Ý: Cần sửa gì trong kho thì viết SQL đưa tôi dán; kèm luôn câu SQL để tôi
nâng một tài khoản lên vai admin
```

Agent đưa bạn một câu SQL ngắn kiểu "nâng tài khoản email này lên admin". Dán vào SQL Editor của Supabase, sửa email thành email bạn vừa đăng ký, bấm Run. Bạn vừa tự bổ nhiệm mình làm chủ cửa hàng.

Kiểm tra: Đăng nhập bằng tài khoản admin, vào khu quản trị bình thường. Đăng xuất, đăng ký thêm một tài khoản thường bằng email khác, thử vào đúng địa chỉ khu quản trị: bị chặn. Hàng rào đã dựng đúng chỗ.

6.5 Khu quản trị ghi thật

Còn hai động tác **sửa** và **xóa**, giao nốt cho khu quản trị:

```
VIỆC: Cho khu quản trị làm việc thật với kho Supabase
NGUỒN: Các màn quản trị hiện có: sản phẩm, đơn hàng
KẾT QUẢ: Admin thêm, sửa, xóa sản phẩm thì bảng trên Supabase đổi theo và trang
khách hiện đúng như vậy; màn đơn hàng liệt kê đơn thật từ kho, đổi được trạng
thái
đơn từ mới sang đang giao, đã giao
LƯU Ý: Sản phẩm mới dùng ảnh có sẵn trong thư mục assets; xóa sản phẩm thì hỏi
xác nhận trước khi xóa
```

Kiểm tra: Đây là màn nghiệm thu vui nhất từ đầu sách. Mở hai cửa sổ trình duyệt cạnh nhau: một bên là khu quản trị đăng nhập admin, một bên là trang khách. Bên quản trị sửa giá Đèn tre thủ công, bên khách tải lại trang: giá mới hiện ra. Thêm một sản phẩm mới bên quản trị, bên khách thấy món mới lên kệ. Mở màn đơn hàng, thấy đơn bạn đặt ở mục 6.2, đổi trạng thái nó sang đang giao.

Xong, đừng quên câu quen thuộc:

```
Lưu mốc lên GitHub, ghi chú: cửa hàng vận hành thật với kho Supabase.
```

6.6 Nhìn lại

Điểm lại ba chỗ đuối treo từ bài 2, giờ đã gỡ được mấy?

Chỗ lặp menu, bài 3 gỡ bằng component. Chỗ dữ liệu tạm bợ, bài 5 và bài này gỡ nốt: sản phẩm nằm trong sổ cái, đơn hàng ghi về một chỗ, người bán nhìn thấy và xử lý được, người mua có tài khoản thật. Cửa hàng của bạn, chạy trên máy bạn, đã là một phần mềm vận hành đúng nghĩa.

Chỉ còn đúng một chuyện: nó vẫn đang ở `localhost`, sau cánh cửa nhà bạn. Trước khi mở cửa cho cả thế giới, người cẩn thận sẽ đi khám tổng quát một lượt. Đó là bài 7.

Tóm tắt bài

- Làm việc với kho dữ liệu gói trong bốn động tác thêm, đọc, sửa, xóa; bài 5 lo đọc, bài này lo ba động tác còn lại
- Đơn đặt hàng giờ ghi thẳng vào kho, đóng máy mở lại vẫn còn, người bán nhìn thấy
- Chuyện mật khẩu không bao giờ tự chế; dùng hệ Auth có sẵn của Supabase
- Phân vai bằng một cột trong kho: khách mua hàng, admin vào khu quản trị, rào chặn đặt ở địa chỉ quản trị
- Khu quản trị thêm sửa xóa thật: đổi bên quản trị, trang khách đổi theo
- Cửa hàng đã vận hành thật trên máy bạn; trước khi lên mạng còn một lượt khám tổng quát

BÀI 7: KHÁM TỔNG QUÁT TRƯỚC KHI RA MẮT

Mục tiêu bài học:

- Hiểu vì sao phải tự kiểm thử trước khi cho người khác vào
- Đi hết kịch bản người mua và kịch bản người bán, ghi lỗi cho đúng cách
- Thử web như một người lạ: trên điện thoại, chưa đăng nhập, nhập liệu ẩu
- Đưa danh sách lỗi cho AI sửa an toàn, có mốc để quay lại
- Chốt danh sách sẵn sàng trước khi lên mạng ở bài 8

7.1 Khách không báo lỗi, khách bỏ đi

Cửa hàng chạy tốt dưới tay bạn. Nhưng bạn là người dựng ra nó, bạn bấm đúng chỗ theo thói quen, nhập đúng kiểu dữ liệu nó chờ. Khách thì khác: họ bấm lung tung, nhập thiếu, vào bằng điện thoại, và khi gặp một trang lỗi họ lặng lẽ đóng tab. Rất hiếm ai nhắn cho bạn "web bị hỏng chỗ này". Lỗi trên web bán hàng trả giá bằng đơn hàng mất đi trong im lặng.

Vì vậy trước khi mở cửa, chủ cửa hàng tự đóng vai khách đi mua một vòng từ đầu tới cuối. Bài này làm đúng việc đó, theo ba lượt: đi vai người mua, đi vai người bán, rồi thử như một người lạ. Gặp lỗi thì ghi lại, cuối bài đưa cả danh sách cho AI sửa một thể.

Cách ghi một lỗi cho ra hồn

Ghi "trang giỏ hàng bị lỗi" thì AI cũng bó tay như một người thợ nghe chủ nhà nói "nhà có chỗ hỏng". Một dòng lỗi tốt gồm bốn ý: **ở màn nào, bạn làm gì, thấy gì, đáng lẽ phải thế nào.** Ví dụ: "Ở trang giỏ hàng, tôi bấm nút trừ khi số lượng đang là 1, món hàng biến mất luôn, đáng lẽ phải hỏi tôi có muốn xóa không."

Mở một file ghi chú, mỗi lỗi một dòng theo bốn ý đó. Danh sách này chính là phần NGUỒN cho prompt sửa lỗi ở cuối bài.

7.2 Kịch bản người mua

Đi chậm từng bước như một khách thật, đối chiếu từng dấu hiệu:

Bước	Làm gì	Dấu hiệu đạt
1	Mở trang chủ	Banner, sản phẩm, giá hiện đủ, không ô ảnh vỡ
2	Gõ "đèn" vào ô tìm kiếm	Chỉ còn các sản phẩm đèn
3	Lọc danh mục Đồ thủ công	Chỉ còn nhóm đồ thủ công
4	Mở một sản phẩm	Đúng ảnh, đúng giá, đúng mô tả món đó
5	Thêm vào giỏ hai lần	Số trên biểu tượng giỏ tăng đúng

Bước	Làm gì	Dấu hiệu đạt
6	Vào giỏ, tăng giảm số lượng	Tổng tiền tính lại đúng
7	Thả tim hai món	Trang yêu thích hiện đúng hai món
8	Đăng ký một tài khoản mới	Đăng ký xong, tên hiện trên menu
9	Đặt hàng với giỏ đang có	Báo thành công, giỏ về không
10	Sang Supabase mở bảng đơn hàng	Đơn vừa đặt nằm đó, đủ thông tin

Bước nào lệch, ghi vào danh sách lỗi theo bốn ý, rồi cứ đi tiếp cho hết kịch bản. Đừng dừng lại sửa vặt giữa chừng, khám xong toàn thân rồi chữa một thể.

7.3 Kịch bản người bán

Đăng nhập bằng tài khoản admin, đi tiếp lượt của chủ cửa hàng:

Bước	Làm gì	Dấu hiệu đạt
1	Vào khu quản trị	Vào được, số liệu tổng quan hiện ra
2	Mở màn đơn hàng	Thấy đơn khách vừa đặt ở kịch bản trước
3	Đổi trạng thái đơn sang đang giao	Trạng thái đổi và giữ nguyên sau khi tải lại
4	Thêm một sản phẩm thử	Trang khách thấy món mới
5	Sửa giá món vừa thêm	Trang khách thấy giá mới
6	Xóa món thử đó	Có hỏi xác nhận, xóa xong trang khách hết món đó

7.4 Thử như một người lạ

Hai kịch bản trên là đường đi của người ngoan. Giờ thử kiểu người lạ, ba phép thử:

Thử trên điện thoại. Mở trang web trên điện thoại của bạn (cùng mạng Wi-Fi, Agent sẽ chỉ địa chỉ nếu bạn hỏi), hoặc thu nhỏ cửa sổ trình duyệt về cỡ màn hình dọc. Menu, lưới sản phẩm, giỏ hàng phải xếp lại gọn gàng, chữ không tràn, nút không chồng lên nhau.

Thử khi chưa đăng nhập. Mở một cửa sổ trình duyệt ở chế độ ẩn danh, gõ thẳng địa chỉ khu quản trị. Phải bị chặn và đưa về trang đăng nhập. Nếu vào được, đây là lỗi nặng nhất buổi khám, ghi ngay đầu danh sách.

Thử nhập ẩu. Ở form đặt hàng, bỏ trống hết rồi bấm Đặt hàng: web phải nhắc điền, không được tạo đơn rỗng. Nhập số lượng âm hoặc chữ vào ô số: web phải từ chối lịch sự thay vì hiện màn lỗi trắng.

7.5 Đưa danh sách lỗi cho AI sửa

Danh sách lỗi trong tay, giờ chữa. Trước khi gõ câu đúng, xem câu viết vội hỏng ra sao.

Yêu cầu chưa tốt

```
Kiểm tra web giúp tôi xem có lỗi gì không rồi sửa hết đi.
```

Chỗ hỏng	AI sẽ làm gì
Không đưa danh sách lỗi của bạn	Đọc lướt code, khen "nhìn chung ổn", bỏ sót lỗi thật bạn đã thấy
Không giới hạn phạm vi	Nhân tiện "cải thiện" lung tung, hỏng cả chỗ đang tốt
Không có mốc quay lại	Sửa hỏng nặng hơn thì hết đường lùi
Không nói cách báo kết quả	Sửa xong bạn chẳng biết thử lại chỗ nào

Yêu cầu tốt

Lưu mốc trước, rồi giao đúng danh sách:

```
VIỆC: Sửa các lỗi theo danh sách tôi liệt kê dưới đây
NGUỒN: Danh sách lỗi, mỗi dòng gồm: ở màn nào, tôi làm gì, thấy gì, đáng lẽ thế nào
KẾT QUẢ: Từng lỗi được sửa, báo lại đã sửa ra sao để tôi thử lại đúng chỗ đó
LƯU Ý: Trước khi sửa hãy lưu mốc lên GitHub; chỉ sửa các lỗi trong danh sách, không tự ý đổi giao diện hay thêm tính năng khác
<dán danh sách lỗi vào đây>
```

Agent sửa xong lỗi nào, bạn mở đúng chỗ đó thử lại theo kịch bản. Hết danh sách thì chạy lại trọn hai kịch bản một lần chót. Vòng lặp khám rồi chữa này lặp tới khi đi hết kịch bản mà danh sách lỗi trống trơn.

Muốn kỹ thêm một nấc, nhờ AI tự khám phần của nó:

```
Tự rà toàn bộ dự án, liệt kê những chỗ dễ gây lỗi hoặc còn thiếu sót, chưa cần sửa.
Tôi sẽ chọn chỗ nào đáng sửa.
```

Nó sẽ liệt ra một danh sách, thường lẫn cả những góp ý kiểu làm màu. Bạn đọc và chỉ chọn những mục thấy đúng là vấn đề, quyền quyết định luôn ở người chủ cửa hàng.

7.6 Danh sách sẵn sàng ra mắt

Khép bài bằng bốn câu hỏi, trả lời đủ "rồi" thì cửa hàng được phép lên mạng:

- Hai kịch bản người mua, người bán đi trọn không còn lỗi?

- Cửa sổ ẩn danh vào khu quản trị đã bị chặn?
- Khóa Supabase vẫn nằm ngoài GitHub (đã kiểm ở bài 5)?
- Bản mới nhất đã lưu mốc lên GitHub?

Bốn chữ "rồi" trong tay, hẹn bạn ở bài cuối: mở cửa cho cả thế giới.

Tóm tắt bài

- Khách gặp lỗi thì bỏ đi trong im lặng, nên chủ cửa hàng phải tự khám trước khi mở cửa
- Một dòng lỗi tốt gồm bốn ý: màn nào, làm gì, thấy gì, đáng lẽ thế nào
- Khám theo kịch bản trọn vòng: người mua mười bước, người bán sáu bước, lỗi ghi lại chữa một thể
- Thử như người lạ: trên điện thoại, ẩn danh vào khu quản trị phải bị chặn, nhập ẩu phải bị từ chối lịch sự
- Giao AI sửa theo đúng danh sách, lưu mốc trước khi sửa, sửa xong thử lại từng chỗ
- Đủ bốn chữ "rồi" trong danh sách sẵn sàng thì mới lên mạng

BÀI 8: ĐƯA CỬA HÀNG LÊN MẠNG

Mục tiêu bài học:

- Bật khóa an toàn RLS cho kho dữ liệu trước khi mở cửa, trả lời hẹn của bài 5
- Đưa cửa hàng lên mạng bằng Vercel, ai có link cũng vào được
- Nghiệm thu trên web thật: tự đặt hàng bằng điện thoại, nhờ người thân mua thử
- Nắm nhịp làm việc về sau: sửa, lưu mốc, web tự cập nhật
- Biết cần cất giữ những gì để cửa hàng sống lâu dài

8.1 Giờ mở cửa đã tới

Từ bài 1, bạn đã biết ranh giới giữa hai thế giới: web chạy `localhost` trong máy mình, và web trên mạng ai cũng vào được. Suốt bảy bài, cửa hàng của bạn sống ở thế giới thứ nhất. Hôm nay nó bước sang thế giới thứ hai.

Dịch vụ đưa nó sang là **Vercel**, hiểu đơn giản là một bãi đỗ trên mạng cho các web dựng bằng Next.js, mức khởi đầu miễn phí. Vercel có một điểm ăn khớp tuyệt vời với thứ bạn đã có: nó **nối thẳng vào GitHub**. Bạn chỉ nó tới kho `mini-shop`, nó tự lấy code về, tự dựng, tự phát hành. Và từ đó về sau, mỗi lần bạn lưu mốc phiên bản mới lên GitHub, Vercel tự động cập nhật web trên mạng. Thói quen lưu mốc của bài 4 giờ trả lãi.

Nhưng khoan bấm nút. Còn một lời hẹn phải trả trước đã.

8.2 Khóa kho trước, mở cửa sau

Bài 5 có một lời hẹn: các bảng trong kho đang **mở**, vì lúc đó cửa hàng còn chạy trong nhà. Đưa web lên mạng khi kho còn mở giống như khai trương cửa hàng mà cửa kho không khóa: ai tinh ý đều tự vào đọc, thậm chí sửa sổ sách của bạn. Vậy nên thứ tự bắt buộc là **khóa kho trước, mở cửa sau**.

Khóa ở đây là **RLS**, khóa an toàn trên từng bảng mà bạn đã gặp. Bật khóa nghĩa là đặt luật cho từng bảng: ai được đọc, ai được ghi. Trước khi gõ câu đúng, xem câu viết vội hỏng ra sao.

Yêu cầu chưa tốt

```
Bật bảo mật cho database đi.
```

Chỗ hổng	AI sẽ làm gì
Không nói luật cho từng bảng	Khóa tuốt mọi thứ, web trắng trơn vì chính nó cũng hết đọc được
Không nói cách làm	Loay hoay sửa lung tung thay vì đưa SQL cho bạn dán
Không nghiệm thu sau khóa	Khóa xong hỏng chỗ nào không ai biết

Chỗ hổng	AI sẽ làm gì
Không có mốc quay lại	Lỡ khóa sai không biết đường lùi

Yêu cầu tốt

Luật của cửa hàng nói bằng tiếng người như sau: ai cũng được **xem** sản phẩm; người mua **tạo được đơn của mình**; chỉ **admin** được thêm sửa xóa sản phẩm và xử lý mọi đơn. Đưa nguyên văn cho AI:

```
VIỆC: Viết SQL bật khóa RLS cho toàn bộ các bảng trong kho, trả lời hẹn ở bài 5
NGUỒN: Các bảng danh mục, sản phẩm, đơn hàng, người dùng đang có trên Supabase
KẾT QUẢ: Một đoạn SQL để tôi dán vào SQL Editor, theo luật: ai cũng xem được
sản phẩm và danh mục; người đăng nhập tạo được đơn của mình; chỉ admin được
thêm sửa xóa sản phẩm và xem, xử lý mọi đơn
LƯU Ý: Lưu mốc lên GitHub trước; sau khi tôi chạy SQL xong, nhắc tôi đi lại
kịch bản bài 7 để chắc không chỗ nào bị khóa nhầm
```

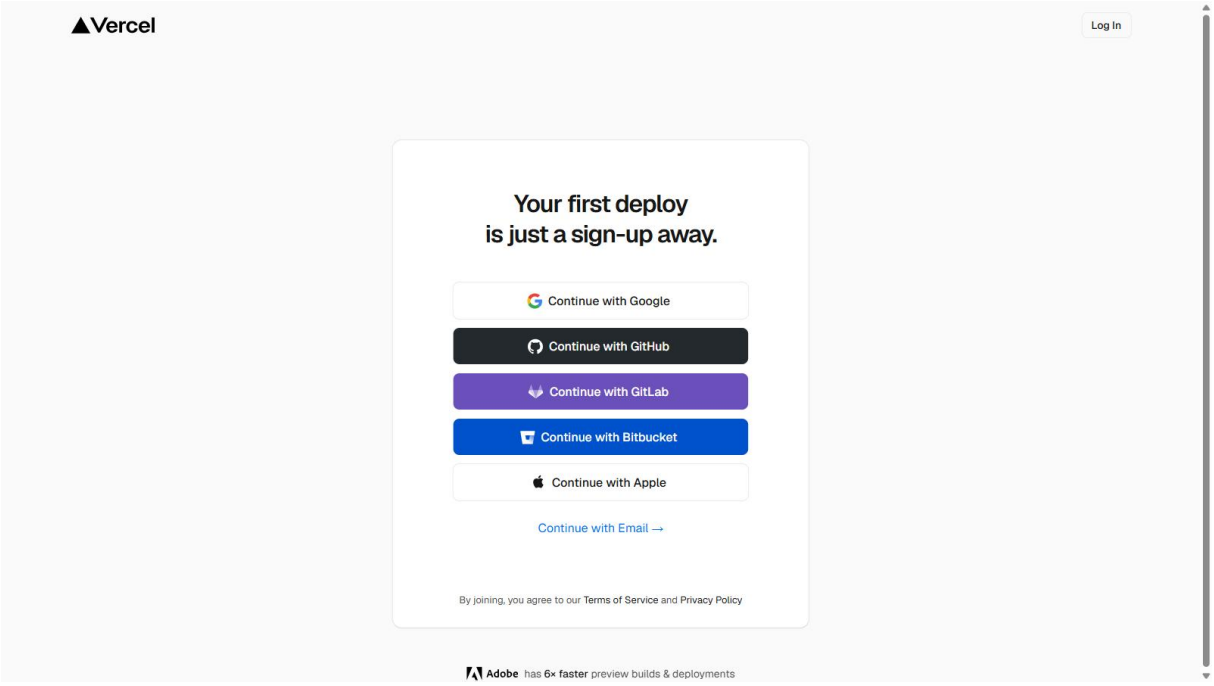
Chép đoạn SQL, dán vào **SQL Editor** của Supabase, bấm **Run**, thấy **Success**. Lần này chạy có khóa đàng hoàng, đúng nghĩa một cái kho trước giờ khai trương.

Kiểm tra: Đi nhanh lại kịch bản bài 7: khách xem hàng, đặt một đơn; admin vào quản trị sửa giá. Tất cả phải chạy như trước. Nếu chỗ nào bỗng trắng trơn hay báo không có quyền, tả đúng hiện tượng cho Agent, nó chỉnh lại luật, dán chạy lại là xong.

8.3 Đưa lên Vercel

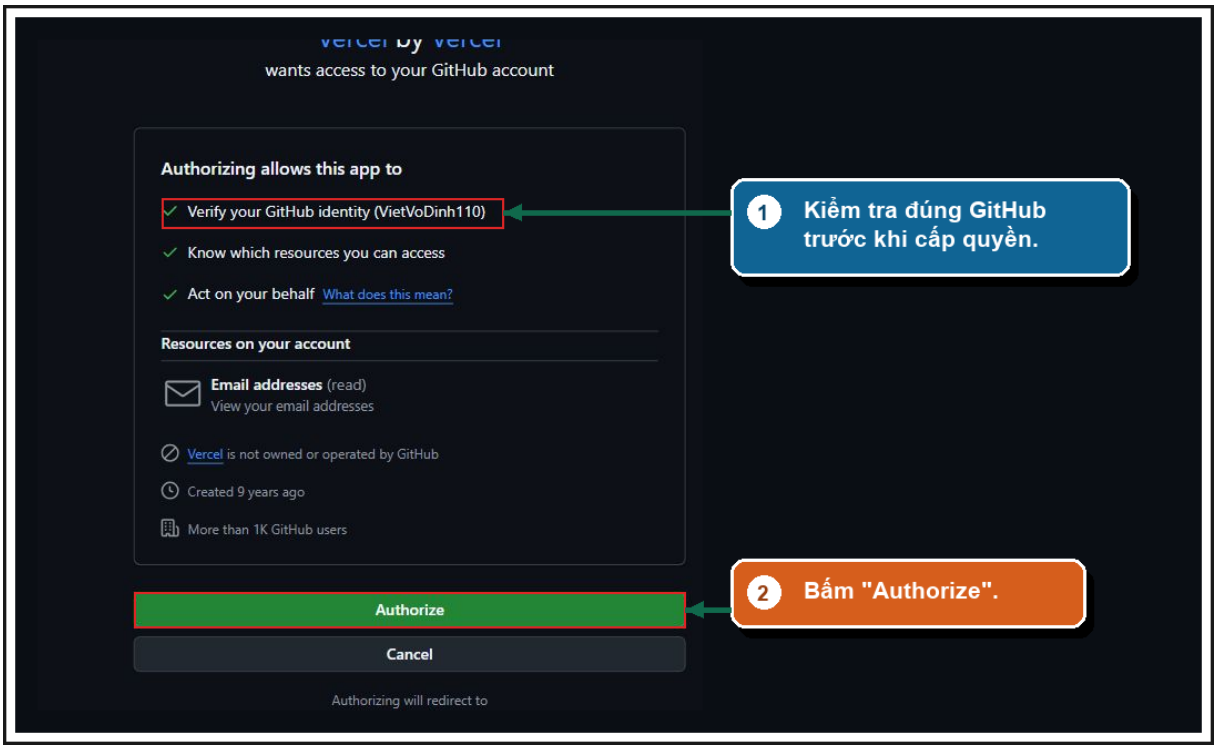
Tới nghi thức chính. Phần này là việc tay trên trang Vercel, có ảnh dẫn từng chặng.

Bước 1. Tạo tài khoản Vercel. Vào **vercel.com**, bấm **Sign Up**, và ở màn chọn cách đăng nhập, chọn **Continue with GitHub**. Vẫn là tài khoản GitHub quen thuộc, lần thứ ba nó mở cửa cho bạn.



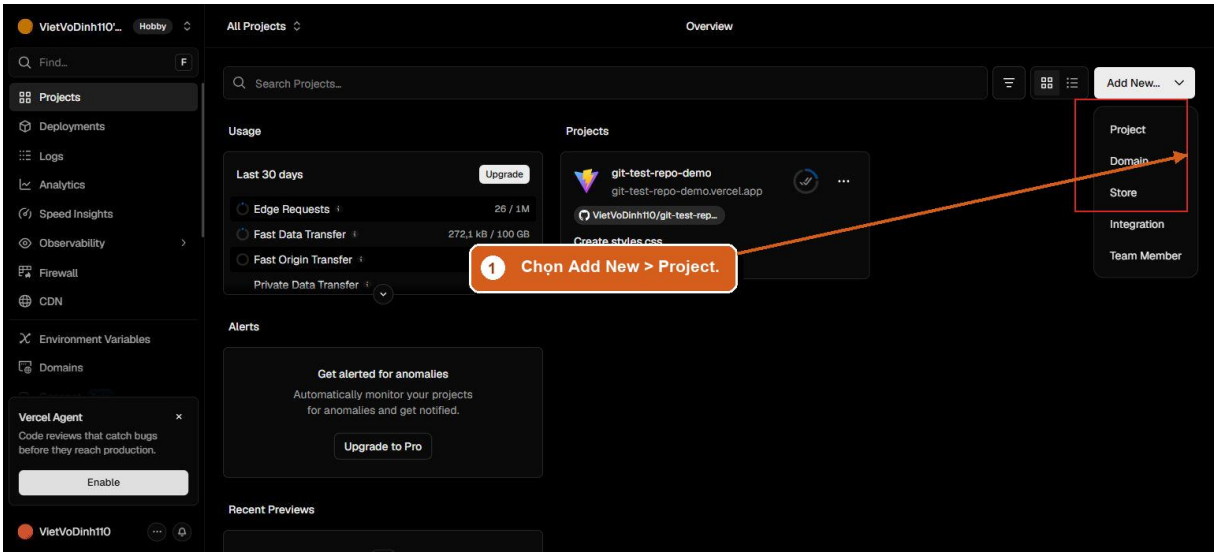
Đăng ký Vercel bằng tài khoản GitHub

GitHub hỏi có cho phép Vercel không, bấm **Authorize**:



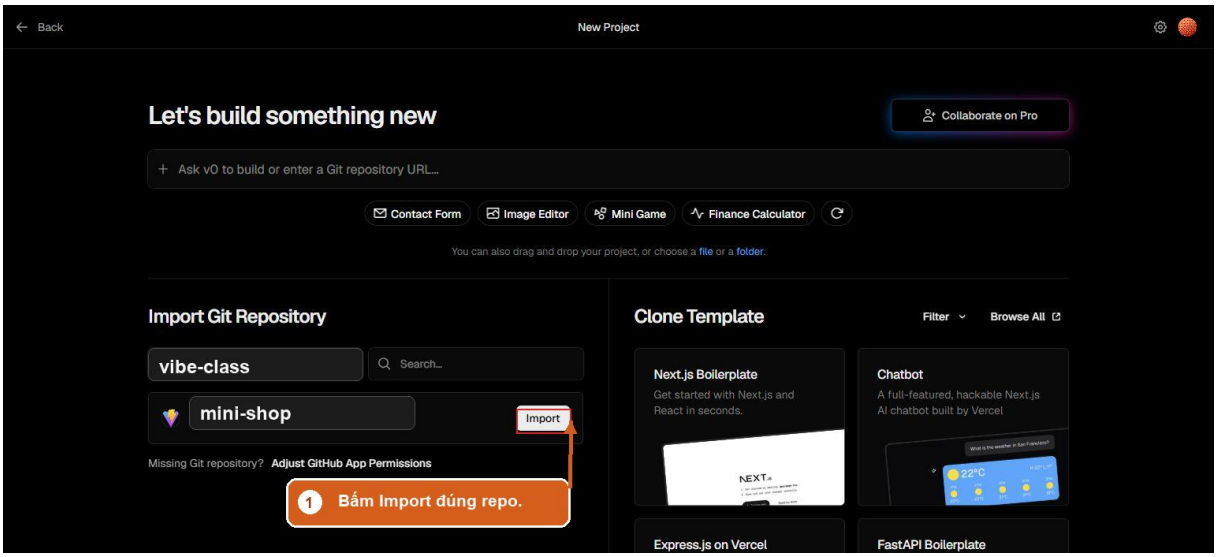
Cho phép Vercel dùng tài khoản GitHub

Bước 2. Tạo dự án từ kho GitHub. Vào trang quản lý của Vercel, bấm **Add New** rồi chọn **Project**:



Bấm Add New rồi chọn Project

Danh sách kho GitHub của bạn hiện ra. Tìm dòng **mini-shop**, bấm **Import**:



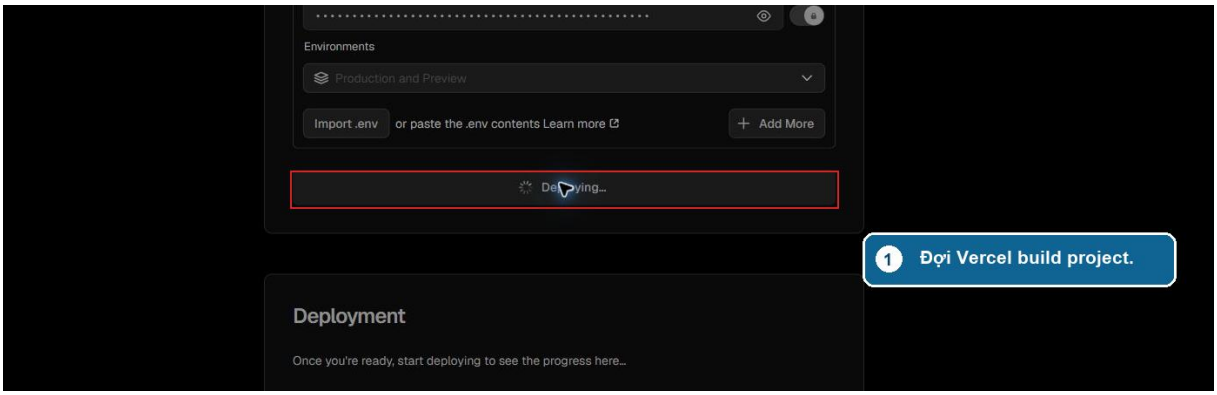
Import đúng kho mini-shop

Bước 3. Khai hai chuỗi bí mật. Màn cấu hình hiện ra. Phần khung dựng, Vercel tự nhận diện đây là dự án Next.js, để nguyên như nó chọn. Chỗ duy nhất cần tay bạn là mục **Environment Variables**.

Nhớ lại bài 5: địa chỉ kho và chìa khóa nằm trong file môi trường, và file đó cố tình **không** được đẩy lên GitHub. Vercel vì vậy chưa biết hai chuỗi này, bạn phải trao tận tay. Mở file môi trường trong dự án (khung Explorer của Antigravity, file tên bắt đầu bằng `.env`), với **từng dòng** trong đó: chép phần tên biến bên trái dấu bằng dán vào ô Name, chép phần giá trị bên phải dấu bằng dán vào ô Value, bấm **Add**. Làm đủ các dòng.

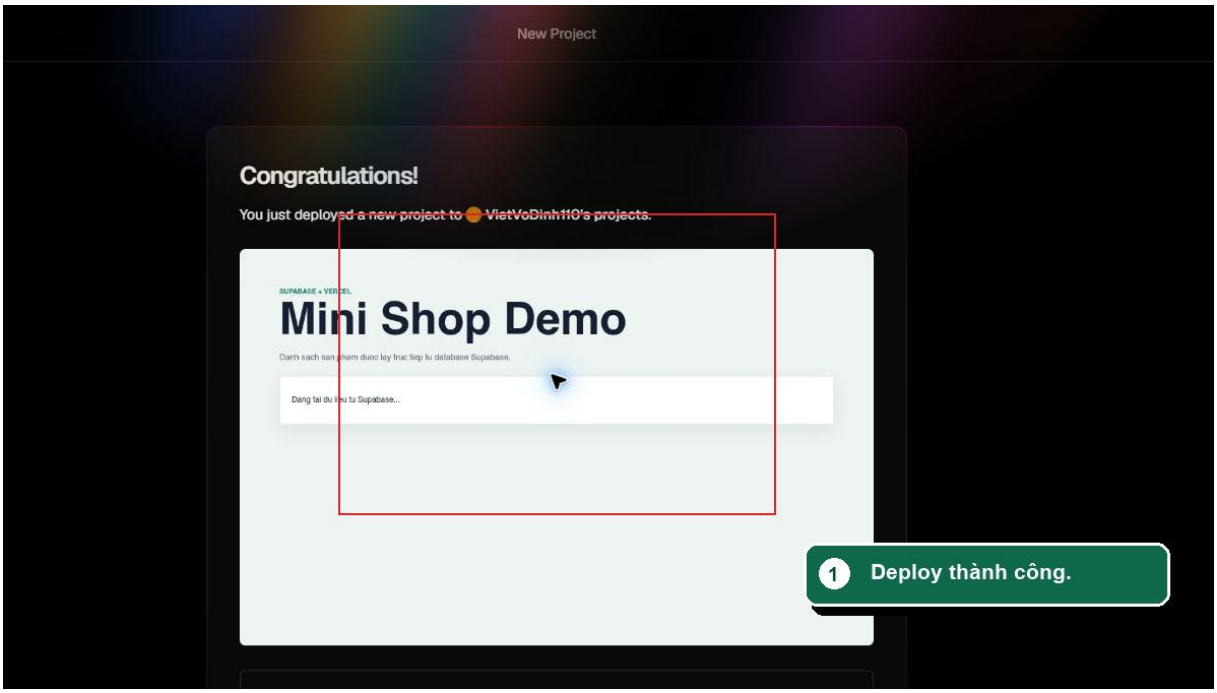
Đây chính là cách chuyên nghiệp để đem bí mật lên dịch vụ: đi thẳng từ tay bạn vào két của Vercel, không đi qua chỗ công khai nào.

Bước 4. Deploy. Bấm nút **Deploy**. Vercel lấy code từ GitHub về và dựng, mất chừng một hai phút:



Vercel đang dựng web

Dựng xong, màn hình chúc mừng hiện ra kèm ảnh chụp trang web của bạn:



Deploy thành công

Bấm vào ảnh chụp đó, trình duyệt mở cửa hàng của bạn ở một địa chỉ dạng `mini-shop-xxxx.vercel.app`. Nhìn kỹ thanh địa chỉ: không còn `localhost`. Đây là web thật, trên mạng thật.

8.4 Đi một vòng trên web thật

Nghiệm thu lần này có nghi thức riêng, vì nó xứng đáng.

Rút điện thoại, **tắt Wi-Fi đi cho dùng mạng di động**, gõ địa chỉ cửa hàng. Trang chủ hiện ra trên một thiết bị chẳng dây dợ gì với máy tính của bạn. Đặt thử một đơn ngay trên điện thoại.

Rồi làm điều mà cả tám bài hướng tới: **gửi đường link cho một người thân**, nhờ họ chọn một món và đặt thử. Vài phút sau, mở khu quản trị: đơn của họ nằm đó, tên thật, món thật. Người cách bạn cả thành phố vừa dùng phần mềm bạn dựng. Cảm giác này, người làm nghề lâu năm vẫn nhớ lần đầu của họ.

Điện thoại có chỗ hiển thị lệch, form có chỗ khó bấm? Ghi lại theo đúng kiểu bài 7, về Antigravity sửa, rồi xem mục tiếp theo để hiểu vì sao bản sửa tự lên mạng.

8.5 Nhịp làm việc từ nay về sau

Cửa hàng đã sống trên mạng, và nhịp chăm nó gói trong một vòng bốn nhịp:

1. **Sửa** trên máy bằng prompt, như vẫn làm suốt tám bài
2. **Nghiệm thu** bản local, tự tay bấm thử
3. **Lưu mốc** lên GitHub bằng một câu
4. **Chờ vài phút**, Vercel tự dựng và web trên mạng thành bản mới

Không phải làm lại thao tác deploy nào nữa. Đường ống từ máy bạn ra thế giới đã nối xong, van tự động. Mọi tính năng bạn thêm về sau, từ đổi banner tới làm chương trình giảm giá, đều chảy qua đúng vòng bốn nhịp này.

8.6 Giữ gìn và đi tiếp

Cửa hàng sống lâu dài dựa trên một ít của cải cần cất kỹ. Ghi vào chỗ an toàn của bạn:

Thứ cần giữ	Là gì
Ba tài khoản	GitHub, Supabase, Vercel, và mật khẩu của chúng
Mật khẩu kho dữ liệu	Đặt ra ở bài 5 khi tạo dự án Supabase
File môi trường	File `.env` trong dự án, thứ duy nhất không nằm trên GitHub
Đường link cửa hàng	Địa chỉ vercel.app của bạn

Muốn địa chỉ đẹp kiểu `cuahangcuaban.com`? Mua một tên miền rồi nhờ Agent hướng dẫn trỏ về Vercel, việc chừng mười lăm phút, làm lúc nào cũng được.

Còn cánh cửa lớn hơn thì bạn đã tự mở rồi: một ý tưởng bất kỳ, một bộ ảnh, và tám bài vừa rồi làm khuôn. Vòng lặp là như nhau cho mọi sản phẩm.

Tóm tắt bài

- Thứ tự bắt buộc: khóa kho trước, mở cửa sau; bật RLS bằng SQL do AI viết theo luật nói bằng tiếng người
- Khóa xong phải đi lại kịch bản bài 7 để chắc không khóa nhầm chỗ
- Vercel lấy code từ GitHub, tự dựng và phát hành; tài khoản đăng nhập bằng GitHub
- Hai chuỗi bí mật trao tận tay Vercel qua mục Environment Variables, không đi qua GitHub
- Nghiệm thu web thật bằng điện thoại dùng mạng di động và một đơn hàng do người thân đặt
- Nhịp về sau: sửa, nghiệm thu, lưu mốc, web tự cập nhật; cất kỹ ba tài khoản, mật khẩu kho, file môi trường và đường link

LỜI KẾT

Hãy quay lại bức tường ở đầu cuốn sách.

Bạn có một ý tưởng, hình dung được từng màn hình, và dừng lại ở câu "tôi không biết viết code". Tám bài trước, bức tường đó còn đứng nguyên. Bây giờ, trong điện thoại của bạn có một đường link. Bấm vào là một cửa hàng thật, chạy trên mạng thật, có kho dữ liệu thật, và một đơn hàng do người thân của bạn đặt.

Bạn vẫn chưa viết một dòng code nào. Hóa ra cánh cửa chưa bao giờ đòi hỏi điều đó.

Điều thật sự ở lại

Tám bài có rất nhiều thao tác: kéo ảnh, gõ prompt, dán SQL, bấm deploy. Những thao tác rồi sẽ cũ. Công cụ đổi giao diện, dịch vụ đổi tên nút, vài năm nữa có khi cả cái tên Antigravity cũng thành kỷ niệm.

Thứ ở lại là cách bạn làm việc với AI, gói trong bốn thói quen:

- **Mô tả có khuôn**: việc gì, dựa vào đâu, ra cái gì nhìn thấy được, ràng buộc nào. Có ảnh mẫu thì đưa ảnh.
- **Việc lớn thì kiểm kê trước, bắt lên kế hoạch, duyệt rồi mới cho chạy.**
- **File là trí nhớ**: dự án nằm trong thư mục và trên GitHub, ảnh mẫu nằm cạnh code, khóa bí mật nằm ngoài chỗ công khai.
- **Nghiệm thu bằng mắt và tay**: tự bấm thử theo kịch bản trước khi tin là xong, và lưu mốc trước mỗi lần sửa lớn.

Bốn thói quen này không thuộc về công cụ nào. Chúng đi theo bạn sang mọi AI, mọi nền tảng về sau.

Sản phẩm tiếp theo là của riêng bạn

Mini Shop là sản phẩm để học. Sản phẩm đáng giá nhất là cái ý tưởng vẫn nằm trong đầu bạn từ trước khi mở cuốn sách này.

Đừng chọn ý tưởng to nhất. Chọn cái nhỏ mà bạn thật sự cần: trang giới thiệu cho cửa hàng nhà mình, công cụ ghi lịch hẹn, bảng theo dõi thu chi của đội nhóm. Kiếm vài tấm ảnh mẫu ưng mắt, mở thư mục mới, và đi lại đúng con đường tám bài: dựng thô, cho chạy thật, khám, rồi mở cửa. Con đường giờ đã quen chân.

Sản phẩm đầu tiên chạy được sẽ dạy bạn nhiều hơn mọi trang sách, kể cả những trang này.

Chia lại cho người bên cạnh

Điều bạn vừa học quý nhất khi được chia. Chỉ cho đồng nghiệp cách viết một yêu cầu bốn phần. Gửi người bạn đang ấp ủ ý tưởng cái link cửa hàng của bạn kèm câu

"tôi tự làm đấy". Ngồi cạnh con em mình trong buổi đầu nó dựng thử một trang web bằng lời.

Bức tường "phải biết code mới làm được phần mềm" đứng vững bao năm vì ai cũng tin nó. Mỗi người bước qua và quay lại kể, bức tường thấp đi một chút.

Học cùng Sao Việt

Trung tâm Đào tạo AI Sao Việt đồng hành cùng bạn trên chặng đường này. Ngoài hành trình bạn vừa hoàn thành, chúng tôi còn những chương trình khác cho người muốn đưa AI vào công việc và vận hành hàng ngày.

Liên hệ: tinhocsaoviet.com

Cảm ơn bạn đã đi cùng tới trang cuối. Chúc sản phẩm tiếp theo của bạn, cái của riêng bạn, sớm có đường link của nó.