

**KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO MÔN HỌC
CÔNG NGHỆ PHẦN MỀM
HỌC KỲ II, NĂM HỌC 2024 – 2025
ĐỀ TÀI
XÂY DỰNG WEBSITE CAFE**

Giáo viên hướng dẫn: Nguyễn Bảo Ân

Sinh viên thực hiện:

Nguyễn Nhật Hóa	110122006	DA22TTA
Bùi Quốc Anh	110122032	DA22TTA
Mai Tuấn Đạt	110122048	DA22TTA

Trà Vinh, tháng 7 năm 2025

This image shows a full page of a handwriting practice worksheet. It consists of numerous horizontal dotted lines spaced evenly down the page, providing a guide for letter height and placement. The background is plain white, and there are no other markings or text present.

Giáo viên hướng dẫn
(Ký tên và ghi rõ họ tên)

This image shows a full page of a document template designed for handwriting practice or general note-taking. It features a series of evenly spaced, horizontal dotted lines that run across the entire width of the page. The background is plain white, providing a clear contrast for the grey dots. There are no margins, text, or other markings present on the page.

Thành viên hội đồng
(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Lời nói đầu, chúng em xin cảm ơn Thầy Nguyễn Bảo Ân đã hỗ trợ, giảng dạy tận tình trong thời gian chúng em học tập môn CÔNG NGHỆ PHẦN MỀM, những kiến thức mà Thầy đã dạy sẽ là hành trang quý báu trên con đường học vấn và phát triển sự nghiệp tương lai rộng mở của chúng em. Thầy đã luôn kiên nhẫn, nhiệt tình trong việc truyền đạt kiến thức và kinh nghiệm quý báu, giúp chúng em vượt qua những khó khăn và thử thách trong quá trình học tập và nghiên cứu làm đồ án.

Những lời khuyên, góp ý của Thầy không chỉ là kim chỉ nam cho sự phát triển của đồ án kết thúc môn này mà còn là nguồn động viên, khích lệ tinh thần lớn lao cho chúng em.

Mặc dù đã cố gắng hết sức, nhưng với kinh nghiệm còn hạn chế, báo cáo này không thể tránh được những thiếu sót. Chúng em rất mong nhận được sự chỉ bảo, góp ý từ Thầy để từ đó chúng em có thể bổ sung, nâng cao kiến thức của mình, phục vụ tốt hơn cho những báo cáo sau này.

Chúng em xin chân thành cảm ơn.

MỤC LỤC

MỤC LỤC.....	5
DANH MỤC HÌNH ẢNH.....	6
DANH MỤC BẢNG BIỂU.....	7
CHƯƠNG 1 GIỚI THIỆU.....	8
1.1. Lý do chọn đề tài.....	8
1.2. Mục tiêu của ứng dụng.....	8
1.3. Tính cấp thiết của đề tài	9
1.4. Phạm vi và đối tượng nghiên cứu.....	9
1.5. Ý nghĩa thực tiễn của đề tài.....	10
1.6. Công nghệ và công cụ sử dụng.....	11
CHƯƠNG 2: PHÂN TÍCH YÊU CẦU.....	12
2.1. Khảo sát hiện trạng.....	12
2.2. Yêu cầu chức năng.....	12
2.2.1. Chức năng dành cho khách hàng.....	12
2.2.2. Chức năng dành cho quản trị viên.....	13
2.3. Yêu cầu phi chức năng.....	13
2.3.1. Hiệu năng.....	13
2.3.2. Bảo mật.....	13
2.3.3. Khả dụng và Tính ổn định.....	14
2.3.4. Khả năng mở rộng.....	14
2.3.5. Tính khả dụng.....	14
2.3.6. Khả năng bảo trì.....	14
2.3.7. Khả năng kiểm thử.....	14
2.4. Đặc tả Use Case.....	15
2.4.1. Biểu đồ Use Case tổng quan.....	15
2.5. Product Backlog.....	15
CHƯƠNG 3: THIẾT KẾ HỆ THỐNG.....	17
3.1. Kiến trúc tổng thể hệ thống.....	17
3.1.1. Mô hình kiến trúc (Client-Server, REST API).....	17
3.1.2. Nguyên tắc RESTful áp dụng.....	18

3.2. Thiết kế cơ sở dữ liệu.....	19
3.2.1. Sơ đồ ERD.....	19
3.2.2. Mô hình quan hệ và mô tả bảng.....	20
3.3. Thiết kế API.....	20
3.3.1. Danh sách Endpoint chính.....	21
3.4. Thiết kế giao diện người dùng (UI/UX).....	22
3.4.1. Figma mô phỏng.....	22
3.4.2. Các giao diện chính.....	28
CHƯƠNG 4: TRIỂN KHAI VÀ CÔNG NGHỆ.....	35
4.1. Công nghệ và công cụ sử dụng.....	35
4.1.1. Công nghệ sử dụng.....	35
4.1.2 Công cụ hỗ trợ phát triển.....	36
4.2. Triển khai quy trình CI/CD với GitHub Actions.....	37
4.2.1. Mục tiêu triển khai CI/CD.....	37
4.2.2. Quy trình CI/CD trong dự án.....	37
4.2.3. Vai trò của GitHub Actions trong dự án.....	38
4.2.4. Lợi ích đạt được.....	39
4.3. Docker hóa ứng dụng.....	39
4.3.1. Dockerfile Backend & Frontend.....	39
4.3.2. docker-compose.yml.....	40
CHƯƠNG 5: QUẢN LÝ DỰ ÁN.....	42
5.1. Quy trình phát triển phần mềm.....	42
5.1.1. Product Backlog.....	42
5.1.2. Sprint Backlog.....	43
5.2. Sử dụng Jira để quản lý dự án.....	44
5.2.1 Sprint 1.....	44
5.2.2 Sprint 2.....	45
5.2.3 Sprint 3.....	46
5.2.4 Sprint 4.....	46
5.2.5 Sprint 5.....	47
5.3. Phân công công việc cho các thành viên.....	48

CHƯƠNG 6: KIỂM THỬ	49
6.1. Chiến lược kiểm thử	49
6.1.1 Mục tiêu kiểm thử	49
6.1.2 Phạm vi kiểm thử	49
6.1.3 Phương pháp và kỹ thuật kiểm thử	49
6.2. Kết quả kiểm thử	50
CHƯƠNG 7: ĐÁNH GIÁ VÀ KẾT LUẬN	53
7.1. Đánh giá kết quả đạt được	53
7.2. Khó khăn gặp phải	53
7.3. Bài học rút ra	54
7.4. Hướng phát triển trong tương lai	54

DANH MỤC HÌNH ẢNH

Hình 2.1 Biểu đồ Use Case tổng quan.....	18
Hình 3.1 Sơ đồ ERD.....	22
Hình 3.2 Trang mô phỏng form đăng nhập bằng Figma.....	25
Hình 3.3 Trang mô phỏng form đăng kí bằng Figma.....	26
Hình 3.4 Trang mô phỏng của trang chủ bằng Figma.....	27
Hình 3.5 Trang mô phỏng của trang menu bằng Figma.....	28
Hình 3.6 Trang mô phỏng của trang blog bằng Figma.....	29
Hình 3.7 Trang mô phỏng của trang giỏ hàng bằng Figma.....	30
Hình 3.8 Trang mô phỏng của trang thông tin cá nhân bằng Figma.....	30
Hình 3.9 Giao diện đăng nhập.....	31
Hình 3.10 Giao diện đăng kí.....	32
Hình 3.11 Giao diện trang chủ.....	33
Hình 3.12 Giao diện menu.....	33
Hình 3.13 Giao diện trang blog.....	34
Hình 3.14 Giao diện trang quản lí người dùng.....	35
Hình 3.15 Giao diện quản lí đơn hàng.....	35
Hình 3.16 Danh sách các đơn hàng.....	36
Hình 3.17 Giao diện quản lí đồ uống.....	36
Hình 3.18 Giao diện trang thêm món mới vào menu.....	37
Hình 5.1 Sprint Backlog của Sprint 1.....	47
Hình 5.2 Sprint Backlog của Sprint 2.....	48
Hình 5.3 Sprint Backlog của Sprint 3.....	49
Hình 5.4 Sprint Backlog của Sprint 4.....	49
Hình 5.5 Sprint Backlog của Sprint 5.....	50
Hình 6.1 Kiểm thử đăng nhập đúng.....	53
Hình 6.2 Kiểm thử khi đăng nhập sai.....	54
Hình 6.3 Kiểm thử hiển thị danh sách sản phẩm.....	54
Hình 6.4 Kiểm thử thông tin người dùng.....	55
Hình 6.5 Kiểm thử xem đơn hàng.....	55

DANH MỤC BẢNG BIỂU

Bảng 2.1 Product Backlog chi tiết.....	20
Bảng 5.1 Product Backlog chi tiết.....	47
Bảng 5.1 Phân công công việc các thành viên	52

CHƯƠNG 1 GIỚI THIỆU

1.1. Lý do chọn đề tài

Trong bối cảnh công nghệ thông tin ngày càng phát triển, các hoạt động mua sắm trực tuyến đã trở nên phổ biến và là xu hướng tất yếu trong đời sống hiện đại. Người tiêu dùng ngày nay ưu tiên sự nhanh chóng, tiện lợi và tiết kiệm thời gian khi lựa chọn sản phẩm, đặc biệt đối với các mặt hàng tiêu dùng nhanh như đồ uống và đồ ăn nhẹ.

Tuy nhiên, thực tế cho thấy phần lớn các quán nước nhỏ vẫn đang áp dụng hình thức nhận đơn hàng truyền thống như gọi điện thoại hoặc nhắn tin qua mạng xã hội. Phương pháp này tồn tại nhiều hạn chế:

- **Thiếu tính chuyên nghiệp:** Không có giao diện tập trung để khách hàng đặt hàng.
- **Khó kiểm soát dữ liệu:** Quản lý đơn hàng và sản phẩm thủ công dẫn đến sai sót.
- **Không có hệ thống báo cáo:** Doanh thu, số lượng đơn hàng không được thống kê một cách khoa học.
- Từ những vấn đề trên, nhóm quyết định thực hiện đề tài “**Xây dựng website Cafe**” nhằm:
- Hỗ trợ các cửa hàng tối ưu quy trình bán hàng, giảm thời gian và chi phí vận hành.
- Tạo sự tiện lợi cho khách hàng, cải thiện trải nghiệm người dùng.
- Ứng dụng kiến thức lập trình web và các công cụ hiện đại đã học vào một sản phẩm thực tế, đảm bảo tính ứng dụng cao.

1.2. Mục tiêu của ứng dụng

Dự án hướng đến việc xây dựng một hệ thống phần mềm với các mục tiêu sau:

- **Đối với khách hàng:**
 - + Đăng ký và đăng nhập tài khoản bảo mật.
 - + Xem danh sách sản phẩm và tìm kiếm nhanh chóng.
 - + Thêm sản phẩm vào giỏ hàng và thực hiện đặt hàng.
 - + Theo dõi trạng thái đơn hàng và xem lịch sử giao dịch.
- **Đối với quản trị viên (Admin):**

-
- + Quản lý danh sách sản phẩm (thêm, chỉnh sửa, xóa).
 - + Quản lý đơn hàng (cập nhật trạng thái, xử lý khiếu nại).
 - + Quản lý thông tin người dùng.
- **Về mặt kỹ thuật:**
- + Xây dựng hệ thống **theo kiến trúc Client – Server** với **Frontend ReactJS** và **Backend Node.js + Express**.
 - + Giao tiếp giữa các thành phần thông qua **RESTful API**, đảm bảo tính mở rộng và bảo trì dễ dàng.
 - + Thiết kế giao diện hiện đại với **UI/UX thân thiện**, tối ưu cho nhiều thiết bị (Responsive Design).
 - + **Docker** để triển khai, và **GitHub Actions** để thực hiện CI/CD.

1.3. Tính cấp thiết của đề tài

Đề tài mang tính cấp thiết vì những nguyên nhân sau:

- **Xu hướng thị trường:** Ngày càng nhiều khách hàng chọn mua hàng online, đặc biệt trong bối cảnh dịch vụ giao hàng phát triển mạnh.
- **Tính cạnh tranh:** Các cửa hàng không áp dụng công nghệ sẽ bị hạn chế khả năng cạnh tranh so với đối thủ.
- **Chuyển đổi số trong kinh doanh:** Việc ứng dụng công nghệ trong quy trình bán hàng là bước quan trọng giúp doanh nghiệp tối ưu chi phí và tăng doanh thu.
- **Khả năng triển khai thực tế:** Giải pháp này không chỉ phục vụ một quán nước mà có thể mở rộng cho các mô hình khác như quán cà phê, đồ ăn nhanh, nhà hàng quy mô nhỏ.

1.4. Phạm vi và đối tượng nghiên cứu

Phạm vi nghiên cứu của đề tài tập trung vào việc xây dựng một hệ thống website bán cafe , với mục tiêu chính là hỗ trợ quy trình đặt hàng trực tuyến và quản lý kinh doanh cho các cửa hàng quy mô nhỏ. Hệ thống được thiết kế để cung cấp một nền tảng đơn giản, dễ sử dụng cho khách hàng, đồng thời đảm bảo khả năng quản lý dữ liệu tập trung cho người quản trị.

Về phạm vi chức năng, ứng dụng cho phép khách hàng thực hiện các thao tác cơ bản như đăng ký, đăng nhập, tìm kiếm sản phẩm, thêm sản phẩm vào giỏ hàng, đặt

hàng và theo dõi trạng thái đơn hàng. Đồng thời, hệ thống cũng hỗ trợ chức năng quản trị cho chủ cửa hàng như quản lý thông tin sản phẩm (thêm, chỉnh sửa, xóa), xử lý và cập nhật trạng thái đơn hàng, cũng như quản lý thông tin khách hàng. Các tính năng này được xây dựng dựa trên nguyên tắc tối ưu hóa trải nghiệm người dùng, đảm bảo quy trình mua bán diễn ra nhanh chóng và thuận tiện.

Phạm vi kỹ thuật của đề tài bao gồm việc phát triển ứng dụng theo kiến trúc Client-Server với giao tiếp thông qua API chuẩn REST. Phía giao diện người dùng (frontend) được xây dựng bằng ReactJS kết hợp với TailwindCSS để đảm bảo tính hiện đại và tương thích đa nền tảng. Phần xử lý nghiệp vụ (backend) sử dụng Node.js với framework Express, cùng với hệ quản trị cơ sở dữ liệu quan hệ MySQL. Toàn bộ hệ thống được triển khai trên môi trường container hóa bằng Docker, kết hợp với quy trình CI/CD sử dụng GitHub Actions nhằm nâng cao tính tự động hóa trong triển khai và kiểm thử.

Đối tượng nghiên cứu chủ yếu của hệ thống gồm hai nhóm chính: khách hàng và quản trị viên. Khách hàng là những người có nhu cầu đặt đồ uống hoặc đồ ăn nhẹ trực tuyến thông qua giao diện web, trong khi quản trị viên là người điều hành cửa hàng, có quyền quản lý toàn bộ dữ liệu liên quan đến sản phẩm, đơn hàng và thông tin khách hàng. Các chức năng được thiết kế nhằm đáp ứng nhu cầu thực tế của hai nhóm đối tượng này, đồng thời bảo đảm tính bảo mật và dễ sử dụng.

1.5. Ý nghĩa thực tiễn của đề tài

Đề tài mang lại giá trị thiết thực cả trong lĩnh vực kinh doanh và giáo dục. Về phía người tiêu dùng, việc triển khai một nền tảng bán hàng trực tuyến giúp khách hàng tiết kiệm thời gian và công sức khi mua sắm, đặc biệt trong bối cảnh công nghệ ngày càng phát triển và nhu cầu tiện lợi ngày càng cao. Người dùng có thể dễ dàng tìm kiếm sản phẩm, so sánh giá, đặt hàng và theo dõi tình trạng đơn hàng mà không cần phải liên hệ trực tiếp hoặc đến cửa hàng, qua đó cải thiện đáng kể trải nghiệm mua sắm.

Đối với chủ cửa hàng, hệ thống hỗ trợ quản lý quy trình kinh doanh một cách khoa học và chính xác. Các chức năng như quản lý tồn kho, xử lý đơn hàng và thống kê doanh thu giúp tối ưu hóa hoạt động kinh doanh, giảm thiểu sai sót do thao tác thủ công, đồng thời cung cấp dữ liệu cần thiết để đưa ra quyết định chiến lược. Việc áp

dụng công nghệ vào hoạt động kinh doanh không chỉ tăng hiệu quả quản lý mà còn góp phần nâng cao tính cạnh tranh trên thị trường, đặc biệt trong giai đoạn chuyển đổi số hiện nay.

Về phương diện giáo dục, đề tài giúp sinh viên áp dụng kiến thức lý thuyết về công nghệ phần mềm vào thực tiễn, từ khâu phân tích yêu cầu, thiết kế kiến trúc hệ thống đến triển khai và kiểm thử. Sinh viên được trải nghiệm các công cụ và quy trình phát triển phần mềm chuyên nghiệp như quản lý dự án bằng Jira, thiết kế giao diện với Figma, kiểm thử API bằng Postman và triển khai với Docker. Đây là những kỹ năng quan trọng trong ngành công nghệ thông tin, giúp sinh viên nâng cao năng lực chuyên môn và sẵn sàng đáp ứng yêu cầu của thị trường lao động.

1.6. Công nghệ và công cụ sử dụng

Để đảm bảo tính hiện đại, dễ bảo trì và mở rộng, đề tài lựa chọn các công nghệ và công cụ phổ biến, được sử dụng rộng rãi trong ngành phát triển phần mềm hiện nay. Về phía frontend, ứng dụng được xây dựng bằng ReactJS, một thư viện JavaScript mạnh mẽ, hỗ trợ xây dựng giao diện động và tối ưu hiệu suất. ReactJS được kết hợp với TailwindCSS để tạo giao diện trực quan, đẹp mắt và thân thiện với người dùng.

Về backend, hệ thống sử dụng Node.js làm nền tảng chạy phía máy chủ, kết hợp với ExpressJS để xây dựng API theo chuẩn RESTful. Cơ sở dữ liệu lựa chọn là MongoDB, một hệ quản trị cơ sở dữ liệu quan hệ phổ biến, đảm bảo khả năng lưu trữ và truy xuất dữ liệu hiệu quả. Để kiểm thử API và đảm bảo tính chính xác trong trao đổi dữ liệu, Postman được sử dụng như một công cụ hỗ trợ đắc lực, trong khi Swagger được tích hợp để tự động tạo tài liệu API, giúp cho việc quản lý và phát triển sau này trở nên dễ dàng hơn.

Trong khâu triển khai, Docker được áp dụng nhằm đóng gói ứng dụng và triển khai trên môi trường container hóa, đảm bảo tính đồng nhất giữa các môi trường phát triển, kiểm thử và sản xuất. Quy trình CI/CD được thiết lập thông qua GitHub Actions, cho phép tự động hóa kiểm thử và triển khai, từ đó nâng cao hiệu quả làm việc và giảm thiểu rủi ro khi đưa hệ thống vào vận hành. Ngoài ra, để quản lý dự án và lập kế hoạch theo phương pháp Agile, nhóm sử dụng Jira, trong khi GitHub đóng vai trò là nền tảng quản lý mã nguồn, hỗ trợ làm việc nhóm và kiểm soát phiên bản hiệu quả.

Việc lựa chọn các công nghệ này không chỉ đáp ứng yêu cầu chức năng và phi chức năng của đề tài mà còn phù hợp với xu hướng phát triển hiện đại, đảm bảo khả năng mở rộng và bảo trì hệ thống trong tương lai.

CHƯƠNG 2: PHÂN TÍCH YÊU CẦU

2.1. Khảo sát hiện trạng

Hiện nay, nhu cầu mua nước giải khát và đồ ăn nhẹ ngày càng gia tăng do sự phát triển mạnh mẽ của đời sống xã hội và xu hướng tiêu dùng tiện lợi. Tuy nhiên, đa số các cửa hàng, quán nước vẫn áp dụng phương thức kinh doanh truyền thống hoặc thông qua các kênh mạng xã hội như Facebook, Zalo, vốn tồn tại nhiều hạn chế. Người tiêu dùng thường gặp khó khăn trong việc tiếp cận thông tin sản phẩm, phải liên hệ trực tiếp qua điện thoại hoặc tin nhắn để đặt hàng, điều này không chỉ mất thời gian mà còn dễ gây nhầm lẫn hoặc sai sót trong quá trình trao đổi. Đồng thời, khách hàng không có khả năng theo dõi trạng thái đơn hàng, dẫn đến trải nghiệm mua sắm chưa tối ưu.

Đối với chủ cửa hàng, việc quản lý sản phẩm, đơn hàng và thông tin khách hàng phần lớn vẫn thực hiện thủ công, gây tốn thời gian và dễ xảy ra sai sót. Bên cạnh đó, thiếu một hệ thống báo cáo thống kê doanh thu khiến việc phân tích và dự báo kinh doanh trở nên khó khăn, ảnh hưởng đến hiệu quả vận hành. Trong bối cảnh xu hướng số hóa và thương mại điện tử đang phát triển mạnh mẽ, việc xây dựng một hệ thống bán hàng trực tuyến hiện đại là yêu cầu cấp thiết nhằm giải quyết các hạn chế nói trên. Một website chuyên biệt cho phép khách hàng đặt hàng nhanh chóng, quản lý đơn hàng dễ dàng, đồng thời hỗ trợ chủ cửa hàng kiểm soát toàn bộ hoạt động kinh doanh một cách hiệu quả sẽ là giải pháp tối ưu và phù hợp với yêu cầu thực tiễn.

2.2. Yêu cầu chức năng

2.2.1. Chức năng dành cho khách hàng

Hệ thống cần cung cấp các chức năng cơ bản và nâng cao cho khách hàng:

- **Đăng ký/Đăng nhập:**
 - + Cho phép người dùng tạo tài khoản mới hoặc đăng nhập bằng tài khoản hiện có.
 - + Hỗ trợ lưu mật khẩu an toàn (mã hóa).
- **Xem danh sách sản phẩm:**
 - + Hiện thị sản phẩm kèm hình ảnh, giá, mô tả chi tiết.

-
- **Tìm kiếm & lọc sản phẩm:**
 - + Tìm theo từ khóa, phân loại theo giá, loại sản phẩm (nước ngọt, cà phê, bánh snack).
 - **Quản lý giỏ hàng:**
 - + Thêm sản phẩm, cập nhật số lượng, xóa sản phẩm.
 - **Đặt hàng & thanh toán:**
 - + Xác nhận thông tin giao hàng, phương thức thanh toán (tiền mặt khi nhận hàng hoặc ví điện tử).
 - **Theo dõi trạng thái món:**
 - + Hiện thị tiến trình: Đang xử lý → Đang giao → Hoàn tất.

2.2.2. Chức năng dành cho quản trị viên

- **Quản lý sản phẩm:**
 - + Thêm, sửa, xóa sản phẩm trong danh mục.
- **Quản lý đơn hàng:**
 - + Xác nhận đơn mới, cập nhật trạng thái, hủy đơn hàng.
- **Quản lý khách hàng:**
 - + Xem thông tin khách hàng, lịch sử mua hàng, hỗ trợ khi cần.
- **Xem thống kê:**
 - + Báo cáo doanh thu theo ngày, tháng, năm, thống kê sản phẩm bán chạy.

2.3. Yêu cầu phi chức năng

2.3.1. Hiệu năng

- Hệ thống phải đáp ứng tối thiểu 100 yêu cầu truy cập đồng thời mà không gây ra hiện tượng nghẽn hoặc chậm phản hồi.
- Thời gian tải trang chính không được vượt quá 3 giây trên đường truyền internet tiêu chuẩn.
- Truy vấn cơ sở dữ liệu cho các thao tác tìm kiếm và lọc sản phẩm phải trả về kết quả trong vòng 2 giây.

2.3.2. Bảo mật

- Toàn bộ dữ liệu truyền tải giữa client và server phải được mã hóa bằng giao thức HTTPS.

-
- Mật khẩu người dùng được mã hóa bằng thuật toán an toàn như bcrypt hoặc SHA-256.
 - Hệ thống phải có cơ chế ngăn chặn tấn công phổ biến như SQL Injection, Cross-Site Scripting (XSS) và Cross-Site Request Forgery (CSRF).
 - Tích hợp cơ chế xác thực và phân quyền người dùng rõ ràng: Admin và Khách hàng.

2.3.3. Khả dụng và Tính ổn định

- Hệ thống phải đảm bảo thời gian hoạt động (uptime) đạt tối thiểu 99% trong tháng.
- Khi xảy ra sự cố, phải có cơ chế khôi phục dữ liệu trong vòng tối đa 1 giờ.
- Website hoạt động ổn định trên các trình duyệt phổ biến (Chrome, Firefox, Edge) và hệ điều hành khác nhau (Windows, Android, iOS).

2.3.4. Khả năng mở rộng

- Thiết kế hệ thống theo kiến trúc **Client-Server** và **RESTful API** để dễ dàng mở rộng thêm tính năng hoặc tích hợp với các nền tảng khác.
- Cho phép bổ sung phương thức thanh toán điện tử, ví điện tử hoặc tích hợp chatbot hỗ trợ khách hàng mà không cần thay đổi toàn bộ kiến trúc.

2.3.5. Tính khả dụng

- Giao diện thân thiện, dễ sử dụng cho cả người dùng mới.
- Hỗ trợ đa ngôn ngữ nếu mở rộng quy mô kinh doanh.
- Thiết kế responsive để tương thích trên máy tính, máy tính bảng và điện thoại thông minh.

2.3.6. Khả năng bảo trì

- Mã nguồn phải được viết theo chuẩn coding convention, dễ đọc và dễ mở rộng.
- Tách biệt **Frontend (ReactJS)** và **Backend (NodeJS + Express)** giúp bảo trì và phát triển nhanh chóng.

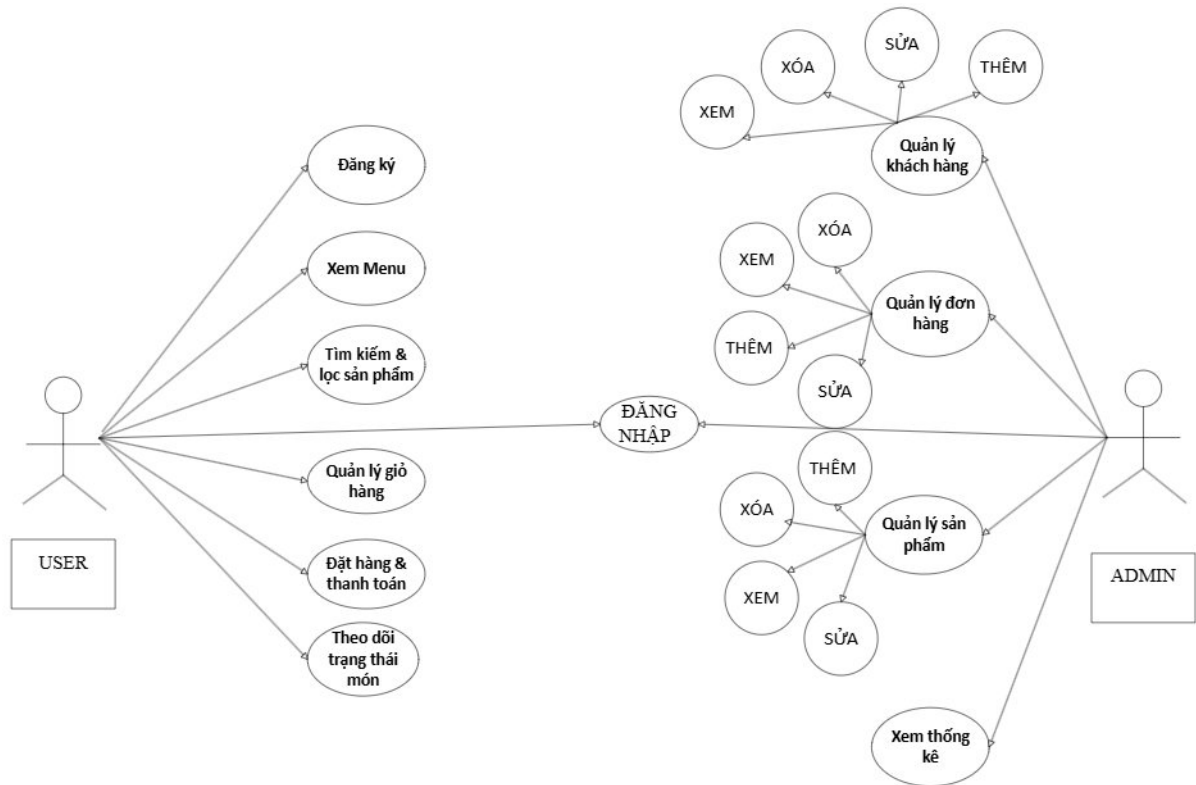
2.3.7. Khả năng kiểm thử

- Tích hợp công cụ kiểm thử API (Postman) và kiểm thử tự động (GitHub Actions).

- Có bộ kiểm thử đơn vị (unit test) cho các chức năng quan trọng như đăng nhập, đặt hàng, thanh toán.

2.4. Đặc tả Use Case

2.4.1. Biểu đồ Use Case tổng quan



Hình 2.1 Biểu đồ Use Case tổng quan

2.5. Product Backlog

ID	User Story	Ưu tiên	Điểm
1	Là khách hàng, tôi muốn đăng ký tài khoản để có thể đặt hàng.	Cao	3
2	Là khách hàng, tôi muốn đăng nhập để quản lý giỏ hàng và đơn hàng.	Cao	3
3	Là khách hàng, tôi muốn xem danh sách sản phẩm để chọn mua.	Cao	5
4	Là khách hàng, tôi muốn tìm kiếm sản phẩm theo từ khóa.	Trung bình	3
5	Là khách hàng, tôi muốn quản lý giỏ hàng	Cao	8

	của mình.		
6	Là khách hàng, tôi muốn đặt hàng trực tuyến dễ dàng.	Cao	8
7	Là khách hàng, tôi muốn theo dõi trạng thái đơn hàng.	Trung bình	5
8	Là admin, tôi muốn thêm và chỉnh sửa sản phẩm.	Cao	8
9	Là admin, tôi muốn quản lý đơn hàng của khách hàng.	Cao	8
10	Là admin, tôi muốn xem báo cáo doanh thu chi tiết.	Trung bình	5

Bảng 2.1 Product Backlog chi tiết

CHƯƠNG 3: THIẾT KẾ HỆ THỐNG

3.1. Kiến trúc tổng thể hệ thống

3.1.1. Mô hình kiến trúc (Client-Server, REST API)

Hệ thống được thiết kế theo mô hình **Client-Server**, trong đó **Frontend** (giao diện người dùng) và **Backend** (xử lý logic và dữ liệu) được tách biệt rõ ràng, giao tiếp với nhau thông qua **RESTful API**. Kiến trúc này đảm bảo tính linh hoạt, dễ mở rộng và dễ dàng bảo trì khi triển khai thực tế.

Frontend: Được xây dựng bằng **ReactJS**, đóng vai trò chịu trách nhiệm hiển thị giao diện người dùng, tiếp nhận các thao tác từ khách hàng như chọn sản phẩm, đặt hàng và gửi yêu cầu đến server. ReactJS hỗ trợ cơ chế component-based giúp tái sử dụng mã nguồn, cải thiện tốc độ phản hồi và mang lại trải nghiệm mượt mà cho người dùng.

Backend: Được phát triển bằng **Node.js** kết hợp **Express**, chịu trách nhiệm xử lý các yêu cầu từ client, quản lý dữ liệu, xác thực người dùng và thực hiện các nghiệp vụ như quản lý sản phẩm, đơn hàng, người dùng. Backend thực hiện các thao tác CRUD (Create, Read, Update, Delete) và trả kết quả về cho client dưới dạng **JSON**, đảm bảo sự tương thích và dễ tích hợp với nhiều nền tảng khác.

Database: Hệ thống sử dụng **MongoDB** để lưu trữ dữ liệu, đảm bảo tính toàn vẹn và nhất quán. Cơ sở dữ liệu được thiết kế theo mô hình quan hệ với các bảng chính như **Users** (thông tin người dùng), **Products** (thông tin sản phẩm), **Orders** (đơn hàng), **OrderDetails** (chi tiết đơn hàng) và **Categories** (danh mục sản phẩm). Việc chuẩn hóa cơ sở dữ liệu giúp giảm thiểu trùng lặp và tối ưu hiệu suất truy vấn.

Mô hình kiến trúc tổng thể của hệ thống được chia thành **3 lớp chính**:

- **Presentation Layer:** Lớp hiển thị giao diện web cho khách hàng và quản trị viên, đảm bảo trải nghiệm người dùng thân thiện và trực quan.
- **Application Layer:** Lớp xử lý nghiệp vụ, nơi triển khai API, thực hiện xác thực, xử lý logic đặt hàng và quản lý thông tin sản phẩm.
- **Data Layer:** Lớp lưu trữ dữ liệu trong hệ quản trị cơ sở dữ liệu quan hệ, đảm bảo an toàn và toàn vẹn thông tin.

Việc áp dụng kiến trúc này không chỉ giúp hệ thống hoạt động ổn định mà còn dễ dàng mở rộng trong tương lai, chẳng hạn như tích hợp với ứng dụng di động hoặc triển khai trên môi trường điện toán đám mây.

3.1.2. Nguyên tắc RESTful áp dụng

Hệ thống được thiết kế tuân thủ theo các nguyên tắc của **REST (Representational State Transfer)** nhằm đảm bảo API hoạt động hiệu quả, dễ sử dụng và có khả năng mở rộng trong tương lai. Việc áp dụng kiến trúc RESTful giúp cho việc giao tiếp giữa Client và Server trở nên đơn giản, nhất quán và dễ dàng tích hợp với các ứng dụng hoặc dịch vụ khác.

Các tài nguyên trong hệ thống được định danh rõ ràng thông qua các **URI** cụ thể. Ví dụ: `/api/products` được sử dụng để truy xuất danh sách sản phẩm, `/api/orders` để quản lý thông tin đơn hàng. Các thao tác với tài nguyên được thực hiện thông qua các phương thức HTTP tiêu chuẩn:

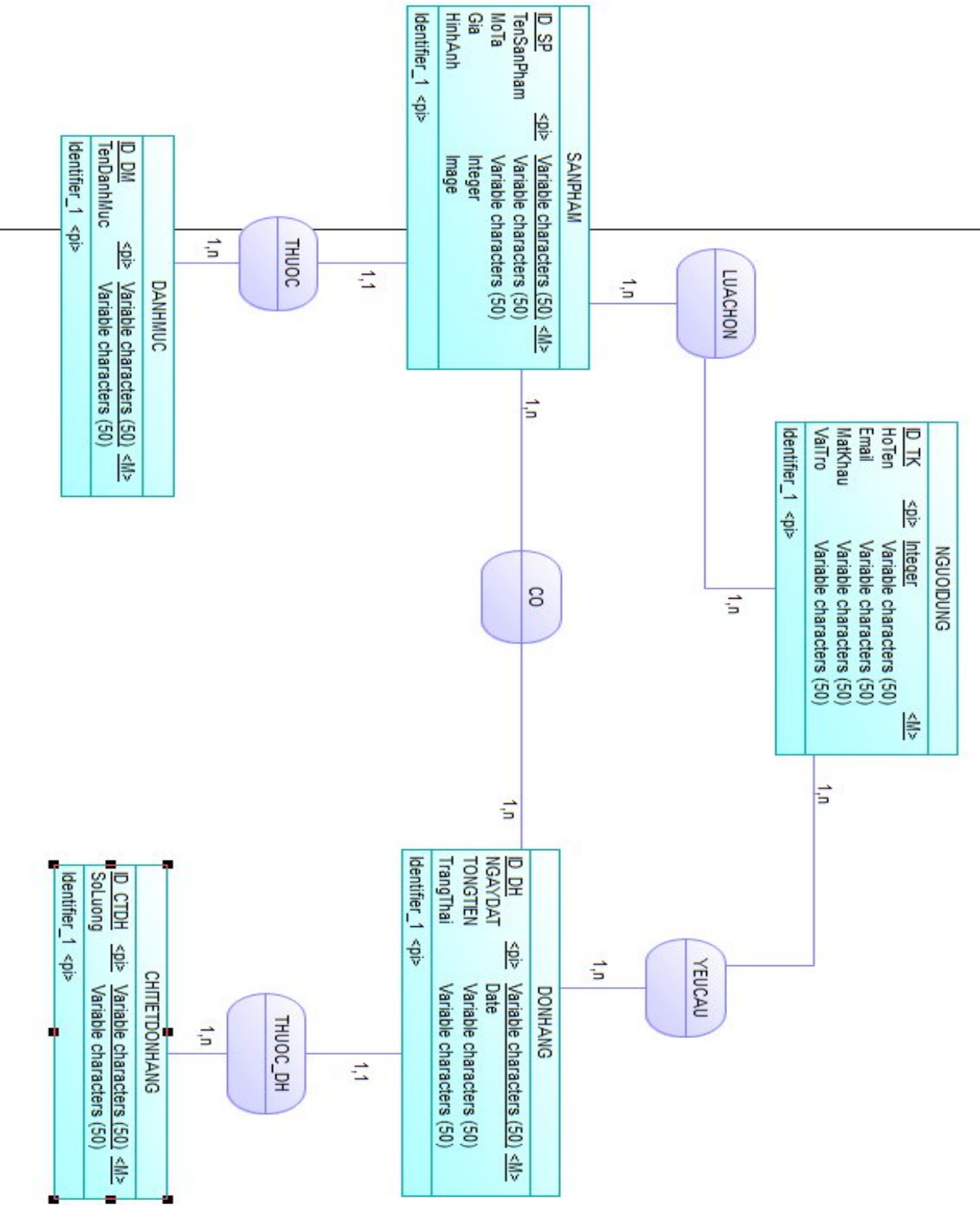
- **GET**: Lấy dữ liệu từ máy chủ, chẳng hạn như danh sách sản phẩm hoặc chi tiết đơn hàng.
- **POST**: Gửi dữ liệu mới để tạo tài nguyên, ví dụ tạo tài khoản hoặc đặt hàng mới.
- **PUT/PATCH**: Cập nhật thông tin của tài nguyên đã tồn tại, chẳng hạn chỉnh sửa thông tin sản phẩm.
- **DELETE**: Xóa tài nguyên khỏi hệ thống, ví dụ xóa sản phẩm hoặc đơn hàng.

Tất cả dữ liệu phản hồi từ server được trả về ở định dạng **JSON**, một chuẩn phổ biến, dễ xử lý và tương thích với nhiều nền tảng khác nhau. Ngoài ra, hệ thống đảm bảo **tính stateless**, nghĩa là mỗi yêu cầu từ Client đến Server phải chứa đầy đủ thông tin cần thiết để xử lý, không phụ thuộc vào trạng thái của các yêu cầu trước đó. Điều này giúp hệ thống dễ mở rộng, tăng tính ổn định và giảm thiểu rủi ro khi triển khai trên môi trường phân tán.

Việc tuân thủ các nguyên tắc RESTful không chỉ đảm bảo tính đơn giản và hiệu quả trong thiết kế API mà còn tạo điều kiện thuận lợi cho việc tích hợp với các ứng dụng di động hoặc các hệ thống khác trong tương lai, đồng thời hỗ trợ mở rộng sang các kiến trúc nâng cao như microservices hoặc triển khai trên điện toán đám mây.

3.2. Thiết kế cơ sở dữ liệu

3.2.1. Sơ đồ ERD



Hình 3.1 Sơ đồ ERD

3.2.2. Mô hình quan hệ và mô tả bảng

- NGUOIDUNG (ID_TK, HoTen, Email, MatKhau, VaiTro)
 - + Khóa chính (PK): ID_TK
 - + Quan hệ: 1 – N với DONHANG (một người dùng có nhiều đơn hàng).
- DANHMUC (ID_DM, TenDanhMuc)
 - + PK: ID_DM
 - + Quan hệ: 1 – N với SANPHAM (một danh mục có nhiều sản phẩm).
- SANPHAM (ID_SP, TenSanPham, MoTa, Gia, HinhAnh, ID_DM)
 - + PK: ID_SP
 - + FK: ID_DM → DANHMUC
 - + Quan hệ: 1 – N với CHITIETDONHANG (một sản phẩm có trong nhiều chi tiết đơn hàng).
- DONHANG (ID_DH, NgayDat, TongTien, TrangThai, ID_TK)
 - + PK: ID_DH
 - + FK: ID_TK → NGUOIDUNG
 - + Quan hệ: 1 – N với CHITIETDONHANG.
- CHITIETDONHANG (ID_CTDH, ID_DH, ID_SP, SoLuong)
 - + PK: ID_CTDH
 - + FK: ID_DH → DONHANG
 - + FK: ID_SP → SANPHAM

3.3. Thiết kế API

Hệ thống sử dụng kiến trúc **RESTful API**, giao tiếp giữa **Frontend (ReactJS)** và **Backend (Node.js + Express)**. API được thiết kế theo nguyên tắc **REST**, với các phương thức chuẩn:

- **GET**: Lấy dữ liệu
- **POST**: Thêm dữ liệu
- **PUT**: Cập nhật dữ liệu
- **DELETE**: Xóa dữ liệu

Các API được mô tả rõ ràng bằng **Swagger** để hỗ trợ kiểm thử và tích hợp.

3.3.1. Danh sách Endpoint chính

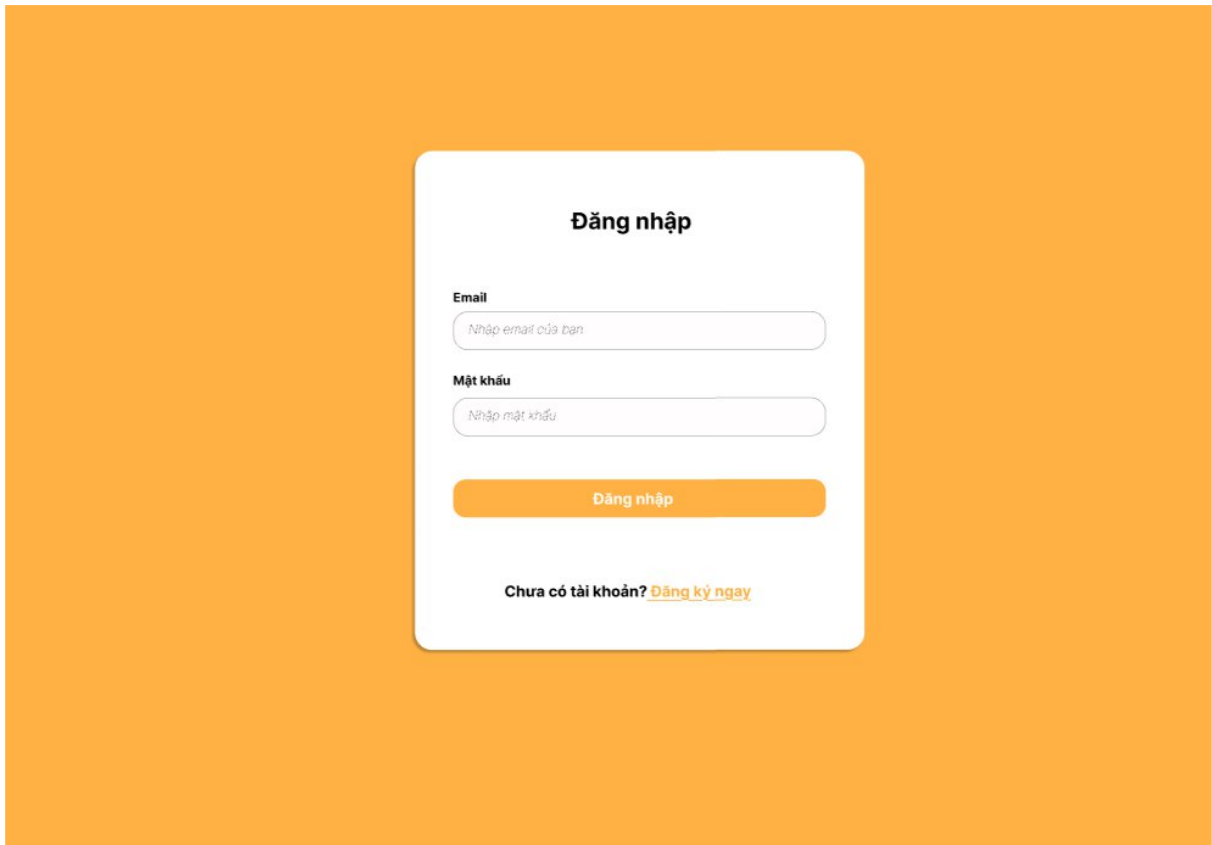
Nhóm chức năng	Phương thức	Endpoint	Mô tả
Xác thực	POST	/api/auth/register	Đăng ký tài khoản mới
	POST	/api/auth/login	Đăng nhập hệ thống
Quản lý sản phẩm	GET	/api/products	Lấy danh sách sản phẩm
	GET	/api/products/:id	Lấy chi tiết sản phẩm
	POST	/api/products	Thêm sản phẩm (Admin)
	PUT	/api/products/:id	Cập nhật sản phẩm (Admin)
	DELETE	/api/products/:id	Xóa sản phẩm (Admin)
Danh mục	GET	/api/categories	Lấy danh sách danh mục
Giỏ hàng	POST	/api/cart	Thêm sản phẩm vào giỏ
	GET	/api/cart	Xem giỏ hàng
	DELETE	/api/cart/:id	Xóa sản phẩm trong giỏ
Đơn hàng	POST	/api/orders	Tạo đơn hàng mới
	GET	/api/orders	Xem danh sách đơn hàng (User)
	GET	/api/orders	Xem chi tiết đơn hàng
Thống kê	GET	/api/statistics	Xem báo cáo doanh thu (Admin)

Bảng 3.1. Các thuộc tính trong các danh sách Endpoint chính

3.4. Thiết kế giao diện người dùng (UI/UX)

3.4.1. Figma mô phỏng

- Trang mô phỏng của form đăng nhập



The image shows a Figma mockup of a login form. The form is a white rounded rectangle centered on a solid orange background. At the top of the form, the title "Đăng nhập" is displayed in bold black text. Below the title, there are two input fields. The first is labeled "Email" and contains the placeholder text "Nhập email của bạn". The second is labeled "Mật khẩu" and contains the placeholder text "Nhập mật khẩu". Below these fields is a large orange button with the text "Đăng nhập" in white. At the bottom of the form, there is a link that says "Chưa có tài khoản? [Đăng ký ngay](#)".

Hình 3.2 Trang mô phỏng form đăng nhập bằng Figma

-
- Trang mô phỏng của form đăng kí

Đăng ký

Họ và tên
Nhập họ và tên

Email
Nhập email của bạn

Mật khẩu
Nhập mật khẩu

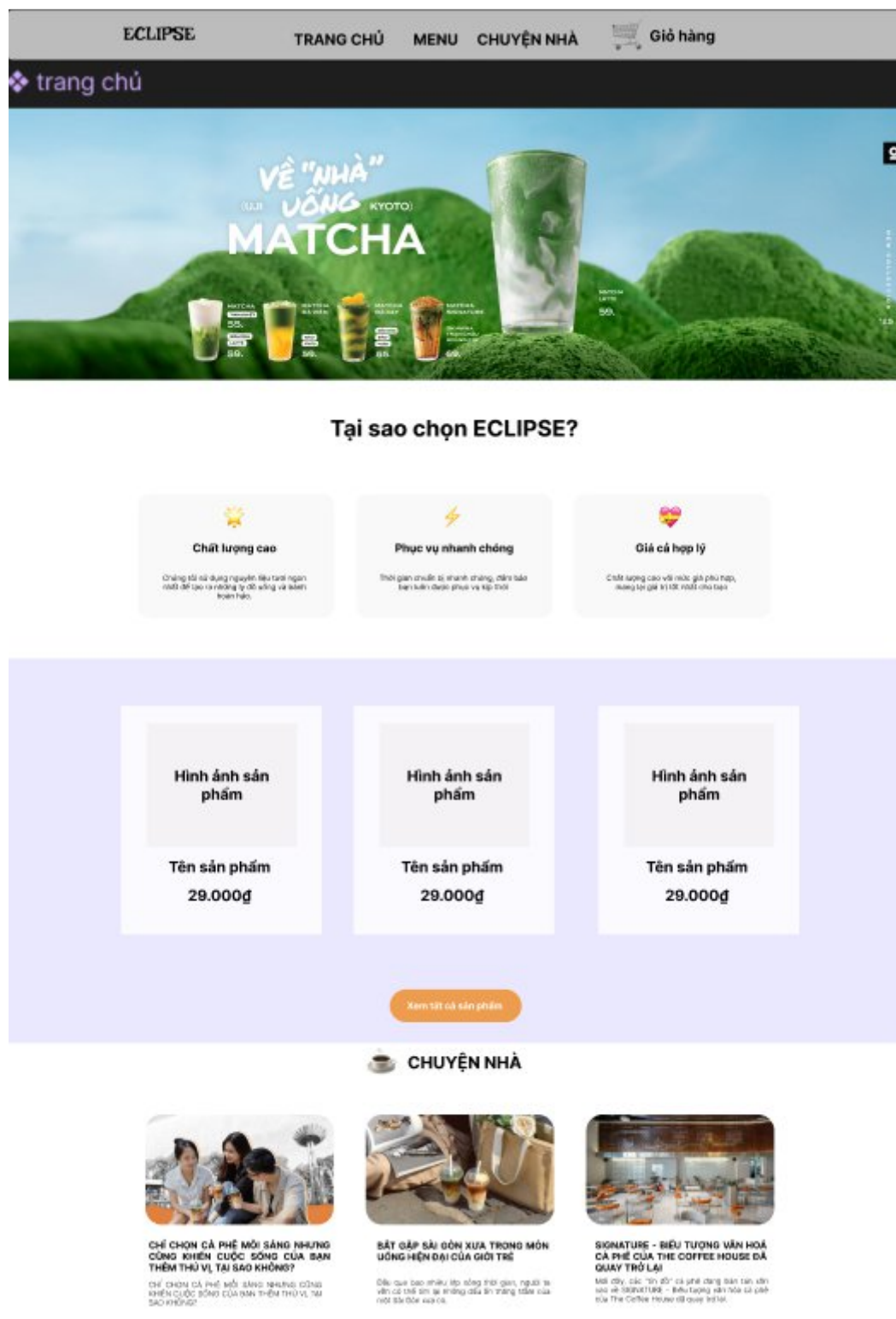
Xác nhận mật khẩu
Nhập lại mật khẩu

Đăng ký

Đã có tài khoản? [Đăng nhập ngay](#)

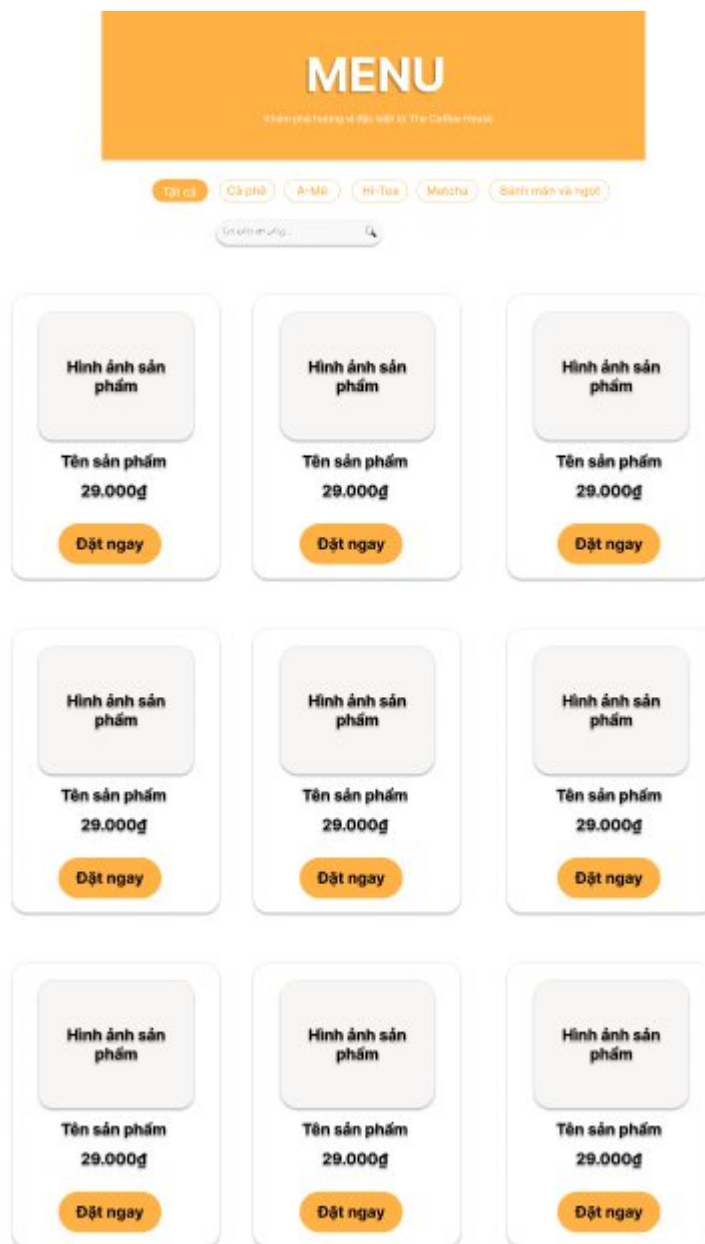
Hình 3.3 Trang mô phỏng form đăng kí bằng Figma

- Trang mô phỏng của trang chủ



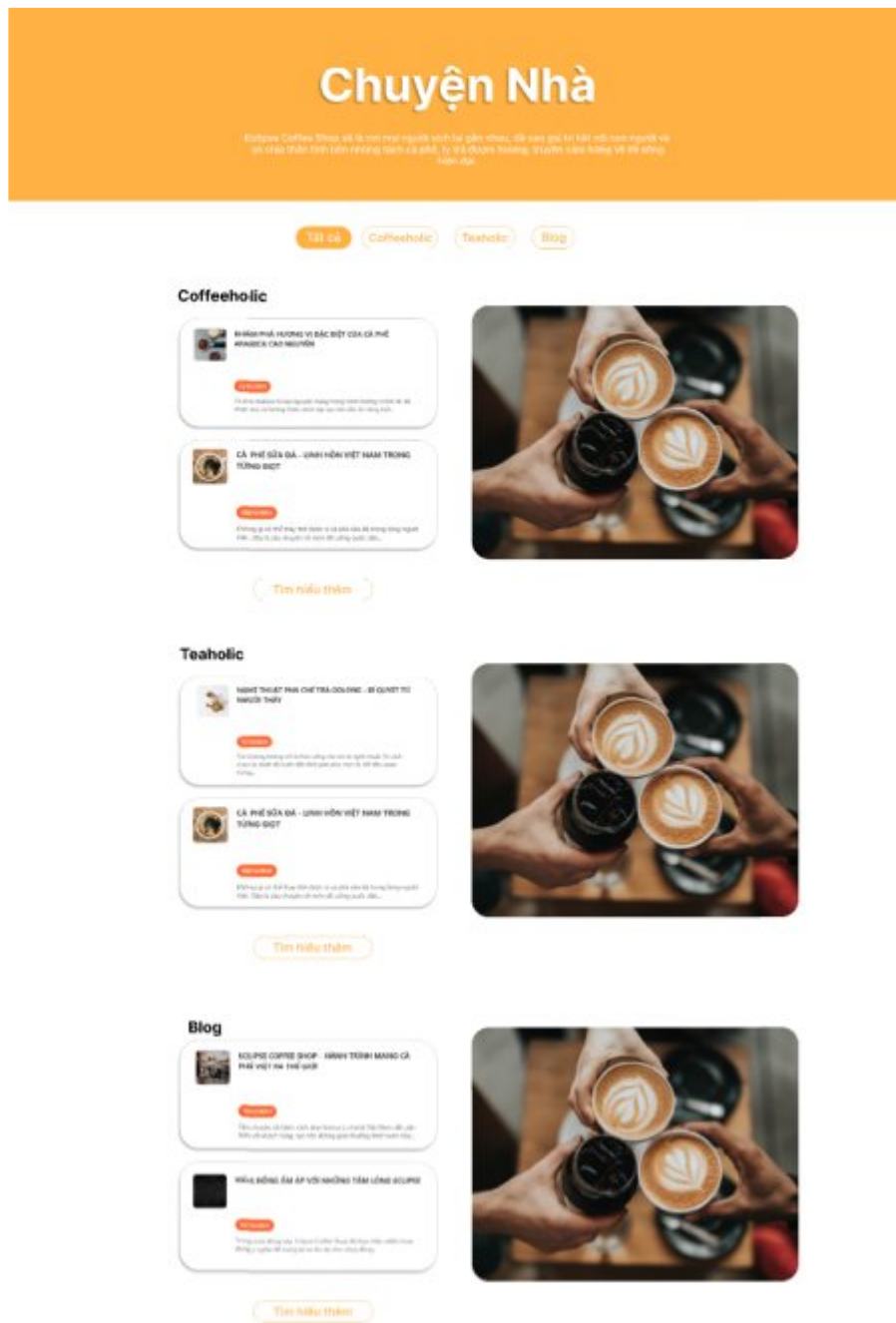
Hình 3.4 Trang mô phỏng của trang chủ bằng Figma

- Trang mô phỏng của trang menu



Hình 3.5 Trang mô phỏng của trang menu bằng Figma

- Trang mô phỏng của trang blog



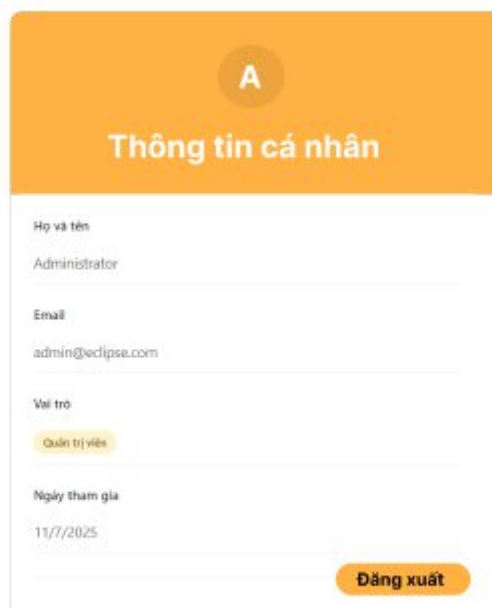
Hình 3.6 Trang mô phỏng của trang blog bằng Figma

-
- Trang mô phỏng của trang giỏ hàng



Hình 3.7 Trang mô phỏng của trang giỏ hàng bằng Figma

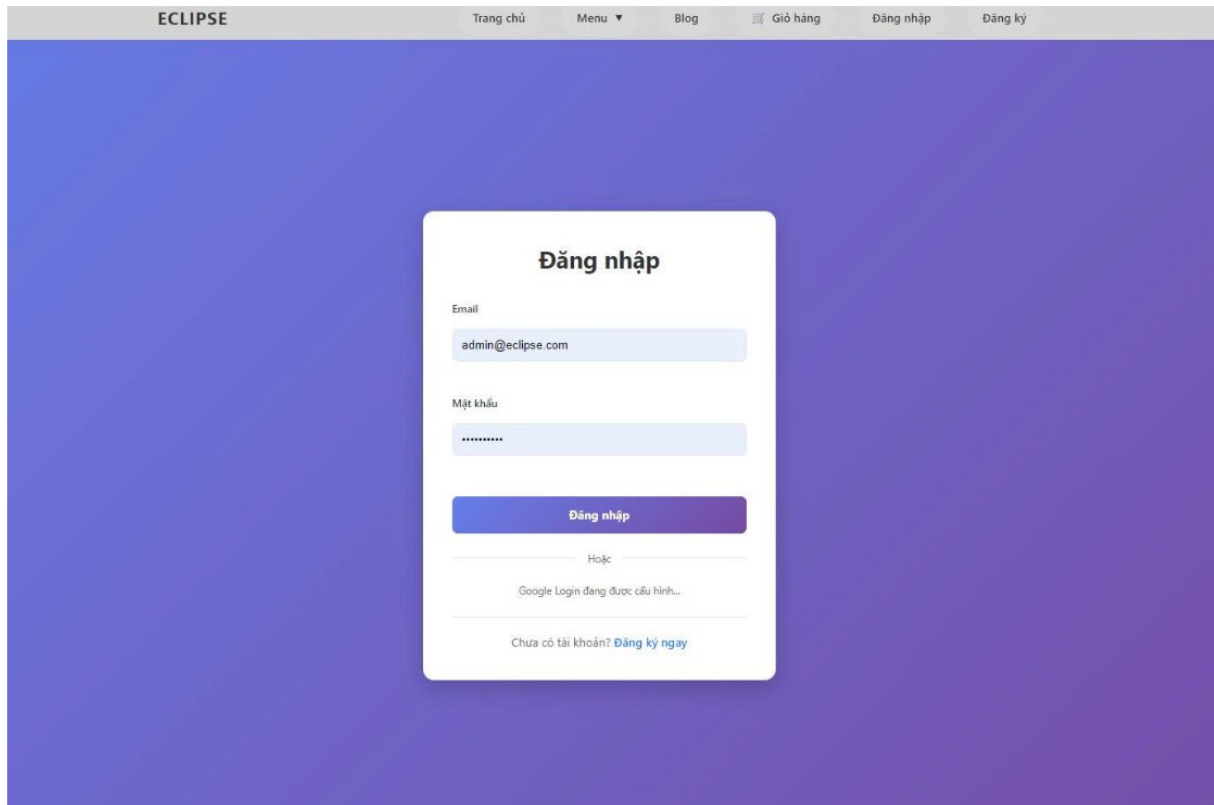
- Trang mô phỏng của trang thông tin cá nhân



Hình 3.8 Trang mô phỏng của trang thông tin cá nhân bằng Figma

3.4.2. Các giao diện chính

- Giao diện của form đăng nhập



The screenshot displays the Eclipse login interface. At the top, a navigation bar includes the 'ECLIPSE' logo and links for 'Trang chủ', 'Menu', 'Blog', 'Giỏ hàng', 'Đăng nhập', and 'Đăng ký'. The main content area features a central white login card with a purple gradient background. The card is titled 'Đăng nhập' and contains input fields for 'Email' (with 'admin@eclipse.com' entered) and 'Mật khẩu' (masked with dots). A purple 'Đăng nhập' button is positioned below the password field. Underneath the button is a horizontal line with the word 'Hoặc' in the center. Below this line, there is a link for 'Google Login đang được cấu hình...' and a link for 'Chưa có tài khoản? Đăng ký ngay'.

Hình 3.9 Giao diện đăng nhập

- Giao diện của form đăng kí

The image shows a web browser displaying the Eclipse website's registration page. The header is a light gray bar with the 'ECLIPSE' logo on the left and navigation links 'Trang chủ', 'Menu', 'Blog', 'Giỏ hàng', 'Đăng nhập', and 'Đăng ký' on the right. The main content area has a purple gradient background. In the center is a white registration form titled 'Đăng ký'. The form contains four input fields: 'Họ và tên' (placeholder: 'Nhập họ và tên'), 'Email' (placeholder: 'Nhập email của bạn'), 'Mật khẩu' (placeholder: 'Nhập mật khẩu (ít nhất 6 ký tự)'), and 'Xác nhận mật khẩu' (placeholder: 'Nhập lại mật khẩu'). Below these fields is a blue 'Đăng ký' button. Under the button is a horizontal line with the word 'Hoặc' in the center. Below that is a link 'Google Login đang được cấu hình...'. At the bottom of the form is a link 'Đã có tài khoản? Đăng nhập ngay'.

Hình 3.10 Giao diện đăng kí

- Giao diện của trang chủ

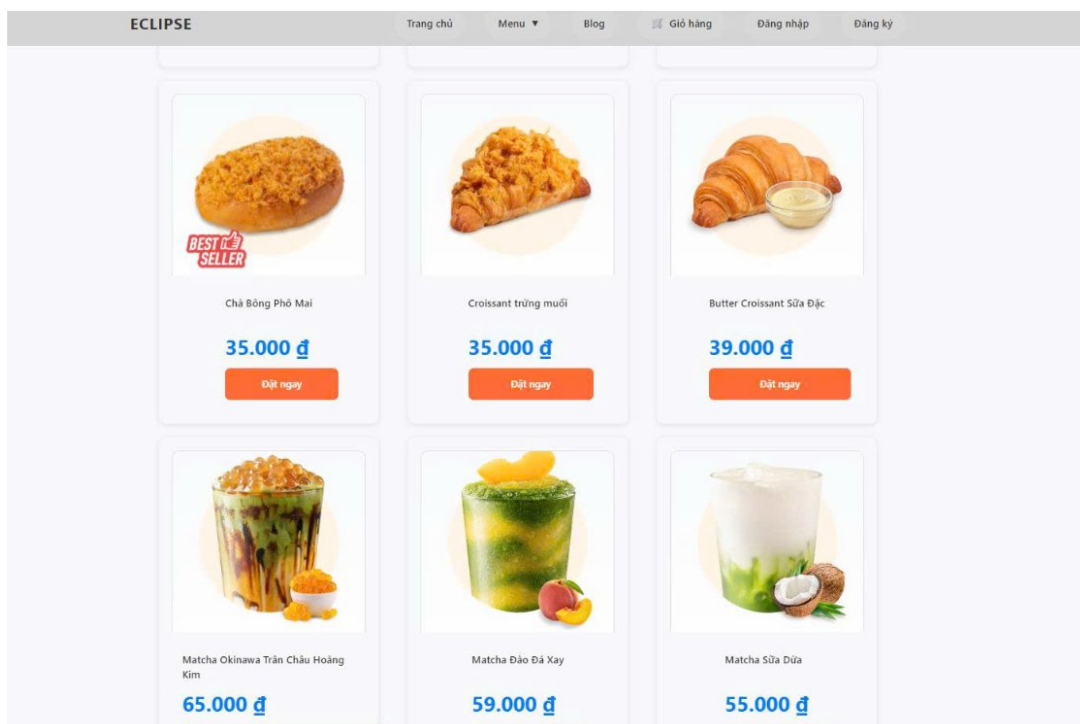


Tại sao chọn ECLIPSE?



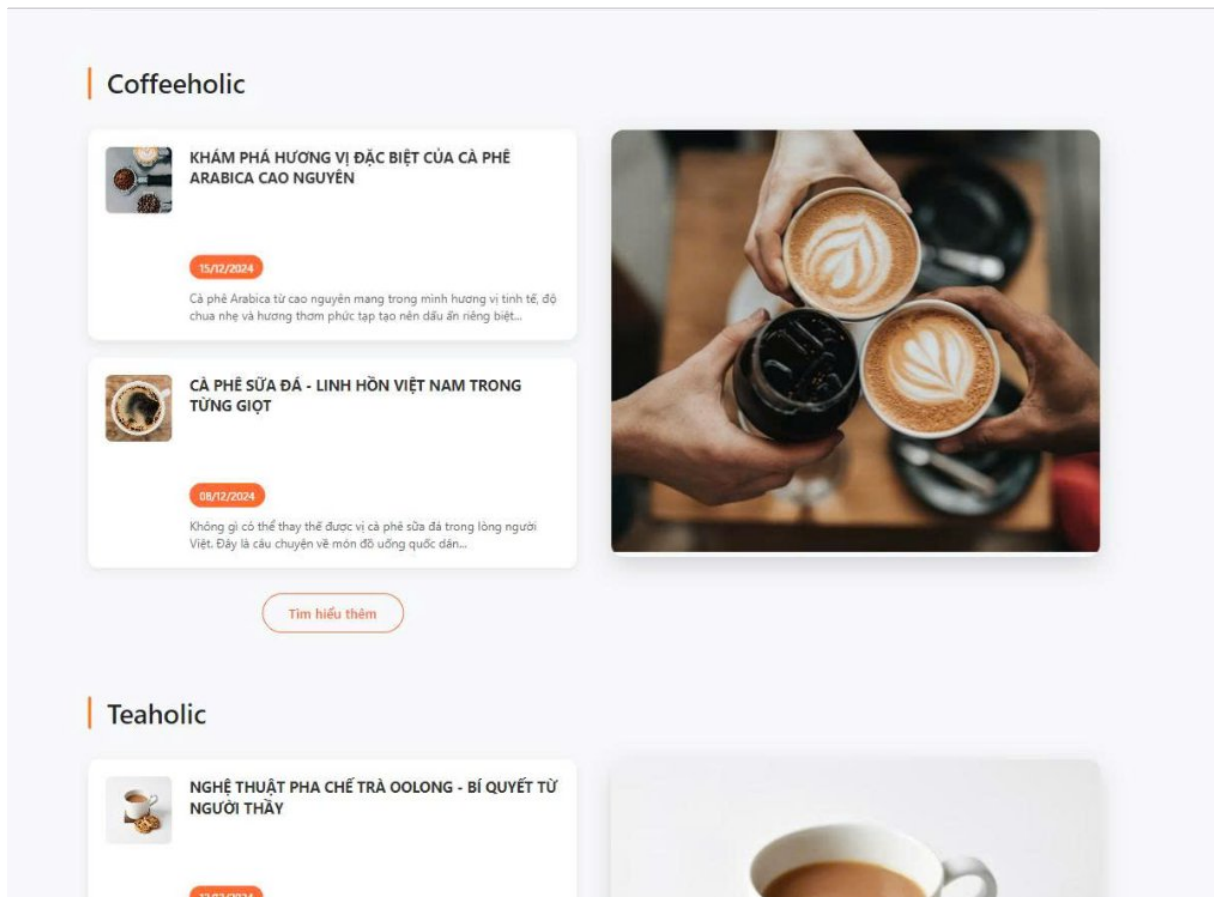
Hình 3.11 Giao diện trang chủ

- Giao diện của trang menu



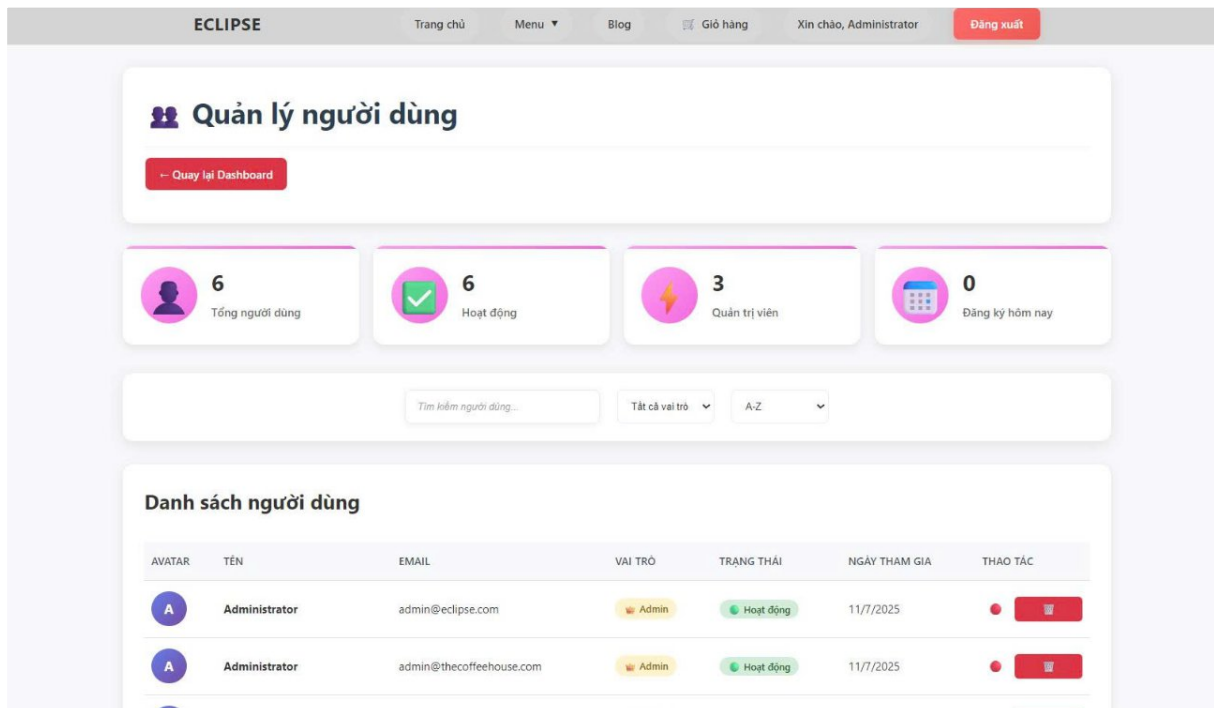
Hình 3.12 Giao diện menu

- Giao diện của trang blog



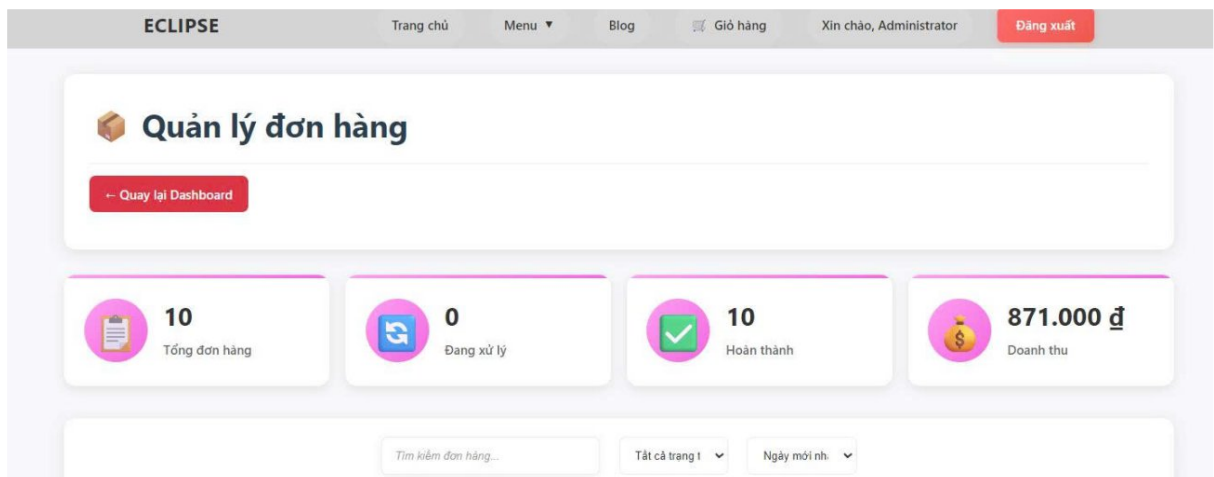
Hình 3.13 Giao diện trang blog

- Giao diện của trang quản lí người dùng



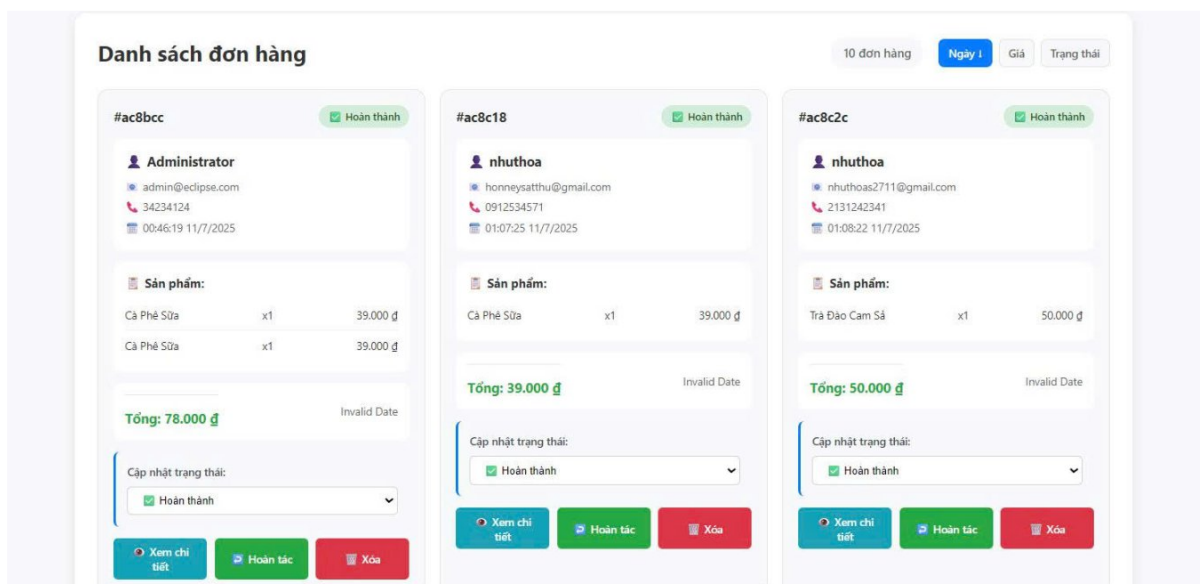
Hình 3.14 Giao diện trang quản lí người dùng

- Quản lí đơn hàng



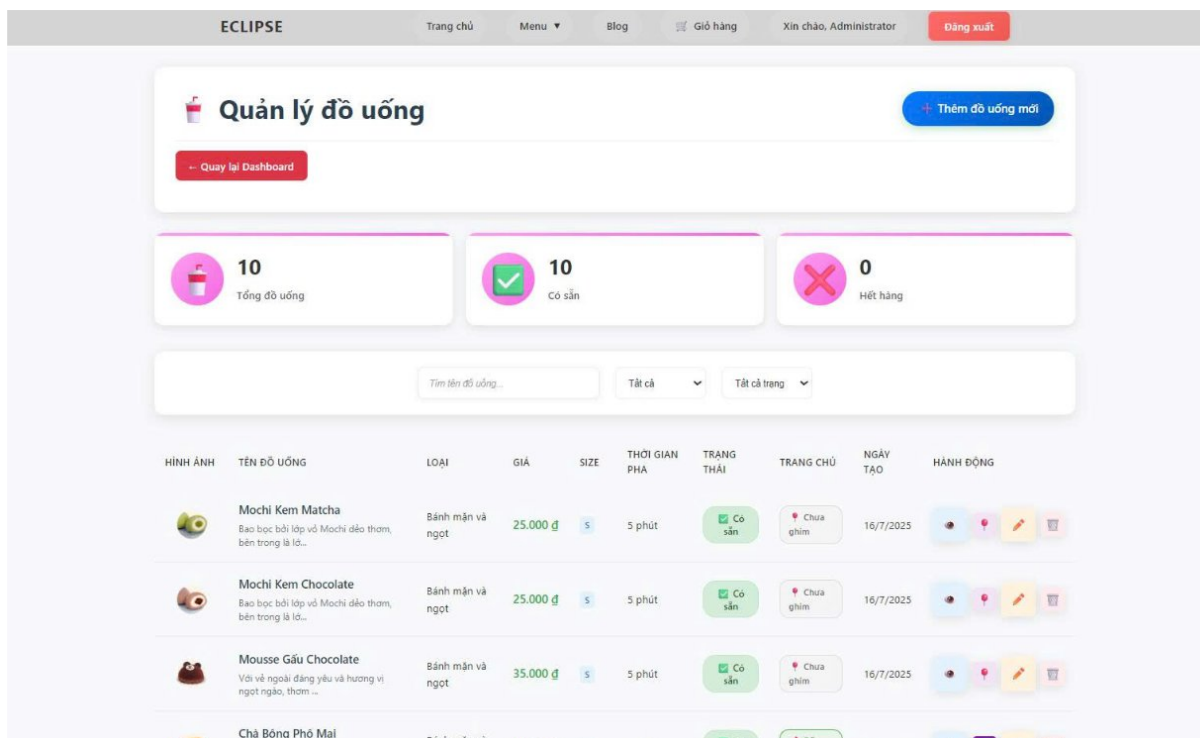
Hình 3.15 Giao diện quản lí đơn hàng

- Các danh sách đơn hàng



Hình 3.16 Danh sách các đơn hàng

- Quản lý đồ uống



Hình 3.17 Giao diện quản lý đồ uống

- Thêm món mới vào menu

The screenshot shows the 'Thêm đồ uống mới' (Add new drink) form in the ECLIPSE application. The form is titled 'Thêm đồ uống mới' and includes a 'Quay lại' (Go back) button. The form fields are as follows:

- Tên đồ uống *** (Drink name): A text input field with the placeholder 'Nhập tên đồ uống'.
- Loại đồ uống *** (Drink type): A dropdown menu with 'Cà phê' (Coffee) selected.
- Giá (VNĐ) *** (Price in VND): A text input field with the value '0'.
- Giảm giá (%)** (Discount %): A text input field with the value '0'.
- Mô tả** (Description): A text area with the placeholder 'Mô tả về đồ uống...'.
- Hình ảnh (tối đa 5 ảnh)** (Image): A file upload section with a 'Choose Files' button and the text 'No file chosen'.
- Kích thước *** (Size): Four radio buttons for 'Size S', 'Size M', 'Size L', and 'Size XL'.
- Nhiệt độ** (Temperature): Three radio buttons for 'Nóng' (Hot), 'Lạnh' (Cold), and 'Thường' (Room temperature).
- Độ ngọt** (Sweetness): A section with three radio buttons, partially visible at the bottom.

Hình 3.18 Giao diện trang thêm món mới vào menu

CHƯƠNG 4: TRIỂN KHAI VÀ CÔNG NGHỆ

4.1. Công nghệ và công cụ sử dụng

4.1.1. Công nghệ sử dụng

- **Ngôn ngữ lập trình:**

- **JavaScript (ES6+):** Là ngôn ngữ phổ biến và linh hoạt, sử dụng cho cả frontend và backend, giúp đồng bộ hóa công nghệ giữa hai phía, giảm chi phí học tập và phát triển.
- **HTML5 & CSS3:** Được sử dụng để xây dựng cấu trúc và định dạng giao diện web, đảm bảo tính thẩm mỹ và khả năng responsive.

- **Frontend:**

- **ReactJS:**

- + Lý do chọn ReactJS: Là thư viện JavaScript mạnh mẽ do Facebook phát triển, ReactJS giúp xây dựng giao diện người dùng (UI) nhanh chóng, dễ bảo trì nhờ kiến trúc component-based.
- + Các tính năng nổi bật: **Virtual DOM**, khả năng tối ưu render, hỗ trợ **React Router** cho điều hướng, và tích hợp tốt với các thư viện UI như **Material-UI**, **TailwindCSS**.
- + Ưu điểm: Hiệu suất cao, cộng đồng lớn, hỗ trợ SEO tốt nếu triển khai cùng Next.js.

- **Backend:**

- **Node.js và Express.js:**

- + Node.js: Cho phép chạy JavaScript phía server với hiệu suất cao nhờ kiến trúc **non-blocking I/O**.
- + Express.js: Framework tối ưu cho việc xây dựng RESTful API, hỗ trợ middleware, routing, và dễ tích hợp với cơ sở dữ liệu.
- + Lý do chọn: Gọn nhẹ, dễ triển khai, cộng đồng mạnh, tài liệu phong phú.

- **Cơ sở dữ liệu:**

- **MongoDB:**

- + Lý do chọn: MongoDB là cơ sở dữ liệu phù hợp cho các ứng dụng web thương mại với dữ liệu phi cấu trúc hoặc thay đổi linh hoạt.
- + Ưu điểm: Khả năng mở rộng cao, truy vấn nhanh, hỗ trợ tích hợp dễ dàng với Node.js thông qua **Mongoose**.

4.1.2 Công cụ hỗ trợ phát triển

Trong quá trình phát triển hệ thống, nhóm đã sử dụng một số công cụ hỗ trợ để đảm bảo quy trình được thực hiện hiệu quả, dễ quản lý và đạt chất lượng cao. Các công cụ chính bao gồm:

- **Figma** được sử dụng cho khâu thiết kế giao diện (UI/UX), giúp nhóm có cái nhìn trực quan về bố cục và trải nghiệm người dùng trước khi triển khai thực tế. Công cụ này hỗ trợ cộng tác trực tuyến, cho phép các thành viên cùng làm việc và đóng góp ý kiến trên một bản thiết kế chung.

- Để quản lý mã nguồn và hợp tác nhóm, **Git** và **GitHub** được lựa chọn. Git giúp kiểm soát phiên bản mã nguồn một cách an toàn, hỗ trợ làm việc nhóm hiệu quả thông qua các nhánh (branch) và chức năng merge. GitHub đóng vai trò là kho lưu trữ trực tuyến, tích hợp các tính năng hỗ trợ kiểm soát phiên bản và triển khai CI/CD thông qua GitHub Actions.

- **Postman** là công cụ được sử dụng trong việc kiểm thử các API RESTful của hệ thống. Nhờ Postman, nhóm có thể gửi các yêu cầu (request) và nhận phản hồi (response) từ server để đảm bảo các chức năng hoạt động đúng như đặc tả. Song song với đó, **Swagger** cũng được áp dụng để sinh tài liệu API tự động, giúp việc tích hợp và bảo trì trở nên dễ dàng, đồng thời cung cấp một giao diện thân thiện cho các nhà phát triển kiểm thử trực tiếp trên web.

- Trong giai đoạn triển khai, nhóm sử dụng **Docker** để đóng gói ứng dụng thành các container, đảm bảo tính đồng nhất giữa các môi trường phát triển và triển khai thực tế. Kết hợp với đó là **docker-compose**, cho phép nhóm chạy nhiều container (frontend, backend và cơ sở dữ liệu) đồng thời chỉ với một tệp cấu hình duy nhất, giúp đơn giản hóa quy trình triển khai.

- Bên cạnh đó, để quản lý công việc theo mô hình Agile/Scrum, nhóm đã sử dụng **Jira**. Đây là công cụ quản lý dự án mạnh mẽ, hỗ trợ lập kế hoạch sprint, theo dõi tiến độ qua biểu đồ Burndown chart và phân công công việc rõ ràng cho từng thành viên. Việc tích hợp Jira với GitHub giúp dễ dàng liên kết các commit với các nhiệm vụ cụ thể, đảm bảo tính minh bạch và kiểm soát tiến độ hiệu quả.

- Cuối cùng, để tự động hóa quy trình tích hợp và triển khai liên tục (CI/CD), nhóm thiết lập **GitHub Actions**. Với công cụ này, mỗi khi có thay đổi trong mã nguồn được đẩy lên nhánh chính (main), hệ thống sẽ tự động thực hiện các bước như kiểm thử mã nguồn, build ứng dụng và triển khai, giảm thiểu lỗi thủ công và rút ngắn thời gian phát hành sản phẩm.

4.2. Triển khai quy trình CI/CD với GitHub Actions

Trong quá trình phát triển phần mềm hiện đại, việc tích hợp liên tục (Continuous Integration - CI) và triển khai liên tục (Continuous Deployment - CD) là yếu tố quan trọng nhằm đảm bảo chất lượng sản phẩm và tối ưu thời gian phát hành. Dự án đã áp dụng **GitHub Actions** làm công cụ chính để xây dựng quy trình tự động hóa CI/CD, giúp đảm bảo mã nguồn được kiểm thử, đóng gói và triển khai một cách nhanh chóng, chính xác.

4.2.1. Mục tiêu triển khai CI/CD

Việc triển khai CI/CD nhằm đạt được các mục tiêu sau:

- **Tự động hóa quy trình phát triển phần mềm**, giảm thiểu thao tác thủ công.
- **Đảm bảo chất lượng mã nguồn** thông qua việc kiểm thử trước khi đưa lên môi trường triển khai.
- **Rút ngắn thời gian phát hành** và giảm thiểu lỗi phát sinh trong quá trình triển khai.
- **Tăng tính minh bạch và đồng bộ** trong nhóm phát triển khi làm việc theo mô hình Agile.

4.2.2. Quy trình CI/CD trong dự án

Quy trình triển khai CI/CD được nhóm thiết kế theo các giai đoạn chính sau:

- **Phát hiện thay đổi mã nguồn:** Khi có commit hoặc pull request trên nhánh chính, hệ thống sẽ tự động kích hoạt pipeline.
- **Thiết lập môi trường:** Tạo môi trường phát triển đồng nhất, cài đặt các thư viện và công cụ cần thiết cho cả frontend và backend.
- **Kiểm tra mã nguồn (Linting):** Đảm bảo mã nguồn tuân thủ các quy chuẩn coding, tránh lỗi cú pháp.
- **Chạy kiểm thử tự động:** Thực hiện các bài kiểm thử chức năng (unit test) để phát hiện lỗi trước khi build ứng dụng.
- **Đóng gói và build ứng dụng:** Sau khi vượt qua bước kiểm thử, hệ thống sẽ build ứng dụng thành các gói sẵn sàng triển khai.
- **Triển khai ứng dụng:** Ứng dụng được triển khai lên môi trường cục bộ hoặc nền tảng đám mây (Cloud) thông qua tích hợp với Docker hoặc các dịch vụ như AWS, Heroku.

4.2.3. Vai trò của GitHub Actions trong dự án

GitHub Actions là một công cụ CI/CD tích hợp trực tiếp trên nền tảng GitHub, cho phép tạo các pipeline tự động hóa dựa trên các sự kiện của repository (như push, pull request). Công cụ này đóng vai trò quan trọng trong việc:

- **Tự động hóa toàn bộ quy trình phát triển**, từ kiểm tra mã nguồn đến triển khai.
- **Đảm bảo tính nhất quán và ổn định**, vì mọi bước kiểm thử và triển khai đều được thực hiện theo kịch bản định sẵn.
- **Hỗ trợ cộng tác hiệu quả:** Khi có pull request, hệ thống sẽ tự động thông báo trạng thái kiểm thử, giúp phát hiện sớm lỗi trước khi merge.
- **Dễ dàng tích hợp với Docker và dịch vụ Cloud**, cho phép triển khai ứng dụng trong môi trường container hoặc đám mây.

4.2.4. Lợi ích đạt được

Nhờ việc triển khai quy trình CI/CD với GitHub Actions, dự án đạt được các lợi ích rõ rệt:

- **Tốc độ phát triển và triển khai nhanh hơn**, giảm thiểu sai sót do thao tác thủ công.
- **Chất lượng sản phẩm được nâng cao**, nhờ kiểm thử tự động ngay trong giai đoạn tích hợp.
- **Tối ưu hóa quy trình làm việc nhóm**, đảm bảo mỗi thay đổi trong mã nguồn đều được kiểm tra và triển khai một cách đồng bộ.

4.3. Docker hóa ứng dụng

4.3.1. Dockerfile Backend & Frontend

- **Dockerfile cho Backend**

Backend được triển khai trên nền tảng **Node.js**. Nội dung Dockerfile thực hiện:

- Sử dụng image **node:18-alpine** để tối ưu kích thước.
- Cài đặt các dependencies từ package.json.

```
FROM node:18-alpine
WORKDIR /app
COPY package*.json ./
RUN npm install --production
COPY . .
EXPOSE 5000
CMD ["npm", "start"]
```

Dockerfile cho Frontend

Frontend sử dụng **ReactJS**. Quy trình triển khai:

- Build mã nguồn React thành static files.
- Sử dụng **Nginx** để phục vụ nội dung tĩnh.

```
FROM node:18-alpine AS build
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
```

```
RUN npm run build
```

```
FROM nginx:alpine
```

```
COPY --from=build /app/build /usr/share/nginx/html
```

```
EXPOSE 80
```

```
CMD ["nginx", "-g", "daemon off;"]
```

4.3.2. docker-compose.yml

Để quản lý các container của **Frontend**, **Backend** và **MongoDB**, nhóm sử dụng **docker-compose**. Tập docker-compose.yml định nghĩa các dịch vụ, mạng và volume để các thành phần có thể giao tiếp với nhau dễ dàng.

```
version: "3.8"
services:
  backend:
    build: ./backend
    container_name: backend_service
    ports:
      - "5000:5000"
    environment:
      - MONGO_URI=mongodb://mongo:27017/doan_db
    depends_on:
      - mongo

  frontend:
    build: ./frontend
    container_name: frontend_service
    ports:
      - "3000:80"
    depends_on:
      - backend

  mongo:
    image: mongo:6.0
    container_name: mongo_db
    restart: always
    ports:
      - "27017:27017"
    volumes:
```

- mongo_data:/data/db

volumes:

mongo_data:

- **backend**: Chạy dịch vụ API với Node.js, kết nối cơ sở dữ liệu qua biến môi trường MONGO_URI.
- **frontend**: Xây dựng giao diện người dùng, phục vụ thông qua Nginx.
- **mongo**: Chạy dịch vụ MongoDB với volume mongo_data để lưu trữ dữ liệu lâu dài.
- **depends_on**: Đảm bảo thứ tự khởi động các dịch vụ.

CHƯƠNG 5: QUẢN LÝ DỰ ÁN

5.1. Quy trình phát triển phần mềm

5.1.1. Product Backlog

Product Backlog bao gồm toàn bộ yêu cầu chức năng và phi chức năng của hệ thống, được sắp xếp theo thứ tự ưu tiên.

STT	User Story	Mức độ ưu tiên	Ghi chú
1	Người dùng có thể đăng ký tài khoản	Cao	Chức năng cơ bản
2	Người dùng có thể đăng nhập vào hệ thống	Cao	
3	Người dùng xem danh sách sản phẩm	Cao	
4	Người dùng tìm kiếm và lọc sản phẩm	Trung bình	
5	Người dùng thêm sản phẩm vào giỏ hàng	Cao	
6	Người dùng đặt hàng và thanh toán	Cao	
7	Người dùng theo dõi trạng thái đơn hàng	Trung bình	
8	Admin quản lý sản phẩm (thêm, sửa, xóa)	Cao	
9	Admin quản lý đơn	Cao	

	hàng		
10	Admin xem thống kê doanh thu	Thấp	

Bảng 5.1 Product Backlog chi tiết

5.1.2. Sprint Backlog

Dự án được chia thành **4 Sprint**, mỗi Sprint kéo dài 2 tuần (14 ngày), tuân thủ nguyên tắc Agile Scrum. Các Sprint được xây dựng dựa trên ưu tiên trong Product Backlog, đảm bảo hoàn thiện dần các tính năng quan trọng trước khi phát triển các tính năng nâng cao.

Chi tiết các Sprint:

- **Sprint 1: Thiết lập nền tảng và chức năng xác thực người dùng**

Mục tiêu: Hoàn thiện hạ tầng dự án, môi trường phát triển, cơ sở dữ liệu và các chức năng cơ bản liên quan đến đăng ký/đăng nhập.

Nhiệm vụ:

- Cài đặt môi trường phát triển (Node.js, ReactJS, MySQL).
- Thiết kế cơ sở dữ liệu: ERD, mô hình quan hệ.
- Tạo cấu trúc dự án Backend (Express) và Frontend (ReactJS).
- Xây dựng API đăng ký, đăng nhập với mã hóa mật khẩu.
- Thiết kế giao diện đăng ký/đăng nhập trên Frontend.
- Kiểm thử chức năng đăng ký/đăng nhập bằng Postman.

- **Sprint 2: Hiện thị và tìm kiếm sản phẩm**

Mục tiêu: Hoàn thiện phần giao diện và API phục vụ người dùng tra cứu sản phẩm.

Nhiệm vụ:

- Thiết kế API lấy danh sách sản phẩm (GET).
- Thiết kế API tìm kiếm, lọc sản phẩm theo tên hoặc danh mục.
- Thiết kế giao diện trang chủ hiển thị danh sách sản phẩm.
- Tạo chức năng tìm kiếm và bộ lọc trên giao diện.
- Kiểm thử API bằng Postman, đảm bảo tốc độ phản hồi dưới 2 giây.

• Sprint 3: Quản lý giỏ hàng và đặt hàng

Mục tiêu: Cho phép người dùng thêm sản phẩm vào giỏ, quản lý số lượng, và đặt hàng.

Nhiệm vụ:

- Xây dựng API quản lý giỏ hàng (Thêm, xóa, cập nhật số lượng).
- Thiết kế API đặt hàng (POST), lưu thông tin đơn hàng vào DB.
- Thiết kế giao diện giỏ hàng và trang thanh toán.
- Tích hợp chức năng xác nhận đơn hàng và hiển thị trạng thái.
- Kiểm thử quy trình đặt hàng và lưu dữ liệu đơn hàng.

• Sprint 4: Chức năng Admin và thống kê

Mục tiêu: Hoàn thiện chức năng quản trị hệ thống và tối ưu toàn bộ sản phẩm.

Nhiệm vụ:

- Xây dựng API quản lý sản phẩm cho Admin (CRUD).
- Xây dựng API quản lý đơn hàng cho Admin.
- Tạo giao diện quản lý sản phẩm và đơn hàng cho Admin.
- Thiết kế màn hình báo cáo thống kê doanh thu.
- Triển khai ứng dụng với Docker và tạo tài liệu API bằng Swagger.
- Kiểm thử toàn bộ hệ thống (End-to-End Testing).

5.2. Sử dụng Jira để quản lý dự án

5.2.1 Sprint 1

SCRUM-2	Story 1.1: Thu thập yêu cầu khách hàng	EPIC 1: PHÂN TÍCH & ...	DONE	2	0
SCRUM-3	Task 1.1.1: Phỏng vấn chủ quản	EPIC 1: PHÂN TÍCH & ...	DONE	1	0
SCRUM-4	Task 1.1.2: Xác định yêu cầu chức năng	EPIC 1: PHÂN TÍCH & ...	DONE	2	0
SCRUM-5	Task 1.1.3: Xác định yêu cầu phi chức năng	EPIC 1: PHÂN TÍCH & ...	DONE	2	0
SCRUM-6	Story 1.2: Thiết kế giao diện	EPIC 1: PHÂN TÍCH & ...	DONE	3	0
SCRUM-7	Task 1.2.1: Thiết kế wireframe (Figma)	EPIC 1: PHÂN TÍCH & ...	DONE	1	0
SCRUM-8	Task 1.2.2: Thiết kế UI/UX chi tiết	EPIC 1: PHÂN TÍCH & ...	DONE	1	0
SCRUM-9	Task 1.2.3: Review và chỉnh sửa	EPIC 1: PHÂN TÍCH & ...	DONE	1	0

Hình 5.1 Sprint Backlog của Sprint 1

Thời gian thực hiện: 10/06/2025 – 22/06/2025

Mục tiêu Sprint: Phân tích và thiết kế hệ thống

Kế hoạch công việc:

- **Story 1.1: Thu thập yêu cầu khách hàng**

Mục tiêu: Xác định yêu cầu chức năng và phi chức năng thông qua trao đổi với khách hàng.

- **Task 1.1.1: Phỏng vấn chủ quán**

Thu thập thông tin về nhu cầu, quy trình và mong đợi của khách hàng.

- **Task 1.1.2: Xác định yêu cầu chức năng**

Liệt kê các chức năng chính như quản lý thực đơn, đơn hàng, người dùng.

- **Task 1.1.3: Xác định yêu cầu phi chức năng**

Xác định tiêu chí về hiệu năng, bảo mật, khả năng mở rộng và dễ sử dụng.

- **Story 1.2: Thiết kế giao diện**

Mục tiêu: Xây dựng giao diện trực quan, thân thiện với người dùng.

- **Task 1.2.1: Thiết kế Wireframe (Figma)**

Tạo bố cục và luồng tương tác cơ bản.

- **Task 1.2.2: Thiết kế UI/UX chi tiết**

Hoàn thiện màu sắc, bố cục và trải nghiệm người dùng.

- **Task 1.2.3: Review và chỉnh sửa**

Đánh giá, nhận phản hồi và tối ưu thiết kế.

5.2.2 Sprint 2

✓ SERUM-11	Story 2.1: Khởi tạo dự án	EPIC 2: CÀI ĐẶT FRONTEND	DONE	2	=	N
✓ SERUM-12	Khởi tạo dự án		DONE	-	=	0
✓ SERUM-13	Task 2.1.1: Cấu trúc HTML/CSS/JS hoặc React		DONE	2	=	GM
✓ SERUM-14	Task 2.1.2: Setup GitHub/GitLab		DONE	2	=	N
✓ SERUM-15	Story 2.2: Trang chủ		DONE	-	=	GM
✓ SERUM-16	Task 2.2.1: Code header, footer		DONE	1	=	0
✓ SERUM-17	Task 2.2.2: Banner và thông tin quán		DONE	1	=	N

Hình 5.2 Sprint Backlog của Sprint 2

Mục tiêu: Thiết lập cấu trúc dự án và phát triển giao diện cơ bản.

Story 2.1: Khởi tạo dự án

Mục tiêu: Tạo nền tảng ban đầu cho việc phát triển hệ thống.

- **Task 2.1.1: Cấu trúc HTML/CSS/JS hoặc React**

Thiết lập khung mã nguồn và bố cục cơ bản.

- **Task 2.1.2: Setup GitHub/GitLab**

Tạo kho lưu trữ mã nguồn và thiết lập quy trình làm việc nhóm.

Story 2.2: Trang chủ

Mục tiêu: Xây dựng giao diện trang chủ cơ bản.

- **Task 2.2.1: Code header, footer**
Thiết kế phần đầu và cuối trang theo bố cục chuẩn.
- **Task 2.2.2: Banner và thông tin quán**
Hiển thị hình ảnh và giới thiệu về quán trên trang chủ.

5.2.3 Sprint 3

Hình 5.3 Sprint Backlog của Sprint 3

Mục tiêu: Hoàn thiện trang menu với danh sách đồ uống và bộ lọc.

Story 2.3: Trang Menu

Mục tiêu: Cung cấp tính năng xem và lọc danh sách đồ uống.

- **Task 2.3.1: Hiển thị danh sách đồ uống (HTML/CSS/JS)**
Hiển thị hình ảnh, tên và giá sản phẩm.
- **Task 2.3.2: Thêm bộ lọc theo loại đồ uống**
Cho phép người dùng lọc sản phẩm theo danh mục.

5.2.4 Sprint 4

Hình 5.4 Sprint Backlog của Sprint 4

Mục tiêu: Tích hợp chức năng đặt hàng và quản lý giỏ hàng.

Story 3.1: Xây dựng form đặt hàng

Mục tiêu: Cho phép khách hàng nhập thông tin và tính toán tổng tiền.

- **Task 3.1.1: Form nhập thông tin khách hàng**
Tạo form với các trường cơ bản như tên, số điện thoại, địa chỉ.

- **Task 3.1.2: Tính tổng tiền và hiển thị**

Tính toán chi phí dựa trên giỏ hàng và hiển thị cho người dùng.

Story 3.2: Xử lý giỏ hàng

Mục tiêu: Quản lý sản phẩm trong giỏ hàng.

- **Task 3.2.1: Thêm sản phẩm vào giỏ**

Cập nhật giỏ hàng khi người dùng chọn sản phẩm.

- **Task 3.2.2: Cập nhật số lượng**

Cho phép chỉnh sửa số lượng sản phẩm trong giỏ.

5.2.5 Sprint 5



☐	▼	Kiểm thử, tối ưu & Triển khai	10 Jul – 22 Jul (4 work items)	0	0	7	Complete sprint	...
🔍	SCRUM-30	Story 4.1: Kiểm thử giao diện trên PC/Mobile	DONE▼	1	=	Q		
🔍	SCRUM-31	Story 4.2: Kiểm thử chức năng đặt hàng	DONE▼	2	=	N		
🔍	SCRUM-32	Story 4.3: Tối ưu tốc độ & SEO cơ bản	DONE▼	2	=	QM		
🔍	SCRUM-33	Story 4.4: Deploy website lên hosting hoặc GitHub Pages	DONE▼	2	=	N		

Hình 5.5 Sprint Backlog của Sprint 5

Mục tiêu: Đảm bảo hệ thống ổn định và triển khai sản phẩm hoàn thiện.

- **Story 4.1: Kiểm thử giao diện trên PC/Mobile**

Kiểm tra tính tương thích và khả năng hiển thị.

- **Story 4.2: Kiểm thử chức năng đặt hàng**

Đảm bảo quy trình đặt hàng hoạt động đúng logic.

- **Story 4.3: Tối ưu tốc độ & SEO cơ bản**

Cải thiện hiệu năng tải trang và chuẩn SEO cơ bản.

- **Story 4.4: Deploy website lên hosting hoặc GitHub Pages**

Đưa sản phẩm hoàn thiện lên môi trường triển khai.

5.3. Phân công công việc cho các thành viên

Tên thành viên	Công việc thực hiện
Nguyễn Nhật Hóa	<ul style="list-style-type: none">- Thiết kế cấu trúc dự án.- Xây dựng chức năng đặt hàng và giỏ hàng.- Hỗ trợ tích hợp API với Backend.- Tham gia kiểm thử hệ thống.- Triển khai Docker- Xây dựng và triển khai Backend (Node.js, API).- Viết báo cáo
Bùi Quốc Anh	<ul style="list-style-type: none">- Thu thập và phân tích yêu cầu của khách hàng.- Thiết kế giao diện người dùng (UI/UX) bằng Figma.- Thiết kế wireframe và nguyên mẫu giao diện.- Đảm bảo tính thẩm mỹ và trải nghiệm người dùng.- Kiểm thử giao diện trên các thiết bị- Viết báo cáo- Làm slide
Mai Tuấn Đạt	<ul style="list-style-type: none">- Triển khai ứng dụng lên môi trường thực tế (Hosting/GitHub Pages).- Thiết kế kiến trúc tổng thể hệ thống.- Thiết kế và phát triển trang Menu.- Tích hợp API giữa Frontend và Backend.- Phát triển trang chủ của website.- Viết báo cáo

Bảng 5.1 Phân công công việc các thành viên

6.1. Chiến lược kiểm thử

6.1.1 Mục tiêu kiểm thử

Mục tiêu của quá trình kiểm thử là đảm bảo hệ thống website vận hành đúng theo yêu cầu nghiệp vụ, cung cấp đầy đủ các chức năng đặt hàng, quản lý giỏ hàng, và đảm bảo tính ổn định, bảo mật cơ bản trước khi triển khai.

6.1.2 Phạm vi kiểm thử

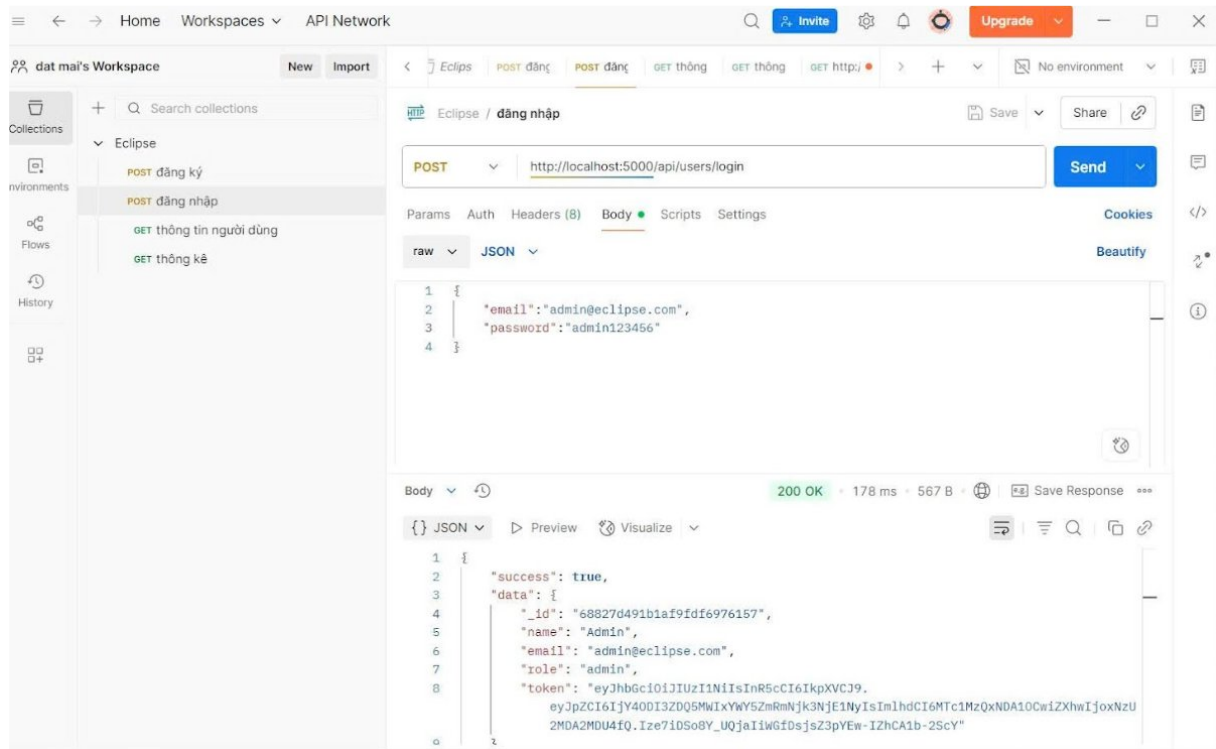
- **Frontend:** Kiểm tra giao diện người dùng, khả năng tương thích trên các thiết bị khác nhau (desktop, mobile).
- **Backend:** Kiểm tra hoạt động của các API (CRUD sản phẩm, đơn hàng, người dùng) bằng Postman và Swagger.
- **Tích hợp:** Kiểm tra luồng đặt hàng từ giao diện người dùng đến cơ sở dữ liệu (MongoDB).
- **Hiệu năng và bảo mật cơ bản:** Đánh giá tốc độ phản hồi API và kiểm tra mã hóa mật khẩu.

6.1.3 Phương pháp và kỹ thuật kiểm thử

- **Phương pháp kiểm thử:** Áp dụng kiểm thử hộp đen (Black-box Testing), tập trung vào đánh giá đầu vào và đầu ra mà không quan tâm đến mã nguồn.
- **Kỹ thuật kiểm thử:**
 - + Kiểm thử thủ công (Manual Testing) cho giao diện và luồng đặt hàng.
 - + Kiểm thử tự động một phần (Postman Collection) cho API.
- **Các loại kiểm thử thực hiện:**
 - + Kiểm thử chức năng (Functional Testing): Xác minh các chức năng hoạt động đúng yêu cầu.
 - + Kiểm thử giao diện (UI Testing): Kiểm tra tính thẩm mỹ, bố cục và khả năng tương thích.
 - + Kiểm thử tích hợp (Integration Testing): Đảm bảo các thành phần hệ thống hoạt động nhịp nhàng.
 - + Kiểm thử chấp nhận (Acceptance Testing): Đáp ứng yêu cầu người dùng cuối.

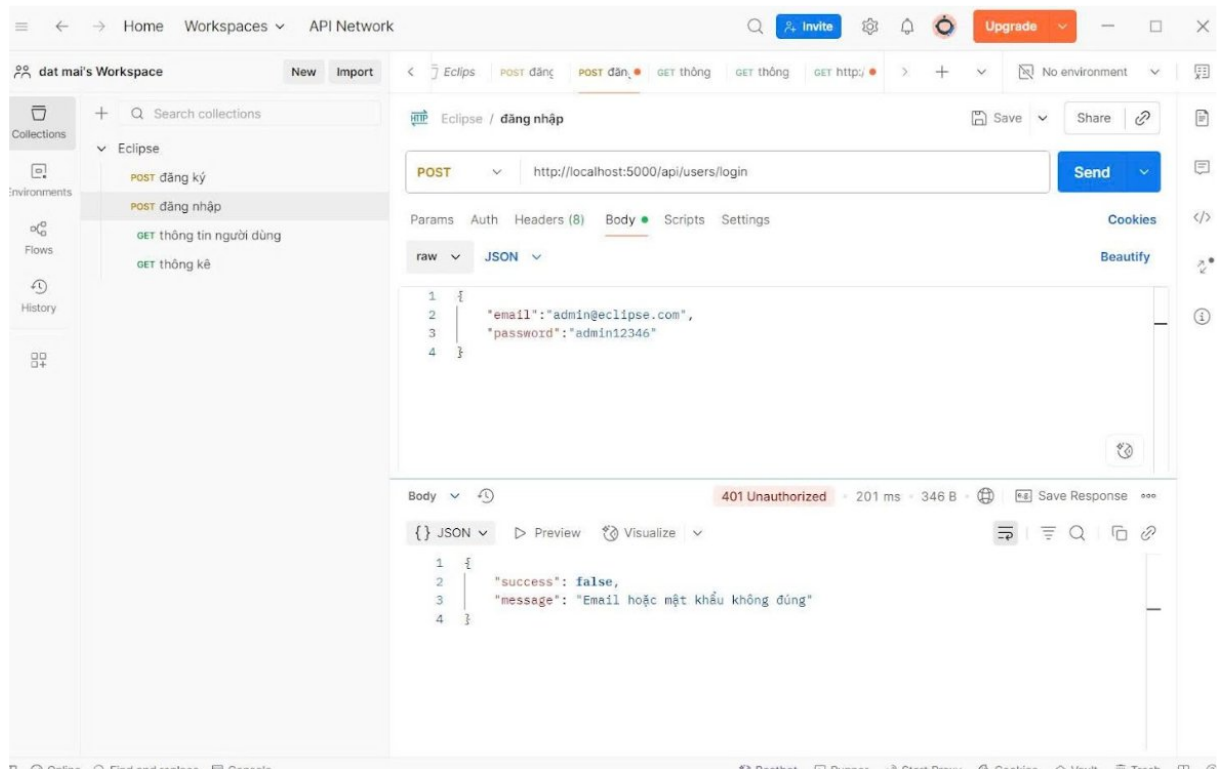
6.2. Kết quả kiểm thử

- Đăng nhập người dùng



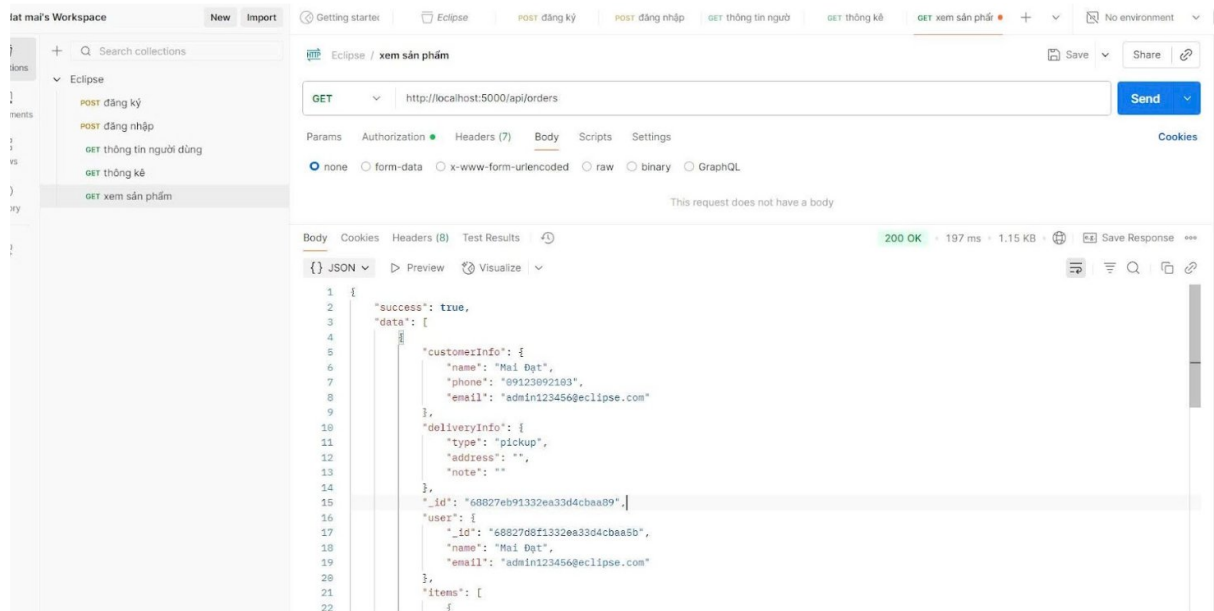
Hình 6.1 Kiểm thử đăng nhập đúng

- Đăng nhập sai mật khẩu



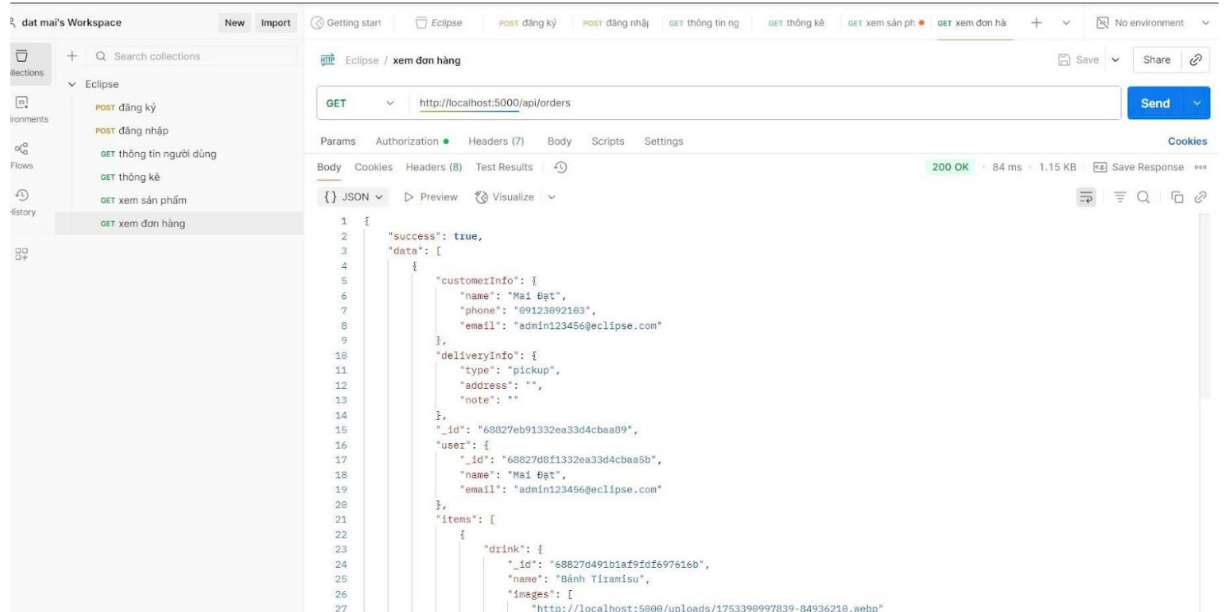
Hình 6.2 Kiểm thử khi đăng nhập sai

- Hiện thị danh sách sản phẩm



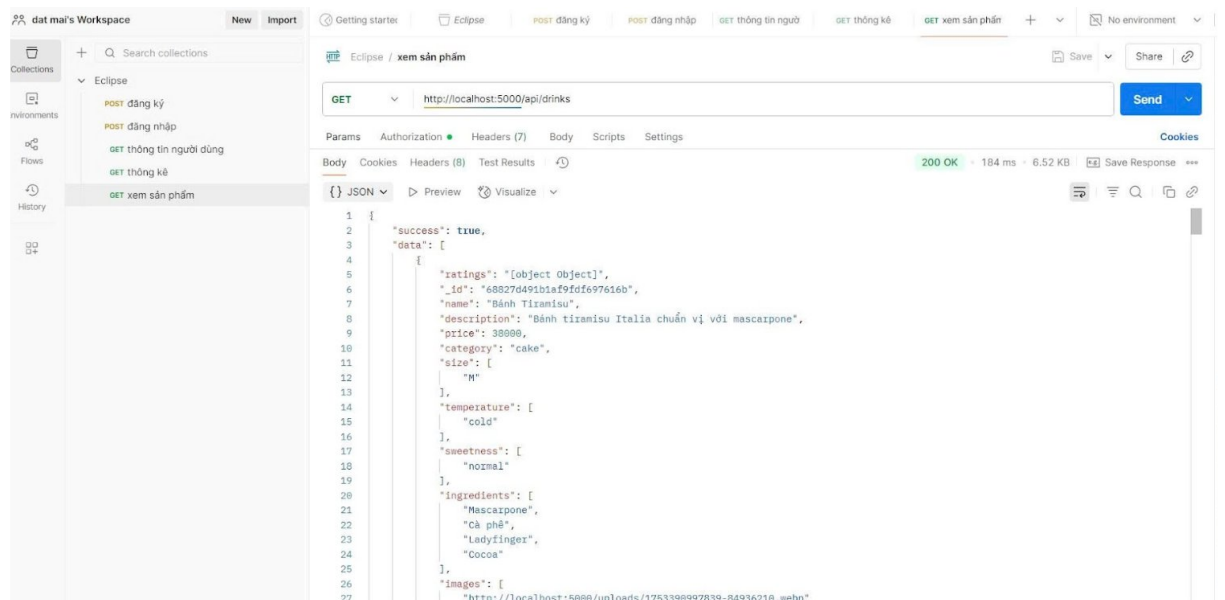
Hình 6.3 Kiểm thử hiện thị danh sách sản phẩm

- Thông tin người dùng



Hình 6.4 Kiểm thử thông tin người dùng

- Xem đơn hàng



Hình 6.5 Kiểm thử xem đơn hàng

CHƯƠNG 7: ĐÁNH GIÁ VÀ KẾT LUẬN

7.1. Đánh giá kết quả đạt được

Qua quá trình nghiên cứu, phân tích và triển khai, nhóm đã hoàn thành các mục tiêu đề ra cho đề tài “Xây dựng website bán nước và đồ ăn nhẹ” với các kết quả chính sau:

- **Xây dựng được một hệ thống website hoàn chỉnh**, đáp ứng yêu cầu nghiệp vụ bao gồm: hiển thị danh sách sản phẩm, quản lý giỏ hàng, đặt hàng trực tuyến, theo dõi đơn hàng và quản trị nội dung (dành cho admin).
- **Áp dụng kiến trúc Client-Server và RESTful API** để tách biệt rõ ràng giữa frontend (ReactJS) và backend (NodeJS), giúp hệ thống có khả năng mở rộng và bảo trì tốt.
- **Triển khai thiết kế giao diện UI/UX trên Figma** với phong cách hiện đại, thân thiện, tối ưu trải nghiệm người dùng trên cả desktop và thiết bị di động.
- **Sử dụng các công cụ quản lý và kiểm thử** như Postman (kiểm thử API), Swagger (tạo tài liệu API), GitHub và GitHub Actions (quản lý mã nguồn, CI/CD), Docker (đóng gói và triển khai), đảm bảo quy trình phát triển phần mềm chuyên nghiệp.
- **Đáp ứng yêu cầu về bảo mật cơ bản**: mã hóa mật khẩu, sử dụng HTTPS, hạn chế các lỗi bảo mật phổ biến.
- **Xây dựng báo cáo và tài liệu đầy đủ**, bao gồm phân tích yêu cầu, thiết kế hệ thống, mô tả API, sơ đồ kiến trúc, và kế hoạch phát triển theo mô hình Agile.

7.2. Khó khăn gặp phải

Trong quá trình thực hiện, nhóm gặp phải một số khó khăn như:

- **Khó khăn trong việc tích hợp nhiều công cụ và công nghệ**: Đặc biệt là thiết lập Docker và GitHub Actions để đảm bảo triển khai CI/CD, do chưa có nhiều kinh nghiệm thực tế.
- **Thời gian hạn chế**: Với khối lượng công việc lớn, việc phân chia và theo dõi tiến độ yêu cầu sự phối hợp chặt chẽ trong nhóm.
- **Vấn đề tương thích giao diện**: Ban đầu giao diện chưa tương thích hoàn toàn trên thiết bị di động, nhóm phải thực hiện nhiều lần điều chỉnh.

-
- **Xử lý bảo mật:** Việc triển khai các cơ chế chống SQL Injection, XSS cần thời gian tìm hiểu và kiểm thử thêm.

7.3. Bài học rút ra

Từ quá trình triển khai dự án, nhóm đã rút ra một số bài học kinh nghiệm quan trọng:

- **Tầm quan trọng của việc phân tích yêu cầu chi tiết:** Một bản phân tích tốt sẽ giảm thiểu sai sót trong quá trình phát triển.
- **Làm việc nhóm hiệu quả là yếu tố then chốt:** Sử dụng Jira để quản lý sprint, phân công công việc rõ ràng giúp nhóm phối hợp nhịp nhàng hơn.
- **Quy trình phát triển phần mềm hiện đại (Agile, CI/CD)** mang lại nhiều lợi ích, giúp sản phẩm dễ bảo trì và triển khai nhanh chóng.
- **Cần ưu tiên vấn đề bảo mật ngay từ đầu**, vì sửa lỗi bảo mật sau khi triển khai sẽ tốn kém hơn nhiều.

7.4. Hướng phát triển trong tương lai

Hệ thống hiện tại đã đáp ứng các chức năng cơ bản, tuy nhiên để nâng cao chất lượng và trải nghiệm người dùng, nhóm dự định phát triển thêm các tính năng sau:

- **Tích hợp cổng thanh toán trực tuyến (VNPay, Momo)** để tăng sự tiện lợi cho khách hàng.
- **Xây dựng ứng dụng di động (Mobile App)** sử dụng React Native để mở rộng phạm vi người dùng.
- **Tích hợp hệ thống đánh giá sản phẩm và phản hồi của khách hàng**, giúp cải thiện chất lượng dịch vụ.
- **Ứng dụng trí tuệ nhân tạo (AI)** trong việc gợi ý sản phẩm dựa trên lịch sử mua hàng.
- **Tăng cường bảo mật** bằng cách triển khai xác thực hai lớp (Two-Factor Authentication) và giám sát hệ thống chống tấn công mạng.
- **Triển khai trên nền tảng điện toán đám mây** như AWS hoặc Google Cloud để đảm bảo khả năng mở rộng và độ tin cậy cao.