

**TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ
KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO KẾT THÚC MÔN HỌC
MÔN: CÔNG NGHỆ PHẦN MỀM**

**THIẾT KẾ VÀ TRIỂN KHAI WEBSITE
BÁN GÀ RÁN TÍCH HỢP DOCKER
VÀ CI/CD THEO MÔ HÌNH RESTFUL**

Giáo viên hướng dẫn:

TS. Nguyễn Bảo Ân

Nhóm sinh viên thực hiện:

Nguyễn Phúc An (110122214 – DA22TTA)

Nguyễn Thiên Ân (110122030 – DA22TTA)

Hứa Khánh Đăng (110122046 – DA22TTA)

Trà Vinh, tháng 7 năm 2025

LỜI CẢM ƠN

Chúng em xin chân thành cảm ơn các thầy cô trường Đại học Trà Vinh, đặc biệt là các thầy cô khoa Kỹ thuật & Công nghệ đã tạo cơ hội cho chúng em được thực tập và giao lưu. Chúng em có thể tránh được các vấn đề và nhầm lẫn trong môi trường làm việc trong tương lai.

Chúng em xin cảm ơn thầy Nguyễn Bảo Ân. Nhờ sự giúp đỡ và hướng dẫn tận tình của thầy từ đầu đến cuối báo cáo, chúng em đã hoàn thành bài báo cáo đúng thời hạn và tích lũy được một vốn kiến thức quý báu.

Mặc dù đã cố gắng hết sức để hoàn thành đề tài này nhưng do hạn chế về thời gian cũng như kiến thức chuyên môn nên trong quá trình nghiên cứu, đánh giá và trình bày đề tài, chúng em còn nhiều thiếu sót. Chúng em rất mong được sự quan tâm, góp ý của các thầy cô bộ môn để bài báo cáo kết thúc môn của chúng em được hoàn thiện và hoàn thiện hơn.

Chúng em xin chân thành cảm ơn.

Trà Vinh, ngày ... tháng 7 năm 2025

Nhóm thực hiện:

Nguyễn Phúc An - 110122214

Nguyễn Thiên Ân - 110122030

Hứa Khánh Đăng - 110122046

Sinh viên ký và ghi rõ họ và tên

Sinh viên 1

Sinh viên 2

Sinh viên 3

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Trà Vinh, ngày tháng năm

Giảng viên hướng dẫn

(Ký tên và ghi rõ họ tên)

NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Trà Vinh, ngày tháng năm

Giảng viên phản biện

(Ký tên và ghi rõ họ tên)

MỤC LỤC

DANH MỤC HÌNH ẢNH	8
DANH MỤC BẢNG BIỂU	10
DANH MỤC KÍ HIỆU CÁC CỤM TỪ VIẾT TẮT	11
BẢNG PHÂN CÔNG CÔNG VIỆC	12
CHƯƠNG 1: GIỚI THIỆU	13
1.1 Lý do chọn đề tài	13
1.2 Tên đề tài và chủ đề	13
1.3 Mục tiêu ứng dụng	13
1.4 Tổng quan về công nghệ phần mềm	14
CHƯƠNG 2: PHÂN TÍCH YÊU CẦU	15
2.1 Các chức năng chính (Functional Requirements)	15
2.1.1 Chức năng dành cho khách hàng	15
2.1.2 Chức năng dành cho quản trị viên	16
2.2 Các yêu cầu phi chức năng (Non-functional Requirements)	17
2.2.1 Hiệu suất	17
2.2.2 Tính bảo mật và quyền riêng tư	17
2.2.3 Tính sẵn sàng	17
2.2.4 Khả năng mở rộng	18
2.2.5 Tính dễ sử dụng	18
2.2.6 Tính duy trì và phát triển	18
2.3 Mô hình phát triển phần mềm	18
CHƯƠNG 3: THIẾT KẾ HỆ THỐNG	20
3.1 Kiến trúc tổng thể	20
3.1.1 Frontend (Client Layer)	20
3.1.2 Server (Node.js + ExpressJS)	20
3.1.3 Cơ sở dữ liệu	21
3.2 Thiết kế cơ sở dữ liệu	23
3.2.1 Mô hình thực thể - quan hệ (ERD)	23
3.2.2 Sơ đồ quan hệ dữ liệu MySQL	25
3.3 Thiết kế API	26
3.3.1 Tổng quan	26

3.3.2 Nhóm endpoint chính	26
3.3.3 Cấu trúc request/response	27
3.4 Thiết kế giao diện (UI/UX)	32
3.4.1 Trang chủ website	32
3.4.2 Giao diện đăng nhập, đăng ký	33
3.4.3 Trang Giới thiệu	34
3.4.4 Giao diện trang Liên hệ	35
3.4.5 Giao diện Overlay khi thêm món ăn vào giỏ hàng	36
3.4.6 Giao diện trang Giỏ hàng	36
3.4.7 Giao diện trang thông tin cá nhân	37
3.4.8 Giao diện đổi mật khẩu	39
3.4.9 Giao diện các trang quản trị	39
CHƯƠNG 4: TRIỂN KHAI VÀ CÔNG NGHỆ SỬ DỤNG	46
4.1 Danh sách công nghệ sử dụng	46
4.2 Quy trình CI/CD với GitHub Actions	47
4.3 Cấu hình Docker và quy trình triển khai ứng dụng	48
4.3.1 Cấu hình Docker	48
4.3.2 Quy trình triển khai	48
CHƯƠNG 5: QUẢN LÝ DỰ ÁN	50
5.1 Sử dụng Jira để lập kế hoạch và theo dõi tiến độ	50
5.1.1 Thiết lập Jira Board	50
5.1.2 Quy trình theo dõi tiến độ	50
5.1.3 Metrics và Báo cáo	51
5.2 Phân công nhiệm vụ của các thành viên trong nhóm	51
CHƯƠNG 6: KIỂM THỬ	53
6.1 Chiến lược kiểm thử và công cụ sử dụng	53
6.1.1 Chiến lược kiểm thử tổng thể	53
6.1.2 Công cụ kiểm thử	53
6.2 Kết quả kiểm thử	55
6.2.1 Đăng ký người dùng mới	55
6.2.2 Lấy danh sách Nhóm sản phẩm	57
6.2.3 Cập nhập thông tin các nhân (khách hàng)	58

CHƯƠNG 7: ĐÁNH GIÁ VÀ KẾT LUẬN	61
7.1 Những khó khăn gặp phải trong quá trình thực hiện	61
7.1.1 Khó khăn về công nghệ	61
7.1.2 Khó khăn về quy trình làm việc nhóm	61
7.2 Bài học rút ra	62
7.3 Đề xuất trong tương lai	62
CHƯƠNG 8: PHỤ LỤC	63
8.1 Hướng dẫn cài đặt và chạy ứng dụng	63
8.1.1 Yêu cầu hệ thống	63
8.1.2 Cài đặt thủ công (Manual Setup)	63
8.1.3 Cài đặt bằng Docker (Khuyến nghị)	64
8.1.4 Tài khoản mặc định	65
8.1.5 Kiểm tra cài đặt	65
8.2 Liên kết GitHub repository và link demo	66
8.2.1 GitHub Repository	66
8.2.2 Demo Links	67
TÀI LIỆU THAM KHẢO	68

DANH MỤC HÌNH ẢNH

Hình 2.1. Sơ đồ use-case chức năng	15
Hình 3.1 Sơ đồ ERD được thiết kế trong PowerDesign	23
Hình 3.2 Sơ đồ cơ sở dữ liệu quan hệ MySQL	25
Hình 3.3 Request gửi dữ liệu đăng ký	28
Hình 3.4 Response khi đăng ký thành công	28
Hình 3.5 Response khi lấy danh sách món ăn thành công	30
Hình 3.6 Request cập nhật thông tin một người dùng	31
Hình 3.7 Response sau khi cập nhật thành công	32
Hình 3.8 Giao diện trang chủ website	33
Hình 3.9 Giao diện trang đăng ký tài khoản	34
Hình 3.10 Giao diện trang đăng nhập	34
Hình 3.11 Giao diện trang Giới thiệu	35
Hình 3.12 Giao diện trang Liên hệ	35
Hình 3.13 Overlay thêm món ăn vào giỏ hàng	36
Hình 3.14 Giao diện trang Giỏ hàng	37
Hình 3.15 Overlay hiện trạng thái đơn hàng	37
Hình 3.16 Giao diện trang đổi mật khẩu	38
Hình 3.17 Giao diện đổi thông tin cá nhân	38
Hình 3.18 Giao diện Đổi mật khẩu	39
Hình 3.19 Giao diện menu thống kê	40
Hình 3.20 Giao diện menu Đơn đặt hàng	40
Hình 3.21 Overlay hiện Ghi chú đơn hàng	41
Hình 3.22 Thông báo xác nhận xóa Đơn hàng	41
Hình 3.23. Giao diện menu Sản phẩm	42

Hình 3.24 Overlay thêm Sản phẩm	42
Hình 3.25. Overlay sửa Sản phẩm	43
Hình 3.26 Giao diện menu Nhóm sản phẩm	43
Hình 3.27 Giao diện menu Người dùng	44
Hình 3.28 Giao diện menu Liên hệ	45
Hình 6.1 Request gửi dữ liệu đăng ký	56
Hình 6.2 Response khi đăng ký thành công	57
Hình 6.3 Response khi lấy danh sách phim đang chiếu thành công	58
Hình 6.4 Request cập nhật thông tin một người dùng	59
Hình 6.5 Response sau khi cập nhật thành công	60

DANH MỤC BẢNG BIỂU

Bảng 3.1 Mô tả các quan hệ giữa các thực thể trong ERD	24
Bảng 3.2 Cấu hình truy cập API	26
Bảng 3.3 Danh sách nhóm endpoint chính	27

DANH MỤC KÍ HIỆU CÁC CỤM TỪ VIẾT TẮT

Số thứ tự	Kí hiệu	Nội dung viết tắt
1	UI/UX	User Interface/User Experience
2	CI/CD	Continuous Integration/Continuous Deployment
3	JWT	JSON Web Token
4	API	Application Programming Interface
5	MERN	Bộ công nghệ phát triển web
6	REST	Representational State Transfer
7	CRUD	Create, Read, Update, Delete
8	JSON	JavaScript Object Notation
9	HTTP	HyperText Transfer Protocol
10	CORS	Cross-Origin Resource Sharing
11	URL	Uniform Resource Locator

BẢNG PHÂN CÔNG CÔNG VIỆC

Công việc	Người thực hiện
<ul style="list-style-type: none"> - Wireframe: Trang chủ, menu, giỏ hàng, đặt hàng - Lấy feedback + chỉnh thiết kế - Cấu trúc frontend (React/Vue) - Thiết kế schema (user, sp, đơn hàng, liên hệ) - API: user, sp, nhóm sp, order, liên hệ, auth - Header, Footer, slogan - Admin: tổng quan, order, user + API - Giao diện login + API auth - Hồ sơ cá nhân + PrivateRoute - Docker Compose toàn hệ thống - CI/CD với GitHub Actions 	Nguyễn Phúc An
<ul style="list-style-type: none"> - Tạo Jira + Scrum board - Nghiên cứu UI web đồ ăn nhanh - Giao diện trang chủ tích hợp menu và tìm kiếm - Admin: sản phẩm, nhóm, liên hệ + API - Test API bằng Postman - Viết báo cáo 	Nguyễn Thiên Ân
<ul style="list-style-type: none"> - Tạo workspace Figma - Tạo GitHub repo + nhánh - Thiết kế UI hi-fi - Tích hợp Swagger - Thêm vào giỏ hàng - Trang đặt hàng + gọi API - Trang Giới thiệu, Liên hệ - Dockerfile backend + frontend - Slide thuyết trình 	Hứa Khánh Đăng

CHƯƠNG 1: GIỚI THIỆU

1.1 Lý do chọn đề tài

- Tính thực tiễn cao: Dịch vụ đặt món trực tuyến đang trở nên phổ biến, đặc biệt trong thời đại công nghệ và sau giai đoạn giãn cách xã hội do dịch bệnh.
- Cơ hội áp dụng kiến thức môn học: Dự án là môi trường thực tế để áp dụng đầy đủ các kiến thức đã học trong môn Công nghệ phần mềm: từ thiết kế hệ thống, quản lý dự án với Jira, kiểm thử API bằng Postman, thiết kế UI/UX với Figma, đến CI/CD và container hoá bằng Docker.
- Nâng cao kỹ năng làm việc nhóm: Dự án yêu cầu chia nhiệm vụ rõ ràng, phối hợp chặt chẽ giữa các thành viên, tăng cường kỹ năng giao tiếp và quản lý tiến độ.
- Hướng đến khả năng triển khai thực tế: Nếu phát triển đầy đủ, hệ thống có thể triển khai cho cửa hàng thật, hoặc mở rộng thành app di động trong tương lai.

1.2 Tên đề tài và chủ đề

- Tên dự án: Thiết kế và triển khai website bán gà rán tích hợp Docker và CI/CD theo mô hình Restful. (A.D.A FastFood - Website thương mại điện tử).
- Chủ đề: Xây dựng một ứng dụng web thương mại điện tử cho phép người dùng đặt món ăn nhanh trực tuyến, thanh toán điện tử, theo dõi đơn hàng và quản lý quy trình bán hàng của cửa hàng thức ăn nhanh.

1.3 Mục tiêu ứng dụng

Mục tiêu chính của dự án là xây dựng một nền tảng bán hàng trực tuyến chuyên biệt cho lĩnh vực thức ăn nhanh. Ứng dụng sẽ giúp:

- Người dùng dễ dàng tìm kiếm và đặt món ăn mọi lúc, mọi nơi.
- Quản lý giỏ hàng và thanh toán trực tuyến nhanh chóng, an toàn.
- Theo dõi trạng thái đơn hàng theo thời gian thực.
- Tối ưu hóa quy trình bán hàng và giao nhận cho cửa hàng.

Ngoài ra, hệ thống cũng hỗ trợ nhân viên và quản trị viên theo dõi doanh thu, quản lý thực đơn, đơn hàng và phản hồi khách hàng một cách hiệu quả.

1.4 Tổng quan về công nghệ phần mềm

Công nghệ phần mềm (Software Engineering) là lĩnh vực nghiên cứu và ứng dụng các phương pháp và công cụ để phát triển, vận hành và bảo trì phần mềm hiệu quả, ổn định và dễ mở rộng. Một nội dung cốt lõi là vòng đời phát triển phần mềm (SDLC), gồm các giai đoạn: khảo sát, phân tích, thiết kế, lập trình, kiểm thử, triển khai và bảo trì. Quy trình này giúp đảm bảo phần mềm đáp ứng đúng yêu cầu, có tính hệ thống và dễ nâng cấp trong tương lai.

Trong dự án này, công nghệ phần mềm được áp dụng xuyên suốt từ phân tích yêu cầu người dùng đến triển khai và vận hành hệ thống. Dự án đáp ứng các nhu cầu cụ thể:

- Khách hàng: Cần một nền tảng đặt đồ ăn tiện lợi, có thể xem menu sản phẩm, chọn món và thanh toán nhanh chóng.
- Chủ cửa hàng (Admin): Cần hệ thống hỗ trợ thống kê doanh thu, quản lý sản phẩm, nhóm sản phẩm, quản lý đơn hàng và có khả năng mở rộng khi số lượng sản phẩm hoặc người dùng tăng lên.

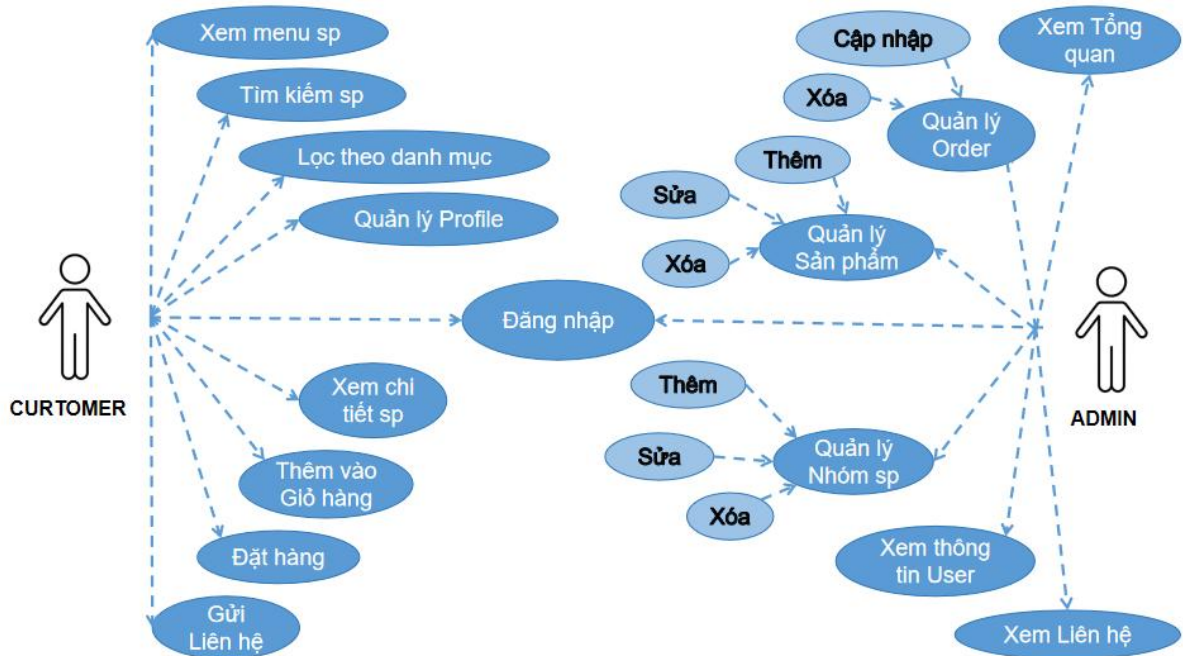
Để đáp ứng các yêu cầu trên, nhóm đã lựa chọn kiến trúc client-server, tách biệt rõ ràng giữa giao diện người dùng (frontend) và phần xử lý nghiệp vụ (backend). Các thành phần này được container hóa bằng Docker, dễ dàng triển khai và quản lý. Ngoài ra, backend được thiết kế theo hướng module service-based đơn giản, giúp từng chức năng như quản lý người dùng, sản phẩm, đơn hàng có thể phát triển và bảo trì độc lập. Việc triển khai bằng Docker Compose trên môi trường phát triển cũng đảm bảo hiệu suất cao, dễ mở rộng và tiết kiệm chi phí vận hành.

Tổng thể, việc ứng dụng các nguyên lý của công nghệ phần mềm vào dự án giúp nhóm xây dựng được một hệ thống quản lý nhanh chóng và hoàn chỉnh, có tính thực tiễn cao và phù hợp với nhu cầu phát triển trong bối cảnh số hóa ngành dịch vụ ăn uống tại Việt Nam.

CHƯƠNG 2: PHÂN TÍCH YÊU CẦU

2.1 Các chức năng chính (Functional Requirements)

Hệ thống bao gồm hai nhóm chức năng chính: dành cho người dùng (khách hàng) và dành cho quản trị viên (admin).



Hình 2.1 Sơ đồ use-case chức năng

2.1.1 Chức năng dành cho khách hàng

- Đăng ký tài khoản và đăng nhập

+ Đăng ký tài khoản khách hàng bằng tên người dùng và mật khẩu.

+ Đăng nhập/đăng xuất.

+ Cập nhật thông tin cá nhân, đổi mật khẩu.

+ Xem lịch sử đặt hàng.

- Xem và tìm kiếm món ăn:

+ Xem danh sách món ăn theo danh mục: Gà rán, Burger, Pizza, Đồ uống, Combo...

+ Tìm kiếm món ăn theo tên.

+ Xem chi tiết món ăn: mô tả, hình ảnh, giá.

- Chọn món ăn và thêm vào giỏ hàng:

- + Người dùng chọn món ăn mong muốn.
- + Hiển thị hiển thị outlay cho phép người dùng xem trước số lượng và giá cả trước khi thêm vào giỏ hàng.
- Giỏ hàng và thanh toán:
 - + Thêm/xoá món ăn trong giỏ hàng.
 - + Cập nhật số lượng món ăn.
 - + Hiển thị tổng tiền.
 - + Xác nhận đơn hàng khi thanh toán.
 - + Thanh toán khi nhận hàng.
- Quản lý đơn hàng:
 - + Khách hàng xem trạng thái đơn hàng:
Chờ xác nhận → Đang chuẩn bị → Đang giao → Đã giao.
 - + Nhân viên/cửa hàng xác nhận, chuẩn bị, cập nhật trạng thái đơn hàng.
- Xem lại thông tin đơn hàng đã đặt và trạng thái: Người dùng có thể xem lịch sử giao dịch và tình trạng đơn hàng.

2.1.2 Chức năng dành cho quản trị viên

- Đăng nhập quản trị: Chỉ admin mới được phép truy cập vào giao diện quản trị.
- Quản lý danh mục nhóm sản phẩm:
 - + Xem danh sách nhóm sản phẩm có trong hệ thống.
 - + Thêm, sửa và xóa các nhóm sản phẩm không còn bán nữa.
- Quản lý sản phẩm:
 - + Xem danh sách sản phẩm có trong hệ thống.
 - + Thêm, sửa các sản phẩm với các thông tin: Tên món ăn, hình ảnh, giá, và thuộc nhóm sản phẩm nào.
 - + Xóa các sản phẩm không còn bán nữa.

- Quản lý đơn hàng: Xem danh sách đơn hàng, đồng thời cập nhập tình trạng đơn hàng.
- Quản lý người dùng: Xem danh sách người dùng đăng ký.
- Xem báo cáo thống kê: Thống kê số lượng đơn hàng, doanh thu theo ngày/ tuần/ tháng giúp hệ thống nắm bắt được tình trạng cửa hàng.

2.2 Các yêu cầu phi chức năng (Non-functional Requirements)

Các yêu cầu phi chức năng đề cập đến chất lượng hệ thống, hiệu suất, tính bảo mật, và khả năng mở rộng:

2.2.1 Hiệu suất

- Hệ thống cần phản hồi nhanh trong các thao tác tìm kiếm, thêm vào giỏ hàng, đặt hàng và xem trạng thái đơn hàng.
- Thời gian tải trang không vượt quá 3 giây đối với kết nối mạng trung bình.

2.2.2 Tính bảo mật và quyền riêng tư

- Thông tin tài khoản người dùng được bảo mật bằng cách mã hóa mật khẩu (sử dụng bcrypt).
- Sử dụng JWT để xác thực và phân quyền truy cập API.
- Dữ liệu cá nhân của người dùng được giới hạn hiển thị và chỉ truy cập bởi chính chủ tài khoản (hướng tiếp cận tương đương nguyên tắc GDPR).
- Có cơ chế kiểm tra quyền truy cập tài nguyên (middleware bảo vệ route).
- Chỉ người dùng có vai trò admin mới được phép truy cập giao diện quản trị.

2.2.3 Tính sẵn sàng

- Hệ thống có thể hoạt động liên tục 24/7 (nếu triển khai thật).
- Khả năng xử lý nhiều người dùng cùng lúc đặt vé mà không xung đột dữ liệu.

2.2.4 Khả năng mở rộng

- Hệ thống được thiết kế theo mô hình client-server tách biệt (React frontend và Node.js backend), dễ tích hợp với mobile app trong tương lai.
- Sử dụng Docker giúp triển khai linh hoạt trên môi trường cloud.
- Sử dụng Swagger để tạo tài liệu API tự động.

2.2.5 Tính dễ sử dụng

- Giao diện người dùng nhất quán, dễ sử dụng cho cả người không am hiểu công nghệ.
- Giao diện quản trị đơn giản, phân loại chức năng theo danh mục.

2.2.6 Tính duy trì và phát triển

- Mã nguồn được tổ chức rõ ràng theo mô hình RESTful API, dễ bảo trì và phát triển thêm tính năng.
- Tách biệt rõ frontend – backend để các thành viên nhóm có thể làm việc song song.
- Nhóm có kiểm thử API bằng Postman để đảm bảo chất lượng hệ thống và phát hiện lỗi sớm.

2.3 Mô hình phát triển phần mềm

Phát triển phần mềm thường dựa trên các mô hình khác nhau, mỗi mô hình có ưu điểm và hạn chế tùy thuộc vào yêu cầu dự án. Hai mô hình phổ biến được nhắc đến là Waterfall và Agile. Mô hình Waterfall là phương pháp tuyến tính, trong đó các giai đoạn (yêu cầu, thiết kế, triển khai, kiểm thử) được thực hiện tuần tự và không cho phép quay lại sau khi hoàn thành. Ngược lại, mô hình Agile tập trung vào phát triển lặp lại, chia dự án thành các chu kỳ ngắn gọi là "sprint", cho phép điều chỉnh yêu cầu dựa trên phản hồi và thay đổi thực tế.

Nhóm đã lựa chọn mô hình Agile cho dự án "Hệ thống quản lý Website thương mại điện tử bán thức ăn nhanh" vì tính linh hoạt và khả năng thích nghi cao, phù hợp với việc phát triển một hệ thống web với nhiều chức năng (đặt đồ ăn, quản lý sản phẩm, xử lý đơn hàng, quản lý admin). Agile cho phép nhóm phản ứng nhanh với các yêu cầu mới hoặc cải tiến từ feedback trong quá trình phát triển, đặc biệt khi thiết kế giao diện menu và tích hợp API quản lý đơn hàng.

Trong quá trình thực hiện, Agile được áp dụng để chia dự án thành các giai đoạn ngắn, cho phép nhóm điều chỉnh yêu cầu dựa trên phản hồi từ thử nghiệm và ý kiến từ giảng viên.

Phương pháp Agile cũng được hỗ trợ bởi các phương pháp phát triển hiện đại: CI/CD với GitHub Actions để tích hợp liên tục, Docker container hóa cho môi trường phát triển nhất quán, quy trình làm việc Git cho cộng tác nhóm và kiến trúc mô-đun cho phát triển tính năng độc lập. Phương pháp này không chỉ phù hợp với tiến độ học tập mà còn có thể được hoàn thành với phản hồi và thay đổi yêu cầu.

CHƯƠNG 3: THIẾT KẾ HỆ THỐNG

3.1 Kiến trúc tổng thể

Hệ thống được phát triển theo mô hình Client–Server, chia thành 3 thành phần chính:

3.1.1 Frontend (Client Layer)

- Sử dụng công nghệ: React.js 17, CSS3, React Router
- Vai trò: Cung cấp giao diện người dùng để tương tác, như xem sản phẩm, chọn món ăn, thêm vào giỏ hàng, đặt hàng, quản lý hệ thống...
- Gửi yêu cầu HTTP tới server thông qua các API RESTful.
- Port: 3000

3.1.2 Server (Node.js + ExpressJS)

Hệ thống backend được xây dựng theo mô hình phân lớp gồm các thành phần chính như sau:

- Cơ sở dữ liệu:
 - + Lưu trữ thông tin người dùng (User), sản phẩm (Product), đơn hàng (Order) thông qua MongoDB
 - + Truy xuất dữ liệu thông qua Mongoose ODM với các schema được định nghĩa trong thư mục “models.js”.
- API Layer: Các logic xử lý nghiệp vụ được tổ chức trong thư mục api/ gồm:
 - + login-register.js: Xử lý đăng ký, đăng nhập người dùng
 - + products.js: Quản lý sản phẩm (CRUD operations)
 - + orders.js: Xử lý đơn hàng và trạng thái
 - + users.js: Quản lý thông tin người dùng
 - + categories.js: Quản lý danh mục sản phẩm
 - + overview.js: Dashboard và thống kê

- Routes Configuration: Được định nghĩa trong app.js với các endpoint RESTful:

+ /api/login-register: Authentication endpoints (đăng ký, đăng nhập)

+ /api/products: Product management (CRUD sản phẩm)

+ /api/orders: Order processing (xử lý đơn hàng)

+ /api/users: User management (quản lý người dùng)

+ /api/categories: Category management (quản lý danh mục)

+ /api/overview: Dashboard và thống kê tổng quan

+ /api/contact: Contact form và customer support

+ /api-docs: Swagger API documentation

+ /uploads: Static file serving cho hình ảnh

- Middleware & Additional Features: Bao gồm các chức năng như kiểm tra xác thực người dùng (session-based authentication), phân quyền (admin/customer), validation dữ liệu đầu vào, xử lý upload file, CORS handling, và error handling middleware.

3.1.3 Cơ sở dữ liệu

Hệ thống sử dụng MongoDB làm cơ sở dữ liệu NoSQL với Mongoose ODM để quản lý dữ liệu. Database được thiết kế với 4 collection chính:

- Users Collection: Lưu trữ thông tin người dùng bao gồm username, password (được hash), email, phone, address, role (customer/admin), và timestamps. Có unique constraints trên username và email để đảm bảo tính duy nhất.

- Products Collection: Quản lý thông tin sản phẩm với các trường name, price, category, và createdAt. Được liên kết với Orders thông qua productId reference.

- Orders Collection: Lưu trữ thông tin đơn hàng với userId reference đến Users, mảng products chứa productId, quantity và note, totalAmount, status (pending/completed/cancelled), và timestamps. Sử dụng MongoDB references để tạo mối quan hệ one-to-many.

- Contact Collection: Quản lý tin nhắn liên hệ từ khách hàng với name, email, message và createdAt.

Database connection được cấu hình kết nối đến MongoDB container thông qua connection string “*mongodb://mongo:27017/ada_fastfood*”, hỗ trợ môi trường Docker và có fallback về localhost cho development.

Sơ đồ mô tả kiến trúc hệ thống sử dụng mô hình Client - Server, trong đó:

- Client: là một ứng dụng React SPA (Single Page Application) chạy trên port 3000, được build và serve thông qua React development server hoặc production build.

- Server: gồm Express.js backend chạy trên port 5000 với các API endpoints (/api/login-register, /api/products, /api/orders, /api/users, /api/categories, /api/overview, /api/contact), tích hợp Swagger UI documentation tại /api-docs.

- Database: MongoDB NoSQL database chạy trên port 27017 với database name "ada_fastfood", sử dụng Mongoose ODM để quản lý 4 collections chính (Users, Products, Orders, Contact).

- File Storage: Static file serving cho uploads (images) thông qua Express static middleware tại endpoint /uploads..

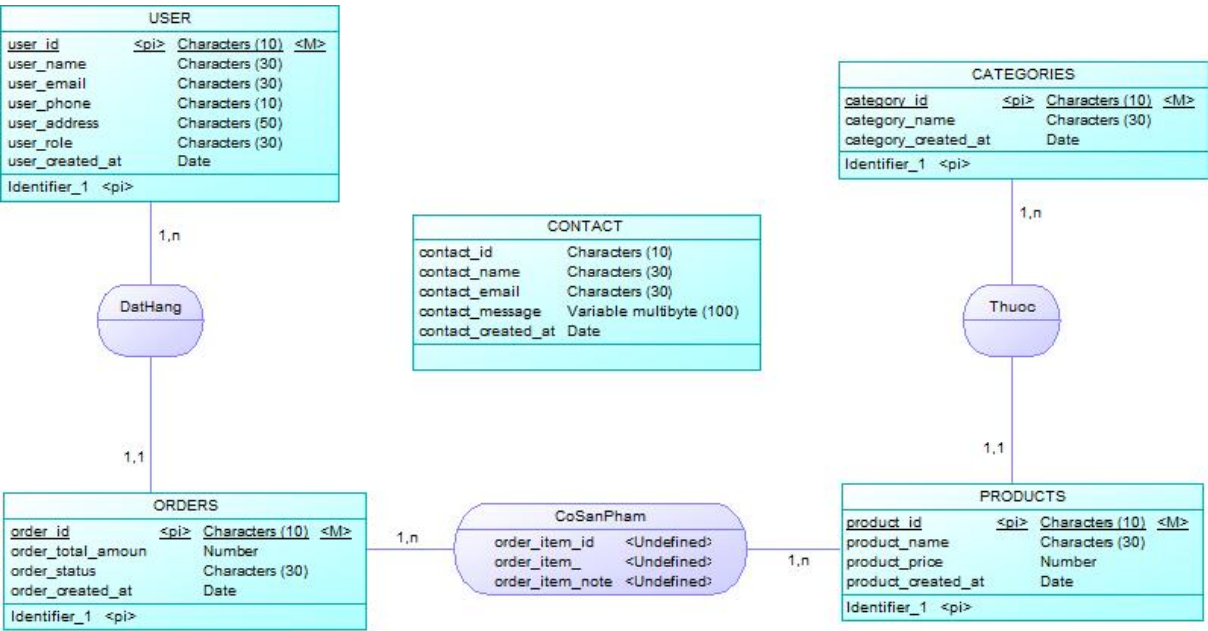
Các thành phần giao tiếp qua REST API với format JSON, hỗ trợ CORS cho cross-origin requests.

Hệ thống được containerized bằng Docker với:

- Frontend container (React app)
- Backend container (Express.js + Node.js)
- MongoDB container
- Shared volumes cho file uploads.

3.2 Thiết kế cơ sở dữ liệu

3.2.1 Mô hình thực thể - quan hệ (ERD)



Hình 3.1 Sơ đồ ERD được thiết kế trong PowerDesign

Thực thể	Thuộc tính	Khóa chính	Khóa ngoại	Mối quan hệ
USER	user_id, user_name, user_email, user_phone, user_address, user_role, user_created_at	user_id		1-N với Orders (Một user có nhiều orders)
CATEGORIES	category_id, category_name, category_created_at	category_id		1-N với Products (Một nhóm sp có nhiều sản phẩm)

PRODUCTS	product_id, product_name, product_price, category_id, product_created_at	product_id	category_id (String ref đến Categories.name)	1-1 với Categories (Một sản phẩm thuộc về một nhóm) 1-N với Orders (Một sản phẩm có trong nhiều Order)
ORDERS	order_id, user_id, order_total_amoun, order_status, order_created_at	Order_id	user_id	1-N với Products (Một đơn hàng có nhiều sản phẩm) 1-1 với Users (Một Order thuộc về một Users)
ORDERS_ITEMS	order_item_id, order_id, product_id, order_item_quantit, order_item_note	order_item_id	order_id, product_id	Được tạo ra từ Products và Order
CONTACT	contact_id, contact_name, contact_email, contact_message, contact_created_at	contact_id		Độc lập (Không có mối quan hệ với các thực thể khác)

Bảng 3.1 Mô tả các quan hệ giữa các thực thể trong ERD

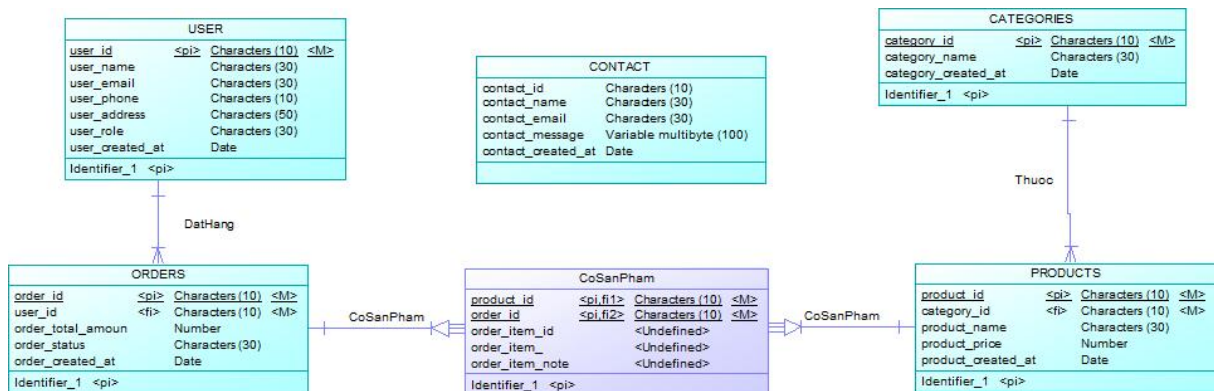
- Môi quan hệ:

- + 1-N: Quan hệ một-nhiều (ví dụ: Một Category có nhiều Products).
- + 1-1: Quan hệ nhiều-một (ví dụ: Một Orders thuộc về một User).
- + N-N: Quan hệ nhiều-nhiều (ví dụ: Một order chứa nhiều products, một product có thể xuất hiện trong nhiều orders).

3.2.2 Sơ đồ quan hệ dữ liệu MySQL

Cơ sở dữ liệu được xây dựng dựa trên mô hình ERD thiết kế ở mục trên bằng PowerDesigner, sau đó được triển khai thực tế bằng MongoDB với Mongoose ODM. Các collection được tạo ra thông qua định nghĩa Schema trong file models.js và tự động khởi tạo khi ứng dụng chạy lần đầu.

Cơ sở dữ liệu bao gồm các collection chính như: users (quản lý người dùng), categories (phân loại sản phẩm), products (danh mục món ăn), orders (đơn hàng với embedded order_items), và contact (tin nhắn liên hệ từ khách hàng). MongoDB tự động tạo các chỉ mục trên các trường _id và hỗ trợ các ràng buộc như cho use_rname, email và category_name.



Hình 3.2 Sơ đồ cơ sở dữ liệu quan hệ MySQL

3.3 Thiết kế API

3.3.1 Tổng quan

Hệ thống cung cấp RESTful API theo chuẩn OpenAPI 3.0, tài liệu hóa bằng Swagger, dưới đây là bảng mô tả tổng quan:

Mục	Giá trị
Base URL	http://localhost:5000/api
Swagger UI	http://localhost:5000/api-docs
Xác thực	Bearer Token (JWT) – header Authorization: Bearer <token> Xác thực dựa trên phân quyền truy cập dựa trên vai trò.
Định dạng	application/json (trừ các endpoint upload hình sử dụng multipart/form-data)
CORS	Được cấu hình cho phép cross-origin requests từ frontend
File Upload	Tệp tĩnh phục vụ tại điểm cuối /uploads

Bảng 3.2 Cấu hình truy cập API

Cấu hình Swagger nằm trong src/swagger.js; tài liệu được sinh tự động từ các chú thích @swagger trong các file API router (login-register.js, users.js, products.js, categories.js, orders.js, overview.js, contact.js). Swagger UI được mount tại endpoint /api-docs thông qua swagger-ui-express middleware trong app.js.

3.3.2 Nhóm endpoint chính

Hệ thống cung cấp các nhóm API chính phục vụ các chức năng CRUD (Create, Read, Update, Delete) như quản lý người dùng, sản phẩm, danh mục, đơn hàng, thống kê,... Mỗi nhóm được tổ chức theo chuẩn RESTful và định nghĩa rõ ràng bằng Swagger để dễ tích hợp frontend và kiểm thử.

Nhóm chức năng	Đường dẫn (prefix)	Các thao tác nổi bật
Authentication	/login-register	POST /register, POST /login,...
Người dùng	/users U	GET /, GET /:id, POST /, PUT /:id, DELETE /:id
Danh mục sản phẩm	/categories	GET /, POST /, PUT /:id, DELETE /:id
Sản phẩm	/products	GET /, GET /:id, POST /, PUT /:id, DELETE /:id, POST /upload (hình ảnh)
Đơn hàng	/orders	GET /, GET /:id, POST /, PUT /:id, DELETE /:id
Thống kê tổng quan	/overview	GET /stats (doanh thu, đơn hàng, sản phẩm bán chạy)
Liên hệ	/contact	GET /, POST /, DELETE /:id

Bảng 3.3 Danh sách nhóm endpoint chính

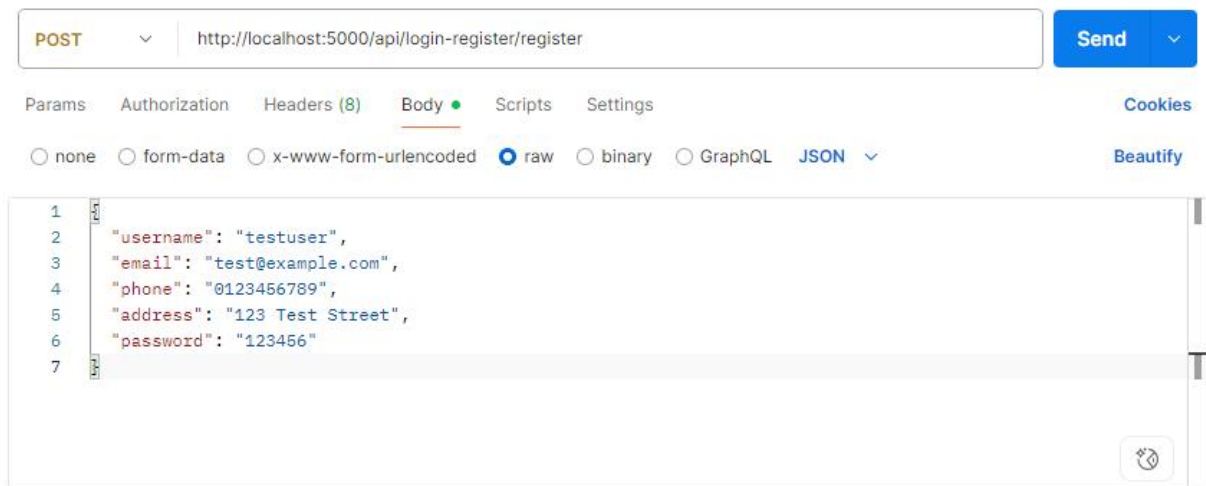
3.3.3 Cấu trúc request/response

Dưới đây là một số ví dụ minh họa cách thức giao tiếp giữa client và server, bao gồm phần request từ phía người dùng và phản hồi trả về từ hệ thống.

Đăng ký người dùng mới:

- Phương thức: POST
- URL: `http://localhost:5000/api/login-register/register`
- Request (yêu cầu):

```
{
  "username": "testuser",
  "email": "test@example.com",
  "phone": "0123456789",
  "address": "123 Test Street",
  "password": "123456"
}
```



Hình 3.3 Request gửi dữ liệu đăng ký

- Response (phản hồi):

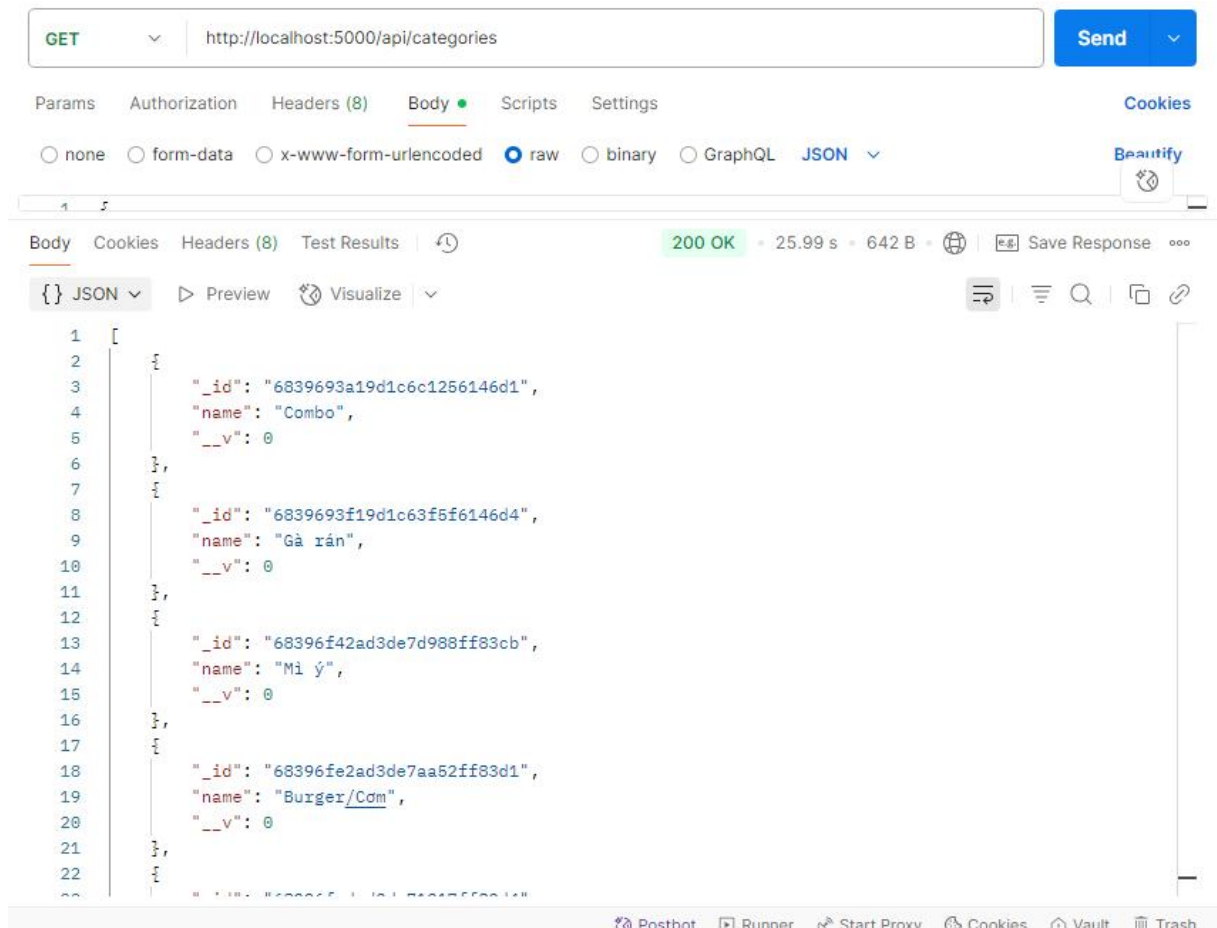


Hình 3.4 Response khi đăng ký thành công

Lấy danh sách Nhóm sản phẩm:

- Phương thức: POST
- URL: <http://localhost:5000/api/categories>
- Response:

```
[
  {
    "_id": "6839693a19d1c6c1256146d1",
    "name": "Combo",
    "__v": 0
  },
  {
    "_id": "6839693f19d1c63f5f6146d4",
    "name": "Gà rán",
    "__v": 0
  },
  ...
  {
    "_id": "6839724bad3de73c77ff842d",
    "name": "Nước ngọt",
    "__v": 0
  }
]
```



Hình 3.5 Response khi lấy danh sách món ăn thành công

Cập nhật thông tin các nhân (khách hàng):

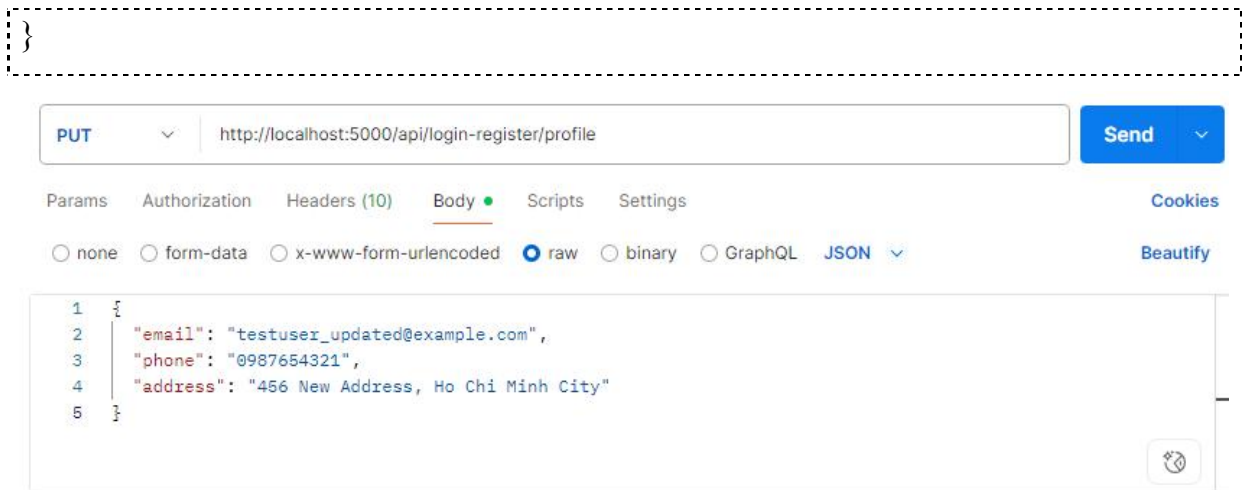
- Phương thức: PUT
- URL: http://localhost:5000/api/login-register/profile
- Header:

Content-Type: application/json

Authorization: Bearer YOUR_JWT_TOKEN

- Request (chỉ cập nhật email, phone và address):

```
{
  "email": "testuser_updated@example.com",
  "phone": "0987654321",
  "address": "456 New Address, Ho Chi Minh City"
}
```



Hình 3.6 Request cập nhật thông tin một người dùng

- Response:

```

{
  "message": "Cập nhật thông tin thành công!",
  "user": {
    "role": "user",
    "_id": "687de0bb3d428a5d14772f66",
    "username": "testuser",
    "email": "testuser_updated@example.com",
    "phone": "0987654321",
    "address": "456 New Address, Ho Chi Minh City",
    "createdAt": "2025-07-21T06:39:55.165Z",
    "__v": 0
  }
}
    
```



Hình 3.7 Response sau khi cập nhật thành công

3.4 Thiết kế giao diện (UI/UX)

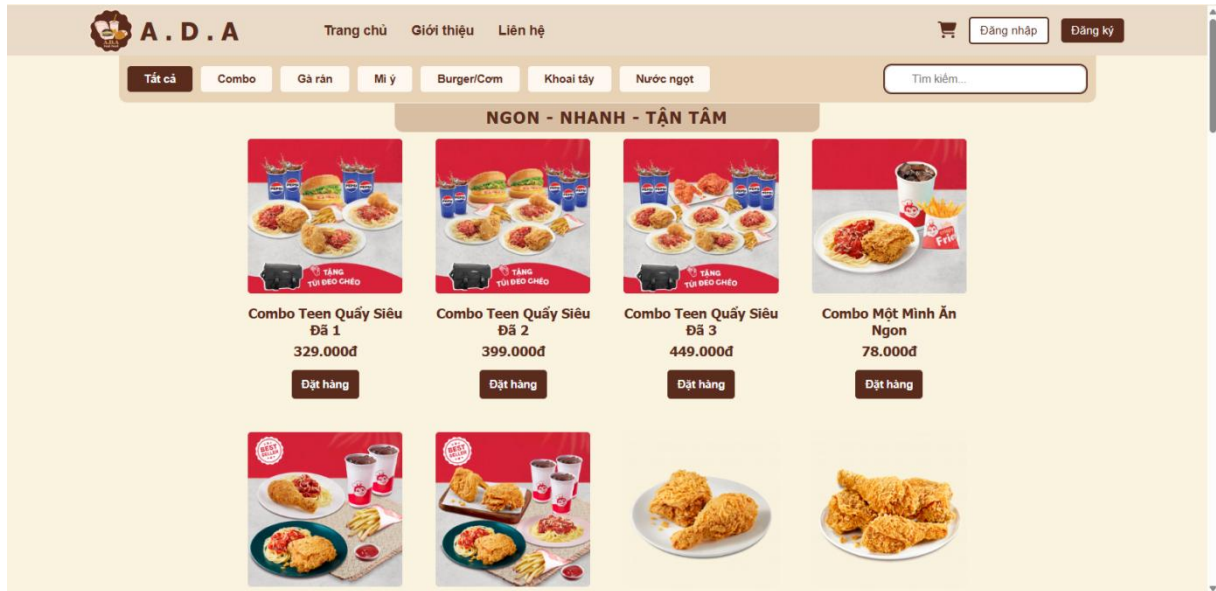
Giao diện người dùng được thiết kế bằng Figma, đảm bảo trực quan, dễ sử dụng và nhất quán. Các yếu tố như màu sắc, bố cục và tương tác được tối ưu cho trải nghiệm người dùng. Giao diện được chia thành hai nhóm chính: người dùng đặt vé và quản trị viên quản lý hệ thống.

3.4.1 Trang chủ website

Giao diện chính khi truy cập website, hiển thị slogan "NGON - NHANH - TẬN TÂM", hình ảnh logo chính của A.D.A được đặt ở bên trái trang tạo điểm nhấn thị giác cho thương hiệu. Danh sách sản phẩm theo danh mục với chức năng tìm kiếm và đặt hàng. Trên đầu trang và cuối trang là thanh header và footer dùng chung cho tất cả trang giao diện người dùng.

Thanh header có các mục như "Trang chủ", "Menu", "Giới thiệu", thanh tìm kiếm sản phẩm với biểu tượng kính lúp, và các nút Đăng nhập/Đăng ký nếu chưa đăng nhập. Nếu đã đăng nhập thì hiển thị tên người dùng và dropdown menu dành cho người dùng như xem thông tin cá nhân, cập nhật thông tin, đổi mật khẩu, xem lịch sử đơn hàng và đăng xuất. Nếu đăng nhập bằng tài khoản admin thì sẽ có thêm mục "Trang quản trị" để điều hướng đến trang chỉ dành cho admin quản lý sản phẩm, danh mục, đơn hàng và thống kê.

Trang chủ hiển thị các danh mục sản phẩm dưới dạng menu bar ngang với nút "Tất cả" mặc định. Người dùng có thể lọc sản phẩm theo danh mục, tìm kiếm theo tên, và thêm sản phẩm vào giỏ hàng với overlay modal để chọn số lượng và ghi chú. Giỏ hàng được lưu trữ tạm thời trong localStorage cho đến khi người dùng thực hiện checkout.



Hình 3.8 Giao diện trang chủ website

3.4.2 Giao diện đăng nhập, đăng ký

Người dùng bấm vào hai nút Đăng nhập hoặc đăng kí để chuyển đến trang tương ứng, sau khi đăng nhập thành công sẽ chuyển đến trang chủ nếu là người dùng và chuyển đến trang quản trị nếu là admin website.

URL của trang đăng nhập: <http://localhost:3000/login>

URL của trang đăng ký: <http://localhost:3000/Register>

Hình 3.9 Giao diện trang đăng ký tài khoản

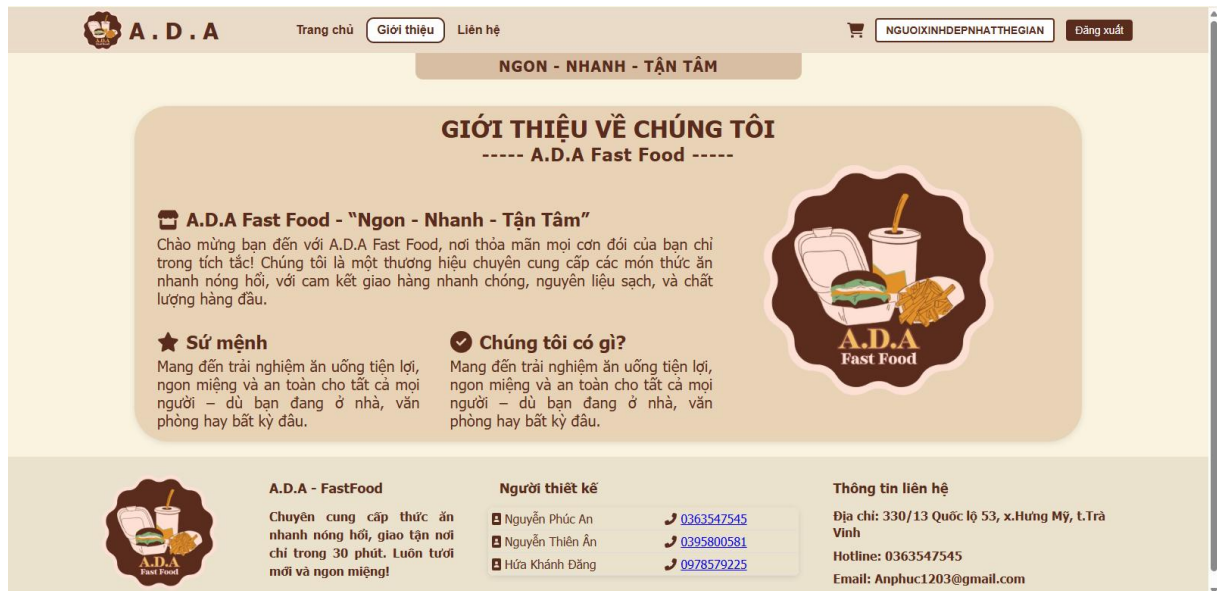
Sau khi bấm nút “ĐĂNG KÝ” sẽ chuyển sang form đăng nhập.

Hình 3.10 Giao diện trang đăng nhập

3.4.3 Trang Giới thiệu

Khi người dùng chọn vào nút “Giới thiệu” trên thanh header sẽ hiện ra trang thông tin về cửa hàng A.D.A FastFood. Trang này được thiết kế với layout table để trình bày thông tin một cách có tổ chức và dễ đọc.

URL của trang: <http://localhost:3000/gioithieu>



Hình 3.11 Giao diện trang Giới thiệu

3.4.4 Giao diện trang Liên hệ

Khi người dùng chọn vào nút "Liên hệ" trên thanh header sẽ hiện ra trang thông tin liên hệ với A.D.A FastFood. Trang này được thiết kế để cung cấp đầy đủ thông tin liên lạc và cho phép khách hàng gửi tin nhắn trực tiếp.

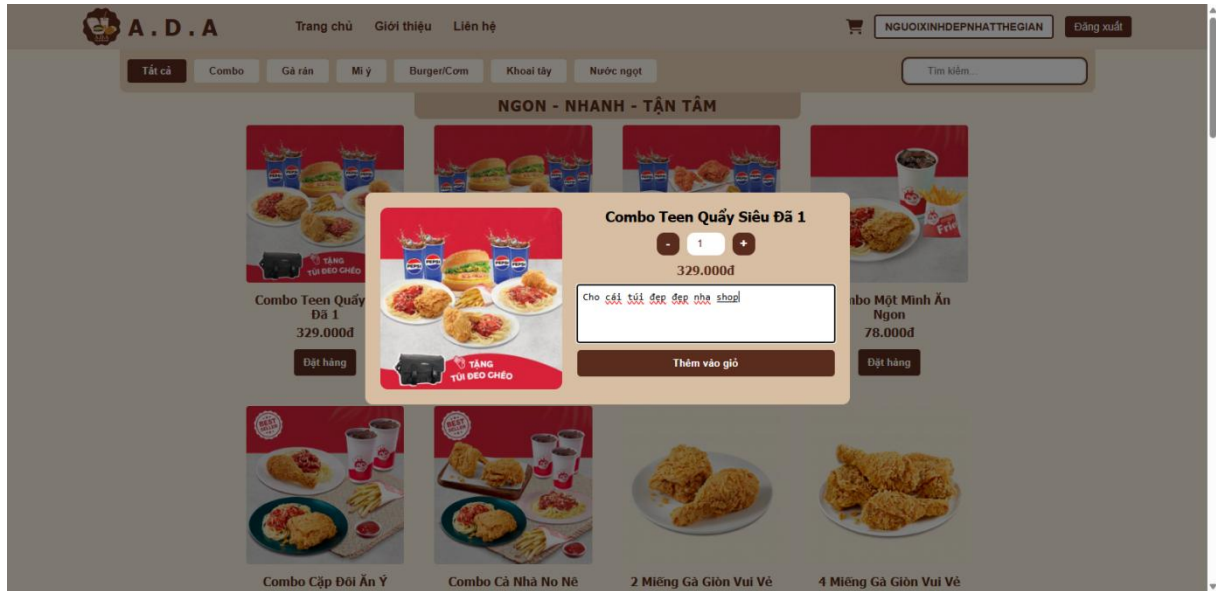
URL của trang: <http://localhost:3000/lienhe>



Hình 3.12 Giao diện trang Liên hệ

3.4.5 Giao diện Overlay khi thêm món ăn vào giỏ hàng

Khi người dùng nhấn nút "Đặt hàng" trên bất kỳ sản phẩm nào, hệ thống sẽ hiển thị một overlay modal cho phép người dùng tùy chỉnh đơn hàng trước khi thêm vào giỏ hàng. Có thể tăng giảm sản phẩm, xem số tiền, ghi chú cho sản phẩm và thêm vào giỏ hàng.

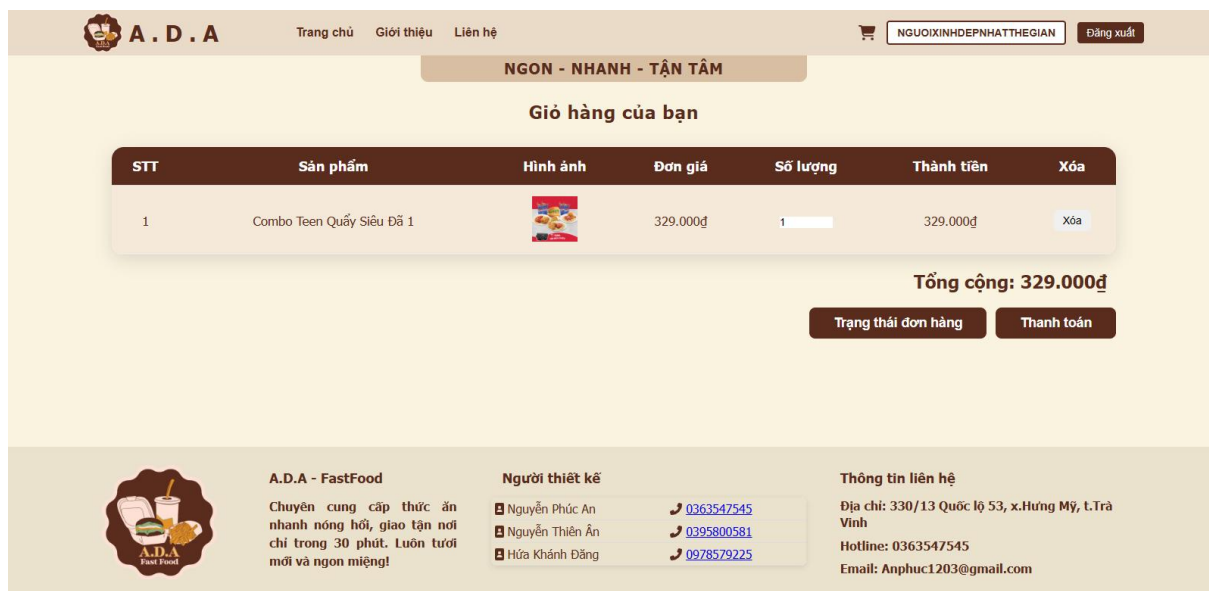


Hình 3.13 Overlay thêm món ăn vào giỏ hàng

3.4.6 Giao diện trang Giỏ hàng

Trang giỏ hàng được truy cập khi người dùng nhấn vào biểu tượng giỏ hàng trên header. Đây là nơi người dùng quản lý các sản phẩm đã chọn trước khi tiến hành thanh toán.

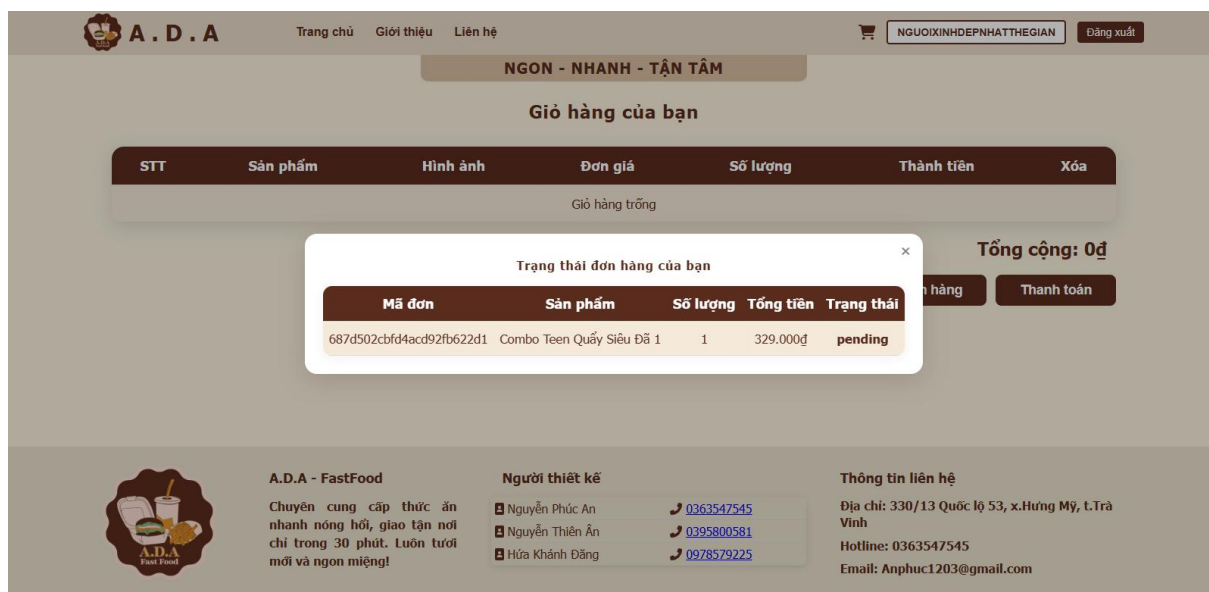
URL của trang: <http://localhost:3000/giohang>.



Hình 3.14 Giao diện trang Giỏ hàng

Sau khi nhấn nút “Đặt hàng”, website sẽ hiển thị thông báo “Đặt hàng thành công!”. (Bị lỗi nếu giỏ hàng trống hoặc chưa đăng nhập)

Sau khi nhấn nút “Trạng thái đơn hàng”, một Overlay sẽ hiển thị trạng thái đơn hàng và lịch sử đơn hàng nếu có.



Hình 3.15 Overlay hiện trạng thái đơn hàng

3.4.7 Giao diện trang thông tin cá nhân

Trang thông tin cá nhân hiển thị và cho phép người dùng cập nhật các thông tin như tên đăng nhập, email, số điện thoại và địa chỉ. Giao diện đơn giản, rõ ràng, phù hợp cho việc quản lý thông tin cá nhân.

URL của trang thông tin cá nhân: <http://localhost:3000/profile>.

A.D.A. Trang chủ Giới thiệu Liên hệ

NGON - NHANH - TẬN TÂM

Hồ sơ tài khoản

Quyền: Khách hàng

Tên đăng nhập: nguoixinhdepnhatthegian

Email: gian@gmail.com

Số điện thoại: 0363547545

Địa chỉ: Chà Vinh

Sửa thông tin

Đổi mật khẩu

A.D.A - FastFood
Chuyên cung cấp thức ăn nhanh nóng hổi, giao tận nơi chỉ trong 30 phút. Luôn tươi

Người thiết kế
Nguyễn Phúc An
Nguyễn Thiên Ân

Thông tin liên hệ
Địa chỉ: 330/13 Quốc lộ 53, x.Hưng Mỹ, t.Trà Vinh
Hotline: 0363547545

Hình 3.16 Giao diện trang đổi mật khẩu

Người dùng có thể chỉnh sửa trực tiếp bằng cách nhấn nút “Sửa thông tin” => sử dụng thông tin trực tiếp => nhấn nút "Lưu" để lưu thay đổi.

Hồ sơ tài khoản

Quyền: Khách hàng

Tên đăng nhập: nguoixinhdepnhatthegian

Email: gian@gmail.com

Số điện thoại: 0363547545

Địa chỉ: Vĩnh Long

Sửa thông tin

Đổi mật khẩu

Cập nhật thông tin thành công!

Hồ sơ tài khoản

Quyền: Khách hàng

Tên đăng nhập: nguoixinhdepnhatthegian

Email: gian@gmail.com

Số điện thoại: 0363547545

Địa chỉ: Vĩnh Long

Lưu

Hủy

Đổi mật khẩu

Cập nhật thông tin thành công!

Hình 3.17 Giao diện đổi thông tin cá nhân

3.4.8 Giao diện đổi mật khẩu

Để đổi mật khẩu cho tài khoản, ở trang Profile, có nút “Đổi mật khẩu”. Sau khi nhấn vào giao diện sẽ hiển thị 3 ô để nhập gồm: Mật khẩu cũ, mật khẩu mới và nhập lại mật khẩu mới. Sau đó nhấn nút “Xác nhận đổi mật khẩu” để lưu mật khẩu mới. Từ đây, bạn có thể đăng nhập bằng mật khẩu vừa đổi.



Hình 3.18 Giao diện Đổi mật khẩu

3.4.9 Giao diện các trang quản trị

URL của trang Admin: <http://localhost:3000/admin> (chỉ có admin mới có thể truy cập).

3.4.9.1 Giao diện menu Tổng quan

Trang Tổng quan là trang chính của khu vực quản trị, khi đăng nhập bằng tài khoản quản trị, cung cấp cái nhìn tổng thể về tình hình kinh doanh của cửa hàng A.D.A FastFood thông qua các chỉ số thống kê quan trọng.

Hệ thống cung cấp 3 bộ lọc thời gian để admin có thể xem thống kê theo từng khoảng thời gian cụ thể: Ngày / Tuần / Tháng hiện tại.

A.D.A. ADMIN Đăng xuất

NGON - NHANH - TẬN TÂM

A.D.A.

Tổng quan

Đơn đặt hàng

Sản phẩm

Nhóm sản phẩm

Người dùng

Liên hệ

Ngày Tuần Tháng

Tổng số đơn: 1

Tổng số tiền: 329.000đ

Tổng khách hàng: 0

Món bán chạy: Combo Teen Quẩy Siêu Đã 1

A.D.A - FastFood

Chuyên cung cấp thức ăn nhanh nóng hổi, giao tận nơi chỉ trong 30 phút. Luôn tươi mới và ngon miệng!

Người thiết kế

Nguyễn Phúc An 0363547545

Nguyễn Thiên Ân 0395800581

Hứa Khánh Đăng 0978579225

Thông tin liên hệ

Địa chỉ: 330/13 Quốc lộ 53, x.Hưng Mỹ, t.Trà Vinh

Hotline: 0363547545

Email: Anphuc1203@gmail.com

Hình 3.19 Giao diện menu thống kê

3.4.9.2 Giao diện menu Đơn đặt hàng

Trang này cho phép admin theo dõi và quản lý tất cả đơn hàng trong hệ thống, từ việc xem danh sách đến cập nhật trạng thái và xóa đơn hàng.

A.D.A. ADMIN Đăng xuất

NGON - NHANH - TẬN TÂM

A.D.A.

Tổng quan

Đơn đặt hàng

Sản phẩm

Nhóm sản phẩm

Người dùng

Liên hệ

Mã đơn	Tên khách hàng	Món ăn	Tổng tiền	Trạng thái	Xóa
687d502cbfd4acd92fb622d1	nguoioinhdepnhathegian	Combo Teen Quẩy Siêu Đã 1 (1)	329.000đ	Đang chuẩn bị	Xóa

A.D.A - FastFood

Chuyên cung cấp thức ăn nhanh nóng hổi, giao tận nơi chỉ trong 30 phút. Luôn tươi mới và ngon miệng!

Người thiết kế

Nguyễn Phúc An 0363547545

Nguyễn Thiên Ân 0395800581

Hứa Khánh Đăng 0978579225

Thông tin liên hệ

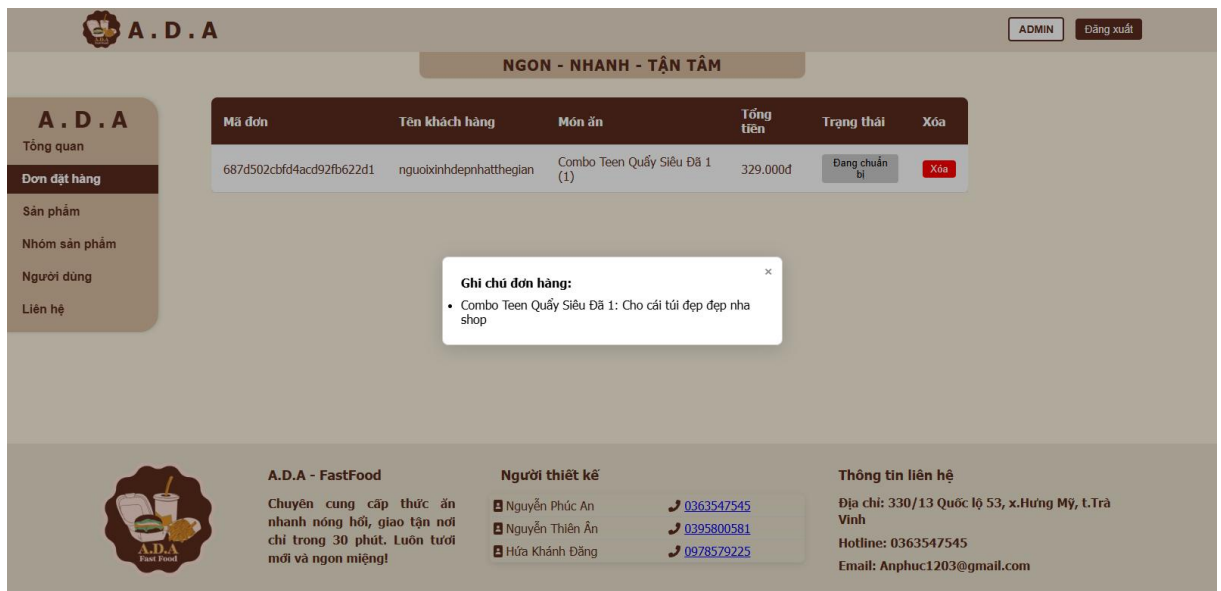
Địa chỉ: 330/13 Quốc lộ 53, x.Hưng Mỹ, t.Trà Vinh

Hotline: 0363547545

Email: Anphuc1203@gmail.com

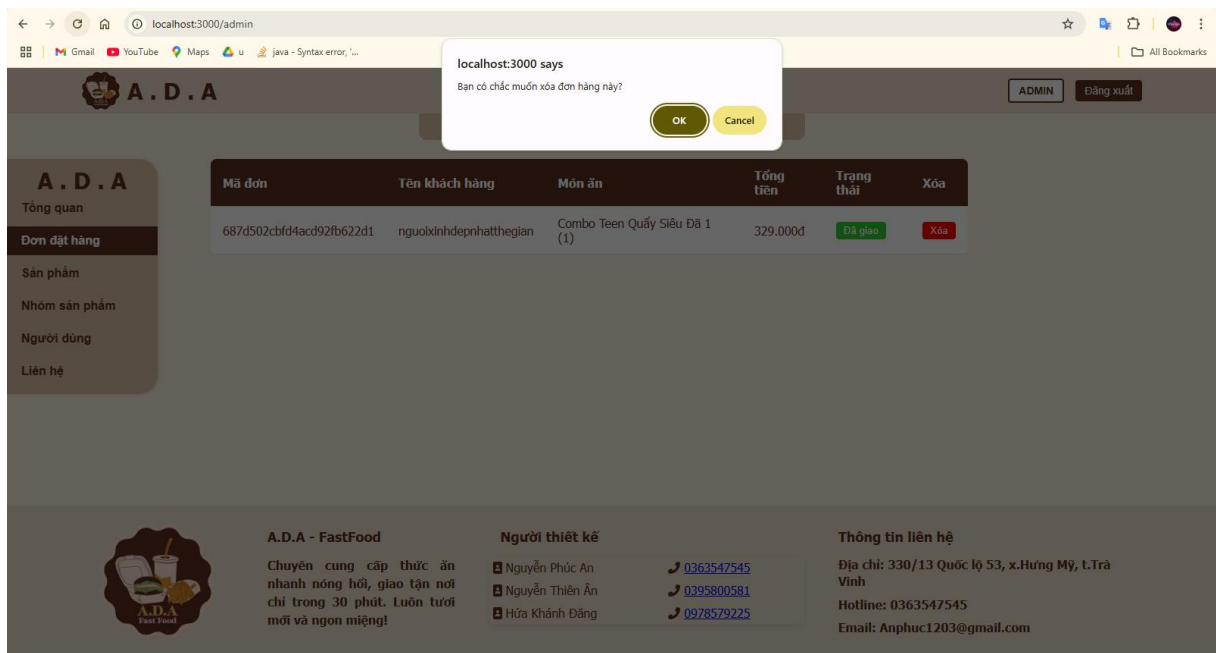
Hình 3.20 Giao diện menu Đơn đặt hàng

Khi bấm vào một đơn hàng bất kỳ, giao diện sẽ hiển thị cho xem “Ghi chú đơn hàng”.



Hình 3.21 Overlay hiện Ghi chú đơn hàng

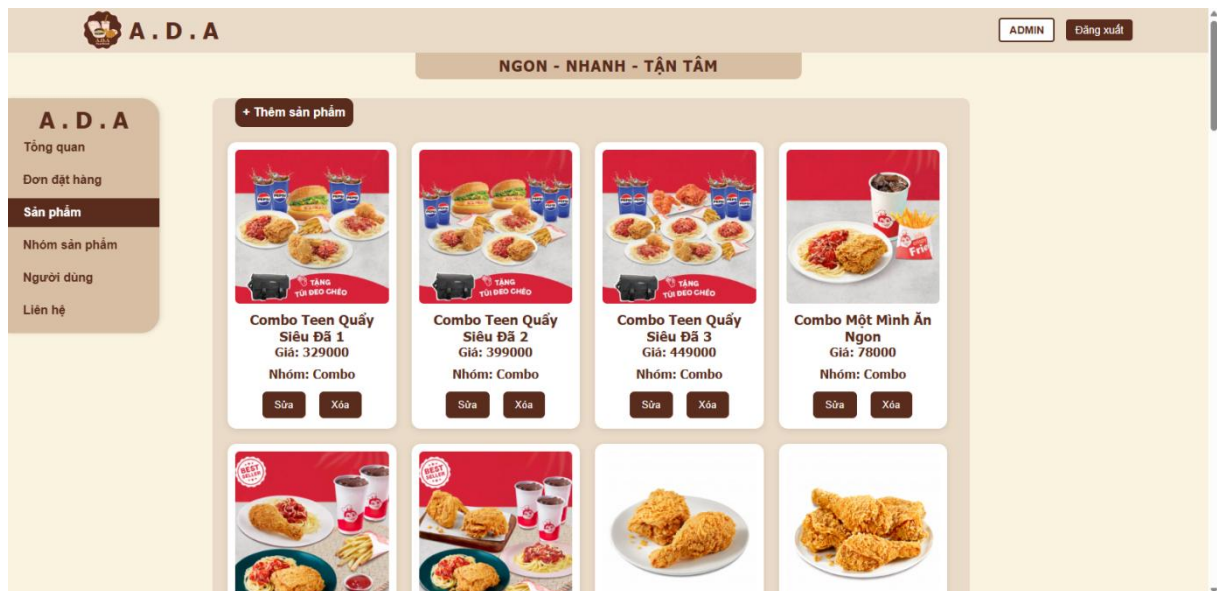
Tương tự như chức năng xem thì chức năng xóa sẽ được thực hiện khi nhấn chọn biểu tượng xóa ở cột “Xóa” sẽ sẽ hiện thông báo xác nhận xóa đơn hàng đó.



Hình 3.22 Thông báo xác nhận xóa Đơn hàng

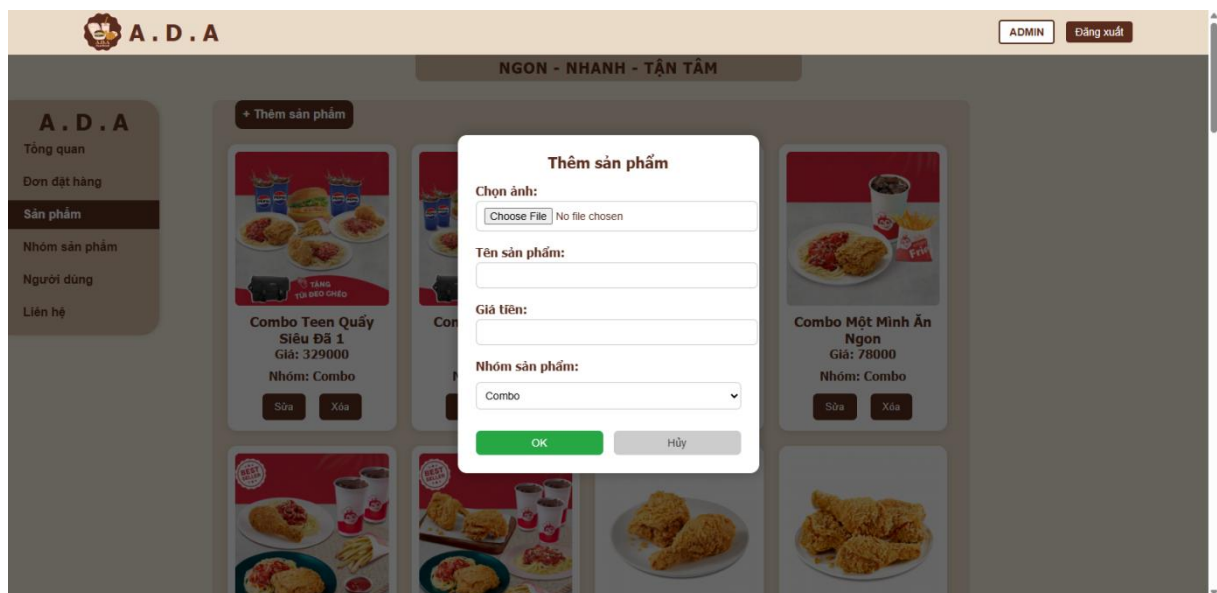
3.4.9.3 Giao diện menu Sản phẩm

Trang này cung cấp giao diện quản lý toàn bộ sản phẩm trong hệ thống với các chức năng CRUD đầy đủ và trực quan. Admin có thể xem danh sách sản phẩm và có thể thêm, sửa và xóa sản phẩm.



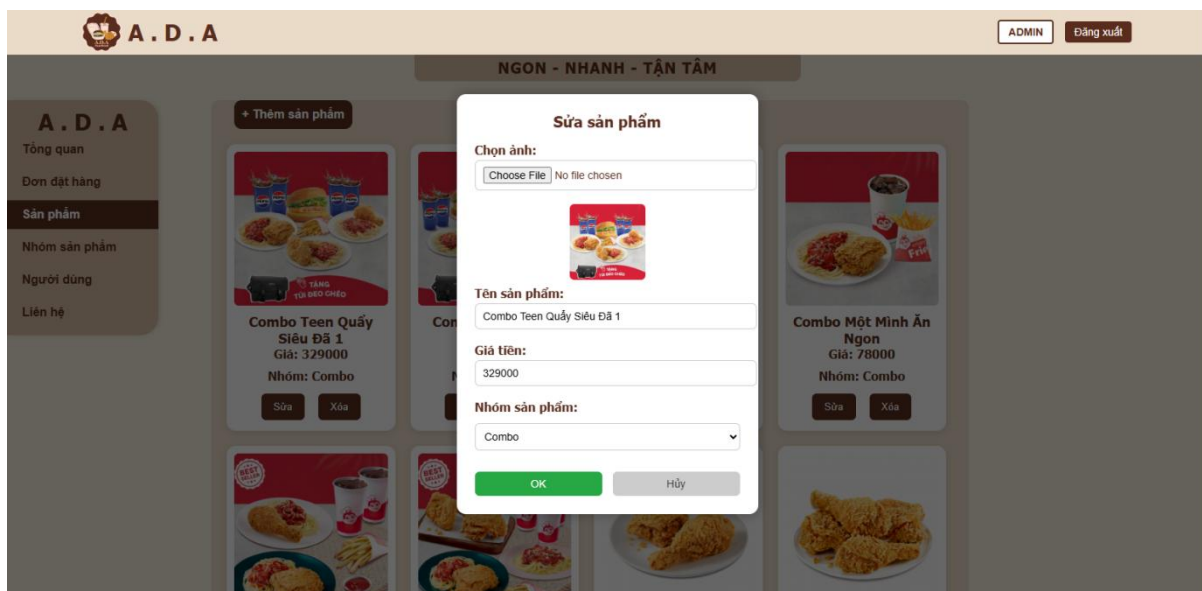
Hình 3.23. Giao diện menu Sản phẩm

Thêm sản phẩm bằng cách nhấn vào nút “+ Thêm sản phẩm” và nhập đầy đủ các thông tin như: chọn ảnh, tên sản phẩm, giá và nhóm sản phẩm. Nhấn nút “Lưu” để thêm sản phẩm vào danh sách và sẽ được hiển thị ở “Trang chủ”.



Hình 3.24 Overlay thêm Sản phẩm

Để sửa thông tin sản phẩm, nhấn vào nút “Sửa” để thay đổi thông tin sau đó nhấn nút “Ok” để lưu.



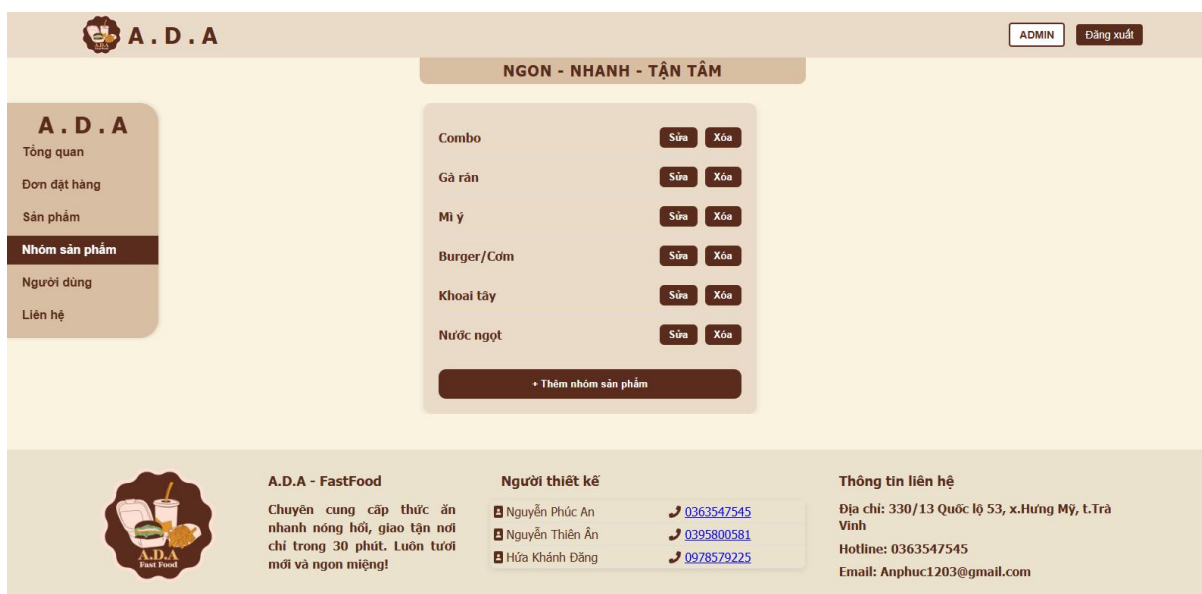
Hình 3.25. Overlay sửa Sản phẩm

Để xóa sản phẩm, nhấn vào nút “Xóa” để xóa.

3.4.9.4 Giao diện menu Nhóm sản phẩm

Trang này cung cấp giao diện quản lý toàn bộ nhóm sản phẩm trong hệ thống. Admin có thể xem danh sách các nhóm sản phẩm và có thể thêm, sửa và xóa sản phẩm.

Chọn “+ Thêm nhóm sản phẩm” => ghi tên nhóm sản phẩm => Chọn “Ok” để thêm nhóm sản phẩm.



Hình 3.26 Giao diện menu Nhóm sản phẩm

3.4.9.5 Giao diện menu Người dùng

Trang này hiển thị danh sách tất cả người dùng đã đăng ký trong hệ thống, cho phép admin xem thông tin chi tiết của từng tài khoản.

Tuân thủ GDPR: Thiết kế theo nguyên tắc "chỉ hiển thị thông tin tối thiểu cần thiết". Password của các users được ẩn hoàn toàn để đảm bảo quyền riêng tư và bảo mật cho khách hàng.

Đảm bảo cân bằng giữa nhu cầu quản lý của admin và quyền riêng tư của khách hàng, tuân thủ các nguyên tắc bảo mật thông tin cá nhân hiện đại.



The screenshot displays the 'A.D.A.' Admin Dashboard. At the top, there's a header with the logo, the text 'A.D.A.', and buttons for 'ADMIN' and 'Đăng xuất'. Below the header, a navigation menu on the left includes 'Tổng quan', 'Đơn đặt hàng', 'Sản phẩm', 'Nhóm sản phẩm', 'Người dùng' (selected), and 'Liên hệ'. The main content area is titled 'NGON - NHANH - TẬN TÂM' and features a table of users.

Tên đăng nhập	Email	Số điện thoại	Địa chỉ	Quyền
thienan	thienan@gmail.com	0363547540	Phước Hảo	user
admin	anphuc1203@gmail.com			admin
khanhdang	khanhdang@gmail.com	0363547541	Hưng Mỹ	user
nguoixinhdepnhaththegian	gian@gmail.com	0363547545	Vinh Long	user

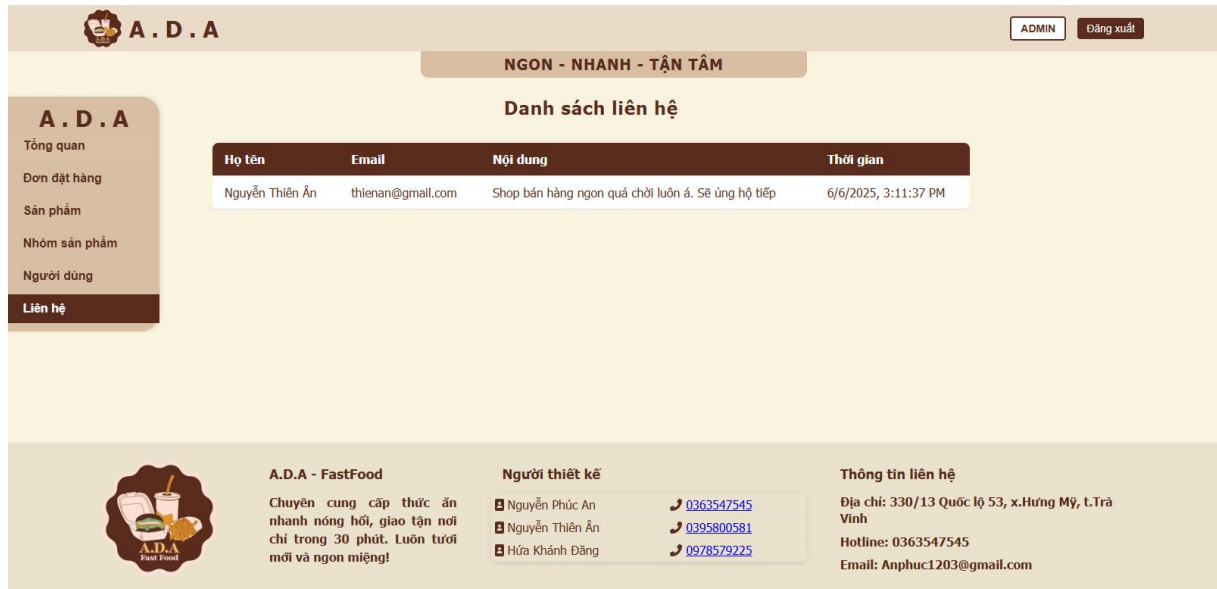
At the bottom of the dashboard, there's a footer section with the A.D.A. logo, contact information for 'A.D.A - FastFood', a list of designers with their contact details, and general contact information including address, hotline, and email.

Hình 3.27 Giao diện menu Người dùng

3.4.9.6 Giao diện menu Liên hệ

Trang này hiển thị danh sách tất cả tin nhắn liên hệ mà khách hàng đã gửi từ trang Liên hệ, giúp admin theo dõi và phản hồi các yêu cầu hỗ trợ.

Có vai trò quan trọng trong việc duy trì mối quan hệ khách hàng và đảm bảo không có tin nhắn nào bị bỏ sót trong quá trình hỗ trợ.



A . D . A

ADMIN Đăng xuất

NGON - NHANH - TẬN TÂM

A . D . A

Tổng quan
Đơn đặt hàng
Sản phẩm
Nhóm sản phẩm
Người dùng
Liên hệ

Danh sách liên hệ

Họ tên	Email	Nội dung	Thời gian
Nguyễn Thiên Ân	thienan@gmail.com	Shop bán hàng ngon quá chờ luôn á. Sẽ ủng hộ tiếp	6/6/2025, 3:11:37 PM

A.D.A - FastFood
Chuyên cung cấp thức ăn nhanh nóng hổi, giao tận nơi chỉ trong 30 phút. Luôn tươi mới và ngon miệng!

Người thiết kế
 Nguyễn Phúc An ☎ 0363547545
 Nguyễn Thiên Ân ☎ 0395800581
 Hứa Khánh Đăng ☎ 0978579225

Thông tin liên hệ
 Địa chỉ: 330/13 Quốc lộ 53, x.Hưng Mỹ, t.Trà Vinh
 Hotline: 0363547545
 Email: Anphuc1203@gmail.com

Hình 3.28 Giao diện menu Liên hệ

CHƯƠNG 4: TRIỂN KHAI VÀ CÔNG NGHỆ SỬ DỤNG

4.1 Danh sách công nghệ sử dụng

Dự án "Hệ thống quản lý bán hàng Ada FastFood" được phát triển dựa trên các công nghệ chính, được gom nhóm theo vai trò và chức năng để đảm bảo hiệu quả trong thiết kế, triển khai và quản lý hệ thống:

- Nhóm công nghệ giao diện và frontend

ReactJS: Thư viện JavaScript hàng đầu dùng để xây dựng giao diện người dùng (frontend) theo mô hình Single Page Application (SPA). Nó hỗ trợ tạo các thành phần (component) tái sử dụng, như trang danh sách sản phẩm, overlay đặt hàng, và giao diện giỏ hàng, với quản lý trạng thái qua hooks (useState, useEffect). Điều này đảm bảo phản hồi nhanh khi người dùng thêm sản phẩm vào giỏ hàng hoặc cập nhật số lượng, nâng cao trải nghiệm người dùng. FontAwesome được tích hợp để cung cấp các biểu tượng trực quan cho giao diện.

- Nhóm công nghệ backend và API

Node.js/ExpressJS: Node.js là môi trường chạy JavaScript phía server, kết hợp ExpressJS để phát triển backend. ExpressJS cung cấp các API RESTful, xử lý logic như xác thực người dùng (JWT), quản lý sản phẩm, đơn hàng, và tính toán tổng tiền. Nhóm đã tích hợp middleware để kiểm tra quyền (admin/customer), xử lý upload file, CORS handling, và error handling, đảm bảo các endpoint như /api/products, /api/orders hoạt động ổn định. Swagger được sử dụng để tự động tạo documentation cho API..

- Nhóm công nghệ cơ sở dữ liệu

+ MongoDB: Hệ quản trị cơ sở dữ liệu NoSQL, lưu trữ dữ liệu cốt lõi như thông tin người dùng, sản phẩm, danh mục, đơn hàng và liên hệ. MongoDB hỗ trợ lưu trữ linh hoạt với embedded documents, ví dụ order items được lưu trực tiếp trong orders collection, với thiết kế collections như users, products, categories, orders, contact.

+ Mongoose ODM: Object Document Mapping cho Node.js, đơn giản hóa tương tác với MongoDB. Nhóm dùng Mongoose để định nghĩa schema và validation, đồng thời tự động tạo ObjectId và timestamps cho các documents.

- Nhóm công nghệ triển khai và container hóa

Docker: Công cụ container hóa đóng gói ứng dụng thành các container (React frontend, Express backend, MongoDB), tách biệt môi trường phát triển và triển khai. Docker Compose được sử dụng để quản lý và khởi tạo multi-container application. Điều này đảm bảo tính nhất quán, cho phép hệ thống chạy giống nhau trên các máy chủ, đặc biệt hữu ích khi tích hợp với đám mây.

- Nhóm công nghệ quản lý mã nguồn và làm việc nhóm

Git/GitHub: Công cụ quản lý mã nguồn phân tán, giúp theo dõi thay đổi, quản lý phiên bản, và hợp tác nhóm. GitHub lưu trữ kho mã nguồn với repository "CongNghePhanMem", hỗ trợ branch và collaboration, đảm bảo quy trình phát triển chuyên nghiệp.

4.2 Quy trình CI/CD với GitHub Actions

CI/CD, viết tắt của Continuous Integration/Continuous Delivery (hoặc Continuous Deployment), là một phương pháp phát triển phần mềm giúp tự động hóa quá trình tích hợp, kiểm thử và triển khai mã nguồn. Mục tiêu chính là tăng tốc độ phát hành phần mềm, giảm lỗi và cải thiện tính ổn định của sản phẩm.

Nhóm đã triển khai quy trình CI đơn giản bằng GitHub Actions để tự động hóa một số bước cơ bản. Cụ thể, hệ thống tự động build mã nguồn mỗi khi có commit mới, giúp phát hiện lỗi sớm. Quá trình này bao gồm:

- Tự động build: Biên dịch mã nguồn frontend (React) và backend (Node.js).
- Test code: Chạy các kiểm thử đơn vị cơ bản để đảm bảo tính toàn vẹn.
- Docker build: Kiểm tra khả năng containerization của ứng dụng.

Do giới hạn thời gian, quy trình CD (Continuous Deployment) chưa được triển khai đầy đủ, nhưng CI đã hỗ trợ nhóm tối ưu hóa quy trình phát triển và đảm bảo chất lượng code.

4.3 Cấu hình Docker và quy trình triển khai ứng dụng

4.3.1 Cấu hình Docker

Ứng dụng "Hệ thống quản lý bán hàng Ada FastFood" được container hóa bằng Docker, với cấu hình được định nghĩa trong tệp `docker-compose.yml`. Cấu hình này bao gồm ba container chính::

- Frontend Container: Chạy React SPA, phục vụ giao diện người dùng với các trang như menu sản phẩm, giỏ hàng, và đặt hàng, truy cập tại `localhost:3000`.

- Backend Container: Chạy ExpressJS backend với Node.js, xử lý các API RESTful (`/api/products`, `/api/orders`, `/api/categories`, `/api/users`, `/api/contact`, `/api/overview`) và kết nối với cơ sở dữ liệu, chạy trên port 5000.

- Database Container: Chạy MongoDB, lưu trữ dữ liệu hệ thống như thông tin người dùng, sản phẩm, danh mục, đơn hàng và tin nhắn liên hệ trong các collections tương ứng, với persistent volume để đảm bảo dữ liệu không bị mất.

Các container được kết nối thông qua Docker network bridge, cho phép backend kết nối đến MongoDB thông qua connection string `'mongodb://mongo:27017/ada_fastfood'`. Static file serving cho hình ảnh sản phẩm được mount từ thư mục `uploads` của host vào container backend.

4.3.2 Quy trình triển khai

Dockerize ứng dụng: Nhóm đã đóng gói ứng dụng thành các container bằng Docker. Tệp `docker-compose.yml` định nghĩa các services (frontend, backend, mongo) với các port mappings và dependencies (backend phụ thuộc vào mongo). Lệnh `'docker-compose up --build'` được sử dụng để xây dựng và khởi động các container, đảm bảo môi trường phát triển nhất quán trên các máy khác nhau.

Triển khai và chạy web: Sau khi build thành công, ứng dụng được chạy cục bộ trên máy chủ tại địa chỉ `localhost:3000` cho frontend và `localhost:5000` cho backend API. Người dùng có thể truy cập giao diện web để xem menu sản phẩm, thêm vào giỏ hàng, và đặt hàng. Admin có thể truy cập trang quản trị để quản lý sản phẩm, đơn hàng và xem thống kê. API documentation có thể truy cập tại `localhost:5000/api-docs` thông qua Swagger UI.

Triển khai lên host: Hiện tại, ứng dụng đã được triển khai thành công trên máy chủ cục bộ với MongoDB persistent storage.

Volumes và Persistence: Hệ thống sử dụng Docker volumes để lưu trữ dữ liệu MongoDB và thư mục uploads chứa hình ảnh sản phẩm, đảm bảo dữ liệu không bị mất khi restart containers.

CHƯƠNG 5: QUẢN LÝ DỰ ÁN

5.1 Sử dụng Jira để lập kế hoạch và theo dõi tiến độ

5.1.1 Thiết lập Jira Board

Trong suốt quá trình phát triển dự án, nhóm chúng tôi sử dụng Jira Software làm công cụ chính để lập kế hoạch, phân chia công việc và theo dõi tiến độ theo mô hình Scrum để quản lý dự án "Thiết kế và triển khai website bán gà rán tích hợp Docker và CI/CD theo mô hình Restful".

Ngay từ đầu dự án, chúng tôi đã tạo một Scrum board trong Jira để chia dự án thành 5 Sprint kéo dài với thời gian chủ động. Mỗi Sprint được lập kế hoạch dựa trên mục tiêu rõ ràng, gắn liền với tiến độ phát triển sản phẩm.

Các Sprint được đặt tên theo thứ tự: Sprint 1, Sprint 2,...Sprint 5. Mỗi Sprint bao gồm các User Story mô tả tính năng hoặc công việc cụ thể.

Mỗi User Story được gắn nhãn rõ ràng (UI, Backend, API, Docker, CI/CD,...) và gắn với thành viên phụ trách.

Jira cho phép nhóm quản lý trạng thái công việc với các cột:

- To Do: Các nhiệm vụ chưa bắt đầu.
- In Progress: Nhiệm vụ đang thực hiện.
- Code Review / Test: Chờ kiểm tra hoặc phản hồi.
- Done: Đã hoàn thành.

Cách sử dụng này giúp các thành viên dễ dàng nắm bắt tình hình tổng thể và điều chỉnh tốc độ làm việc khi cần thiết.

5.1.2 Quy trình theo dõi tiến độ

Chúng tôi tổ chức họp đầu Sprint và cuối Sprint để:

- Thảo luận tiến độ, khó khăn gặp phải.
- Đánh giá khối lượng công việc đã hoàn thành.
- Lập kế hoạch cho Sprint tiếp theo.

5.1.3 Metrics và Báo cáo

Jira cung cấp các báo cáo quan trọng:

- Burndown Chart: Theo dõi tiến độ hoàn thành Sprint
- Velocity Chart: Đo lường năng suất nhóm qua các Sprint
- Sprint Report: Tổng kết chi tiết từng Sprint

5.2 Phân công nhiệm vụ của các thành viên trong nhóm

Dự án được chia làm nhiều mảng (UI/UX, Frontend, Backend, DevOps), mỗi thành viên đảm nhận các nhiệm vụ phù hợp với khả năng và kinh nghiệm.

Nguyễn Phúc An - Full-stack Developer & Database Architect:

- Thiết kế wireframe và schema database
- Phát triển backend API và authentication
- Xây dựng giao diện admin và private routes
- Thiết lập CI/CD pipeline

Nguyễn Thiên Ân - UI/UX Researcher & QA Tester:

- Nghiên cứu UI/UX của các website tương tự
- Phát triển giao diện trang menu và tìm kiếm
- Quản lý admin backend integration
- Viết test case và báo cáo

Hứa Khánh Đăng - Frontend Developer & DevOps:

- Quản lý Figma workspace và thiết kế UI
- Phát triển giao diện user (giỏ hàng, đặt hàng, giới thiệu)
- Containerization với Docker
- Chuẩn bị slide thuyết trình

Trong quá trình làm việc, các thành viên cũng thường xuyên hỗ trợ chéo khi cần thiết, đặc biệt trong các Sprint gấp hoặc có yêu cầu gấp từ người dùng thử.

Nhờ sử dụng Jira một cách hiệu quả và phân công công việc rõ ràng, nhóm đã hoàn thành đúng tiến độ, đảm bảo chất lượng sản phẩm. Việc ứng dụng phương pháp Scrum giúp nhóm linh hoạt hơn trong xử lý thay đổi và đảm bảo tính liên tục trong phát triển phần mềm.

CHƯƠNG 6: KIỂM THỬ

6.1 Chiến lược kiểm thử và công cụ sử dụng

6.1.1 Chiến lược kiểm thử tổng thể

Dự án "Thiết kế và triển khai website bán gà rán tích hợp Docker và CI/CD theo mô hình Restful" áp dụng chiến lược kiểm thử đa tầng để đảm bảo chất lượng và độ tin cậy của hệ thống:

Testing Pyramid Strategy:

- Unit Testing (30%): Kiểm thử các function và component riêng lẻ
- Integration Testing (50%): Kiểm thử API endpoints và database interaction
- End-to-End Testing (20%): Kiểm thử user workflow hoàn chỉnh

Test Coverage Target:

- Backend API: 90% cho các chức năng quan trọng
- Frontend Components: 70% cho các logic nghiệp vụ
- Database Operations: 100% cho các hoạt động CRUD

6.1.2 Công cụ kiểm thử

Postman - API Testing:

- Kiểm thử tất cả RESTful API endpoints
- Automation testing với collection runner
- Environment variables để switch giữa dev/prod
- Pre-request scripts và test assertions

GitHub Actions - CI/CD Testing:

Dự án đã thiết lập pipeline CI/CD tự động với file "simple-test.yml", nhằm đảm bảo mọi thay đổi code đều được kiểm tra tự động trước khi merge, giúp phát hiện lỗi sớm và duy trì chất lượng code ổn định. Dưới đây là đoạn mã:

```
name: Simple Tests
```

on:

push:

branches: [main, PhucAn]

pull_request:

branches: [main]

jobs:

test-all:

runs-on: ubuntu-latest

steps:

- name: Checkout

uses: actions/checkout@v3

- name: Setup Node.js

uses: actions/setup-node@v3

with:

node-version: '18'

- name: Install and test backend

run: |

if [-d "backend"] && [-f "backend/package.json"]; then

echo "Testing backend..."

cd backend

npm install

npm test -- --passWithNoTests || echo "Backend test completed"

```
cd ..

fi

- name: Install and test frontend

run: |

    if [ -d "frontend" ] && [ -f "frontend/package.json" ]; then

        echo "Testing frontend..."

        cd frontend

        npm install

        npm test -- --watchAll=false --passWithNoTests || echo "Frontend test
completed"

        npm run build || echo "Build completed"

    fi
```

6.2 Kết quả kiểm thử

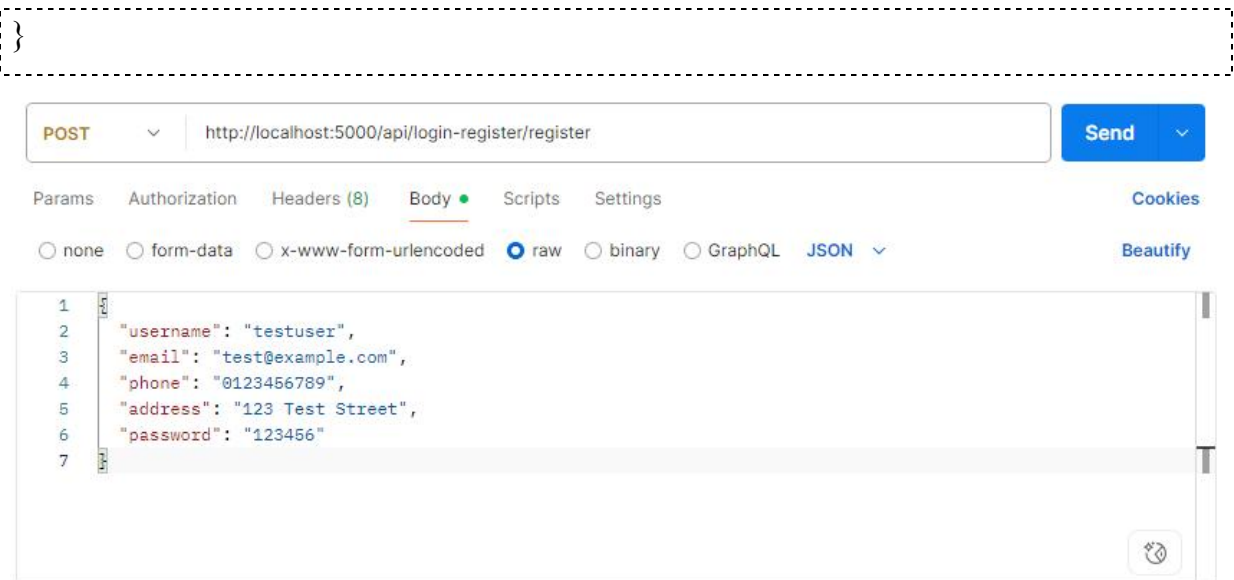
Dưới đây là một số kết quả khi kiểm thử bằng Postman

6.2.1 Đăng ký người dùng mới

- Phương thức: POST
- URL: <http://localhost:5000/api/login-register/register>
- Request (yêu cầu):

```
{

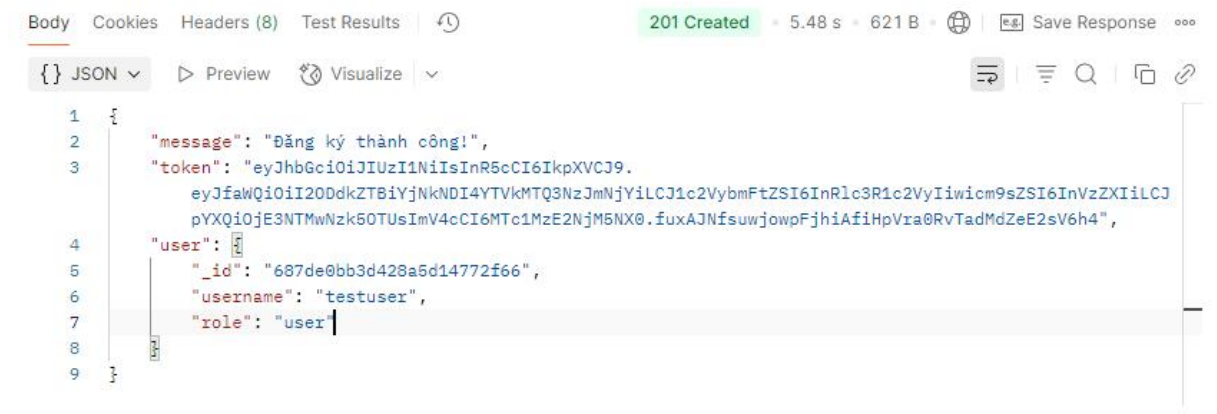
  "username": "testuser",
  "email": "test@example.com",
  "phone": "0123456789",
  "address": "123 Test Street",
  "password": "123456"
```



Hình 6.1 Request gửi dữ liệu đăng ký

- Response (phản hồi):

```
{
  "message": "Đăng ký thành công!",
  "token": "eyJhbGciOiJIUzI1NiIsInR5cGE6IjoiIn0...",
  "user": {
    "_id": "687de0bb3d428a5d14772f66",
    "username": "testuser",
    "role": "user"
  }
}
```

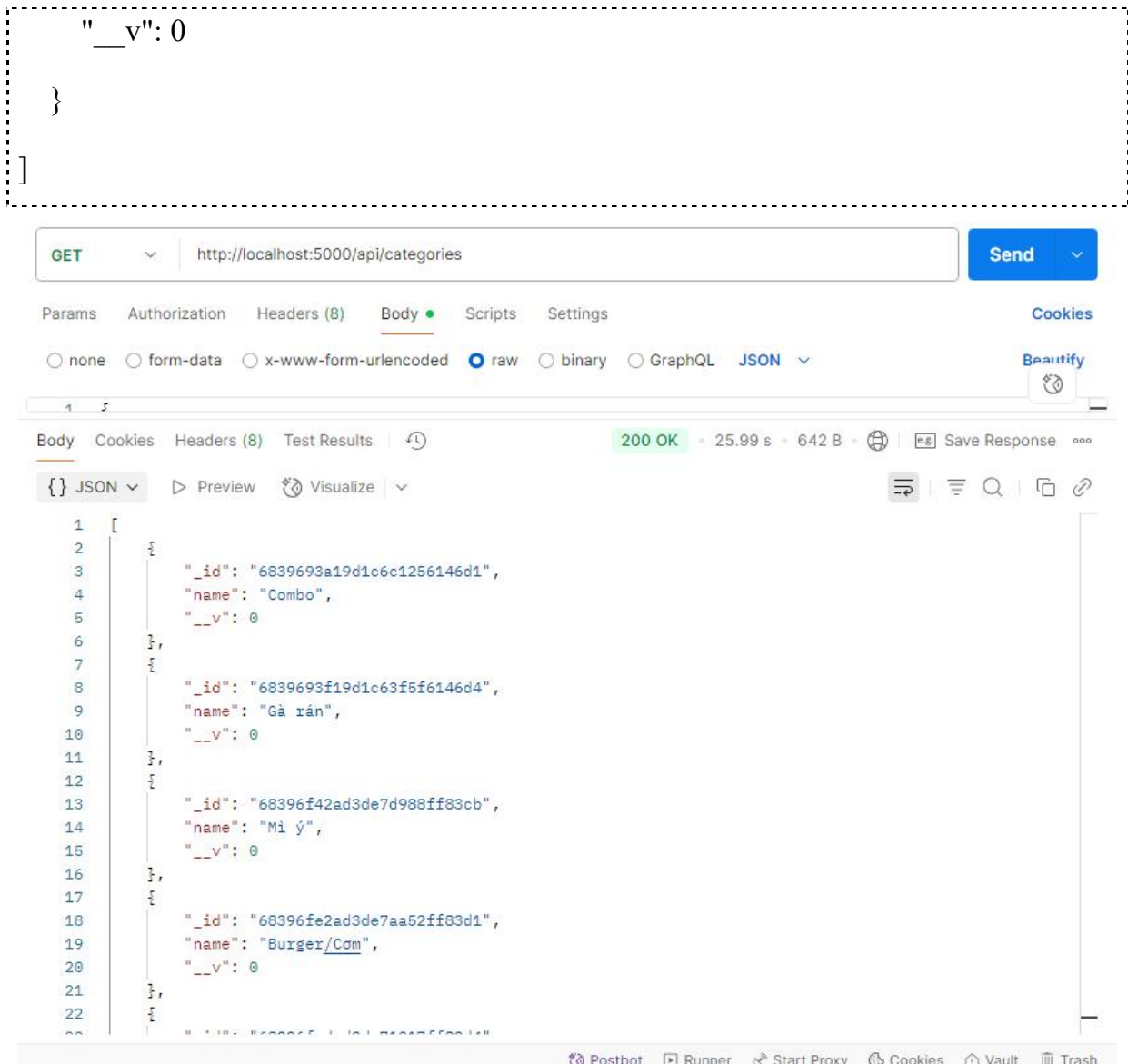
Hình 6.2 Response khi đăng ký thành công

6.2.2 Lấy danh sách Nhóm sản phẩm

- Phương thức: POST
- URL: <http://localhost:5000/api/categories>
- Response:

```

[
  {
    "_id": "6839693a19d1c6c1256146d1",
    "name": "Combo",
    "__v": 0
  },
  {
    "_id": "6839693f19d1c63f5f6146d4",
    "name": "Gà rán",
    "__v": 0
  },
  ...
  {
    "_id": "6839724bad3de73c77ff842d",
    "name": "Nước ngọt",
    ...
  }
]
    
```



Hình 6.3 Response khi lấy danh sách phim đang chiếu thành công

6.2.3 Cập nhật thông tin các nhân (khách hàng)

- Phương thức: PUT
- URL: `http://localhost:5000/api/login-register/profile`
- Header:

Content-Type: application/json

Authorization: Bearer YOUR_JWT_TOKEN

- Request (chỉ cập nhật email, phone và address):

```
{
  "email": "testuser_updated@example.com",

```

```

"phone": "0987654321",

"address": "456 New Address, Ho Chi Minh City"

}
    
```



Hình 6.4 Request cập nhật thông tin một người dùng

- Response:

```

{
  "message": "Cập nhật thông tin thành công!",
  "user": {
    "role": "user",
    "_id": "687de0bb3d428a5d14772f66",
    "username": "testuser",
    "email": "testuser_updated@example.com",
    "phone": "0987654321",
    "address": "456 New Address, Ho Chi Minh City",
    "createdAt": "2025-07-21T06:39:55.165Z",
    "__v": 0
  }
}
    
```



Hình 6.5 Response sau khi cập nhật thành công

CHƯƠNG 7: ĐÁNH GIÁ VÀ KẾT LUẬN

7.1 Những khó khăn gặp phải trong quá trình thực hiện

7.1.1 Khó khăn về công nghệ

- Học công nghệ mới: Nhóm gặp khó khăn khi làm quen với MongoDB vì trước đó chỉ biết MySQL. Cách thiết kế cơ sở dữ liệu NoSQL khác hoàn toàn so với SQL, đặc biệt là việc liên kết dữ liệu giữa các bảng (collections). Ngoài ra, việc xây dựng API với JWT authentication và phân quyền admin/user cũng tốn nhiều thời gian để research và implement.

- Quản lý state trong React: Không sử dụng Redux nên việc quản lý trạng thái giỏ hàng, thông tin đăng nhập và dữ liệu admin trở nên phức tạp. Phải truyền props qua nhiều component, gây ra code rối và khó maintain.

- Upload và xử lý file: Việc upload hình ảnh sản phẩm gặp nhiều vấn đề về validation, resize ảnh và security. Phải học thêm multer middleware và cách serve static files an toàn.

7.1.2 Khó khăn về quy trình làm việc nhóm

- Conflict khi merge code: Do làm việc song song trên các tính năng khác nhau nên thường xuyên bị conflict khi merge code. Team mất nhiều thời gian để resolve conflicts vì chưa thành thạo Git.

- Đồng bộ giữa frontend và backend: Frontend phải chờ backend hoàn thành API mới test được, hoặc phải tạo fake data để làm việc song song. Điều này gây delay và phải refactor code nhiều lần.

- Chia công việc chưa hợp lý: Ban đầu phân công chưa rõ ràng, dẫn đến duplicate work hoặc miss requirement. Sau này mới cải thiện bằng cách sử dụng Jira để track tasks chi tiết hơn.

7.2 Bài học rút ra

- Phương pháp Scrum với 5 sprint giúp nhóm có mục tiêu rõ ràng từng giai đoạn, daily standup meetings phát hiện vấn đề sớm và hỗ trợ kịp thời. Documentation với Swagger và Postman collections tạo điều kiện cho team members làm việc độc lập hiệu quả. Quy trình code review bắt buộc không chỉ giúp phát hiện lỗi sớm mà còn tạo cơ hội học hỏi và chia sẻ kiến thức giữa các thành viên.

- MERN stack (MongoDB, Express, React, Node.js) tỏ ra phù hợp với phương pháp agile development nhờ tính linh hoạt và khả năng thay đổi nhanh chóng. Các vấn đề bảo mật như JWT authentication, mã hóa mật khẩu và validation input cần được triển khai ngay từ đầu dự án thay vì bổ sung sau. Chiến lược testing cần chuyển từ manual testing sang automated testing để đảm bảo chất lượng code và tự tin khi refactor.

7.3 Đề xuất trong tương lai

- Triển khai Redux để quản lý trạng thái ứng dụng hiệu quả hơn, xây dựng hệ thống kiểm thử tự động với unit test và integration test. Tối ưu hiệu suất thông qua đánh chỉ mục cơ sở dữ liệu, tối ưu hóa hình ảnh và bộ nhớ đệm. Nâng cao bảo mật với đăng nhập OAuth, giới hạn tần suất truy cập và triển khai HTTPS.

- Phát triển theo dõi đơn hàng thời gian thực, tích hợp cổng thanh toán trực tuyến, hệ thống quản lý kho hàng tự động và trải nghiệm ứng dụng web di động (PWA) để phục vụ khách hàng tốt hơn.

- Hoàn thiện pipeline CI/CD tự động hóa, áp dụng các công cụ kiểm tra chất lượng code như ESLint và Prettier, xây dựng tài liệu hướng dẫn chi tiết cho người dùng và nhà phát triển nhằm đảm bảo tính bền vững và khả năng mở rộng của hệ thống.

CHƯƠNG 8: PHỤ LỤC

8.1 Hướng dẫn cài đặt và chạy ứng dụng

8.1.1 Yêu cầu hệ thống

Phần mềm cần thiết:

- Node.js version 18.x trở lên
- MongoDB version 5.0 trở lên
- Git version 2.30 trở lên
- Docker và Docker Compose (tùy chọn)

Hệ điều hành hỗ trợ:

- Windows 10/11
- macOS 10.15 trở lên
- Ubuntu 18.04 trở lên

8.1.2 Cài đặt thủ công (Manual Setup)

Bước 1: Clone repository

```
git clone https://github.com/DanghwaPhuocHao/CongNghePhanMem.git  
cd CongNghePhanMem
```

Bước 2: Cài đặt Backend

```
cd backend  
npm install
```

Bước 3: Cấu hình Backend Tạo file .env trong thư mục backend:

```
PORT=5000  
  
MONGODB_URI=mongodb://localhost:27017/ada_fastfood  
  
JWT_SECRET=ada_fastfood_secret_key_2025  
  
NODE_ENV=development
```

Bước 4: Khởi động MongoDB

```
# Windows (MongoDB Service)

net start MongoDB

# macOS (Homebrew)

brew services start mongodb-community

# Ubuntu

sudo systemctl start mongod
```

Bước 5: Chạy Backend

```
cd backend

npm start

# Backend sẽ chạy tại http://localhost:5000

# API Documentation tại http://localhost:5000/api-docs
```

Bước 6: Cài đặt Frontend

```
cd frontend

npm install

npm start

# Frontend sẽ chạy tại http://localhost:3000
```

8.1.3 Cài đặt bằng Docker (Khuyến nghị)

Bước 1: Clone repository

```
git clone https://github.com/DanghwaPhuocHao/CongNghePhanMem.git

cd CongNghePhanMem
```

Bước 2: Chạy bằng Docker Compose

```
docker-compose up --build
```


Các service sẽ chạy tại:

- Frontend: <http://localhost:3000>
- Backend API: <http://localhost:5000>
- API Documentation: <http://localhost:5000/api-docs>
- MongoDB: localhost:27017

Bước 3: Dừng ứng dụng

```
docker-compose down
```

8.1.4 Tài khoản mặc định

Admin Account:

- Username: admin
- Password: 123
- Role: admin (quản lý toàn bộ hệ thống)

8.1.5 Kiểm tra cài đặt

Kiểm tra Backend API:

```
curl http://localhost:5000/api/products
```

```
# Hoặc truy cập Swagger UI: http://localhost:5000/api-docs
```

Kiểm tra Frontend:

- Truy cập: <http://localhost:3000>
- Đăng nhập bằng tài khoản test
- Thử thêm sản phẩm vào giỏ hàng

Kiểm tra Database:

```
# MongoDB Shell
```

```
mongosh
```

```
use ada_fastfood
```

```
db.users.find()
```

8.2 Liên kết GitHub repository và link demo

8.2.1 GitHub Repository

Repository chính:

<https://github.com/PhucAnDangKhoa/CongNghePhanMem>

Cấu trúc repository:

<https://github.com/DanghuaPhuocHao/CongNghePhanMem/tree/main>

Cấu trúc repository:

CongNghePhanMem/

```
├── backend/           # Node.js Express Server
│   ├── src/
│   │   ├── api/       # API routes (29 endpoints)
│   │   │   ├── login-register.js
│   │   │   ├── products.js
│   │   │   ├── categories.js
│   │   │   ├── orders.js
│   │   │   ├── users.js
│   │   │   ├── contact.js
│   │   │   └── overview.js
│   │   ├── models/    # Mongoose schemas
│   │   ├── middleware/ # Auth & validation
│   │   └── utils/      # Helper functions
│   ├── uploads/       # Product images
│   ├── package.json
│   └── server.js
└── frontend/         # React Application
```

```
| |— src/
| | |— components/  # Reusable components
| | |— pages/       # Page components
| | |— contexts/    # State management
| | |— utils/       # Frontend utilities
| |— public/
| |— package.json
|— .github/
| |— workflows/
| | |— simple-test.yml # CI/CD Pipeline
|— docker-compose.yml  # Container setup
|— README.md
|— documentation/     # Project docs
```

8.2.2 Demo Links

Hiện tại dự án chạy local:

Frontend: <http://localhost:3000>

Backend API: <http://localhost:5000>

API Documentation: <http://localhost:5000/api-docs>

TÀI LIỆU THAM KHẢO

- [1] "Database schema là gì? Cách tổ chức lược đồ cơ sở dữ liệu tối ưu," 11 7 2024. [Online]. Available: <https://itviec.com/blog/database-schema-luoc-do-co-so-du-lieu/>. [Accessed 29 5 2025].
- [2] "CI/CD," 17 5 2021 [Online]. Available: <https://viblo.asia/p/cau-hinh-cicd-voi-github-phan-2-tao-mot-workflow-bang-git-actions-Az45bLrLZxY>. [Accessed 29 6 2025].
- [3] "ChatGPT," [Online]. Available: <https://chatgpt.com/>. [Accessed 29 5 2016].
- [4] "Tìm hiểu về json web token (JWT)," 23 11 2010. [Online]. Available: <https://viblo.asia/p/tim-hieu-ve-json-web-token-jwt-7rVRqp73v4bP>. [Accessed 10 6 2025].
- [5] "RESTful API là gì ?," 09 5 2020. [Online]. Available: <https://viblo.asia/p/restful-api-la-gi-1Je5EDJ4lnL>. [Accessed 28 5 2025].