

TRƯỜNG ĐẠI HỌC TRÀ VINH
TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO KẾT THÚC MÔN
MÔN: CÔNG NGHỆ PHẦN MỀM

HỆ THỐNG CỦA HÀNG BÁN ĐỒ ĂN
TRỰC TUYẾN

Giảng viên hướng dẫn:

TS. Nguyễn Bảo Ân

Sinh viên thực hiện:

Lâm Văn Na

(110122116 – DA22TTB)

Kim Ngọc Thiện

(110122164 – DA22TTC)

Vĩnh Long, tháng 7 năm 2025

TRƯỜNG ĐẠI HỌC TRÀ VINH
TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO KẾT THÚC MÔN
MÔN: CÔNG NGHỆ PHẦN MỀM

HỆ THỐNG CỦA HÀNG BÁN ĐỒ ĂN
TRỰC TUYẾN

Giảng viên hướng dẫn:

TS. Nguyễn Bảo An

Sinh viên thực hiện:

Lâm Văn Na

(110122116 – DA22TTB)

Kim Ngọc Thiện

(110122164 – DA22TTC)

Vĩnh Long, tháng 7 năm 2025

LỜI CẢM ƠN

Chúng em xin chân thành cảm ơn thầy vì những kiến thức quý báu và sự tận tâm trong suốt khoá học "**Công nghệ phần mềm**".

Nhờ sự hướng dẫn của thầy, chúng em đã có cơ hội áp dụng lý thuyết vào thực tế qua dự án xây dựng **hệ thống website bán đồ ăn**. Từ việc thiết kế giao diện, xử lý dữ liệu đến triển khai hệ thống, tất cả đều trở nên rõ ràng và thực tiễn hơn nhờ những bài giảng của thầy.

Quá trình thực hiện dự án giúp chúng em hiểu sâu hơn về cách phát triển một phần mềm có tính ứng dụng cao. Những góp ý và định hướng từ thầy chính là nền tảng để nhóm hoàn thiện sản phẩm tốt hơn mỗi ngày.

Chúng em đặc biệt trân trọng môi trường học tập thân thiện và truyền cảm hứng mà thầy đã tạo ra, giúp quá trình học trở nên thú vị và hiệu quả.

Một lần nữa, xin gửi lời cảm ơn chân thành đến thầy vì những đóng góp và nhiệt huyết trong việc giảng dạy.

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

Trà Vinh, ngày.....tháng.....năm 2025
GIÁO VIÊN CHẤM BÁO CÁO
(ký, ghi rõ họ tên)

NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

MỤC LỤC

Chương 1. GIỚI THIỆU	6
1.1 Giới thiệu chủ đề	6
1.2 Mục tiêu của ứng dụng.....	6
1.3 Lý do chọn đề tài.....	7
1.4 Khái quát về môn học	9
1.4.1. Các khái niệm cơ bản về công nghệ phần mềm.....	9
1.4.2. Các mô hình phát triển phần mềm	9
1.4.3 Kiến trúc phần mềm dựa trên đám mây và kiến trúc Microservices	10
Chương 2. PHÂN TÍCH YÊU CẦU.....	12
2.1. Các chức năng chính của hệ thống (Functional Requirements)	12
2.1.1. Đối với người dùng thông thường (user)	12
2.1.2. Đối với quản trị viên hệ thống (admin)	12
2.2. Các yêu cầu phi chức năng (Non-functional Requirements)	13
Chương 3. THIẾT KẾ HỆ THỐNG	15
3.1. Kiến trúc tổng thể.....	15
3.2.1. Mô hình quan hệ dữ liệu ERD	17
3.2.2. Mô tả chi tiết các thực thể.....	17
3.2.3. Mô tả chi tiết các thực thể	18
3.3. Thiết kế API	19
3.3.1. Mô tả các endpoint chính.....	19
3.3.2. Cấu trúc request/response	22
3.4. Figma thiết kế giao diện (UI/UX)	26
Chương 4. TRIỂN KHAI VÀ CÔNG NGHỆ SỬ DỤNG.....	33
4.1 Các công nghệ đã sử dụng	33
4.1.1 Ngôn ngữ lập trình chính.....	33
4.1.1.1 JavaScript (ESNext).....	33
4.1.2 Frontend	34
4.1.2.1 React 18.3.1	34
4.1.2.2 Vite 5.4.19	36
4.1.2.3 Tailwind CSS 3	37
4.1.2.4 Thư viện Thành phần UI (UI Component Libraries)	38
4.1.3 Backend.....	38
4.1.3.1 Node.js với Express.js	38
4.1.3.2 Cơ sở dữ liệu	40

4.1.4 Xác thực và bảo mật.....	40
4.2 Quy trình CI/CD với GitHub Actions.....	43
4.3 Cấu hình Docker và quy trình triển khai ứng dụng	44
4.3.1 Docker.....	44
4.3.2 Quy trình triển khai ứng dụng.....	48
Chương 5. QUẢN LÝ DỰ ÁN	50
5.1 Cách sử dụng Jira để lập kế hoạch và theo dõi tiến độ	50
5.2 Phân công nhiệm vụ của từng thành viên trong nhóm.....	51
5.2.1. Khởi tạo dự án, thiết lập môi trường.....	51
5.2.2. Chức năng cơ bản.....	51
5.2.3. Hoàn thiện chức năng và UI	52
5.2.4. Testing, Fix bug, Triển khai demo.....	52
Chương 6. KIỂM THỬ	53
6.1 Chiến lược kiểm thử và công cụ sử dụng	53
6.1.1. Chiến lược kiểm thử.....	53
6.1.2. Công cụ sử dụng	53
6.2 Kết quả kiểm thử API	54
Chương 7. ĐÁNH GIÁ VÀ KẾT LUẬN	55
7.1 Những khó khăn gặp phải trong quá trình thực hiện	55
7.2 Bài học rút ra và đề xuất cải thiện trong tương lai.....	55
DANH MỤC TÀI LIỆU THAM KHẢO	56

DANH MỤC HÌNH ẢNH

Hình 3.1 Kiến trúc tổng thể của hệ thống	15
Hình 3.2 Mô hình quan hệ dữ liệu ERD	17
Hình 3.3. Request gửi dữ liệu đăng ký người dùng	23
Hình 3.4 Response khi đăng ký thành công.....	24
Hình 3.5 Request gửi dữ liệu đăng nhập.....	24
Hình 3.6 Response khi đăng nhập thành công.....	25
Hình 3.9 Giao diện trang chủ	26
Hình 3.10 Giao diện trang thực đơn.....	27
Hình 3.11 Giao diện về chúng tôi	27
Hình 3.12 Giao diện đơn hàng của tôi	28
Hình 3.13 Giao diện trang đăng nhập	28
Hình 3.14 Giao diện trang đăng ký	29
Hình 3.15 Giao diện quản trị viên.....	29
Hình 3.16 Giao diện trang quản lý sản phẩm.....	30
Hình 3.17 Giao diện quản lý đơn hàng	30
Hình 3.18 Giao diện quản lý đơn hàng	30
Hình 3.17 Giao diện quản lý đánh giá	31
Hình 3.18 Giao diện quản lý banner	31
Hình 3.19 Giao diện trang thông kê.....	31
Hình 3.20 Giao diện trang tài khoản của tôi	32

DANH MỤC TỪ VIẾT TẮT

Từ viết tắt	Diễn giải
API	Application Programming Interface – Giao diện lập trình ứng dụng
CI/CD	Continuous Integration / Continuous Deployment – Tích hợp liên tục / Triển khai liên tục
CRUD	Create, Read, Update, Delete – Tạo, Đọc, Cập nhật, Xóa
ERD	Entity Relationship Diagram – Sơ đồ thực thể quan hệ
JWT	JSON Web Token – Chuỗi mã xác thực người dùng
UI/UX	User Interface / User Experience – Giao diện người dùng / Trải nghiệm người dùng
HTTP	HyperText Transfer Protocol – Giao thức truyền siêu văn bản
JSON	JavaScript Object Notation – Định dạng dữ liệu dạng đối tượng
SQL	Structured Query Language – Ngôn ngữ truy vấn có cấu trúc
NoSQL	Not only SQL – Cơ sở dữ liệu không quan hệ
RESTful	Representational State Transfer – Kiến trúc dịch vụ web
IDE	Integrated Development Environment – Môi trường phát triển tích hợp
CSS	Cascading Style Sheets – Ngôn ngữ định kiểu
HMR	Hot Module Replacement – Thay thế mô-đun nóng (không cần reload trang)
NPM	Node Package Manager – Trình quản lý gói cho Node.js
CORS	Cross-Origin Resource Sharing – Chia sẻ tài nguyên khác nguồn

Hệ thống cửa hàng bán đồ ăn trực tuyến

CRUD	Create, Read, Update, Delete – Các thao tác cơ bản với dữ liệu
JWT	JSON Web Token – Token xác thực người dùng
ESLint	ECMAScript Lint – Công cụ kiểm tra và định dạng mã JavaScript

Chương 1. GIỚI THIỆU

1.1 Giới thiệu chủ đề

Trong thời đại hiện nay, nhu cầu ăn uống không chỉ dừng lại ở việc đáp ứng dinh dưỡng mà còn mang yếu tố tiện lợi, nhanh chóng và trải nghiệm người dùng. Sự phát triển mạnh mẽ của công nghệ thông tin và truyền thông đã tạo điều kiện thuận lợi cho việc chuyển đổi số trong lĩnh vực dịch vụ ăn uống. Các cửa hàng thực phẩm hiện đại không chỉ cạnh tranh về chất lượng món ăn, giá cả mà còn chú trọng vào khả năng phục vụ khách hàng thông qua các nền tảng trực tuyến.

Hệ thống đặt món ăn trực tuyến ra đời như một giải pháp tối ưu nhằm nâng cao trải nghiệm của người tiêu dùng, giúp tiết kiệm thời gian, giảm tải cho nhân viên và hạn chế sai sót trong quá trình phục vụ truyền thống. Người dùng có thể dễ dàng tìm kiếm các món ăn theo nhiều tiêu chí như loại món (chay, mặn, ăn vặt...), khẩu vị, giá cả hoặc theo vị trí cửa hàng gần nhất.

Đặc biệt, hệ thống còn tích hợp chức năng định vị và gợi ý thực đơn phô biến tại từng khu vực, giúp khách hàng dễ dàng lựa chọn món ăn phù hợp với nhu cầu cá nhân. Ngoài ra, các tính năng như theo dõi trạng thái đơn hàng, đặt trước giờ giao nhận, hay thanh toán không tiền mặt cũng được phát triển đồng bộ nhằm mang đến một quy trình đặt món hiện đại và chuyên nghiệp.

Tổng thể, hệ thống bán đồ ăn trực tuyến không chỉ giúp các cửa hàng nâng cao hiệu quả quản lý và vận hành mà còn góp phần xây dựng trải nghiệm dịch vụ thân thiện, tiện lợi và đáp ứng tốt nhu cầu ngày càng cao của người tiêu dùng hiện đại.

1.2 Mục tiêu của ứng dụng

Ứng dụng **đặt món ăn trực tuyến** được xây dựng nhằm mang đến một giải pháp công nghệ tiện lợi và hiện đại cho cả người dùng lẫn doanh nghiệp trong lĩnh vực ẩm thực. Với giao diện thân thiện, thao tác mượt mà và hỗ trợ đa nền tảng, người dùng có thể dễ dàng truy cập để đăng nhập, lựa chọn món ăn, thêm vào giỏ hàng và tiến hành thanh toán chỉ trong vài bước đơn giản.

Không dừng lại ở đó, hệ thống còn tích hợp đầy đủ chức năng quản lý dành cho nhân viên và quản trị viên. Người quản trị có thể kiểm soát danh sách món ăn, xử lý đơn hàng, phê duyệt đánh giá của khách hàng, cập nhật banner trang chủ và theo dõi hoạt động kinh doanh theo thời gian thực thông qua biểu đồ thống kê.

Về mặt kỹ thuật, hệ thống được tổ chức theo mô hình client/server, sử dụng RESTful API làm cầu nối giao tiếp giữa các thành phần. Phần giao diện người dùng (frontend) được phát triển bằng ReactJS, cho phép hiển thị nội dung linh hoạt, tối ưu tốc độ phản hồi và hỗ trợ các thao tác động như tìm kiếm, lọc, phân trang. Trong khi đó, backend được vận hành bởi Node.js với Express, đảm nhiệm các chức năng quan trọng như xác thực người dùng, xử lý nghiệp vụ đơn hàng, quản lý sản phẩm và phân quyền hệ thống. Toàn bộ dữ liệu được lưu trữ bằng MongoDB, với mô hình linh hoạt phù hợp với dữ liệu dạng tài liệu và có khả năng mở rộng tốt.

Để đảm bảo chất lượng và sự ổn định trong triển khai, hệ thống tích hợp quy trình CI/CD thông qua GitHub Actions, giúp kiểm thử và cập nhật phần mềm một cách tự động sau mỗi lần thay đổi mã nguồn. Đồng thời, việc sử dụng Docker cho phép đóng gói và triển khai hệ thống trong môi trường container hoá, giúp giảm thiểu các lỗi liên quan đến sự khác biệt cấu hình giữa môi trường phát triển và thực tế.

Các công cụ và công nghệ chính được sử dụng bao gồm:

- **Jira:** Quản lý tiến độ và công việc nhóm theo phương pháp Agile/Scrum.
- **Figma:** Thiết kế và chia sẻ giao diện người dùng (UI/UX).
- **GitHub:** Lưu trữ mã nguồn, kiểm soát phiên bản và cộng tác nhóm hiệu quả.
- **GitHub Actions:** Thiết lập pipeline CI/CD để tự động kiểm thử và triển khai hệ thống.
- **Postman:** Công cụ kiểm thử API, mô phỏng dữ liệu để kiểm tra logic backend.
- **Swagger:** Tài liệu hóa các API, hỗ trợ phát triển frontend và backend đồng bộ.
- **Docker:** Đóng gói các thành phần hệ thống thành container, dễ dàng triển khai và nhân bản.

1.3 Lý do chọn đề tài

Trong bối cảnh chuyển đổi số ngày càng mạnh mẽ, việc ứng dụng công nghệ vào hoạt động kinh doanh không còn là xu hướng mà đã trở thành nhu cầu tất yếu, đặc biệt trong lĩnh vực dịch vụ ăn uống. Tuy nhiên, trên thực tế, nhiều cửa hàng và quán ăn hiện nay vẫn đang vận hành theo phương pháp thủ công hoặc sử dụng các hệ thống quản lý đơn giản, thiếu tính linh hoạt và khó mở rộng. Điều này dẫn đến nhiều bất cập

nhiều: khách hàng phải chờ đợi lâu để gọi món, dễ xảy ra sai sót trong việc ghi nhận đơn hàng, quá tải vào giờ cao điểm và khó khăn trong việc kiểm soát tình trạng đơn, thực đơn hay doanh thu.

Ngoài ra, ở góc độ quản lý, các chủ cửa hàng thường gặp khó khăn trong việc giám sát hoạt động bán hàng, quản lý nhân viên, điều chỉnh thực đơn hay đánh giá hiệu quả kinh doanh nếu không có một hệ thống hỗ trợ chuyên biệt, đồng bộ và có khả năng phân quyền rõ ràng.

Nhận thấy những tồn tại trên, nhóm quyết định lựa chọn đề tài “**Xây dựng hệ thống đặt món ăn trực tuyến**” với mong muốn thiết kế và hiện thực hóa một nền tảng công nghệ hỗ trợ toàn diện cho cả người dùng và người quản lý trong hoạt động bán hàng ngành F&B (Food & Beverage). Hệ thống không chỉ đáp ứng nhu cầu đặt món nhanh chóng, thanh toán tiện lợi và theo dõi đơn hàng của khách hàng, mà còn cung cấp công cụ quản lý hiệu quả cho nhân viên và quản trị viên như: xử lý đơn, cập nhật món ăn, kiểm tra đánh giá, quản lý người dùng và theo dõi doanh thu trực quan qua biểu đồ.

Bên cạnh giá trị thực tiễn, đề tài cũng là cơ hội để nhóm áp dụng tổng hợp các kiến thức đã học trong môn **Công nghệ phần mềm**, từ phân tích và xác định yêu cầu hệ thống, thiết kế kiến trúc phần mềm, xây dựng giao diện người dùng (frontend), lập trình API (backend), kiểm thử phần mềm, cho đến triển khai ứng dụng và trình bày tài liệu dự án một cách bài bản.

Thông qua quá trình làm việc nhóm, các thành viên không chỉ trau dồi kỹ năng kỹ thuật mà còn phát triển các kỹ năng mềm cần thiết như giao tiếp, quản lý thời gian, phân chia công việc, xử lý tình huống và giải quyết vấn đề trong môi trường phát triển phần mềm thực tế.

Với tinh thần học hỏi, nghiêm túc và sáng tạo, nhóm hy vọng rằng hệ thống sẽ không chỉ là một bài tập học thuật mà còn có tiềm năng ứng dụng thực tiễn trong các mô hình kinh doanh ăn uống vừa và nhỏ, góp phần số hóa lĩnh vực dịch vụ ẩm thực trong thời đại công nghệ.

1.4 Khái quát về môn học

1.4.1. Các khái niệm cơ bản về công nghệ phần mềm

Công nghệ phần mềm (Software Engineering) là lĩnh vực kết hợp giữa khoa học máy tính và kỹ thuật phần mềm, tập trung vào việc xây dựng, phát triển, vận hành và bảo trì phần mềm theo cách có hệ thống, hiệu quả và có thể kiểm soát được. Mục tiêu chính là tạo ra các sản phẩm phần mềm chất lượng cao, đáp ứng đúng yêu cầu người dùng, có khả năng bảo trì, mở rộng và thích ứng tốt với các thay đổi trong thực tế.

Công nghệ phần mềm bao gồm ba thành phần chính:

Quy trình phát triển phần mềm (Software Process): Là chuỗi các hoạt động xuyên suốt vòng đời phần mềm, bao gồm phân tích yêu cầu, thiết kế, hiện thực (lập trình), kiểm thử và bảo trì.

Kỹ thuật phần mềm (Engineering Techniques): Gồm các phương pháp và công cụ hỗ trợ trong việc xác định yêu cầu, thiết kế hệ thống, kiểm thử phần mềm và quản lý tiến độ, rủi ro trong quá trình phát triển.

Công cụ phần mềm (Software Tools): Các công cụ hỗ trợ tự động hóa trong quy trình phát triển phần mềm như: môi trường lập trình tích hợp (IDE), hệ thống quản lý mã nguồn (Git), công cụ triển khai tự động (CI/CD)...

1.4.2. Các mô hình phát triển phần mềm

Mô hình phát triển phần mềm là các phương pháp luận tổ chức hoạt động trong quy trình xây dựng hệ thống. Mỗi mô hình có đặc trưng riêng, phù hợp với từng loại dự án và yêu cầu cụ thể.

Mô hình thác nước (Waterfall): Là mô hình tuyến tính truyền thống, gồm các bước thực hiện tuần tự từ phân tích → thiết kế → lập trình → kiểm thử → triển khai → bảo trì.

- **Ưu điểm:** Dễ hiểu, dễ quản lý tiến độ.
- **Nhược điểm:** Khó linh hoạt, không thích hợp với dự án thay đổi yêu cầu liên tục.

Mô hình chữ V (Verification & Validation): Là phiên bản mở rộng của mô hình thác nước, nhấn mạnh vào kiểm thử ở từng giai đoạn. Phù hợp với các hệ thống yêu cầu độ chính xác và độ tin cậy cao.

Mô hình xoắn ốc (Spiral): Kết hợp giữa mô hình lặp và quản lý rủi ro. Mỗi vòng lặp đại diện cho một giai đoạn phát triển, từ xác định mục tiêu → đánh giá rủi ro → thiết kế → kiểm thử.

- **Ưu điểm:** Linh hoạt, phù hợp với các dự án lớn và phức tạp.

Mô hình Agile: Là phương pháp phát triển linh hoạt, chú trọng giao tiếp nhóm, phản hồi liên tục từ khách hàng và phát triển phần mềm qua các chu kỳ ngắn gọi là sprint.

Các phương pháp phổ biến: Scrum, Kanban.

1.4.3 Kiến trúc phần mềm dựa trên đám mây và kiến trúc Microservices

1.4.3.1. Kiến trúc phần mềm dựa trên đám mây (Cloud-based Architecture)

Đây là mô hình phát triển phần mềm sử dụng nền tảng và dịch vụ từ các nhà cung cấp điện toán đám mây như Amazon Web Services (AWS), Microsoft Azure, Google Cloud... Dạng kiến trúc này mang lại nhiều lợi ích như:

- **Khả năng mở rộng tài nguyên linh hoạt (auto-scaling).**
- **Hỗ trợ triển khai liên tục thông qua CI/CD.**
- **Dễ tích hợp và phục hồi khi có sự cố.**
- **Giảm chi phí vận hành phần cứng, tăng tính sẵn sàng.**

Phần mềm có thể triển khai theo các mô hình dịch vụ như:

- **IaaS (Infrastructure as a Service)**
- **PaaS (Platform as a Service)**
- **SaaS (Software as a Service)**

1.4.3.2. Kiến trúc Microservices

Microservices là kiến trúc phần mềm hiện đại, trong đó hệ thống được chia thành các dịch vụ nhỏ, độc lập. Mỗi dịch vụ đảm nhận một chức năng riêng biệt và giao tiếp với nhau thông qua các API (thường là REST hoặc gRPC).

Đặc điểm nổi bật:

- **Tách biệt chức năng:** Mỗi dịch vụ là một module riêng biệt, dễ phát triển và kiểm soát.
- **Triển khai độc lập:** Có thể cập nhật một phần hệ thống mà không ảnh hưởng đến toàn bộ ứng dụng.

Hệ thống cửa hàng bán đồ ăn trực tuyến

- Khả năng mở rộng cao: Dễ dàng scale theo từng dịch vụ tùy vào nhu cầu sử dụng.
- Phù hợp với các hệ thống phức tạp, quy mô lớn như: thương mại điện tử, nền tảng đặt hàng trực tuyến, mạng xã hội, hệ thống thanh toán...

Chương 2. PHÂN TÍCH YÊU CẦU

2.1. Các chức năng chính của hệ thống (Functional Requirements)

Hệ thống đặt món ăn trực tuyến được thiết kế nhằm đáp ứng nhu cầu gọi món từ xa một cách nhanh chóng, tiện lợi và chính xác cho người dùng cuối, đồng thời cung cấp công cụ quản lý toàn diện cho các bên vận hành như nhân viên và quản trị viên. Các chức năng được chia thành hai nhóm chính:

2.1.1. Đối với người dùng thông thường (user)

Đăng ký và đăng nhập: Người dùng có thể tạo tài khoản cá nhân bằng thông tin cơ bản như họ tên, email và mật khẩu. Hệ thống thực hiện xác thực thông qua mã hóa mật khẩu và sử dụng token để quản lý phiên hoạt động an toàn.

Tìm kiếm và duyệt món ăn: Cho phép người dùng tìm món theo từ khóa, loại món (chay, mặn, nước, ăn vặt...), khoảng giá hoặc mức độ phổ biến. Danh sách được trình bày trực quan, hỗ trợ lọc và phân trang.

Xem thông tin chi tiết món ăn: Hiển thị mô tả, hình ảnh, thành phần, giá bán và đánh giá từ người dùng khác, giúp đưa ra quyết định lựa chọn phù hợp.

Thêm món vào giỏ hàng và tạo đơn đặt hàng: Cho phép thêm nhiều món vào giỏ, điều chỉnh số lượng và ghi chú yêu cầu riêng trước khi tiến hành đặt hàng.

Thanh toán và xác nhận đơn: Hỗ trợ nhiều hình thức thanh toán như tiền mặt, chuyển khoản hoặc ví điện tử. Hệ thống gửi thông tin xác nhận đơn sau khi hoàn tất giao dịch.

Theo dõi đơn hàng: Người dùng có thể xem tình trạng đơn theo thời gian thực, từ khi đang xử lý cho đến khi giao thành công. Cho phép hủy đơn nếu chưa được xử lý.

Đánh giá sau khi nhận món: Sau khi đơn hoàn tất, người dùng có thể để lại nhận xét, đánh giá chất lượng món ăn và trải nghiệm dịch vụ.

2.1.2. Đối với quản trị viên hệ thống (admin)

Quản lý người dùng: Cho phép xem danh sách tài khoản, phân quyền truy cập (người dùng, nhân viên, quản trị), chỉnh sửa hoặc khóa tài khoản khi cần.

Quản lý món ăn: Tạo mới, chỉnh sửa, xóa món ăn; cập nhật giá, loại món, trạng thái kinh doanh và hình ảnh minh họa.

Xử lý đơn hàng: Theo dõi tất cả các đơn hàng đang xử lý; cập nhật trạng thái theo các bước: *Chờ xác nhận → Đang xử lý → Đang giao → Hoàn thành*. Hệ thống hỗ trợ xuất đơn ra file hoặc hủy đơn nếu cần.

Kiểm soát đánh giá: Duyệt, ẩn hoặc xóa đánh giá không phù hợp. Đảm bảo các đánh giá hiển thị là xác thực và có chất lượng.

Quản lý giao diện trang chủ: Cập nhật nội dung như banner, chương trình khuyến mãi, điều chỉnh vị trí hiển thị và trạng thái hoạt động.

Thống kê – báo cáo: Hiển thị báo cáo doanh thu, số đơn theo ngày/tháng, số lượng món bán ra, các món được yêu thích... dưới dạng bảng dữ liệu hoặc biểu đồ.

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

Để hệ thống cửa hàng bán đồ ăn hoạt động hiệu quả, ổn định và dễ dàng mở rộng trong tương lai, các yêu cầu phi chức năng được đặt ra như sau:

Hiệu năng (Performance): Hệ thống cần phản hồi nhanh, đặc biệt với các thao tác như duyệt thực đơn, lọc món ăn, thêm vào giỏ hàng hoặc xác nhận đơn. Thời gian xử lý nên được giới hạn trong vòng 2 giây để đảm bảo trải nghiệm mượt mà. Ngoài ra, hệ thống phải có khả năng phục vụ nhiều khách hàng cùng lúc mà không bị gián đoạn, nhất là vào giờ cao điểm như buổi trưa và tối.

Tính sẵn sàng (Availability): Hệ thống cần duy trì trạng thái hoạt động liên tục 24/7 để đáp ứng nhu cầu đặt món mọi lúc. Cần có cơ chế tự động sao lưu dữ liệu định kỳ, giám sát lỗi và khả năng phục hồi nhanh để hạn chế thời gian gián đoạn khi có sự cố xảy ra.

Bảo mật (Security): Tài khoản người dùng được xác thực thông qua JWT (JSON Web Token). Mật khẩu cần được mã hóa bằng các thuật toán an toàn như bcrypt. Hệ thống cần trang bị các biện pháp chống lại các hình thức tấn công phổ biến như SQL Injection, XSS và CSRF. Các quyền truy cập của người dùng, nhân viên và quản trị viên cần được kiểm soát chặt chẽ theo phân quyền.

Khả năng mở rộng (Scalability): Kiến trúc hệ thống cần tách biệt frontend và backend, sử dụng chuẩn RESTful API để thuận tiện cho việc tích hợp thêm các tính năng như thanh toán trực tuyến, chatbot tư vấn món ăn, hoặc gợi ý thực đơn thông minh. Hệ thống cũng cần mở rộng linh hoạt theo số lượng người dùng hoặc chi nhánh cửa hàng.

Tính tương thích (Compatibility): Giao diện người dùng phải hiển thị tốt trên nhiều thiết bị khác nhau: điện thoại, máy tính bảng, máy tính để bàn... Đồng thời, hệ thống cần hoạt động ổn định trên các trình duyệt phổ biến như Chrome, Firefox, Edge, Safari để đảm bảo mọi khách hàng đều có thể sử dụng dễ dàng.

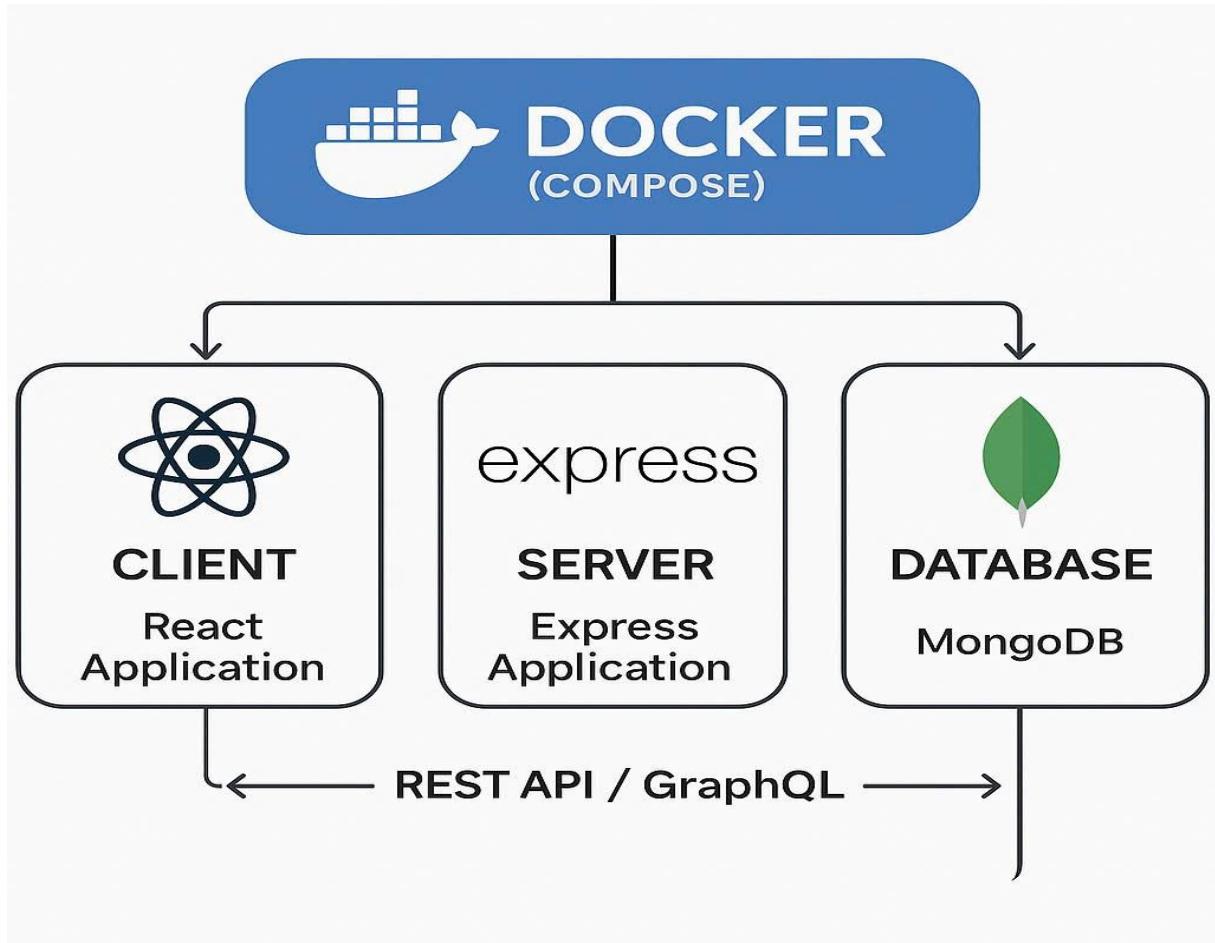
Dễ triển khai và bảo trì (Deployability & Maintainability): Việc đóng gói ứng dụng bằng Docker giúp chuẩn hóa môi trường triển khai trên nhiều nền tảng. Hệ thống tích hợp quy trình CI/CD bằng GitHub Actions để kiểm thử và triển khai tự động. Các API được tài liệu hóa bằng Swagger giúp việc phát triển frontend và bảo trì backend trở nên dễ dàng hơn.

Trải nghiệm người dùng (User Experience): Giao diện cần đơn giản, trực quan, giúp khách hàng dễ dàng tìm món, đặt hàng và thanh toán chỉ trong vài thao tác. Các thông báo như xác nhận đơn, thay đổi trạng thái giao hàng hay đánh giá món ăn cần hiển thị kịp thời để tạo cảm giác tin cậy và chuyên nghiệp.

Chương 3. THIẾT KẾ HỆ THỐNG

3.1. Kiến trúc tổng thể

Sơ đồ trên minh họa kiến trúc tổng thể của hệ thống theo mô hình client-server, trong đó các thành phần chính được tổ chức như sau:



Hình 3.1 Kiến trúc tổng thể của hệ thống

Giao diện người dùng (Frontend – React): Đây là lớp đầu tiên mà người dùng tiếp xúc, được xây dựng bằng React để đảm bảo sự linh hoạt và mượt mà khi tương tác. Tại đây, khách hàng có thể thực hiện các hành động như duyệt thực đơn, tìm món theo danh mục, thêm sản phẩm vào giỏ hàng, tiến hành đặt món, xem lịch sử đơn hàng... Mọi thao tác từ phía client sẽ được gửi đến máy chủ thông qua các lời gọi API.

API trung gian: Lớp API đóng vai trò là “cầu nối” giúp frontend và backend giao tiếp hiệu quả với nhau. Tại đây, các đường dẫn (endpoint) được định nghĩa rõ ràng để phục vụ các thao tác như đăng nhập, thêm món mới, đặt đơn hàng, cập nhật trạng thái, đánh giá món ăn... API tuân thủ nguyên tắc RESTful, giúp hệ thống dễ dàng mở rộng và tích hợp thêm các tính năng mới trong tương lai.

Xử lý nghiệp vụ (Backend – Node.js + Express): Đây là bộ não của hệ thống, nơi thực hiện tất cả các xử lý logic phía máy chủ. Khi nhận được yêu cầu từ API, server sẽ thực hiện các tác vụ như xác thực người dùng, xử lý thanh toán, lưu đơn hàng, cập nhật trạng thái vận chuyển hoặc quản lý thực đơn. Backend được xây dựng bằng Node.js với Express – một framework mạnh mẽ, nhẹ và dễ mở rộng.

Hệ quản trị dữ liệu (MongoDB): Mọi dữ liệu từ người dùng, sản phẩm, đơn hàng, phản hồi, đến các thông tin cấu hình đều được lưu trữ trong MongoDB – hệ quản trị cơ sở dữ liệu NoSQL có khả năng mở rộng linh hoạt và tốc độ xử lý nhanh. Server kết nối đến MongoDB thông qua Mongoose để thao tác dữ liệu hiệu quả, đảm bảo độ toàn vẹn và chính xác cao.

3.2 Thiết kế cơ sở dữ liệu

Dữ liệu được tổ chức theo từng nhóm chức năng để hỗ trợ vận hành cửa hàng một cách rõ ràng và hợp lý:

Bảng người dùng: Chứa thông tin cá nhân (tên, email, mật khẩu mã hóa), phân quyền (khách, nhân viên, quản trị), và trạng thái hoạt động của từng tài khoản.

Bảng sản phẩm: Lưu trữ chi tiết các món ăn đang được cung cấp như tên món, mô tả ngắn, hình ảnh minh họa, đơn giá, loại món (chay, mặn, nước, lẩu...), và trạng thái hiển thị.

Bảng đơn hàng: Ghi lại toàn bộ các đơn đã đặt, bao gồm danh sách sản phẩm, thông tin người mua, phương thức thanh toán, thời gian đặt và trạng thái đơn (đang xử lý, đang giao, hoàn tất...).

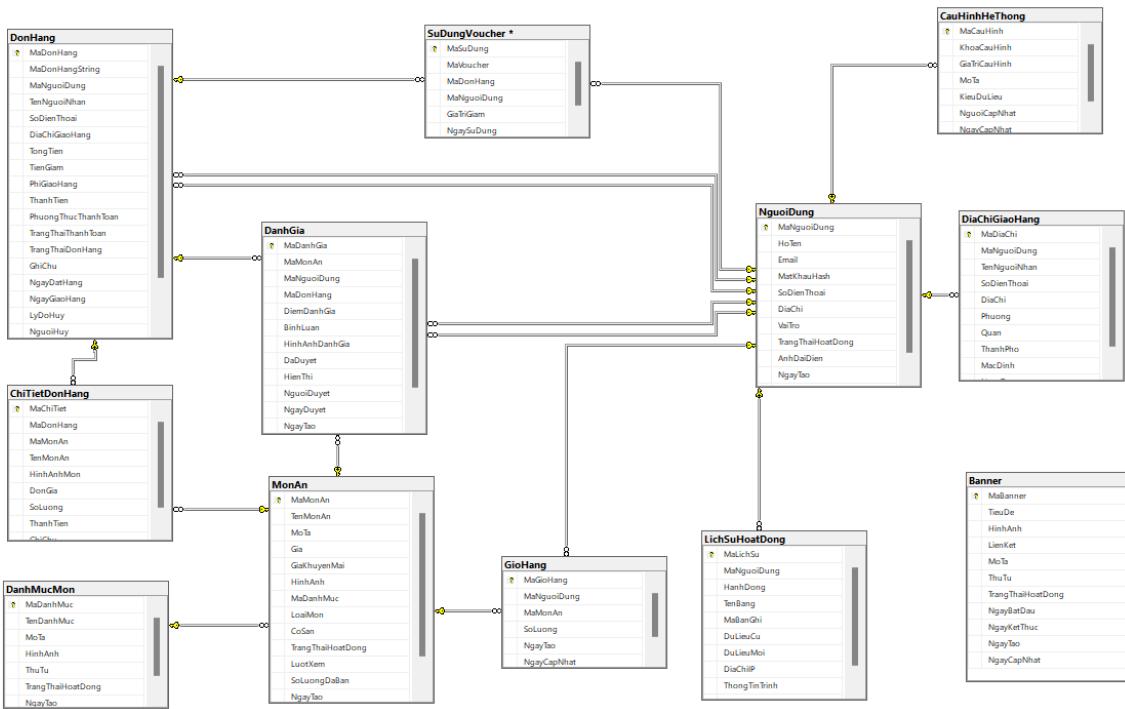
Bảng giỏ hàng: Quản lý tạm thời những món người dùng đã thêm vào trước khi xác nhận đơn hàng chính thức.

Bảng đánh giá: Tập hợp các phản hồi, đánh giá chất lượng món ăn từ người dùng sau khi nhận hàng thành công.

Bảng danh mục: Phân loại sản phẩm theo nhóm để hỗ trợ người dùng dễ lọc và tìm kiếm món phù hợp.

Bảng banner hoặc slider: Lưu các hình ảnh quảng cáo được hiển thị tại trang chủ, thường dùng cho mục đích khuyến mãi hoặc giới thiệu món mới.

3.2.1. Mô hình quan hệ dữ liệu ERD



Hình 3.2 Mô hình quan hệ dữ liệu ERD

3.2.2. Mô tả chi tiết các thực thể

NGƯỜI DÙNG & PHÂN QUYỀN

- **NguoiDung:** Lưu trữ thông tin người dùng như: họ tên, email, mật khẩu (mã hóa), số điện thoại, địa chỉ, vai trò, ảnh đại diện và trạng thái hoạt động.
- **VaiTro:** Định nghĩa các loại vai trò người dùng như: khách hàng, nhân viên, quản trị viên.
- **LichSuHoatDong:** Ghi lại lịch sử hoạt động của người dùng: hành động thực hiện, thời gian, địa chỉ IP, thiết bị,...

ĐẶT HÀNG & GIAO HÀNG

- **DonHang:** Quản lý đơn hàng bao gồm thông tin người nhận, tổng tiền, trạng thái thanh toán, phương thức thanh toán, thời gian đặt và giao hàng.
- **ChiTietDonHang:** Chi tiết từng món trong đơn hàng: tên món, đơn giá, số lượng, hình ảnh và tổng tiền từng món.
- **DiaChiGiaoHang:** Danh sách địa chỉ giao hàng của người dùng, hỗ trợ lưu nhiều địa chỉ khác nhau theo khu vực (tỉnh, huyện, phường).

- SuDungVoucher: Ghi lại việc sử dụng mã giảm giá theo đơn hàng, liên kết với người dùng và mã giảm giá cụ thể.

SẢN PHẨM

- MonAn: Chứa thông tin các món ăn đang kinh doanh: tên món, mô tả, hình ảnh, giá gốc, giá khuyến mãi, danh mục, trạng thái hiển thị,...
- DanhMucMon: Danh mục phân loại món ăn như: món chay, món mặn, nước uống, món lẩu,...
- DanhGia: Đánh giá của người dùng về món ăn sau khi đặt hàng: số sao, bình luận, hình ảnh kèm theo, trạng thái duyệt hiển thị,...
- GioHang: Lưu tạm thời các món được thêm vào giỏ hàng của người dùng trước khi tiến hành thanh toán.

CẤU HÌNH & MARKETING

- CauHinhHeThong: Lưu các cài đặt hệ thống như: giá trị cấu hình, mô tả, kiểu dữ liệu, người cập nhật cuối cùng.
- Banner: Quản lý các banner quảng cáo: tiêu đề, hình ảnh, liên kết, mô tả, trạng thái hiển thị và thời gian bắt đầu/kết thúc chiến dịch.

3.2.3. Mô tả chi tiết các thực thể

1:N (một – nhiều):

- Một người dùng có thể có nhiều đơn hàng, địa chỉ giao hàng, đánh giá, lịch sử hoạt động.
- Một đơn hàng có thể chứa nhiều món (ChiTietDonHang).
- Một danh mục chứa nhiều món ăn.
- Một món ăn có nhiều đánh giá.

N:N (nhiều – nhiều):

- Quan hệ giữa người dùng – voucher – đơn hàng thể hiện qua bảng trung gian SuDungVoucher.

3.2.4. Đánh giá thiết kế

Cơ sở dữ liệu được xây dựng theo hướng logic và khoa học, đảm bảo tuân thủ các nguyên tắc chuẩn hóa (đặc biệt là chuẩn hóa đến mức 3NF). Cách tổ chức bảng và quan hệ giữa các thực thể giúp dữ liệu không bị dư thừa, tránh mâu thuẫn và hỗ trợ bảo trì dễ dàng.

Hệ thống dữ liệu cũng được thiết kế nhằm tối ưu hiệu suất truy vấn, phục vụ tốt cho các chức năng cốt lõi như xử lý đơn hàng, quản lý người dùng, phân quyền truy cập, đánh giá sản phẩm và cập nhật hoạt động người dùng. Nhờ đó, các tác vụ thường xuyên như tìm kiếm món ăn, thống kê đơn hàng, lọc theo danh mục hoặc trạng thái đều được thực hiện hiệu quả.

Bên cạnh đó, mô hình dữ liệu vẫn giữ được tính linh hoạt để mở rộng về sau. Khi cần tích hợp các tính năng mới như lịch sử mua hàng, chương trình khuyến mãi, tích điểm khách hàng hay hệ thống phản hồi – mô hình hiện tại vẫn đáp ứng tốt mà không cần thay đổi quá nhiều cấu trúc cơ bản.

3.3. Thiết kế API

API (Giao diện lập trình ứng dụng) đóng vai trò như cầu nối giữa phần giao diện người dùng (React) và hệ thống xử lý phía máy chủ (Express.js). Các API được thiết kế dựa trên nguyên tắc RESTful, giúp đảm bảo tính nhất quán, dễ mở rộng và thuận tiện cho việc bảo trì hoặc tích hợp thêm các dịch vụ bên ngoài sau này.

Mỗi API đại diện cho một chức năng riêng biệt, được tổ chức khoa học và dễ hiểu, với các phương thức HTTP được sử dụng đúng mục đích như:

- GET để truy vấn dữ liệu,
- POST để thêm mới,
- PUT hoặc PATCH để cập nhật,
- DELETE để xóa dữ liệu.

Cách phân tách tài nguyên rõ ràng giúp frontend dễ dàng gọi và xử lý dữ liệu, đồng thời backend cũng dễ quản lý luồng nghiệp vụ một cách hiệu quả và bảo mật.

3.3.1. Mô tả các endpoint chính

Hệ thống cửa hàng bán đồ ăn được phát triển theo kiến trúc RESTful API, trong đó các endpoint được tổ chức rõ ràng theo từng nhóm chức năng nghiệp vụ. API được

xây dựng trên nền tảng Express.js (JavaScript) và sử dụng JWT cho việc xác thực và phân quyền người dùng theo vai trò như khách hàng, nhân viên và quản trị viên.

Dưới đây là mô tả các endpoint tiêu biểu:

Phân hệ	Endpoint	Phương thức	Mô tả
Xác thực	/api/auth/register	POST	Đăng ký tài khoản người dùng mới
Xác thực	/api/auth/login	POST	Đăng nhập, trả về token JWT
Xác thực	/api/health	GET	Kiểm tra trạng thái hệ thống
Người dùng	/api/users/profile	GET	Truy xuất thông tin người dùng hiện tại
Người dùng	/api/users/profile	PUT	Cập nhật thông tin cá nhân
Người dùng	/api/users	GET	Lấy danh sách tất cả người dùng (Admin)
Người dùng	/api/users/:id/role	PUT	Thay đổi vai trò người dùng (Admin)
Người dùng	/api/users/:id/status	PUT	Thay đổi trạng thái người dùng (Admin)
Sản phẩm	/api/products	GET	Lấy danh sách tất cả sản phẩm
Sản phẩm	/api/products/:id	GET	Lấy chi tiết sản phẩm theo ID
Sản phẩm	/api/products	POST	Tạo sản phẩm mới (Admin/Staff)
Sản phẩm	/api/products/:id	PUT	Cập nhật thông tin sản phẩm (Admin/Staff)

Hệ thống cửa hàng bán đồ ăn trực tuyến

Sản phẩm	/api/products/:id	DELETE	Xóa sản phẩm (Admin)
Đơn hàng	/api/orders	GET	Lấy danh sách đơn hàng
Đơn hàng	/api/orders/:id	GET	Lấy chi tiết đơn hàng theo ID
Đơn hàng	/api/orders	POST	Tạo đơn hàng mới
Đơn hàng	/api/orders/:id	PUT	Cập nhật trạng thái đơn hàng (Staff/Admin)
Đơn hàng	/api/orders/:id	DELETE	Hủy đơn hàng (Admin)
Đánh giá	/api/reviews	GET	Lấy danh sách đánh giá
Đánh giá	/api/reviews	POST	Tạo đánh giá mới (Customer)
Đánh giá	/api/reviews/:id	PUT	Cập nhật trạng thái đánh giá (Admin)
Đánh giá	/api/reviews/:id	DELETE	Xóa đánh giá (Admin)
Banner	/api/banners	GET	Lấy danh sách banner
Banner	/api/banners	POST	Tạo banner mới (Admin)
Banner	/api/banners/:id	PUT	Cập nhật banner (Admin)
Banner	/api/banners/:id	DELETE	Xóa banner (Admin)
Thống kê	/api/stats	GET	Lấy thống kê tổng quan (Admin)

Thống kê	/api/stats/top-products	GET	Lấy danh sách sản phẩm bán chạy
Báo cáo	/api/export/pdf	GET	Xuất báo cáo PDF (Admin)
Báo cáo	/api/export/csv	GET	Xuất báo cáo CSV (Admin)
File	/api/upload	POST	Upload hình ảnh sản phẩm
File	/uploads/:filename	GET	Truy cập file đã upload

Các endpoint được phân quyền cụ thể theo vai trò: khách hàng chỉ có thể thực hiện các thao tác liên quan đến trải nghiệm mua hàng như xem sản phẩm, đặt món, đánh giá đơn hàng; trong khi quản trị viên và nhân viên được cấp quyền truy cập vào các chức năng quản lý như cập nhật sản phẩm, xử lý đơn hàng, điều hành hệ thống và theo dõi thống kê hoạt động.

3.3.2. Cấu trúc request/response

Việc thiết kế cấu trúc request và response trong hệ thống đảm bảo tính nhất quán và dễ hiểu. Dữ liệu trao đổi được định dạng bằng JSON. Dưới đây là một số ví dụ điển hình:

3.3.2.1. Đăng ký người dùng

Phương thức: POST

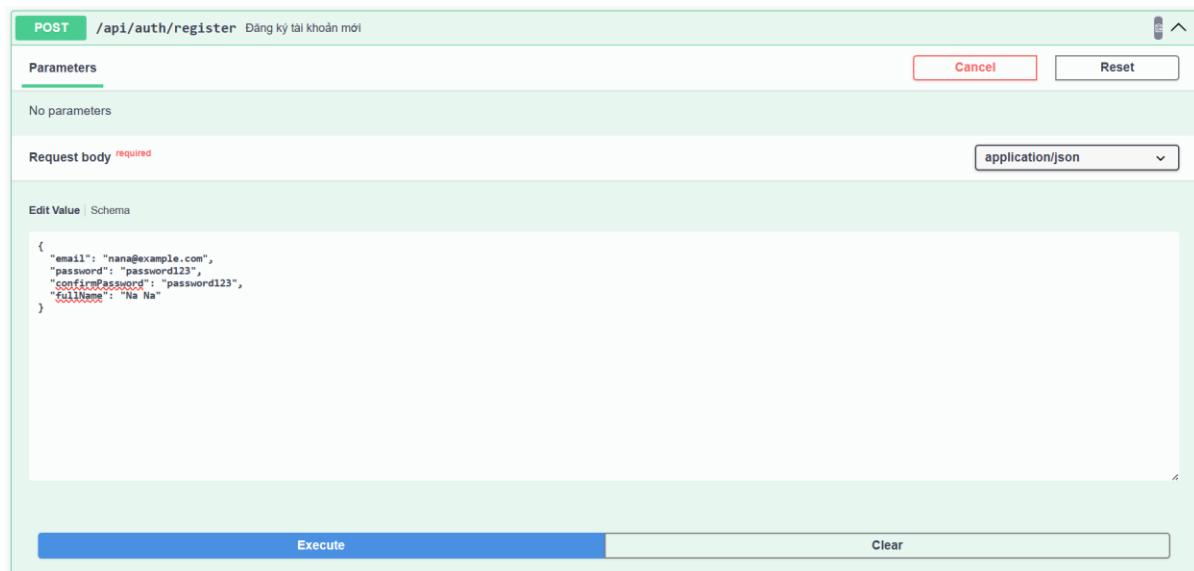
Endpoint: /api/auth/register

Request body:

```
{
    "email": "nana@example.com",
    "password": "password123",
    "confirmPassword": "password123",
```

Hệ thống cửa hàng bán đồ ăn trực tuyến

```
"fullName": "Na Na"  
}
```



Hình 3.3. Request gửi dữ liệu đăng ký người dùng

Response:

```
{  
  
    "message": "Đăng ký thành công",  
  
    "token": "...",  
  
    "user": {  
  
        "id": 16,  
  
        "email": "nana@example.com",  
  
        "fullName": "Na Na",  
  
        "role": "user"  
  
    }  
}
```

Hệ thống cửa hàng bán đồ ăn trực tuyến

Responses

Curl

```
curl -X 'POST' \
'http://localhost:3000/api/auth/register' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
  "email": "nana@example.com",
  "password": "password123",
  "confirmPassword": "password123",
  "fullName": "Na Na"
}'
```

Request URL

```
http://localhost:3000/api/auth/register
```

Server response

Hình 3.4 Response khi đăng ký thành công

3.3.2.2. Đăng nhập và nhận JWT

Phương thức: POST

Endpoint: /api/auth/login

Request body:

```
{
  "email": "nana@example.com",
  "password": "password123"
}
```

Request body required

Edit Value | Schema

```
{
  "email": "nana@example.com",
  "password": "password123"
}
```

Hình 3.5 Request gửi dữ liệu đăng nhập

Response:

```
{
  "message": "Đăng nhập thành công",
  "token": "eyJhbGciOi..",
  "user": {
    "id": 16,
    "email": "nana@example.com",
    "name": "Na Na"
  }
}
```

```
"fullName": "Na Na",
"role": "user"
}
}
```

Details

Response body

```
{
  "message": "Đăng nhập thành công",
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9 .eyJpZCI6MTYsTmVYIisIjoihmFuYUB1eGxJlmNvbSIsInJvbGUiOiJ1c2VyTiwi aWF0TjoxNzUzNDIxNjYwLCT1eHAiOjE3NTQzMjY0NjB9 ._IKI_mn4YA1ybxeqhVT
TyCnjuZt81gBwFQ6PGRnSQg",
  "user": {
    "id": 16,
    "email": "nana@example.com",
    "fullName": "Na Na",
    "role": "user"
  }
}
```

Download

Hình 3.6 Response khi đăng nhập thành công

3.3.2.3. Lấy danh sách rạp phim

Phương thức: GET Endpoint: /api/products

Response:

```
{
  "_id": "687db106ba42acd30ff5dbbf",
  "id": 1,
  "name": "Lẩu thập cẩm đặc biệt",
  "description": "Nồi lẩu hấp dẫn với nước dùng cay đậm đà, kết hợp thịt bò, hải sản, rau tươi, nấm, mì và nhiều loại topping đầy màu sắc, mang đến trải nghiệm tròn vị cho mọi thực khách.",
  "price": 289000,
  "category": "Món chính",
  "image": "/uploads/file-1753417067837-438185745.jpg",
  "isActive": true,
  "createdAt": "2025-07-21T03:16:16.289Z",
  "updatedAt": "2025-07-25T04:17:49.045Z"
```

```
}
```

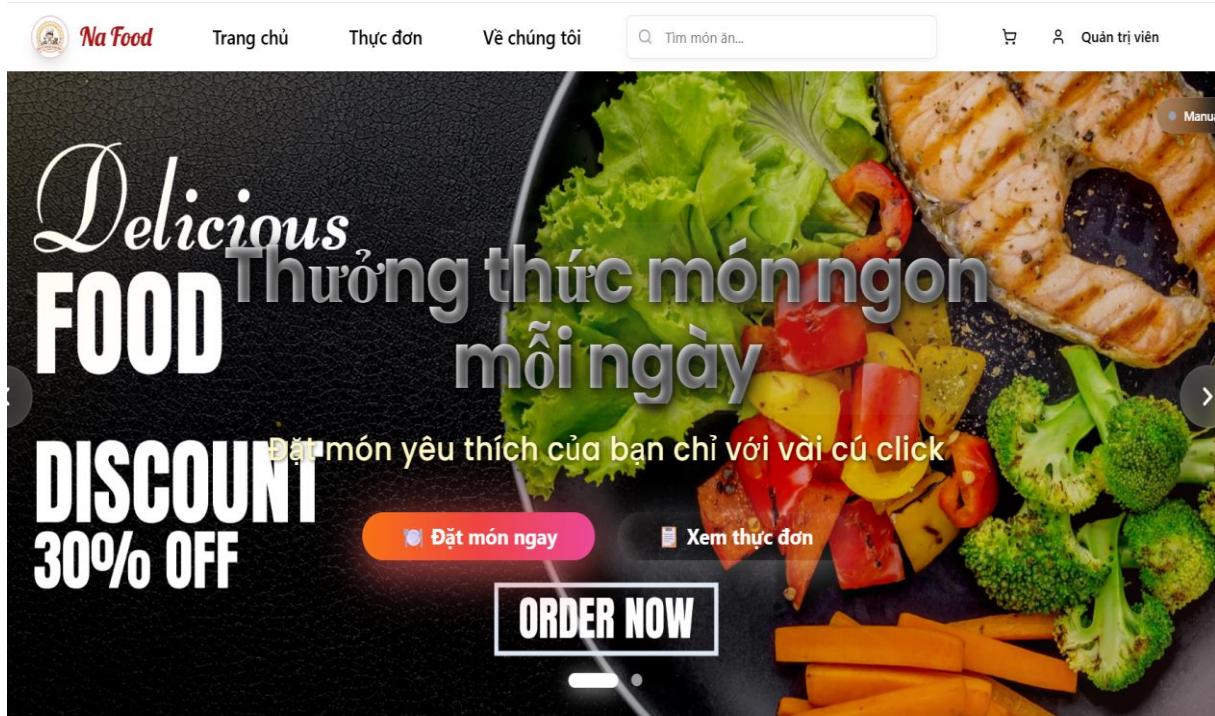
Details

Response body

```
[  
  {  
    "_id": "687db106ba42acd3ff5dbbf",  
    "id": 1,  
    "name": "Lẩu thập cẩm đặc biệt",  
    "description": "Ngoài lẩu hấp dẫn với nước dùng cay đậm đà, kết hợp thịt bò, hải sản, rau tươi, nấm, mì và nhiều loại topping đầy màu sắc, mang đến trải nghiệm tròn vị cho mọi thực khách.",  
    "price": 289000,  
    "category": "Món chính",  
    "image": "/uploads/file-1753417067837-438185745.jpg",  
    "isActive": true,  
    "createdAt": "2025-07-21T03:16:16.289Z",  
    "updatedAt": "2025-07-25T04:17:49.045Z"  
},  
{
```

Hình 3.6 Request gửi dữ liệu get

3.4. Figma thiết kế giao diện (UI/UX)



Hình 3.9 Giao diện trang chủ

Hệ thống cửa hàng bán đồ ăn trực tuyến

The screenshot displays a grid of eight food items from the Na Food website:

- Bánh Mì Thịt Nướng**: Món nhẹ. Bánh mì Việt Nam với thịt nướng thơm ngon, rau sống tươi mát. ★★★★☆ (4.0). Giá: 25.000đ.
- Bún thịt nướng**: Món nhẹ. Dĩa bún tươi kèm chả lụa, thịt nướng, đậu hũ chiên và rau sống như xà lách... ★★★★☆ (4.0). Giá: 30.000đ.
- Bún đậu mắm tôm**: Món nhẹ. Gồm bún tươi, đậu hũ chiên, chả cốt, thịt luộc, dưa leo, rau sống.... ★★★★☆ (4.0). Giá: 65.000đ.
- Chè Ba Màu**: Món tráng miệng. Chè ba màu truyền thống với đậu xanh, đậu đỏ và thạch. ★★★★☆ (4.0). Giá: 20.000đ.
- Cơm Tấm Sườn Nướng**: Món chính. Cơm tấm sườn nướng thơm lừng, ăn kèm chả trứng và bì. ★★★★☆ (4.0). Giá: 55.000đ.
- Lẩu thập cẩm đặc biệt**: Món chính. Nồi lẩu hấp dẫn với nước dùng cay đậm đà, kết hợp thịt bò, hải sản, rau... ★★★★☆ (4.0). Giá: 289.000đ.
- Mỳ cay**: Món nhẹ. Mì xào với nước sốt sa tế, tôm và rau củ, tạo nên món ăn cay nồng và hấp dẫn. ★★★★☆ (4.0). Giá: 55.000đ.
- Trà Đá Chanh**: Đồ uống. Trà đá chanh tươi mát, giải khát tuyệt vời. ★★★★☆ (4.0). Giá: 15.000đ.

Hình 3.10 Giao diện trang thực đơn

The screenshot shows the 'Về chúng tôi' (About Us) page of the Na Food website, featuring the following content:

Về Na Food

Na Food là nền tảng đặt món ăn trực tuyến hàng đầu, mang đến cho bạn những trải nghiệm ẩm thực tuyệt vời với đa dạng món ăn từ khắp nơi. Chúng tôi cam kết về chất lượng món ăn và dịch vụ giao hàng nhanh chóng, đảm bảo sự hài lòng của khách hàng.

Đa dạng món ăn
Hàng trăm món ăn từ khắp nơi

Giao hàng nhanh
Đảm bảo giao hàng trong 30 phút

Chất lượng cao
Cam kết về chất lượng và an toàn

Hình 3.11 Giao diện về chúng tôi

Hệ thống cửa hàng bán đồ ăn trực tuyến

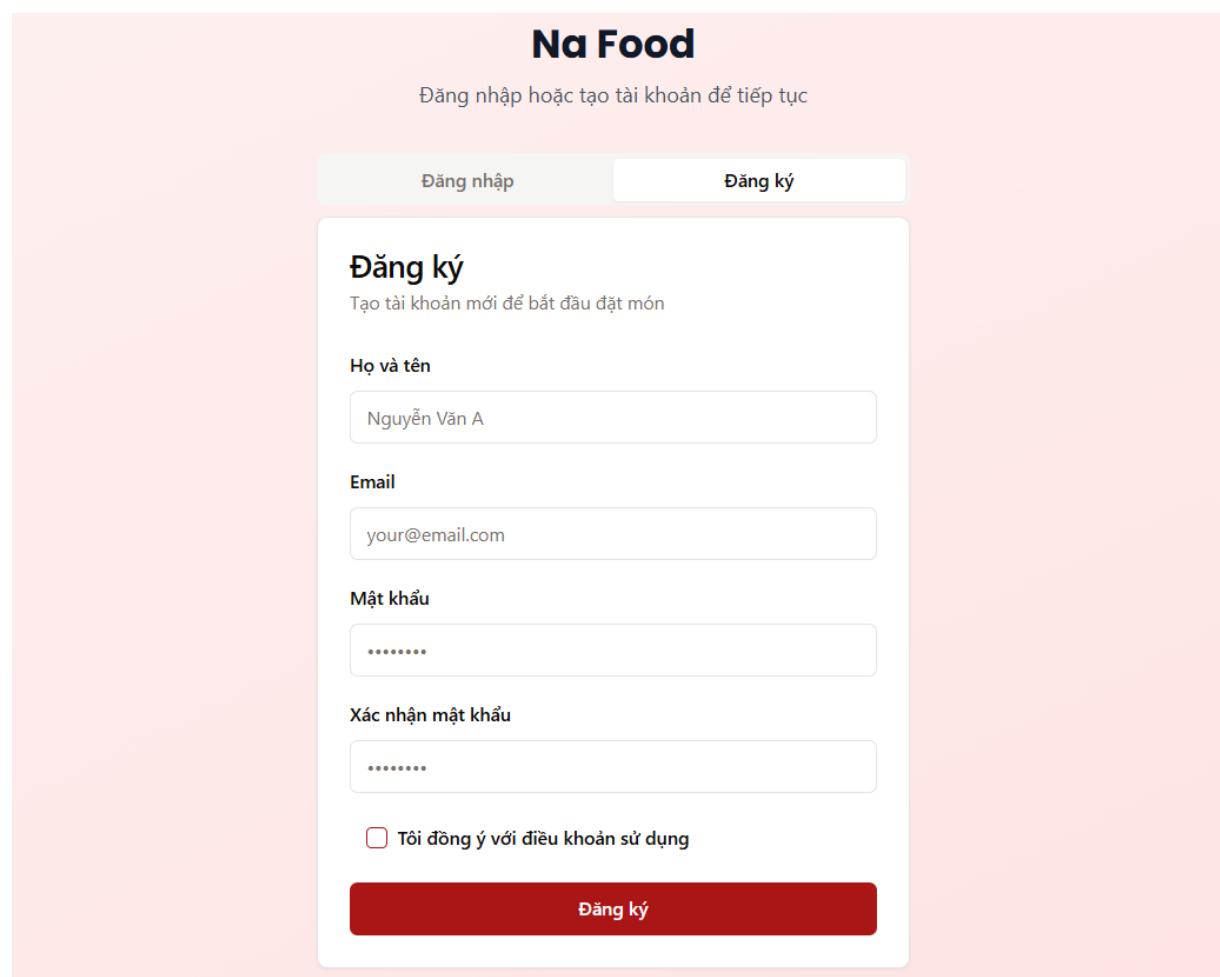
The screenshot shows a modal window titled "Chi tiết đơn hàng #39". Inside, it displays the delivery information for order #39, which was placed at 21:13 on July 24, 2025, by the administrator. The delivery address is Chùa Ông Mèo. The order details show three items: Chè Ba Màu (20,000đ), Bánh Mì Thịt Nướng (25,000đ), and Mỳ cay (55,000đ). The total amount is 100,000đ. Below the modal, there are other order details for order #38, including a total of 40,000đ.

Hình 3.12 Giao diện đơn hàng của tôi

The screenshot shows the login page for the Na Food website. It features a header with the logo "Na Food" and a sub-header "Đăng nhập hoặc tạo tài khoản để tiếp tục". Below this, there are two buttons: "Đăng nhập" (highlighted in red) and "Đăng ký". The main form is titled "Đăng nhập" and contains fields for "Email" (with the value "admin@nafood.com") and "Mật khẩu" (with the value "*****"). There is also a checkbox for "Ghi nhớ đăng nhập" and a large red "Đăng nhập" button at the bottom.

Hình 3.13 Giao diện trang đăng nhập

Hệ thống cửa hàng bán đồ ăn trực tuyến



Hình 3.14 Giao diện trang đăng ký

The screenshot shows the Admin Panel of the Na Food platform. At the top, there is a navigation bar with links for 'Trang chủ', 'Thực đơn', 'Về chúng tôi', a search bar 'Tim món ăn...', and user icons for 'Quản trị viên'. The main dashboard area has a sidebar on the left with links: 'Dashboard' (highlighted in red), 'Sản phẩm', 'Đơn hàng', 'Người dùng', 'Đánh giá', 'Banner', and 'Thống kê'. The main content area features a section titled 'NA FOOD' with four cards: 'Doanh thu hôm nay' (550.000đ, +20.1% so với hôm qua), 'Đơn hàng mới' (2, +180.1% so với tháng trước), 'Tổng đơn hàng' (33, +19% so với tháng trước), and 'Đã giao hàng' (13, +201 so với tháng trước). Below this is a table titled 'Đơn hàng gần đây' showing two recent orders:

Mã đơn	Khách hàng	Tổng tiền	Trạng thái	Thao tác
#22	NA FOOD	255.000đ	Chờ xử lý	
#21	NA FOOD	170.000đ	Chờ xử lý	

Hình 3.15 Giao diện quản trị viên

Hệ thống cửa hàng bán đồ ăn trực tuyến

The screenshot shows the product management section of the application. On the left sidebar, under 'Sản phẩm', the 'Đơn hàng' option is selected. The main area displays a table titled 'Danh sách sản phẩm (8)' with columns: Hình ảnh, Tên sản phẩm, Danh mục, Giá, Trạng thái, and Thao tác. Two products are listed:

- Lẩu thập cẩm đặc biệt: Món chính, 289.000đ, Hoạt động
- Bún thịt nướng: Món nhẹ, 30.000đ, Hoạt động

Hình 3.16 Giao diện trang quản lý sản phẩm

The screenshot shows the order management section of the application. On the left sidebar, under 'Đơn hàng', the 'Đơn hàng' option is selected. The main area displays a summary table 'Tổng đơn' with values: 33, Chờ xử lý: 10, Đang xử lý: 7, Đã giao: 13, Đã hủy: 1, and Doanh thu: 7.245.000đ. Below it is a search bar and a table titled 'Danh sách đơn hàng (33)' with columns: ID, Khách hàng, Sản phẩm, Tổng tiền, Thanh toán, Trạng thái, Ngày tạo, and Thao tác. One order is shown in detail:

#39	Quản trị viên 0333661157	3 món Chè Ba Máu x1 Bánh Mì Thịt Nướng x1	100.000đ	Thanh toán khi nhận hàng	Chờ xử lý	21:13 24/07/2025	
-----	-----------------------------	--	----------	--------------------------	-----------	---------------------	--

Hình 3.17 Giao diện quản lý đơn hàng

The screenshot shows the user management section of the application. On the left sidebar, under 'Người dùng', the 'Người dùng' option is selected. The main area displays a summary table 'Tổng người dùng' with values: 5, Quản trị viên: 2, Nhân viên: 1, and Người dùng: 2. Below it is a search bar and a table titled 'Danh sách người dùng (5)' with columns: Thông tin, Email, Vai trò, Trạng thái, Ngày tạo, and Thao tác. One user is shown in detail:

Quản trị viên 0123456789	admin@nafood.com	Quản trị viên	Hoạt động	21/7/2025	
-----------------------------	------------------	---------------	-----------	-----------	--

Hình 3.18 Giao diện quản lý đơn hàng

Hệ thống cửa hàng bán đồ ăn trực tuyến

The screenshot shows the 'Review' section of the admin panel. It displays statistics: Total reviews (3), Pending review (1), Approved (2), and Rejected (0). A search bar for filtering reviews by content is present. Below this, there are tabs for All reviews, Pending review (1), Approved (2), and Rejected (0). A specific review from 'NA FOOD' is shown as an example.

Hình 3.17 Giao diện quản lý đánh giá

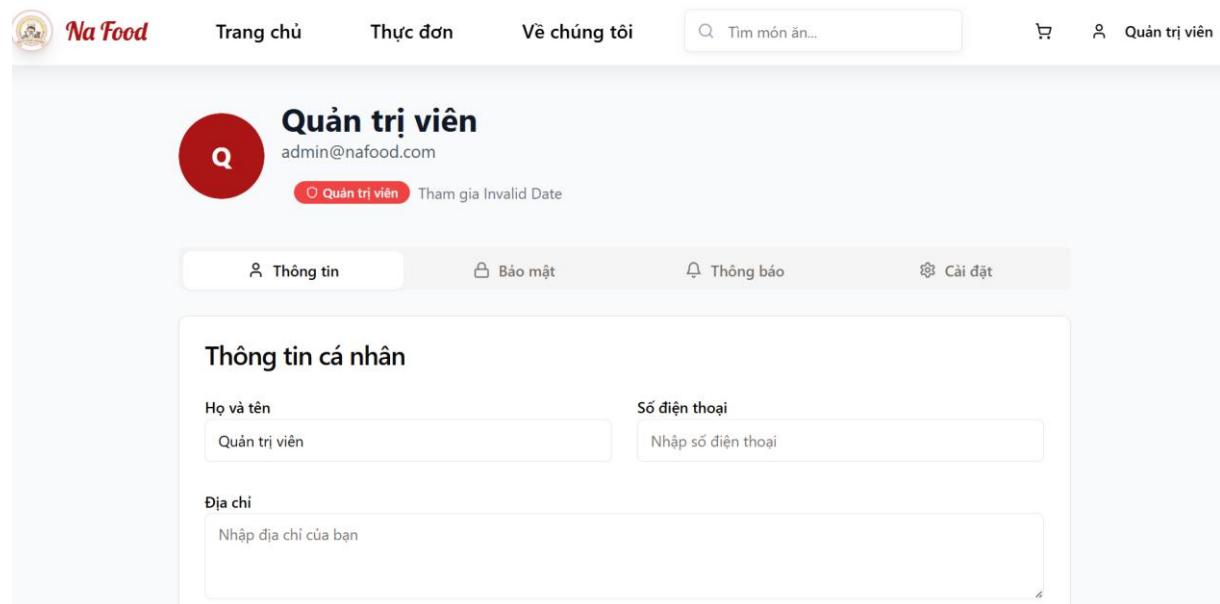
The screenshot shows the 'Banner' section of the admin panel. It displays statistics: Total banners (2), Active (2), and Inactive (0). A table lists the details of each banner, including their status, ID, title, description, link, and position. Both banners are marked as active.

Hình 3.18 Giao diện quản lý banner

The screenshot shows the 'Statistics' section of the admin panel. It features a summary of key performance metrics: Total sales (\$15,750,000đ), Total orders (342), Number of customers (128), and Number of products (45). Below this, there's a chart titled 'Biểu đồ thống kê' (Statistics chart) and a list of top-selling products: Phở Bò Tái (2,500,000đ), Bún Bò Huế (2,100,000đ), and Cơm Tấm Sườn (1,800,000đ).

Hình 3.19 Giao diện trang thống kê

Hệ thống cửa hàng bán đồ ăn trực tuyến



Hình 3.20 Giao diện trang tài khoản của tôi

Chương 4. TRIỂN KHAI VÀ CÔNG NGHỆ SỬ DỤNG

4.1 Các công nghệ đã sử dụng

Trong quá trình phát triển hệ thống ứng dụng web **Hệ Thống Cửa Hàng Bán Đồ Ăn**, nhóm đã lựa chọn và triển khai một số công nghệ chủ chốt nhằm đảm bảo hiệu quả vận hành, khả năng mở rộng và bảo mật của hệ thống. Các công nghệ được phân chia theo ba lớp chính: giao diện người dùng (frontend), xử lý logic nghiệp vụ (backend), cơ sở dữ liệu (database) và xác thực bảo mật (authentication). Cụ thể:

4.1.1 Ngôn ngữ lập trình chính

4.1.1.1 JavaScript (ESNext)

Toàn bộ dự án sử dụng JavaScript là ngôn ngữ lập trình chính cho cả frontend và backend. JavaScript giúp nhóm tận dụng chung ngôn ngữ giữa client và server, đồng thời hỗ trợ nhanh chóng quá trình phát triển, debug và bảo trì.

JavaScript phiên bản ES6+ đóng vai trò là ngôn ngữ lập trình chính cho toàn bộ dự án **HT Cửa Hàng Bán Đồ Ăn**, bao gồm cả các phần frontend và backend. Việc sử dụng JavaScript hiện đại mang lại nhiều lợi ích đáng kể:

- **Tính nhất quán (Consistency):** Sử dụng cùng một ngôn ngữ cho cả frontend và backend, giảm thiểu độ phức tạp trong quá trình phát triển và bảo trì.
- **ES Modules:** Hỗ trợ import/export modules hiện đại, cho phép tổ chức code một cách có cấu trúc và tái sử dụng.
- **Async/Await:** Xử lý bát đồng bộ hiệu quả, đặc biệt quan trọng trong việc tương tác với cơ sở dữ liệu và API.
- **Destructuring và Spread Operator:** Cải thiện khả năng đọc và viết code.
- **Arrow Functions:** Cú pháp ngắn gọn và xử lý context tốt hơn.

Cấu hình JavaScript được thể hiện qua tệp package.json:

```
{  
  "type": "module",  
  "scripts": {
```

```
"dev": "node start.js",

"build": "vite build && esbuild server/index.js --platform=node --packages=external --bundle --format=esm --outdir=dist",
"start": "node start.js"

}
```

4.1.2 Frontend

Hệ thống giao diện người dùng được xây dựng dựa trên React kết hợp với nhiều thư viện hỗ trợ UI hiện đại. Kiến trúc phân tách rõ ràng theo component-based và routing.

Cấu trúc thư mục:

```
client/
  └── src/
    ├── components/  # Các component dùng chung
    ├── pages/       # Các trang chính: Home, Admin, Staff
    ├── lib/          # Utilities, hooks, auth
    ├── styles/      # CSS files
    └── App.jsx      # File khởi động chính
  └── index.html
  └── vite.config.js
```

4.1.2.1 React 18.3.1

React 18.3.1 là thư viện JavaScript chính cho việc xây dựng giao diện người dùng (UI) của **HT Cửa Hàng Bán Đồ Ăn**. Các tính năng nổi bật được áp dụng bao gồm:

- **Functional Components với Hooks:** Giúp quản lý trạng thái và logic trong các thành phần hàm một cách hiệu quả.
- **TanStack Query:** Được sử dụng để quản lý trạng thái máy chủ (server state management), cung cấp khả năng caching, cập nhật nền và xử lý lỗi tối ưu.

- **Context API:** Hỗ trợ quản lý trạng thái toàn cục (global state), đơn giản hóa việc chia sẻ dữ liệu giữa các thành phần.
- **Error Boundaries:** Cải thiện trải nghiệm người dùng bằng cách xử lý các trường hợp lỗi một cách linh hoạt.

App.js

```
import { Switch, Route } from "wouter";

import { QueryClientProvider } from "@tanstack/react-query";

import { AuthProvider } from "@/lib/auth.jsx";
import { CartProvider } from "@/lib/cart.jsx";
import ErrorBoundary from "@/components/error-boundary";

function App() {

  return (
    <ErrorBoundary>
      <QueryClientProvider client={queryClient}>
        <AuthProvider>
          <CartProvider>
            <Router />
          </CartProvider>
        </AuthProvider>
      </QueryClientProvider>
    </ErrorBoundary>
  );
}


```

4.1.2.2 Vite 5.4.19

Vite 5.4.19 đóng vai trò là công cụ xây dựng (build tool) và máy chủ phát triển (development server) chính. Vite mang lại những ưu điểm vượt trội:

- **Hot Module Replacement (HMR)**: Cho phép cập nhật mã nguồn mà không cần tải lại toàn bộ trang, tăng tốc độ phát triển.
- **Thời gian xây dựng nhanh (Fast Build Times)**: Sử dụng ESBUILD để rút ngắn thời gian xây dựng ứng dụng.
- **Hỗ trợ ES Modules**: Tận dụng các tính năng module gốc của JavaScript.
- **Hệ sinh thái plugin**: Cung cấp khả năng mở rộng thông qua các plugin đa dạng.

Cấu hình Vite được định nghĩa trong vite.config.js:

```
export default defineConfig({  
  plugins: [react()],  
  resolve: {  
    alias: {  
      "@": path.resolve(__dirname, "client", "src"),  
      "@shared": path.resolve(__dirname, "shared"),  
      "@assets": path.resolve(__dirname,  
        "attached_assets"),  
    },  
    root: path.resolve(__dirname, "client"),  
    build: {  
      outDir: path.resolve(__dirname, "dist/public"),  
      emptyOutDir: true,  
    }  
  }  
});
```

4.1.2.3 Tailwind CSS 3

Tailwind CSS 3.4.17 là framework CSS theo hướng tiện ích (utility-first), cho phép xây dựng giao diện nhanh chóng và linh hoạt bằng cách sử dụng các lớp tiện ích được định nghĩa sẵn.

Extensions: Dự án tích hợp `@tailwindcss/typography` để xử lý kiểu chữ và `tailwindcss-animate` để tạo các hiệu ứng động mượt mà.

Cấu hình Tailwind CSS:

```
export default {  
  darkMode: ["class"],  
  content: ["./client/index.html",  
            "./client/src/**/*.{js,jsx}"],  
  theme: {  
    extend: {  
      borderRadius: {  
        lg: "var(--radius)",  
        md: "calc(var(--radius) - 2px)",  
        sm: "calc(var(--radius) - 4px)",  
      },  
      colors: {  
        background: "var(--background)",  
        foreground: "var(--foreground)",  
      }  
    }  
  }  
}
```

4.1.2.4 Thư viện Thành phần UI (UI Component Libraries)

Dự án sử dụng kết hợp nhiều thư viện để xây dựng các thành phần UI chất lượng cao:

Radix UI: Một thư viện thành phần toàn diện (hơn 25 thành phần như Dialog, Dropdown, Select, v.v.) tập trung vào khả năng tiếp cận (accessibility-first design) và cung cấp các thành phần không kiểu dáng (unstyled components) để tùy chỉnh hoàn toàn.

Lucide React: Thư viện biểu tượng với hơn 450 biểu tượng, đảm bảo tính nhất quán và đa dạng cho giao diện.

Framer Motion: Thư viện hoạt ảnh mạnh mẽ, giúp tạo ra các chuyển động mượt mà và hiệu ứng động phong phú cho giao diện người dùng.

Wouter: Router nhẹ (2KB) thay thế cho React Router, cung cấp routing đơn giản và hiệu quả.

TanStack Query: Thư viện quản lý server state với khả năng caching thông minh và synchronization tự động.

4.1.3 Backend

Cấu trúc thư mục:

server/

```
|   └── index.js      # Entry point chính  
|   └── routes.js    # Định nghĩa API routes  
|   └── db.js        # Kết nối cơ sở dữ liệu  
|   └── storage.js   # Data access layer  
|   └── vite.js       # Vite integration  
└── middleware/     # Authentication, error handling
```

4.1.2.4.1.3.1 Node.js với Express.js

Backend của Movie Ticket Booking được xây dựng trên Node.js và sử dụng Express.js 4.21.2 làm framework ứng dụng web. Express.js cung cấp một nền tảng mạnh mẽ và linh hoạt để xây dựng các RESTful API.

Middleware Stack: Dự án sử dụng một loạt các middleware để xử lý các chức năng quan trọng:

- express-session: Quản lý phiên người dùng.
- CORS middleware: Cho phép chia sẻ tài nguyên giữa các nguồn gốc khác nhau.
- Custom authentication middleware: Middleware xác thực tùy chỉnh để quản lý quyền truy cập.
- Error handling middleware: Xử lý lỗi tập trung cho toàn bộ ứng dụng.

Thể hiện qua index.js:

```
import express from "express";
import { registerRoutes } from "./routes.js";
import { setupVite, serveStatic } from "./vite.js";

const app = express();

// Cấu hình middleware
app.use(express.json({ limit: '50mb' }));
app.use(express.urlencoded({ extended: false, limit: '50mb' }));
app.use('/uploads',
express.static(path.join(process.cwd(), 'public',
'uploads')));

// Enhanced error handling middleware
app.use((err, _req, res, _next) => {
  const status = err.status || err.statusCode || 500;
  const message = err.message || "Internal Server Error";
```

```
    res.status(status).json({ message, timestamp: new
Date().toISOString() });
}
```

4.1.3.2 Cơ sở dữ liệu

MongoDB 6.17.0: Là cơ sở dữ liệu chính của dự án, được sử dụng để lưu trữ dữ liệu phi cấu trúc (NoSQL data)

Cấu hình MongoDB trong db.js:

```
import { MongoClient } from 'mongodb';

class MongoDB {

  constructor() {

    const options = {

      maxPoolSize: 10,

      serverSelectionTimeoutMS: 5000,

      socketTimeoutMS: 45000,

      heartbeatFrequencyMS: 10000,

    };

    this.client = new MongoClient(process.env.MONGODB_URI, options);

    this.db = this.client.db('nafood');

    this.isConnected = false;

    this.maxRetries = 5;

  }

}
```

4.1.4 Xác thực và bảo mật

JSON Web Tokens (JWT)

jsonwebtoken 9.0.2: Được sử dụng để triển khai xác thực dựa trên token (token-based authentication), cung cấp một phương pháp an toàn và không trạng thái (stateless) để xác minh danh tính người dùng.

bcrypt 6.0.0: Một thư viện mạnh mẽ để băm mật khẩu (password hashing), đảm bảo rằng mật khẩu được lưu trữ an toàn trong cơ sở dữ liệu.

```
// routes.js

import bcrypt from "bcrypt";
import jwt from "jsonwebtoken";


// Registration endpoint

const hashedPassword = await bcrypt.hash(validatedData.password, 10);

// Login verification

app.post("/api/auth/login", async (req, res) => {

    const user = await storage.getUserByEmail(validatedData.email);

    const isValidPassword = await bcrypt.compare(validatedData.password, user.password);

    if (isValidPassword) {

        const token = jwt.sign(
            { id: user.id, email: user.email, role: user.role },
            JWT_SECRET,
            { expiresIn: '7d' }
        )
    }
})
```

```
) ;  
  
    res.json({ token, user }) ;  
  
}  
  
}  
);
```

Passport.js 0.7.0: Một middleware xác thực linh hoạt, với việc sử dụng passport-local cho chiến lược xác thực cục bộ (local authentication strategy).

Cấu hình xác thực bằng JWT (middlewares/routes.js):

```
const authenticateToken = (req, res, next) => {  
  
    const authHeader = req.headers['authorization'];  
  
    const token = authHeader && authHeader.split(' ')[1];  
  
    if (!token) {  
  
        return res.status(401).json({ message: "Access token required" });  
  
    }  
  
    jwt.verify(token, JWT_SECRET, (err, user) => {  
  
        if (err) {  
  
            return res.status(403).json({ message: "Invalid or expired token" });  
  
        }  
  
        req.user = user;  
  
        next();  
    });  
};
```

4.2 Quy trình CI/CD với GitHub Actions

Công cụ được sử dụng cho mục tiêu này là GitHub Actions, một dịch vụ CI/CD tích hợp sẵn trong nền tảng GitHub.

Quy trình CI/CD được thiết kế theo các giai đoạn sau: Continuous Integration (CI): Mỗi khi có thay đổi mã nguồn được đẩy lên nhánh chính (main hoặc develop), GitHub Actions sẽ tự động kích hoạt workflow để thực hiện các bước như kiểm thử mã (unit test), kiểm tra định dạng (linting), và build ứng dụng.

Continuous Deployment (CD): Nếu quy trình CI thành công, workflow tiếp tục thực hiện bước triển khai (deploy) tự động lên môi trường máy chủ đã cấu hình trước đó (có thể là server thực tế hoặc nền tảng cloud). Việc triển khai này có thể bao gồm việc chạy Docker container hoặc cập nhật mã nguồn trên máy chủ đích.

Quy trình CI/CD giúp phát hiện lỗi sớm, tránh xung đột trong quá trình tích hợp mã, và giảm thời gian triển khai, từ đó nâng cao chất lượng sản phẩm.

Cấu hình file:

```
# This workflow will do a clean installation of node
dependencies, cache/restore them, build the source code
and run tests across different versions of node
#
#           For           more           information           see:
#           https://docs.github.com/en/actions/automating-builds-
#           and-tests/building-and-testing-nodejs

name: Node.js CI

on:
  push:
    branches: [ "main" ]
  pull_request:
    branches: [ "main" ]

jobs:
  build:
```

```
runs-on: ubuntu-latest

strategy:
  matrix:
    node-version: [18.x, 20.x, 22.x]
      # See supported Node.js release schedule at
      https://nodejs.org/en/about/releases/

    steps:
      - uses: actions/checkout@v4
      - name: Use Node.js ${{ matrix.node-version }}
        uses: actions/setup-node@v4
        with:
          node-version: ${{ matrix.node-version }}
          cache: 'npm'
      - run: npm ci
      - run: npm run build --if-present
      - run: npm test
```

Nhờ GitHub Actions, nhóm đã có thể rút ngắn đáng kể thời gian triển khai và kiểm thử thủ công, đồng thời đảm bảo rằng mỗi thay đổi đều được kiểm tra tự động trước khi đưa vào sản phẩm chính thức.

4.3 Cấu hình Docker và quy trình triển khai ứng dụng

4.3.1 Docker

Docker là nền tảng đóng gói ứng dụng và các thành phần phụ thuộc vào một container nhẹ, giúp đảm bảo ứng dụng chạy nhất quán trên mọi môi trường. Trong dự án này, nhóm đã sử dụng Docker để xây dựng và triển khai cả frontend và backend của hệ thống.

Cấu trúc triển khai Docker

Ứng dụng được chia thành hai phần chính:

Frontend: Viết bằng React, được đóng gói thành một Docker image sử dụng nginx để phục vụ giao diện người dùng.

Backend: Sử dụng Node.js với Express, đóng gói thành một image riêng để xử lý API.

Database: Sử dụng MongoDB, được khởi chạy thông qua Docker image chính thức mongo.

Toàn bộ kiến trúc được quản lý thông qua tệp docker-compose.yml, đảm bảo các dịch vụ có thể liên kết và khởi động cùng lúc.

Cấu trúc file docker-compose.yml:

```
version: '3.8'

services:
    # MongoDB Database - Cơ sở dữ liệu chính
    mongodb:
        image: mongo:7.0
        container_name: nafood-mongodb
        restart: unless-stopped
        environment:
            MONGO_INITDB_ROOT_USERNAME: admin
            MONGO_INITDB_ROOT_PASSWORD: password123
            MONGO_INITDB_DATABASE: nafood
        ports:
            - "27017:27017" # Port để truy cập MongoDB từ bên ngoài
        volumes:
            - mongodb_data:/data/db      # Lưu trữ dữ liệu persistent
            - ./scripts/init-mongo.js:/docker-entrypoint-initdb.d/init-mongo.js:ro # Script khởi tạo DB
        networks:
            - nafood-network
    healthcheck:
```

```
test:          ["CMD",           "mongosh",           "--eval",
"db.adminCommand('ping')"]

interval: 10s
timeout: 5s
retries: 5
start_period: 30s    # Thời gian chờ MongoDB khởi
động

deploy:
resources:
limits:
memory: 1G        # Giới hạn RAM tối đa

reservations:
memory: 512M      # RAM tối thiểu được đảm bảo

# Backend API - Server ứng dụng chính
backend:
build:
context: ..  # Build từ thư mục gốc
dockerfile: CONGNGHEPHANMEM/Dockerfile
container_name: nafood-backend
restart: unless-stopped
environment:
NODE_ENV: production
DOCKER_ENV: "true"
PORT: 3000
# Kết nối MongoDB Atlas (cloud)
MONGODB_URI:
mongodb+srv://admin:PCO6NePc2Gmcifzt@lamv.tzclslv.mongodb.net/nafood?retryWrites=true&w=majority
# ⚠ QUAN TRỌNG: Thay đổi secrets này trong
production
```

```
JWT_SECRET:      your-super-secret-jwt-key-change-in-
production

SESSION_SECRET:      your-session-secret-change-in-
production

FRONTEND_URL: http://localhost:3000

ports:
  - "3000:3000"  # Port chính của ứng dụng

volumes:
  - ./public/uploads:/app/public/uploads  # Lưu trữ
file upload
  - ./logs:/app/logs  # Lưu trữ log files

# Không cần depends_on vì sử dụng MongoDB Atlas

networks:
  - nafood-network

healthcheck:
  test:          ["CMD",           "curl",           "-f",
"http://localhost:3000/api/health"]
  interval: 30s
  timeout: 10s
  retries: 5
  start_period: 40s  # Thời gian chờ backend khởi
động

deploy:
  resources:
    limits:
      memory: 512M  # Giới hạn RAM cho backend
    reservations:
      memory: 256M  # RAM tối thiểu

  # Redis for Session Store (Tùy chọn - dùng cho cache và
session)

  redis:
```

```
image: redis:7-alpine
container_name: nafood-redis
restart: unless-stopped
ports:
- "6379:6379" # Port Redis
volumes:
- redis_data:/data # Lưu trữ Redis data persistent
networks:
- nafood-network
command: redis-server --appendonly yes # Bật persistence

# Định nghĩa volumes để lưu trữ dữ liệu persistent
volumes:
mongodb_data:
  driver: local # Lưu trữ dữ liệu MongoDB
redis_data:
  driver: local # Lưu trữ dữ liệu Redis
# Định nghĩa network để các container giao tiếp với nhau
networks:
nafood-network:
  driver: bridge # Bridge network cho internal communication
```

4.3.2 Quy trình triển khai ứng dụng

Xây dựng Docker Image: Mỗi phần (frontend/backend) được viết Dockerfile để build image riêng.

Khởi chạy Container: Sử dụng docker-compose up --build để khởi tạo toàn bộ hệ thống với cấu hình đã định nghĩa.

Kiểm tra và giám sát: Theo dõi log bằng docker logs và kiểm tra trạng thái hệ thống qua docker ps.

Triển khai thực tế: Docker giúp nhóm dễ dàng chuyển toàn bộ dự án lên VPS hoặc nền tảng đám mây (như Render, Heroku, AWS...) mà không cần cấu hình lại môi trường.

Nhờ việc áp dụng Docker, hệ thống trở nên dễ triển khai, dễ phục hồi khi có lỗi và có thể chạy ổn định trên bất kỳ máy chủ nào hỗ trợ Docker mà không phụ thuộc vào môi trường cụ thể.

Chương 5. QUẢN LÝ DỰ ÁN

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

Trong quá trình phát triển hệ thống cửa hàng bán đồ ăn, nhóm đã lựa chọn **Jira Software** làm công cụ quản lý dự án chính. Đây là một nền tảng mạnh mẽ, hỗ trợ hiệu quả cho các nhóm làm việc theo phương pháp Agile hoặc Scrum. Việc sử dụng Jira giúp nhóm tổ chức công việc một cách khoa học, dễ theo dõi và đảm bảo tiến độ phát triển luôn được kiểm soát.

Ngay từ đầu, nhóm đã tạo một project mới trên Jira và chọn mẫu phù hợp với Scrum framework. Các **Epic** được xác định tương ứng với các nhóm chức năng lớn trong hệ thống như: quản lý sản phẩm, quản lý đơn hàng, xác thực người dùng, giao diện người dùng, thanh toán và triển khai CI/CD. Từ các Epic này, nhóm xây dựng những **User Story** đại diện cho từng yêu cầu cụ thể của người dùng (chẳng hạn như: thêm món vào giỏ, xem danh sách món, xử lý đơn hàng...). Các User Story sau đó được phân rã thành **Task** hoặc **Sub-task** để phân công rõ ràng cho từng thành viên.

Công việc được tổ chức thành từng **Sprint**. Trước mỗi Sprint, nhóm tiến hành cuộc họp lập kế hoạch, lựa chọn các tác vụ quan trọng từ backlog và thêm vào sprint backlog. Mỗi công việc được đánh giá bằng **Story Point** nhằm ước lượng độ phức tạp và thời gian thực hiện. Jira cung cấp **Sprint Board** với các cột trạng thái như *To Do*, *In Progress*, *In Review* và *Done* giúp nhóm theo dõi tiến độ trực quan.

Trong suốt quá trình phát triển, nhóm thường xuyên cập nhật trạng thái công việc. Các biểu đồ như **Burndown Chart** giúp nhóm theo dõi lượng công việc còn lại, nhận diện sớm nguy cơ trễ hạn. Các báo cáo như **Sprint Report** và **Velocity Chart** cung cấp cái nhìn tổng quan về năng suất làm việc, từ đó giúp nhóm đưa ra những điều chỉnh phù hợp trong các Sprint tiếp theo.

Việc áp dụng Jira vào quản lý dự án đã góp phần quan trọng giúp nhóm triển khai hệ thống một cách hiệu quả, có tổ chức và dễ kiểm soát. Ngoài việc theo dõi tiến độ, Jira còn đóng vai trò như một công cụ hỗ trợ phối hợp nhóm, phản hồi nhanh với thay đổi và cải tiến quy trình làm việc liên tục.

5.2 Phân công nhiệm vụ của từng thành viên trong nhóm

5.2.1. Khởi tạo dự án, thiết lập môi trường

Thành viên 1:

- Tạo project frontend bằng ReactJS và cấu hình Tailwind CSS.
- Thiết kế giao diện trang chủ, trang danh sách món ăn, chi tiết món, giỏ hàng và trang quản trị cơ bản.
- Tạo các component reusable: Header, Footer, ProductCard, Modal, Loading,...
- Kết nối API sử dụng Axios; thực hiện fetch danh sách món ăn, chi tiết món và đơn hàng mẫu.
- Tạo project backend bằng Node.js và Express.
- Cấu hình kết nối MongoDB bằng Mongoose; tạo schema mẫu cho sản phẩm và đơn hàng.
- Thiết kế các API RESTful cho sản phẩm: GET, POST, PUT, DELETE.
- Cấu hình CORS, router và xử lý lỗi cơ bản.
- Tích hợp frontend với backend; kiểm tra khả năng hiển thị dữ liệu thực tế từ server.
- Đóng gói dự án bằng Docker, viết Dockerfile và docker-compose.yml cơ bản.
- Cấu hình GitHub repository, commit các file cần thiết và quản lý source code.

Thành viên 2:

- Tạo file server.js ban đầu để in dòng chữ "Hello World".
- Cài đặt MongoDB trên máy, kiểm tra kết nối thành công.
- Hỗ trợ test API bằng Postman.

5.2.2. Chức năng cơ bản

Thành viên 1: phụ trách khởi tạo project ReactJS, cấu hình Tailwind và thiết lập hệ thống định tuyến với React Router. Thành viên này cũng xây dựng giao diện cho các trang chính như Trang chủ, Chi tiết món ăn, Giỏ hàng, Đăng nhập/Đăng ký, xử lý việc gọi API hiển thị danh sách sản phẩm, chi tiết món, cũng như thực hiện chức năng đăng ký, đăng nhập người dùng bằng JWT. Giao diện được đảm bảo responsive trên các thiết bị khác nhau.

Thành viên 2: đảm nhận thiết kế sơ bộ cơ sở dữ liệu và xây dựng các API đơn giản như đăng ký, đăng nhập, lấy danh sách món ăn, sử dụng Express và MongoDB.

5.2.3. Hoàn thiện chức năng và UI

Thành viên 1: tiếp tục phát triển giao diện đặt món, xử lý thêm vào giỏ hàng, chọn phương thức thanh toán và gửi đơn hàng qua API. Người này cũng thực hiện hiển thị lịch sử đặt hàng của người dùng, thêm các thông báo phản hồi bằng toast và tinh chỉnh UI để giao diện trực quan, hấp dẫn hơn.

Thành viên 2: hỗ trợ viết các API đơn giản như tạo đơn hàng mới, cập nhật trạng thái đơn, đồng thời test thử bằng Postman để đảm bảo hoạt động đúng.

5.2.4. Testing, Fix bug, Triển khai demo

- **Thành viên 1** chịu trách nhiệm kiểm tra và sửa lỗi giao diện, tối ưu responsive trên thiết bị di động, cải thiện hiệu năng tải trang. Ngoài ra, thành viên này còn chuẩn bị bài thuyết trình, nội dung demo, và thực hiện deploy frontend lẫn backend lên nền tảng Render.
- **Thành viên 2** chỉ hỗ trợ kiểm thử các API cơ bản, xác minh hoạt động của JWT và CORS, đảm bảo backend hoạt động ổn định trước khi trình bày.

Chương 6. KIỂM THỬ

6.1 Chiến lược kiểm thử và công cụ sử dụng

6.1.1. Chiến lược kiểm thử

Trong quá trình phát triển hệ thống bán đồ ăn , việc kiểm thử API là bước không thể thiếu nhằm đảm bảo toàn bộ tính năng backend hoạt động chính xác và bảo mật. Nhóm đã sử dụng công cụ **Postman** để thực hiện kiểm thử các API được xây dựng bằng **Express.js**, đảm bảo các thao tác như đặt món, xác thực người dùng, cập nhật đơn hàng... diễn ra trơn tru từ phía frontend.

Việc kiểm thử API nhằm xác nhận các yêu cầu giữa client và server được xử lý đúng logic, giúp hệ thống vận hành ổn định trong thực tế. Các tiêu chí cụ thể gồm:

- Kiểm tra các thao tác HTTP như GET, POST, PUT, DELETE cho từng endpoint hoạt động như kỳ vọng.
- Đảm bảo các API phản hồi đúng mã trạng thái (200 OK, 201 Created, 401 Unauthorized, 403 Forbidden, 500 Internal Server Error,...).
- Định dạng dữ liệu phản hồi là JSON và phù hợp với frontend.
- Xác minh các nghiệp vụ quan trọng: đăng ký tài khoản, đăng nhập, xem danh sách món, đặt đơn hàng, cập nhật trạng thái đơn,... hoạt động mượt mà.
- Đảm bảo chỉ người dùng đã đăng nhập và có token JWT hợp lệ mới truy cập được các tài nguyên yêu cầu xác thực.
- Ghi nhận lỗi chi tiết để frontend dễ dàng xử lý và hiển thị thông báo chính xác.

6.1.2. Công cụ sử dụng

Nhóm sử dụng **Postman** như một công cụ kiểm thử trực quan để mô phỏng các hành vi của người dùng thông qua API. Việc sử dụng Postman giúp dễ dàng phát hiện lỗi logic hoặc lỗi xử lý dữ liệu trong quá trình phát triển backend.

Các bước thực hiện kiểm thử bằng Postman:

6.1.2.1. Gửi yêu cầu API thử nghiệm:

Nhập URL của từng API, chọn đúng phương thức HTTP (GET, POST, PUT, DELETE).

Đối với các yêu cầu có dữ liệu, chuyển sang tab **Body**, chọn định dạng **raw > JSON** và nhập dữ liệu cần gửi (ví dụ thông tin đăng ký, sản phẩm mới, đơn hàng).

6.1.2.2. Xem phản hồi từ server:

Nhấn **Send** để gửi yêu cầu.

Quan sát **status code** (ví dụ: 200, 401, 500) và **nội dung phản hồi** để đánh giá độ chính xác.

6.1.2.3. Kiểm tra phân quyền bằng token JWT:

Với các API yêu cầu xác thực, vào tab **Headers** và thêm:

Authorization: Bearer <token>

để kiểm tra quyền truy cập của user, admin, hoặc nhân viên.

6.1.2.4. Xem phản hồi từ server:

Gom nhóm các API theo chức năng (người dùng, sản phẩm, đơn hàng, đánh giá...).

Thêm mô tả rõ ràng cho từng request để dễ theo dõi, debug và chia sẻ với các thành viên khác.

6.1.2.5. (Tùy chọn) Kiểm thử tự động bằng script:

Sử dụng tab **Tests** để viết đoạn mã kiểm tra điều kiện sau phản hồi (ví dụ: kiểm tra mã trạng thái, thuộc tính JSON có tồn tại hay không...).

6.2 Kết quả kiểm thử API

The screenshot shows the Postman interface with the following details:

- Request URL:** http://localhost:3000/api/products
- Method:** GET
- Response Status:** 200 OK
- Response Time:** 208 ms
- Response Size:** 3.17 KB
- Response Body (JSON):**

```
[{"_id": "607db106ba42acd30ff5dbbf", "id": 1, "name": "Lẩu thập cẩm đặc biệt", "description": "Nồi lẩu hấp dẫn với nước dùng cay đậm đà, kết hợp thịt bò, hải sản, rau tươi, nấm, mì và nhiều loại topping đầy màu sắc, mang đến trải nghiệm tròn vị cho mọi thực khách.", "price": 289000, "category": "Món chính", "image": "/uploads/file-1753417067837-438185745.jpg", "isActive": true, "createdAt": "2025-07-21T03:16:16.289Z", "updatedAt": "2025-07-25T04:17:49.045Z"}, {"_id": "607db106ba42acd30ff5dbc0", "id": 2, "name": "Bún thịt nướng", "description": "Đĩa bún tươi kèm chả lụa, thịt nướng, đậu hũ chiên và rau sống như xà lách, dưa chuột. Nước chấm chua ngọt, tạo hương vị hấp dẫn", "price": 30000, "category": "Món nhẹ", "image": "/uploads/file-1753364516436-256679294.jpg", "isActive": true, "createdAt": "2025-07-21T03:16:16.289Z", "updatedAt": "2025-07-25T04:17:49.045Z"}]
```

Hình 3.5 Kết quả posman

Chương 7. ĐÁNH GIÁ VÀ KẾT LUẬN

7.1 Những khó khăn gặp phải trong quá trình thực hiện

Trong quá trình phát triển hệ thống cửa hàng bán đồ ăn trực tuyến, nhóm gặp phải nhiều khó khăn về mặt kỹ thuật và quản lý. Đầu tiên là việc tích hợp giữa frontend (ReactJS) và backend (Express + MongoDB), đặc biệt là xử lý luồng xác thực, truyền dữ liệu qua API và đồng bộ trạng thái đơn hàng. Bên cạnh đó, việc cấu hình Docker, triển khai CI/CD với GitHub Actions cũng đòi hỏi hiểu biết sâu về môi trường triển khai, quản lý biến môi trường và phân quyền. Ngoài ra, nhóm cũng mất nhiều thời gian để thiết kế giao diện người dùng thân thiện, tương thích tốt với nhiều thiết bị bằng Tailwind CSS và Radix UI. Việc sử dụng Redux Toolkit để đồng bộ hóa trạng thái giữa client và server cũng yêu cầu cấu trúc logic rõ ràng, tránh xung đột dữ liệu trong giỏ hàng hoặc đơn hàng.

7.2 Bài học rút ra và đề xuất cải thiện trong tương lai

Từ những trải nghiệm trên, nhóm rút ra bài học về tầm quan trọng của việc phân công công việc cụ thể ngay từ đầu, chủ động học tập và thử nghiệm trước với các công nghệ DevOps mới, đồng thời duy trì sự phối hợp chặt chẽ giữa các thành viên, đặc biệt là giữa frontend và backend. Trong tương lai, nhóm đề xuất mở rộng quy trình kiểm thử tự động bằng cách tích hợp các công cụ như Jest hoặc Playwright, triển khai giám sát hệ thống bằng PM2 hoặc Grafana để theo dõi hiệu suất, và xây dựng giao diện quản trị nội bộ để tăng tính tự động hóa và khả năng mở rộng cho hệ thống.

DANH MỤC TÀI LIỆU THAM KHẢO

- [1] NodeJS là gì? Tổng quan kiến thức về Node.JS từ A-Z. (2022, April 25).
<https://vietnix.vn/nodejs-la-gi/>
- [2] TopDev. (2019, July 11). Postman là gì? Hướng dẫn API Testing với Postman – API Platform. TopDev; TopDev - Tech Blog. <https://topdev.vn/blog/postman-la-gi/>
- [3] TopDev. (2018, May 25). MongoDB là gì? Định nghĩa đầy đủ và chi tiết nhất về MongoDB. TopDev. <https://topdev.vn/blog/mongodb-la-gi/>
- [4] Tìm hiểu về Tailwind CSS. (2019, December 15). Viblo.asia.
<https://viblo.asia/p/tim-hieu-ve-tailwind-css-924lJp6WKPM>