

This image shows a full page of white paper with horizontal ruling lines. The lines are evenly spaced and extend across the width of the page, providing a template for handwriting practice or general writing. There are no margins, text, or other markings on the page.

(Ký tên và ghi rõ họ tên)

NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Vĩnh Long, ngày ... tháng năm

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

LỜI CẢM ƠN

Tôi xin gửi lời cảm ơn sâu sắc đến Thầy đã hướng dẫn tôi trong quá trình viết bài báo cáo Đồ án “Chuyên ngành” Trong thời gian qua, tôi đã học được nhiều kiến thức mới trong lĩnh vực của ngành Công nghệ thông tin, trong suốt quá trình nghiên cứu cho đến kết thúc đồ án Thông qua bài báo cáo Đồ án này, tôi xin gửi lời cảm ơn đến thầy Phạm Minh Đương giảng viên Khoa Công nghệ thông tin, đã đem lại cho tôi những kiến thức hữu ích thông qua bài báo cáo Đồ án Chuyên ngành với chủ đề **“Xây dựng WebSite Nhà Hàng Ẩm thực Phương Nam Vĩnh Long”** Tôi cũng gửi lời cảm ơn đến Trường Đại học Trà Vinh, Trường Kỹ thuật và Công nghệ, Khoa Công nghệ thông tin đã tạo đủ điều kiện cũng như các kiến thức tài liệu học tập, tài liệu tham khảo để tôi hoàn thành Đồ án Chuyên ngành Trong quá trình viết báo cáo, tôi còn nhiều sai sót và khuyết điểm trong quá trình tìm hiểu và viết báo cáo Tôi rất mong nhận được sự đánh giá đóng góp ý kiến của Thầy để tôi khắc phục và hoàn thiện hơn

Tôi xin chân thành cảm ơn!

MỤC LỤC

LỜI CẢM ƠN	6
MỤC LỤC	7
DANH MỤC BẢNG BIỂU	11
TÓM TẮT ĐỒ ÁN CHUYÊN NGÀNH	12
MỞ ĐẦU	13
CHƯƠNG 1. TỔNG QUAN VỀ NỘI DUNG NGHIÊN CỨU	19
1.1. Bối cảnh và tính cấp thiết	19
1.2. Phạm vi và trọng tâm nghiên cứu	19
1.3. Các thành phần chính của nghiên cứu	20
1.3.1. Phân tích nhu cầu và đặc thù của nhà hàng	20
1.4. Thiết kế kiến trúc và chức năng website	20
1.5. Lựa chọn công nghệ phù hợp	21
1.6. Tối ưu trải nghiệm người dùng (UX/UI)	22
1.7. Phương pháp nghiên cứu	23
1.7.1. Nghiên cứu đối thủ cạnh tranh	23
1.8. Phát triển và kiểm thử	23
1.9. Đánh giá và cải thiện	23
1.10. Mục tiêu cuối cùng	24
CHƯƠNG 2. NGHIÊN CỨU LÝ THUYẾT	25
2.1. TỔNG QUAN VỀ HỆ THỐNG WEB	25
2.1.1. Kiến trúc ClientServer	25
2.1.2. Mô hình MVC (ModelViewController)	26
2.2. CÔNG NGHỆ FRONTEND	27

2.2.1. HTML (HyperText Markup Language)	27
2.2.2. CSS (Cascading Style Sheets)	29
2.2.3. Tailwind CSS	31
2.2.4. JavaScript (ES6+)	33
2.2.5. GSAP (GreenSock Animation Platform)	36
2.3. CÔNG NGHỆ BACKEND	38
2.3.1. Nodejs	38
2.3.2. Expressjs	40
2.3.3. MySQL	43
2.3.4. RESTful API	47
2.3.5. JWT (JSON Web Token)	50
2.3.6. Bcrypt	53
2.4. CÔNG CỤ HỖ TRỢ	54
2.4.1. Visual Studio Code	54
2.4.2. Postman	54
2.4.3. Git & GitHub	55
2.4.4. Docker	56
2.4.5. Nodejs vs Java (Spring Boot)	61
2.5. CÁC NGHIỆP VỤ LIÊN QUAN ĐẾN ĐỀ TÀI	63
2.5.1. Nghiệp vụ quản lý nhà hàng	63
2.5.2. Nghiệp vụ quản lý khách hàng (CRM)	64
2.5.3. Nghiệp vụ marketing và truyền thông	65
2.5.4. Nghiệp vụ báo cáo và thống kê	65
2.6. CÁC CÔNG TRÌNH NGHIÊN CỨU LIÊN QUAN	66

2.6.1. Nghiên cứu trong nước	66
2.6.2. Nghiên cứu quốc tế	67
2.7. KẾT LUẬN CHƯƠNG	68
CHƯƠNG 3. HIỆN THỰC HÓA NGHIÊN CỨU	69
3.1. Mô hình thực thể kết hợp ERD	69
3.2. Mô tả hệ thống	70
3.2.1. Tổng quan hệ thống	70
3.2.2. Kiến trúc hệ thống	71
3.2.3. Cấu trúc dự án	72
3.3. XÁC ĐỊNH CÁC YÊU CẦU CHỨC NĂNG CỦA HỆ THỐNG	73
3.3.1. Sơ đồ Usecase	73
3.4. THIẾT KẾ XỬ LÝ HỆ THỐNG	74
3.4.1. Thiết kế API RESTful	74
3.4.2. Sơ đồ tuần tự (Sequence Diagram)	76
3.5. MÔ HÌNH HOẠT ĐỘNG CỦA HỆ THỐNG	78

DANH MỤC HÌNH ẢNH

Hình 2.1: Ngôn Ngữ HTML	27
Hình 2.2: Bảng định kiểu CSS	30
Hình 2.3: Framework TailWindCSS	32
Hình 2.4: Ngôn ngữ JavaScript	35
Hình 2.5: Framework Backend NodeJs	39
Hình 2.6: Express Js	42
Hình 2.7: Hệ quản trị Cơ sở dữ liệu MySQL	46
Hình 2.9: Phương thức RESTful API	49
Hình 2.10: Visual Studio Code	55
Hình 3.1: Mô hình thực thể kết hợp ERD	69
Hình 3.2: Kiến trúc hệ thống	71
Hình 3.3: Cấu trúc dự án	72
Hình 3.4: Sơ đồ UseCase	73
Hình 3.5: Sơ đồ tuần tự	77
Hình 3.6: Quy trình đặt bàn trực tuyến	78
Hình 3.7: Quy trình đặt hàng	79
Hình 3.8: Quy trình Quản lý đơn hàng	80

DANH MỤC BẢNG BIỂU

<i>Bảng 1: Bảng so sánh Docker và Virtual Machine</i>	<i>56</i>
<i>Bảng 2: Bảng so sánh Nodejs và Java Spring Boot</i>	<i>62</i>

TÓM TẮT ĐỒ ÁN CHUYÊN NGÀNH

Website Nhà hàng Ẩm thực Phương Nam – Vĩnh Long được xây dựng như một không gian số tái hiện trọn vẹn hơi ấm và hương vị miền Tây, mang đến cho người dùng một trải nghiệm trực tuyến hiện đại, trực quan và đầy cảm xúc Dự án hướng đến việc giới thiệu đặc sản miền Tây bằng ngôn ngữ công nghệ, nơi từng món ăn, từng hình ảnh đều được kể lại bằng sự sinh động và tinh tế

Mục tiêu của dự án là tạo nên một hệ thống vừa giới thiệu món ăn vừa hỗ trợ hoạt động của nhà hàng: trình bày thực đơn đa dạng kèm hình ảnh sắc nét; cho phép khách đặt bàn và đặt món trực tuyến nhanh gọn; quản lý album ảnh, tin tức, đơn hàng và dữ liệu vận hành một cách hiệu quả; đồng thời đảm bảo trải nghiệm người dùng mượt mà trên mọi thiết bị

Phần giao diện được tạo dựng bằng HTML, CSS, JavaScript, kết hợp Tailwind CSS để mang lại thiết kế gọn gàng, hiện đại, và GSAP Animation để thêm vào những chuyển động mềm mại, đầy nhịp sống Hệ thống backend sử dụng Nodejs và Expressjs làm lõi xử lý nghiệp vụ, giao tiếp với cơ sở dữ liệu MySQL nhằm quản lý thông tin món ăn, người dùng, đơn hàng Cơ chế xác thực sử dụng JWT đảm bảo an toàn, trong khi Nodemailer hỗ trợ gửi email xác nhận đặt bàn và đơn hàng một cách chuyên nghiệp

Website mang đến bộ tính năng đầy đủ: quản lý thực đơn theo danh mục, đặt bàn trực tuyến, giỏ hàng và đặt món online, album ảnh không gian – món ăn – sự kiện, mục tin tức cập nhật hoạt động, hệ thống đăng ký/đăng nhập bảo mật và giao diện admin toàn diện cho quản lý nội dung

Ý nghĩa của dự án không chỉ nằm ở việc tạo ra một công cụ hỗ trợ vận hành nhà hàng, mà còn ở việc kết nối ẩm thực truyền thống với công nghệ hiện đại Sản phẩm thể hiện tư duy thiết kế, khả năng lập trình và tinh thần đổi mới của sinh viên Công nghệ thông tin, đủ tính thực tiễn để triển khai trong môi trường kinh doanh thực tế và mở ra hướng phát triển cho các giải pháp nhà hàng trong thời đại số

MỞ ĐẦU

LÝ DO CHỌN ĐỀ TÀI

Xu hướng chuyển đổi số: Ngành ẩm thực đang dần số hóa để thích nghi với thói quen tiêu dùng hiện đại Khách hàng ngày càng ưa chuộng đặt bàn, xem thực đơn và đặt món trực tuyến thay vì gọi điện hay đến trực tiếp

Nhu cầu của nhà hàng: Nhiều nhà hàng địa phương, đặc biệt ở vùng miền Tây chưa có website chuyên nghiệp để quảng bá và quản lý kinh doanh hiệu quả

Quảng bá ẩm thực địa phương: Cần một nền tảng để giới thiệu đặc sản, văn hóa ẩm thực Vĩnh Long miền Tây đến du khách trong và ngoài tỉnh

Ứng dụng kiến thức: Đề tài giúp vận dụng kiến thức về lập trình web, cơ sở dữ liệu, thiết kế giao diện vào sản phẩm thực tế

Kỹ năng phát triển: Rèn luyện kỹ năng phân tích, thiết kế hệ thống, lập trình fullstack và triển khai ứng dụng web hoàn chỉnh

MỤC TIÊU

Mục tiêu chung:

Xây dựng website quản lý nhà hàng Ẩm thực Phương Nam Vĩnh Long với đầy đủ chức năng quản lý và hỗ trợ kinh doanh, mang đến trải nghiệm người dùng tốt và giúp nhà hàng hoạt động hiệu quả hơn

Mục tiêu cụ thể:

Phát triển giao diện người dùng:

Thiết kế giao diện responsive, thân thiện, dễ sử dụng

Hiển thị thực đơn, album ảnh, tin tức hấp dẫn và rõ ràng

Xây dựng hệ thống backend:

API RESTful xử lý nghiệp vụ đặt bàn, đặt món, quản lý đơn hàng

Hệ thống xác thực và phân quyền người dùng bảo mật

Quản trị dữ liệu:

Cơ sở dữ liệu lưu trữ thông tin món ăn, khách hàng, đơn hàng

Chức năng CRUD đầy đủ cho admin quản lý

Tối ưu trải nghiệm:

Tốc độ tải trang nhanh

Giao diện mượt mà với animation chuyên nghiệp

Tương thích đa thiết bị (desktop, tablet, mobile)

NỘI DUNG NGHIÊN CỨU

Phần 1: Nghiên cứu lý thuyết

Công nghệ web hiện đại: HTML, CSS, JavaScript, Nodejs, Expressjs

Cơ sở dữ liệu: MySQL, thiết kế cơ sở dữ liệu quan hệ

API RESTful: Thiết kế và triển khai API chuẩn REST

Bảo mật web: JWT, mã hóa mật khẩu, phòng chống SQL Injection, XSS

Phần 2: Phân tích và thiết kế hệ thống

Phân tích yêu cầu: Use case, user story của khách hàng và admin

Thiết kế cơ sở dữ liệu: ER diagram, schema database

Thiết kế giao diện: Wireframe, mockup, UI/UX

Thiết kế kiến trúc: Sơ đồ luồng dữ liệu, API endpoints

Phần 3: Triển khai các chức năng

Chức năng người dùng (Customer)

Xem trang chủ, giới thiệu nhà hàng

Duyệt thực đơn theo danh mục

Xem chi tiết món ăn, album ảnh

Đặt bàn trực tuyến theo ngày giờ

Đặt món, thêm giỏ hàng, thanh toán online

Đăng ký, đăng nhập, quên mật khẩu

Xem lịch sử đơn hàng, đặt bàn

Đọc tin tức, khuyến mãi

Chức năng quản trị (Admin)

Đăng nhập admin với Google OAuth

Quản lý món ăn (thêm, sửa, xóa, cập nhật giá)

Quản lý danh mục món ăn

Quản lý đơn hàng, cập nhật trạng thái

Quản lý đặt bàn, xác nhận/hủy

Quản lý album ảnh nhà hàng

Quản lý tin tức, bài viết

Xem báo cáo doanh thu, thống kê

Phần 4: Kiểm thử và tối ưu

Kiểm thử chức năng (Unit test, Integration test)

Kiểm thử giao diện trên nhiều trình duyệt

Tối ưu hiệu năng (lazy loading, caching, minify)

Kiểm thử bảo mật, xử lý lỗi

ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU

Đối tượng nghiên cứu:

Hệ thống website: Ứng dụng web quản lý nhà hàng fullstack

Công nghệ: Nodejs, Expressjs, MySQL, HTML/CSS/JavaScript, Tailwind CSS

Nhiệm vụ: Quy trình đặt bàn, đặt món, quản lý đơn hàng của nhà hàng

Phạm vi nghiên cứu

Về không gian:

Triển khai cho: Nhà hàng Ẩm thực Phương Nam tại Vĩnh Long

Phục vụ: Khách hàng địa phương và du khách đến Vĩnh Long

Mở rộng: Có thể áp dụng cho các nhà hàng tương tự

Về chức năng:

Triển khai đầy đủ:

Hiển thị thực đơn, album ảnh, tin tức

Đặt bàn, đặt món online

Giỏ hàng và thanh toán

Xác thực người dùng

Quản trị admin cơ bản

PHƯƠNG PHÁP NGHIÊN CỨU

1. Phương pháp nghiên cứu tài liệu:

Tra cứu tài liệu về công nghệ web: Nodejs, Expressjs, MySQL

Nghiên cứu các framework: Tailwind CSS, GSAP Animation

Tham khảo các website nhà hàng hiện có để học hỏi UX/UI

Đọc tài liệu chuẩn về API RESTful, bảo mật web

2. Phương pháp phân tích và thiết kế hệ thống

Phân tích yêu cầu: Khảo sát nhu cầu thực tế của nhà hàng và khách hàng

Thiết kế UML: Vẽ Use case diagram, Activity diagram, ER diagram

Thiết kế UI/UX: Sử dụng Figma để mockup giao diện trước khi code

Thiết kế database: Chuẩn hóa cơ sở dữ liệu (3NF), tối ưu quan hệ bảng

3. Phương pháp lập trình

Agile/Scrum: Chia nhỏ dự án thành các sprint, phát triển từng module

Git version control: Quản lý source code, làm việc nhóm hiệu quả

Code review: Kiểm tra chéo code để đảm bảo chất lượng

4. Phương pháp kiểm thử

Unit testing: Kiểm tra từng function/API riêng lẻ

Integration testing: Kiểm tra tích hợp giữa frontend/backend

Manual testing: Test thủ công các luồng nghiệp vụ chính

Crossbrowser testing: Kiểm tra trên Chrome, Firefox, Safari, Edge

Responsive testing: Kiểm tra trên desktop, tablet, mobile

5. Phương pháp thực nghiệm

Triển khai pilot: Deploy thử nghiệm trên localhost và test server

Thu thập phản hồi: Cho người dùng thử nghiệm và ghi nhận ý kiến

Đo lường hiệu năng: Dùng Lighthouse, PageSpeed Insights để đánh giá

Điều chỉnh và cải tiến: Sửa lỗi, tối ưu dựa trên kết quả test

6. Phương pháp so sánh đánh giá

So sánh với các website nhà hàng khác về tính năng, UX/UI

Đánh giá ưu nhược điểm của từng công nghệ đã chọn

Rút ra bài học và hướng phát triển trong tương lai

CHƯƠNG 1. TỔNG QUAN VỀ NỘI DUNG NGHIÊN CỨU

1.1. Bối cảnh và tính cấp thiết

Sự phát triển mạnh mẽ của Internet và các công nghệ web hiện đại đã thay đổi cách thức kinh doanh của ngành ẩm thực trên toàn cầu. Việc xây dựng website cho nhà hàng không còn là lựa chọn mà đã trở thành yêu cầu cần thiết để tồn tại và phát triển trong thời đại số. Điều này đặc biệt quan trọng đối với Nhà hàng Ẩm thực Phương Nam Vĩnh Long, nơi lưu giữ tinh hoa ẩm thực miền Tây với những món ăn đậm đà bản sắc quê hương.

Một hệ thống website toàn diện không chỉ giúp nhà hàng quảng bá thương hiệu, giới thiệu thực đơn phong phú, mà còn cung cấp các dịch vụ đặt bàn và đặt món trực tuyến, đáp ứng nhu cầu của thực khách hiện đại. Những người ngày càng ưa chuộng sự tiện lợi, nhanh chóng trong việc tìm kiếm thông tin và đặt dịch vụ qua Internet. Đây chính là động lực thúc đẩy việc phát triển đề tài này.

1.2. Phạm vi và trọng tâm nghiên cứu

Đề tài tập trung vào việc xây dựng một hệ thống website fullstack hoàn chỉnh, bao gồm cả Frontend (giao diện người dùng) và Backend (xử lý nghiệp vụ, quản trị dữ liệu). Đây là yếu tố then chốt quyết định sự thành công của nhà hàng trong môi trường số, ảnh hưởng trực tiếp đến:

Ấn tượng ban đầu: Giao diện đẹp mắt, chuyên nghiệp tạo niềm tin với khách hàng.

Trải nghiệm người dùng: Dễ sử dụng, tìm kiếm thông tin nhanh chóng.

Tỷ lệ chuyển đổi: Khách hàng dễ dàng đặt bàn, đặt món từ website.

Quản lý hiệu quả: Hệ thống backend giúp nhà hàng quản lý đơn hàng, thực đơn một cách khoa học.

Giao diện phải được thiết kế sao cho vừa hấp dẫn về mặt thẩm mỹ, vừa thân thiện với người dùng, đồng thời đảm bảo hiệu suất tốt và hiển thị tối ưu trên mọi

thiết bị (máy tính để bàn, máy tính bảng, điện thoại di động) Backend cần xử lý nghiệp vụ chính xác, bảo mật, và hỗ trợ quản trị dễ dàng

1.3. Các thành phần chính của nghiên cứu

1.3.1. Phân tích nhu cầu và đặc thù của nhà hàng

Nhận diện bản sắc: Nghệ thuật ẩm thực miền Nam Việt Nam với hương vị đậm đà, đặc trưng là điểm nhấn cần được thể hiện rõ nét qua màu sắc ấm áp (cam, đỏ, vàng), hình ảnh món ăn chất lượng cao, và bố cục giao diện gần gũi, mộc mạc nhưng hiện đại

Xác định đối tượng khách hàng: Nghiên cứu hướng đến hai nhóm chính:

Cư dân địa phương (Vĩnh Long và vùng lân cận)

Du khách trong nước muốn trải nghiệm ẩm thực miền Tây

Khách hàng trẻ tuổi, quen với công nghệ và ưa thích đặt món online

1.4. Thiết kế kiến trúc và chức năng website

Frontend (Giao diện người dùng):

Trang chủ: Giới thiệu tổng quan, banner hấp dẫn, highlights món ăn nổi bật

Thực đơn: Hiển thị món ăn theo danh mục, hình ảnh đẹp, giá cả rõ ràng

Album ảnh: Trưng bày không gian nhà hàng, món ăn, sự kiện

Tin tức: Cập nhật khuyến mãi, sự kiện, bài viết về ẩm thực

Đặt bàn: Form đặt bàn trực tuyến theo ngày giờ

Đặt món: Giỏ hàng, thanh toán online

Liên hệ: Thông tin địa chỉ, bản đồ, form liên hệ

Backend (Quản trị và xử lý nghiệp vụ):

Hệ thống API RESTful xử lý các yêu cầu từ frontend

Quản lý món ăn, danh mục, giá cả

Quản lý đơn hàng, đặt bàn, trạng thái

Quản lý album ảnh, tin tức, khuyến mãi

Hệ thống xác thực, phân quyền người dùng và admin

Thống kê, báo cáo doanh thu

1.5. Lựa chọn công nghệ phù hợp

Nghiên cứu đánh giá và lựa chọn stack công nghệ tối ưu cho dự án:

Frontend:

HTML: Cấu trúc semantic, hỗ trợ SEO

CSS: Styling hiện đại với animation, transition

Tailwind CSS: Framework CSS utilityfirst giúp phát triển nhanh, giao diện đẹp, responsive tốt

JavaScript (ES6+): Xử lý tương tác, gọi API

GSAP Animation: Tạo hiệu ứng chuyển động mượt mà, chuyên nghiệp

Backend:

Nodejs + Expressjs: Nền tảng JavaScript fullstack, hiệu suất cao, dễ triển khai

MySQL: Cơ sở dữ liệu quan hệ ổn định, phù hợp với dữ liệu có cấu trúc

JWT: Xác thực bảo mật, stateless

Nodemailer: Gửi email xác nhận, thông báo

Lý do lựa chọn:

Chi phí thấp (opensource, miễn phí)

Cộng đồng lớn, tài liệu phong phú

Hiệu năng tốt, dễ mở rộng

Phù hợp với quy mô nhà hàng vừa và nhỏ

1.6. Tối ưu trải nghiệm người dùng (UX/UI)

Nghiên cứu tập trung vào các yếu tố:

Về hình ảnh:

Sử dụng ảnh món ăn chất lượng cao, góc chụp hấp dẫn

Màu sắc ấm áp, phù hợp với bản sắc miền Tây

Icon, illustration gần gũi, dễ hiểu

Về bố cục:

Thiết kế grid system rõ ràng, cân đối

Khoảng trắng hợp lý, không gây rối mắt

Hierarchy thông tin rõ ràng (quan trọng → ít quan trọng)

Về tương tác:

Navigation đơn giản, dễ tìm kiếm thông tin

Form đặt món, đặt bàn ngắn gọn, không phức tạp

Feedback tức thì khi người dùng thao tác

Animation mượt mà, không gây khó chịu

Về hiệu năng:

Lazy loading cho hình ảnh

Minify CSS, JavaScript

Responsive hoàn toàn trên mọi thiết bị

1.7. Phương pháp nghiên cứu

1.7.1. Nghiên cứu đối thủ cạnh tranh

Phân tích thiết kế, chức năng của các website nhà hàng khác (trong nước và quốc tế) để rút ra bài học về:

Điểm mạnh cần học hỏi

Điểm yếu cần tránh

Xu hướng thiết kế hiện đại

1.8. Phát triển và kiểm thử

Phát triển backend API và database

Tích hợp frontend backend

Kiểm thử chức năng, UX trên các thiết bị:

Desktop (Chrome, Firefox, Safari, Edge)

Tablet (iPad, Android tablet)

Mobile (iOS, Android)

Kiểm thử hiệu năng (PageSpeed Insights, Lighthouse)

Kiểm thử bảo mật (SQL Injection, XSS, CSRF)

1.9. Đánh giá và cải thiện

Thu thập phản hồi từ người dùng thử nghiệm

Phân tích số liệu: thời gian truy cập, tỷ lệ thoát, conversion rate

Điều chỉnh, tối ưu dựa trên dữ liệu thực tế

1.10. Mục tiêu cuối cùng

=> Mục tiêu của nghiên cứu không chỉ đơn thuần là tạo ra một giao diện đẹp mắt, mà còn xây dựng một hệ thống website toàn diện, hoạt động hiệu quả, góp phần:

Hỗ trợ chuyển đổi số cho nhà hàng Ẩm thực Phương Nam

Nâng cao khả năng cạnh tranh trên thị trường F&B địa phương

Tăng doanh thu thông qua kênh online

Quảng bá văn hóa ẩm thực miền Tây đến công chúng rộng rãi hơn

Tạo ra sản phẩm thực tế có thể triển khai ngay, không chỉ là đồ án học thuật

CHƯƠNG 2. NGHIÊN CỨU LÝ THUYẾT

2.1. TỔNG QUAN VỀ HỆ THỐNG WEB

2.1.1. Kiến trúc ClientServer

Hệ thống website nhà hàng được xây dựng dựa trên mô hình kiến trúc ClientServer, trong đó:

Client (Phía người dùng):

Trình duyệt web (Chrome, Firefox, Safari, Edge) hiển thị giao diện

Gửi yêu cầu (HTTP Request) đến server

Nhận phản hồi (HTTP Response) và hiển thị cho người dùng

Xử lý tương tác người dùng thông qua JavaScript

Server (Phía máy chủ):

Nhận và xử lý các yêu cầu từ client

Truy vấn cơ sở dữ liệu

Xử lý logic nghiệp vụ (đặt bàn, đặt món, xác thực)

Trả về dữ liệu dạng JSON hoặc HTML

Luồng hoạt động:

- | |
|--|
| 1 User nhập URL → Browser gửi HTTP Request |
| 2 Server nhận request → Xử lý logic → Truy vấn Database |
| 3 Database trả kết quả → Server xử lý → Trả HTTP Response |
| 4 Browser nhận response → Render giao diện → Hiển thị cho user |

2.1.2. Mô hình MVC (ModelViewController)

Dự án áp dụng mô hình MVC để tổ chức code rõ ràng, dễ bảo trì:

Model (Mô hình dữ liệu):

Đại diện cho cấu trúc dữ liệu trong database

Xử lý logic truy vấn, thêm, sửa, xóa dữ liệu

Ví dụ: Model món ăn, Model đơn hàng, Model người dùng

View (Giao diện):

Hiển thị dữ liệu cho người dùng

HTML/CSS/JavaScript ở phía Frontend

Không chứa logic xử lý nghiệp vụ

Controller (Điều khiển):

Nhận request từ client

Gọi Model để xử lý dữ liệu

Trả response về cho View

Ví dụ: MenuController, AuthController, CartController

2.2. CÔNG NGHỆ FRONTEND

2.2.1. HTML (HyperText Markup Language)

Giới thiệu:

HTML là phiên bản mới nhất của ngôn ngữ đánh dấu siêu văn bản, được sử dụng để cấu trúc nội dung trên web

Tính năng chính:

Semantic Elements: <header>, <nav>, <section>, <article>, <footer> giúp cấu trúc rõ ràng, hỗ trợ SEO

Form Controls: Input types mới (email, date, number) với validation tích hợp sẵn

Multimedia: <video>, <audio> hỗ trợ phát media không cần plugin

Canvas & SVG: Vẽ đồ họa, animation trực tiếp trên trình duyệt

Local Storage: Lưu trữ dữ liệu phía client (giỏ hàng, phiên làm việc)



Hình 2.1: Ngôn Ngữ HTML

Ứng dụng trong dự án:

<! Cấu trúc semantic >

```
<header id="navbarcontainer"></header>
```

```
<main>
```

```
    <section class="hero"></section>
```

```
    <section class="menu"></section>
```

```
</main>
```

```
<footer id="footercontainer"></footer>
```

<! Form đặt bàn >

```
<form>
```

```
    <input type="email" required placeholder="Email của bạn">
```

```
    <input type="date" required>
```

```
    <input type="time" required>
```

```
</form>
```

2.2.2. CSS (Cascading Style Sheets)

Giới thiệu:

CSS là ngôn ngữ định dạng giao diện, giúp trang web đẹp mắt và thu hút

Tính năng quan trọng:

Flexbox: Bố cục linh hoạt, căn chỉnh phần tử dễ dàng

Grid Layout: Chia lưới phức tạp cho layout chuyên nghiệp

Transitions & Animations: Hiệu ứng chuyển động mượt mà

Media Queries: Responsive design cho mọi kích thước màn hình

Custom Properties (CSS Variables): Quản lý màu sắc, kích thước tập trung

Ứng dụng trong dự án:

/ Flexbox cho navigation /

```
navbar {  
    display: flex;  
    justifycontent: spacebetween;  
    alignitems: center;  
}
```

/ Grid cho thực đơn /

```
menugrid {  
    display: grid;  
    gridtemplatecolumns: repeat(autofit, minmax(300px, 1fr));  
    gap: 2rem;  
}
```

/ Responsive /

```
@media (maxwidth: 768px) {  
  
    menugrid {  
  
        gridtemplatecolumns: 1fr;  
  
    }  
  
}
```

/ Animation hover /

```
menuitem:hover {  
  
    transform: scale(105);  
  
    transition: transform 0.3s ease;  
  
}
```



Hình 2.2: Bảng định kiểu CSS

2.2.3. Tailwind CSS

Giới thiệu:

Tailwind CSS là framework CSS utilityfirst, cung cấp các class có sẵn để xây dựng giao diện nhanh chóng mà không cần viết CSS custom

Ưu điểm:

Phát triển nhanh: Sử dụng class có sẵn thay vì viết CSS từ đầu

Responsive dễ dàng: Class prefix (sm:, md:, lg:, xl:) cho từng breakpoint

Customizable: Cấu hình màu sắc, spacing, font theo nhu cầu

File size nhỏ: PurgeCSS tự động xóa class không sử dụng

Consistent design: Hệ thống spacing, color palette thống nhất

Ứng dụng trong dự án:

<! Card món ăn >

```
<div class="bgwhite roundedxl shadowlg overflowhidden hover:shadow2xl  
transitionsshadow duration300">
```

```
<img src="" class="wfull h64 objectcover">
```

```
<div class="p6">
```

```
<h3 class="text2xl fontbold textgray800 mb2">Bánh xèo</h3>
```

```
<p class="textgray600 mb4">Món ăn đặc sản miền Tây</p>
```

```
<div class="flex justifybetween itemscenter">
```

```
<span class="textorange600 fontbold textxl">45000đ</span>
```

```
<button class="bgorange600 textwhite px6 py2 roundedfull  
hover:bgorange700">
```

```
Đặt món
```

```
</button>
```

```
</div>
```

```
</div>

</div>

<! Responsive navigation >

<nav class="hidden md:flex spacex8">

  <a class="textwhite hover:textorange300 transition">Trang chủ</a>

  <a class="textwhite hover:textorange300 transition">Thực đơn</a>

</nav>
```



Hình 2.3: Framework TailWindCSS

2.2.4. JavaScript (ES6+)

Giới thiệu:

JavaScript là ngôn ngữ lập trình phía client, tạo tính tương tác cho website ES6+ (ECMAScript 2015+) mang đến nhiều tính năng hiện đại

Tính năng ES6+ sử dụng:

Arrow Functions: Cú pháp ngắn gọn cho function

Template Literals: Nối chuỗi dễ dàng với backtick

Destructuring: Trích xuất giá trị từ object/array

Async/Await: Xử lý bất đồng bộ rõ ràng hơn Promise

Modules: Import/Export chia nhỏ code thành các file

Ứng dụng trong dự án:

```
// Fetch API để lấy dữ liệu món ăn
async function loadMenu() {
  try {
    const response = await fetch('http://localhost:3000/api/menu');
    const result = await response.json();

    if (result.success) {
      displayMenu(result.data);
    }
  } catch (error) {
    console.error('Lỗi load menu:', error);
  }
}
```

```
// Thêm món vào giỏ hàng

const addToCart = (productId, quantity) => {

  const token = localStorage.getItem('token');

  fetch('http://localhost:3000/api/cart/add', {

    method: 'POST',

    headers: {

      'ContentType': 'application/json',

      'Authorization': `Bearer ${token}`

    },

    body: JSON.stringify({ ma_mon: productId, so_luong: quantity })

  })

  then(res => res.json())

  then(data => {

    if (data.success) {

      showNotification('Đã thêm vào giỏ hàng!');

      updateCartBadge();

    }

  });

};

// Event delegation

document.addEventListener('DOMContentLoaded', () => {
```

```
loadMenu();  
  
loadCart();  
  
setupEventListeners();  
  
});
```



Hình 2.4: Ngôn ngữ JavaScript

2.2.5. GSAP (GreenSock Animation Platform)

Giới thiệu:

GSAP là thư viện JavaScript mạnh mẽ để tạo animation mượt mà, hiệu suất cao

Tính năng nổi bật:

Performance: Sử dụng GPU acceleration, chạy 60fps

Timeline: Tạo chuỗi animation phức tạp

ScrollTrigger: Kích hoạt animation khi scroll

Ease functions: Nhiều hàm easing cho animation tự nhiên

Ứng dụng trong dự án:

```
// Animation fade in khi scroll

gsap.registerPlugin(ScrollTrigger);

gsap.from('menuitem', {
  scrollTrigger: {
    trigger: 'menusection',
    start: 'top 80%',
  },
  opacity: 0,
  y: 50,
  duration: 0.8,
  stagger: 0.2
});
```

```
// Animation hero section
```

```
gsapfrom('herotitle', {  
    opacity: 0,  
    y: 50,  
    duration: 1,  
    ease: 'power3out'  
});
```

```
gsapfrom('heroimage', {  
    opacity: 0,  
    scale: 08,  
    duration: 12,  
    delay: 05,  
    ease: 'backout(17)'  
});
```

2.3. CÔNG NGHỆ BACKEND

2.3.1. Nodejs

Giới thiệu:

Nodejs là môi trường runtime cho phép chạy JavaScript ở phía server, được xây dựng trên Chrome V8 Engine

Đặc điểm:

Nonblocking I/O: Xử lý bất đồng bộ, hiệu suất cao với nhiều request đồng thời

Eventdriven: Kiến trúc hướng sự kiện, phù hợp với realtime application

NPM: Hệ sinh thái package phong phú nhất thế giới

Single language: Dùng JavaScript cho cả frontend và backend

Scalable: Dễ dàng mở rộng theo chiều ngang

Ứng dụng trong dự án:

```
// Server cơ bản

const express = require('express');

const app = express();

const PORT = 3000;

app.listen(PORT, () => {
  console.log(`  Server đang chạy tại http://localhost:${PORT}`);
});
```



Hình 2.5: Framework Backend NodeJs

2.3.2. Expressjs

Giới thiệu:

Expressjs là framework web minimal và linh hoạt cho Nodejs, cung cấp các tính năng mạnh mẽ để xây dựng web và mobile application

Tính năng chính:

Routing: Định tuyến URL dễ dàng

Middleware: Xử lý request/response theo chuỗi

Template Engine: Hỗ trợ EJS, Pug, Handlebars

Error Handling: Xử lý lỗi tập trung

Static Files: Serve file tĩnh (HTML, CSS, JS, images)

Ứng dụng trong dự án:

```
const express = require('express');

const cors = require('cors');

const app = express();

// Middleware

app.use(cors());

app.use(express.json());

app.use(express.static('public'));

// Routes

const menuRoutes = require('/routes/menu');

const authRoutes = require('/routes/auth');

const cartRoutes = require('/routes/cart');
```

```
appuse('/api/menu', menuRoutes);

appuse('/api/auth', authRoutes);

appuse('/api/cart', cartRoutes);


// Error handling

appuse((err, req, res, next) => {

    console.error(errstack);

    res.status(500).json({

        success: false,

        message: 'Có lỗi xảy ra!'

    });

});
```

Cấu trúc Routes:

```
// routes/menujs

const express = require('express');

const router = express.Router();

const db = require('/config/database');


// Lấy tất cả món ăn

router.get('/', async (req, res) => {

    try {

        const [rows] = await db.query('SELECT FROM mon_an

        WHERE trang_thai = 1');

        res.json({ success: true, data: rows });

    }
```

```
    } catch (error) {  
        resstatus(500)json({ success: false, message: errormessage });  
    }  
});  
  
// Lấy chi tiết món ăn  
router.get('/:id', async (req, res) => {  
    try {  
        const [rows] = await dbquery('SELECT * FROM mon_an  
WHERE ma_mon = ?', [req.params.id]);  
        res.json({ success: true, data: rows[0] });  
    } catch (error) {  
        resstatus(500)json({ success: false, message: errormessage });  
    }  
});  
  
module.exports = router;
```



Hình 2.6: Express Js

2.3.3. MySQL

Giới thiệu:

MySQL là hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) mã nguồn mở phổ biến nhất thế giới

Đặc điểm:

ACID Compliance: Đảm bảo tính toàn vẹn dữ liệu

High Performance: Xử lý hàng triệu query mỗi giây

Scalability: Dễ dàng mở rộng

Security: Bảo mật mạnh mẽ với user privilege system

Crossplatform: Chạy trên Windows, Linux, macOS

Cấu trúc Database trong dự án:

Bảng người dùng

```
CREATE TABLE nguoi_dung (  
    ma_nguoi_dung INT PRIMARY KEY AUTO_INCREMENT,  
    ten_dang_nhap VARCHAR(50) UNIQUE NOT NULL,  
    mat_khau_hash VARCHAR(255) NOT NULL,  
    email VARCHAR(100) UNIQUE NOT NULL,  
    ho_ten VARCHAR(100),  
    so_dien_thoai VARCHAR(15),  
    dia_chi TEXT,  
    ngay_tao DATETIME DEFAULT CURRENT_TIMESTAMP  
);
```


Bảng danh mục

```
CREATE TABLE danh_muc (  
    ma_danh_muc INT PRIMARY KEY AUTO_INCREMENT,  
    ten_danh_muc VARCHAR(100) NOT NULL,  
    mo_ta TEXT,  
    thu_tu INT DEFAULT 0  
);
```

Bảng món ăn

```
CREATE TABLE mon_an (  
    ma_mon INT PRIMARY KEY AUTO_INCREMENT,  
    ten_mon VARCHAR(200) NOT NULL,  
    ma_danh_muc INT,  
    gia DECIMAL(10,2) NOT NULL,  
    mo_ta TEXT,  
    hinh_anh VARCHAR(500),  
    trang_thai TINYINT DEFAULT 1,  
    ngay_tao DATETIME DEFAULT CURRENT_TIMESTAMP,  
    FOREIGN KEY (ma_danh_muc)  
REFERENCES danh_muc(ma_danh_muc)  
);
```

Kết nối MySQL với Nodejs:

```
// config/databasejs

const mysql = require('mysql2/promise');
require('dotenv').config();

const pool = mysqlcreatePool({
  host: processenvDB_HOST,
  user: processenvDB_USER,
  password: processenvDB_PASSWORD,
  database: processenvDB_NAME,
  port: processenvDB_PORT || 3306,
  waitForConnections: true,
  connectionLimit: 10,
  queueLimit: 0
});

// Test connection
poolgetConnection()
  .then(connection => {
    consolelog('✔ Kết nối database thành công!');
    connectionrelease();
  })
  .catch(err => {
    consoleerror('✗ Lỗi kết nối database:', err);
  });

moduleexports = pool;
```



Hình 2.7: Hệ quản trị Cơ sở dữ liệu MySQL

2.3.4. RESTful API

Giới thiệu:

REST (Representational State Transfer) là kiến trúc thiết kế API, sử dụng HTTP methods để thực hiện CRUD operations

Nguyên tắc REST:

Stateless: Mỗi request độc lập, không lưu trạng thái phiên

ClientServer: Tách biệt frontend và backend

Uniform Interface: Sử dụng chuẩn HTTP methods

Cacheable: Response có thể cache để tăng hiệu năng

HTTP Methods:

GET: Lấy dữ liệu (Read)

POST: Tạo mới dữ liệu (Create)

PUT/PATCH: Cập nhật dữ liệu (Update)

DELETE: Xóa dữ liệu (Delete)

API Endpoints trong dự án:

// Menu API

GET /api/menu // Lấy tất cả món ăn

GET /api/menu/:id // Lấy chi tiết món ăn

GET /api/menu/category/:id // Lấy món ăn theo danh mục

// Auth API

POST /api/auth/register // Đăng ký tài khoản

POST /api/auth/login // Đăng nhập

POST /api/auth/forgotpassword // Quên mật khẩu

POST /api/auth/resetpassword // Đặt lại mật khẩu

// Cart API

GET /api/cart // Lấy giỏ hàng

POST /api/cart/add // Thêm vào giỏ

PUT /api/cart/update/:id // Cập nhật số lượng

DELETE /api/cart/remove/:id // Xóa khỏi giỏ

// Order API

GET /api/orders // Lấy danh sách đơn hàng

POST /api/orders // Tạo đơn hàng mới

GET /api/orders/:id // Chi tiết đơn hàng

PUT /api/orders/:id/status // Cập nhật trạng thái

// Album API

GET /api/albums // Lấy tất cả album

GET /api/albums/category/:loai // Album theo loại

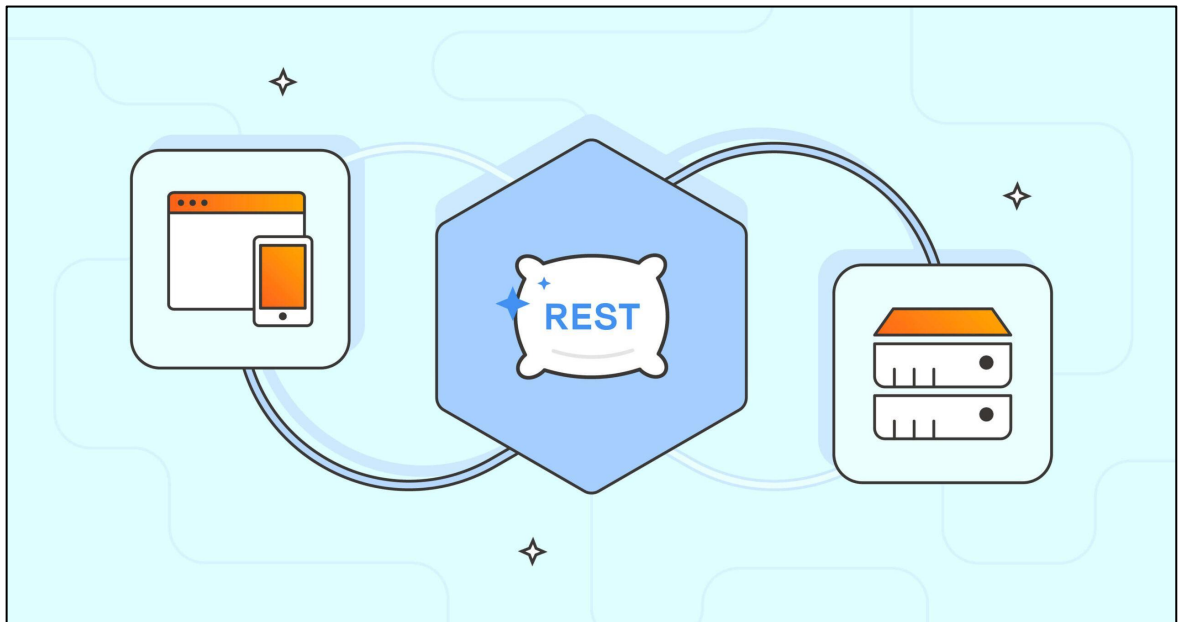
GET /api/albums/:id // Chi tiết album

Response Format:

// Success

```
{  
  "success": true,  
  "data": [],  
  "message": "Thành công"  
}
```

```
// Error  
  
{  
  "success": false,  
  "message": "Mô tả lỗi",  
  "error": "Chi tiết lỗi (chỉ trong development)"  
}
```



Hình 2.9: Phương thức RESTful API

2.3.5. JWT (JSON Web Token)

Giới thiệu:

JWT là chuẩn mở (RFC 7519) để truyền thông tin an toàn giữa các bên dưới dạng JSON object

Cấu trúc JWT:

Header: Loại token và thuật toán mã hóa

Payload: Dữ liệu người dùng (user ID, email)

Signature: Chữ ký bảo mật

Ứng dụng trong dự án:

```
// Tạo token khi đăng nhập

const jwt = require('jsonwebtoken');

const generateToken = (user) => {

  return jwt.sign(

    {

      userId: userma_nguoi_dung,

      email: useremail

    },

    process.env.JWT_SECRET,

    { expiresIn: '7d' }

  );

};
```

```
// Middleware xác thực token

const authenticateToken = (req, res, next) => {

  const authHeader = reqheaders['authorization'];

  const token = authHeader && authHeadersplit(' ')[1];

  if (!token) {

    return resstatus(401)json({

      success: false,

      message: 'Chưa đăng nhập'

    });

  }

  jwtverify(token, processenvJWT_SECRET, (err, user) => {

    if (err) {

      return resstatus(403)json({

        success: false,

        message: 'Token không hợp lệ'

      });

    }

    requser = user;

    next();

  });

};
```



```
// Sử dụng middleware  
router.get('/cart', authenticateToken, async (req, res) => {  
    const userId = req.user.userId;  
    // Xử lý lấy giỏ hàng  
});
```

2.3.6. Bcrypt

Giới thiệu:

Bcrypt là thư viện mã hóa mật khẩu sử dụng thuật toán blowfish, bảo mật cao

Ứng dụng:

```
const bcrypt = require('bcrypt');

// Hash mật khẩu khi đăng ký

const hashPassword = async (password) => {

  const saltRounds = 10;

  return await bcrypt.hash(password, saltRounds);

};

// So sánh mật khẩu khi đăng nhập

const comparePassword = async (password, hash) => {

  return await bcrypt.compare(password, hash);

};

// Trong route đăng ký

router.post('/register', async (req, res) => {

  const { username, password, email } = req.body;

  const hashedPassword = await hashPassword(password);

  await db.query(

    'INSERT INTO nguoi_dung (ten_dang_nhap, mat_khau_hash, email) VALUES (?, ?, ?)',

    [username, hashedPassword, email]

  );

});
```

2.4. CÔNG CỤ HỖ TRỢ

2.4.1. Visual Studio Code

Giới thiệu:

VS Code là code editor miễn phí, mạnh mẽ của Microsoft, hỗ trợ đa ngôn ngữ

Tính năng:

IntelliSense: Gợi ý code thông minh

Debugging: Debug trực tiếp trong editor

Git Integration: Quản lý version control

Extensions: Hàng ngàn extension mở rộng

Extensions sử dụng:

ESLint: Kiểm tra lỗi JavaScript

Prettier: Format code tự động

Live Server: Preview HTML realtime

MySQL: Quản lý database

2.4.2. Postman

Giới thiệu:

Postman là công cụ test API, giúp gửi HTTP request và xem response

Ứng dụng:

Test các endpoint API

Tạo collection các request

Tự động hóa test API

Tạo documentation API

2.4.3. Git & GitHub

Giới thiệu:

Git: Hệ thống quản lý phiên bản phân tán

GitHub: Nền tảng lưu trữ code online

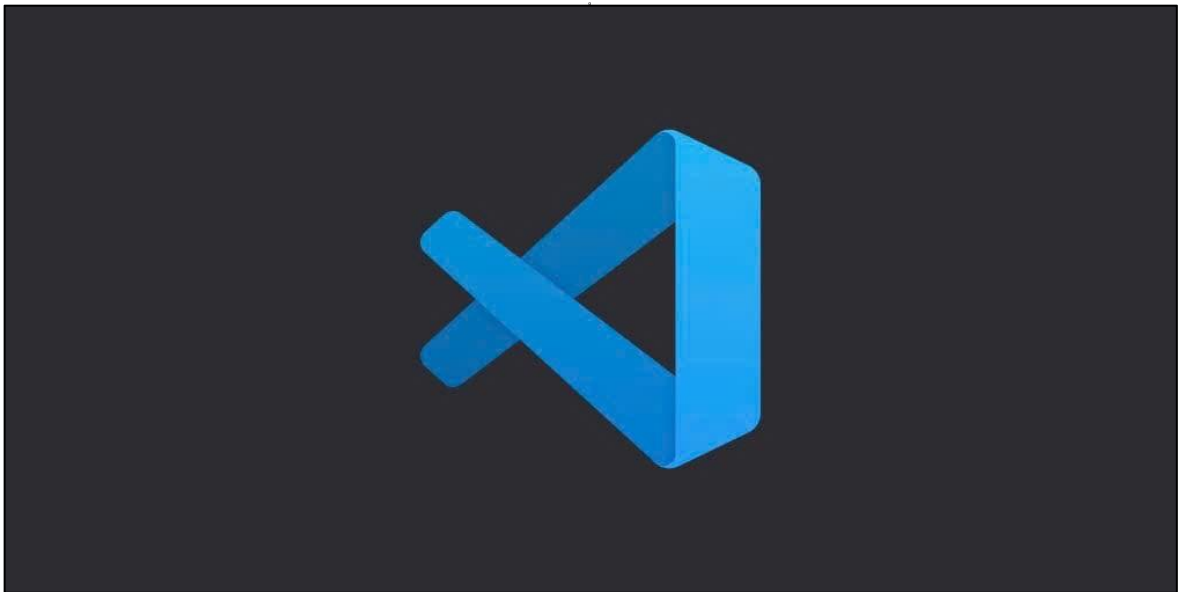
Ứng dụng:

Version control: Lưu lịch sử thay đổi code

Collaboration: Làm việc nhóm

Backup: Sao lưu code an toàn

Deployment: Triển khai lên server



Hình 2.10: Visual Studio Code

2.4.4. Docker

Giới thiệu

Docker là nền tảng mã nguồn mở cho phép đóng gói ứng dụng và các dependencies vào các container đơn vị phần mềm độc lập có thể chạy nhất quán trên mọi môi trường

Khái niệm cốt lõi

- Container vs Virtual Machine

Tiêu chí	Container	Virtual Machine
Kích thước	Nhẹ (MB)	Nặng (GB)
Khởi động	Nhanh (giây)	Chậm (phút)
Tài nguyên	Chia sẻ OS kernel	Mỗi VM có OS riêng
Hiệu năng	Gần native	Overhead cao

Bảng 1: Bảng so sánh Docker và Virtual Machine

Ưu điểm Container:

Khởi động cực nhanh

Tiết kiệm tài nguyên

Triển khai nhất quán

Portable chạy ở mọi nơi

Thành phần chính

1 Docker Image

Template readonly chứa OS, runtime, code để tạo container

Ví dụ: node:18alpine, mysql:8, nginx:latest

2 Docker Container

Instance đang chạy của image, có thể start/stop/delete

3 Dockerfile

File text định nghĩa cách build image

Ví dụ Dockerfile:

```
FROM node:18alpine
WORKDIR /app
COPY packagejson /
RUN npm ci only=production
COPY
EXPOSE 3000
CMD ["node", "serverjs"]
```

4 Docker Compose

Quản lý multicontainer qua file YAML

Ví dụ dockercompose.yml:

```
version: '3.8'

services:
  backend:
    build: /backend
    ports:
      - "3000:3000"
    environment:
      DB_HOST=database
    depends_on:
      - database
  database:
    image: mysql:8
    environment:
      MYSQL_ROOT_PASSWORD=password
      MYSQL_DATABASE=amthuc_phuongnam
    volumes:
      - mysqldata:/var/lib/mysql
volumes:
  mysqldata
```

Lợi ích cho dự án

Development:

Setup nhanh 1 lệnh thay vì cài Node, MySQL, config
dockercompose up d
→ Môi trường đồng nhất cho tất cả developers

Production:

Build và deploy dễ dàng
docker build t amthucbackend:latest
docker push amthucbackend:latest
docker run d p 3000:3000 amthucbackend:latest
Scale nhanh
dockercompose up d scale backend=3

Các lệnh cơ bản:

Quản lý Images
docker images # Liệt kê images
docker build t myapp:10 # Build image
docker pull node:18alpine # Tải image
Quản lý Containers
docker ps # Container đang chạy
docker ps a # Tất cả containers
docker run d p 3000:3000 myapp
docker stop myapp
docker logs f myapp # Xem logs


```
# Docker Compose

dockercompose up d      # Start all services

dockercompose down      # Stop all

dockercompose logs f     # Xem logs

dockercompose restart backend # Restart service
```

Triển khai:

```
# Development

dockercompose up d

# Production

dockercompose f dockercomposeprod.yml up d

# Access:

# Frontend: http://localhost

# Backend: http://localhost:3000

# MySQL: localhost:3306
```

Kết luận

Docker giúp dự án:

- Môi trường development nhất quán
- Setup nhanh chóng
- Deploy dễ dàng
- Isolation giữa các services
- Portable chạy mọi nơi

=> Phù hợp cho quy mô nhà hàng, dễ học, dễ triển khai

2.4.5. Nodejs vs Java (Spring Boot)

Tiêu chí	Nodejs	Java Spring Boot
Ngôn ngữ	JavaScript	Java
Kiến trúc	Eventdriven, Singlethreaded	Multithreaded, Blocking I/O
Hiệu năng	Cao với I/O, kém với CPU	Cao với mọi loại operation
Thời gian phát triển	Nhanh	Chậm (nhiều boilerplate)
Học tập	Trung bình	Khó (cú pháp phức tạp, nhiều concept)
Typing	Dynamic (hoặc TypeScript)	Static typing mạnh mẽ
Memory usage	Nhẹ	Nặng (JVM)
Startup time	Nhanh (< 1s)	Chậm (35s)
Enterprise	Ít phổ biến	Rất phổ biến
Microservices	Tốt	Xuất sắc

Bảng 2: Bảng so sánh Nodejs và Java Spring Boot

Ưu điểm của Nodejs so với Java:

Phát triển nhanh hơn, ít code hơn

Học dễ hơn nhiều

Startup time nhanh

Memory footprint nhỏ hơn

Phù hợp với startup, dự án nhỏ/vừa

Nhược điểm của Nodejs so với Java:

Không mạnh bằng cho hệ thống enterprise lớn

Type safety kém hơn (trừ khi dùng TypeScript)

Xử lý CPUintensive kém hơn

Lý do chọn Nodejs thay vì Java:

Dự án quy mô vừa/nhỏ (nhà hàng), không cần Java

Muốn phát triển nhanh, ra sản phẩm sớm

Team nhỏ, không có chuyên gia Java

Chi phí server thấp hơn (ít RAM hơn)

2.5. CÁC NGHIỆP VỤ LIÊN QUAN ĐẾN ĐỀ TÀI

2.5.1. Nghiệp vụ quản lý nhà hàng

a) Quản lý thực đơn (Menu Management)

Đây là nghiệp vụ cốt lõi của nhà hàng, bao gồm:

Phân loại món ăn: Chia thực đơn theo danh mục (khai vị, món chính, tráng miệng, đồ uống, đặc sản miền Tây...)

Định giá: Xác định giá bán dựa trên chi phí nguyên liệu, chi phí chế biến và lợi nhuận mong muốn

Cập nhật thực đơn: Thêm món mới, ngừng bán món cũ, điều chỉnh giá theo mùa vụ

Quản lý hình ảnh: Chụp ảnh món ăn chất lượng cao để thu hút khách hàng

b) Nghiệp vụ đặt bàn (Table Reservation)

Quy trình đặt bàn truyền thống và trực tuyến:

Tiếp nhận yêu cầu: Khách hàng đặt bàn qua điện thoại, trực tiếp hoặc website

Kiểm tra khả dụng: Xác nhận bàn trống theo ngày giờ, số lượng khách

Xác nhận đặt bàn: Gửi thông báo xác nhận qua email/SMS

Nhắc nhở: Gửi nhắc nhở trước giờ đặt bàn

Check-in: Đón khách và sắp xếp bàn khi khách đến

Hủy/Thay đổi: Xử lý yêu cầu hủy hoặc thay đổi lịch đặt

c) Nghiệp vụ đặt món và giao hàng (Order & Delivery)

Tiếp nhận đơn hàng: Khách chọn món, thêm vào giỏ hàng, đặt hàng

Xác nhận đơn: Kiểm tra thông tin, xác nhận với khách hàng

Chế biến: Chuyển đơn hàng đến bếp, theo dõi tiến độ

Đóng gói: Đóng gói món ăn đảm bảo chất lượng khi giao

Giao hàng: Vận chuyển đến địa chỉ khách hàng

Thanh toán: Thu tiền COD hoặc xác nhận thanh toán online

Đánh giá: Thu thập phản hồi từ khách hàng

d) Nghiệp vụ thanh toán (Payment Processing)

Thanh toán tiền mặt (COD): Khách trả tiền khi nhận hàng

Thanh toán trực tuyến: Tích hợp cổng thanh toán (MoMo, VNPay, ZaloPay)

Chuyển khoản ngân hàng: Khách chuyển khoản trước khi giao hàng

Xuất hóa đơn: Tạo hóa đơn điện tử cho khách hàng

2.5.2. Nghiệp vụ quản lý khách hàng (CRM)

a) Quản lý thông tin khách hàng

Lưu trữ thông tin cá nhân: họ tên, số điện thoại, email, địa chỉ

Lịch sử đặt bàn, đặt món

Sở thích ẩm thực, món ăn yêu thích

Phân loại khách hàng (mới, thường xuyên, VIP)

b) Chăm sóc khách hàng

Gửi email chúc mừng sinh nhật, ngày lễ

Thông báo khuyến mãi, món mới

Xử lý khiếu nại, phản hồi

Chương trình tích điểm, ưu đãi thành viên

2.5.3. Nghiệp vụ marketing và truyền thông

a) Quản lý nội dung (Content Management)

Đăng bài tin tức, sự kiện nhà hàng

Cập nhật album ảnh không gian, món ăn

Viết bài giới thiệu đặc sản miền Tây

Quản lý banner, slider quảng cáo

b) Khuyến mãi và ưu đãi

Tạo mã giảm giá (voucher, coupon)

Chương trình combo, set menu

Ưu đãi theo dịp lễ, sự kiện

Flash sale, happy hour

2.5.4. Nghiệp vụ báo cáo và thống kê

Báo cáo doanh thu: Theo ngày, tuần, tháng, năm

Thống kê đơn hàng: Số lượng, trạng thái, tỷ lệ hủy

Phân tích món ăn: Món bán chạy, món ít được đặt

Thống kê khách hàng: Khách mới, khách quay lại, tỷ lệ chuyển đổi

Đánh giá hiệu quả: So sánh các kênh bán hàng (tại chỗ, online)

2.6. CÁC CÔNG TRÌNH NGHIÊN CỨU LIÊN QUAN

2.6.1. Nghiên cứu trong nước

a) Đề tài: "Xây dựng website quản lý nhà hàng ABC" - Đại học Công nghệ TP.HCM (2022)

Nội dung: Phát triển hệ thống quản lý nhà hàng với chức năng đặt bàn, quản lý thực đơn, thanh toán

Công nghệ: PHP Laravel, MySQL, Bootstrap

Kết quả: Hệ thống hoạt động ổn định, giao diện responsive

Hạn chế: Chưa tích hợp thanh toán online, animation đơn giản

Bài học rút ra: Cần tối ưu UX/UI và tích hợp nhiều phương thức thanh toán

b) Đề tài: "Ứng dụng web đặt món ăn trực tuyến" - Đại học Cần Thơ (2023)

Nội dung: Xây dựng ứng dụng đặt món online cho chuỗi nhà hàng

Công nghệ: React.js, Node.js, MongoDB

Kết quả: Giao diện hiện đại, tốc độ tải nhanh

Hạn chế: Chi phí triển khai cao, yêu cầu kỹ thuật phức tạp

Bài học rút ra: Cần nhắc giữ công nghệ hiện đại và khả năng triển khai thực tế

c) Đề tài: "Hệ thống quản lý đặt bàn nhà hàng" - Đại học Bách khoa Hà Nội (2021)

Nội dung: Phát triển module đặt bàn với calendar view, quản lý sơ đồ bàn

Công nghệ: Java Spring Boot, PostgreSQL, Angular

Kết quả: Hệ thống enterprise-grade, bảo mật cao

Hạn chế: Thời gian phát triển dài, khó bảo trì với team nhỏ

Bài học rút ra: Lựa chọn công nghệ phù hợp với quy mô dự án

2.6.2. Nghiên cứu quốc tế

a) "Design and Implementation of Restaurant Management System" - International Journal of Computer Applications (2020)

Tác giả: Kumar, S., & Patel, R.

Nội dung: Nghiên cứu về thiết kế hệ thống quản lý nhà hàng toàn diện

Đóng góp: Đề xuất mô hình kiến trúc 3-tier cho ứng dụng nhà hàng

Ứng dụng: Áp dụng mô hình phân tầng (Presentation - Business Logic - Data) trong thiết kế hệ thống

b) "User Experience Design for Food Ordering Applications" - ACM Conference on Human Factors in Computing Systems (2021)

Tác giả: Chen, L., Wang, H., & Liu, M.

Nội dung: Nghiên cứu về trải nghiệm người dùng trong ứng dụng đặt món

Phát hiện chính:

78% người dùng quyết định đặt món dựa trên hình ảnh

Thời gian checkout dưới 3 bước tăng tỷ lệ chuyển đổi 40%

Animation mượt mà tăng cảm nhận chất lượng ứng dụng

Ứng dụng: Thiết kế giao diện tập trung vào hình ảnh món ăn, tối ưu luồng đặt hàng

c) "RESTful API Design Best Practices for E-commerce Applications" - IEEE Software Engineering Conference (2022)

Tác giả: Johnson, M., & Smith, A.

Nội dung: Chuẩn hóa thiết kế API cho ứng dụng thương mại điện tử

Đóng góp: Đề xuất cấu trúc response chuẩn, xử lý lỗi, pagination

Ứng dụng: Áp dụng chuẩn RESTful trong thiết kế API backend

**d) "Security Considerations in Web-based Food Ordering Systems" -
Journal of Information Security (2023)**

Tác giả: Williams, D., & Brown, K.

Nội dung: Phân tích các lỗ hổng bảo mật trong hệ thống đặt món online

Khuyến nghị:

Sử dụng JWT với thời hạn ngắn

Mã hóa mật khẩu bằng bcrypt với salt rounds ≥ 10

Validate input phía server để chống SQL Injection, XSS

HTTPS bắt buộc cho mọi giao dịch

Ứng dụng: Triển khai các biện pháp bảo mật trong hệ thống

2.7. KẾT LUẬN CHƯƠNG

Chương 2 đã trình bày tổng quan về các công nghệ được sử dụng trong dự án:

Frontend: HTML, CSS, Tailwind CSS, JavaScript ES6+, GSAP Animation

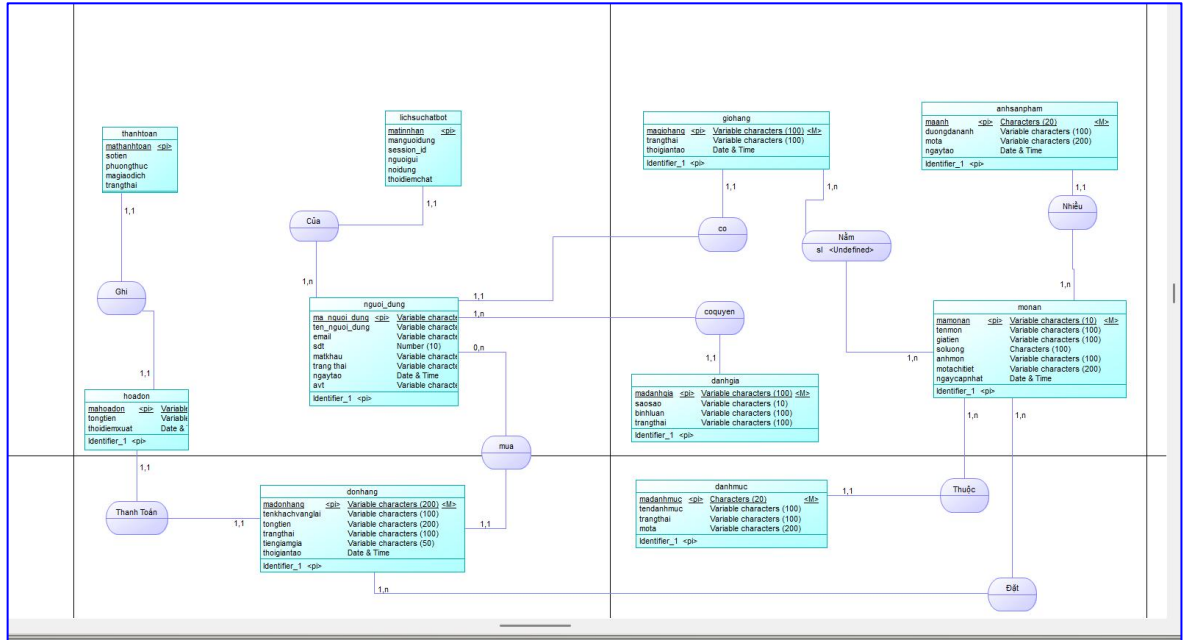
Backend: Nodejs, Expressjs, MySQL, RESTful API, JWT, Bcrypt

Tools: VS Code, Postman, Git/GitHub

=> Sự kết hợp của các công nghệ này tạo nên một hệ thống web hoàn chỉnh, hiệu năng cao, bảo mật tốt và dễ bảo trì. Chương tiếp theo sẽ trình bày chi tiết về phân tích, thiết kế hệ thống dựa trên nền tảng công nghệ này.

CHƯƠNG 3. HIỆN THỰC HÓA NGHIÊN CỨU

3.1. Mô hình thực thể kết hợp ERD



Hình 3.1: Mô hình thực thể kết hợp ERD

3.2. Mô tả hệ thống

3.2.1. Tổng quan hệ thống

Website Nhà hàng Âm thực Phương Nam - Vĩnh Long là hệ thống web fullstack được xây dựng theo kiến trúc Client-Server, bao gồm:

Phía Client (Frontend):

Giao diện người dùng (Customer Interface): Trang chủ, thực đơn, đặt bàn, giỏ hàng, thanh toán

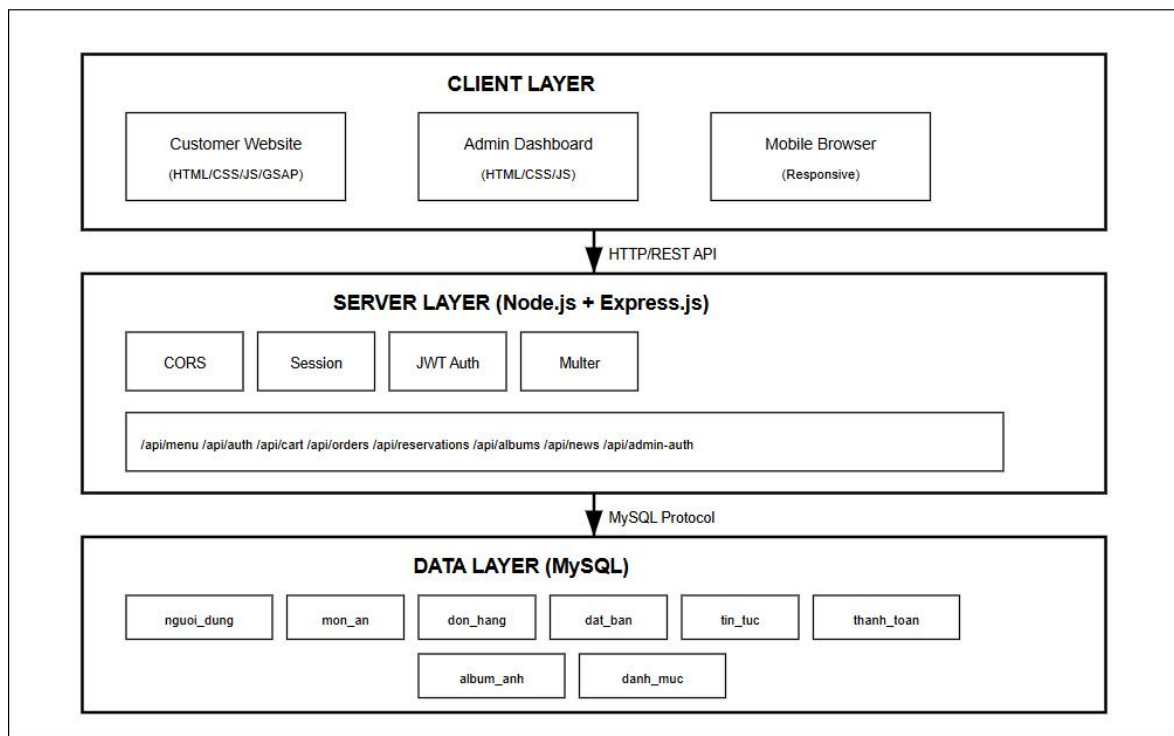
Giao diện quản trị (Admin Interface): Dashboard, quản lý món ăn, đơn hàng, khách hàng, tin tức

Phía Server (Backend):

RESTful API Server: Xử lý nghiệp vụ, xác thực, truy vấn dữ liệu

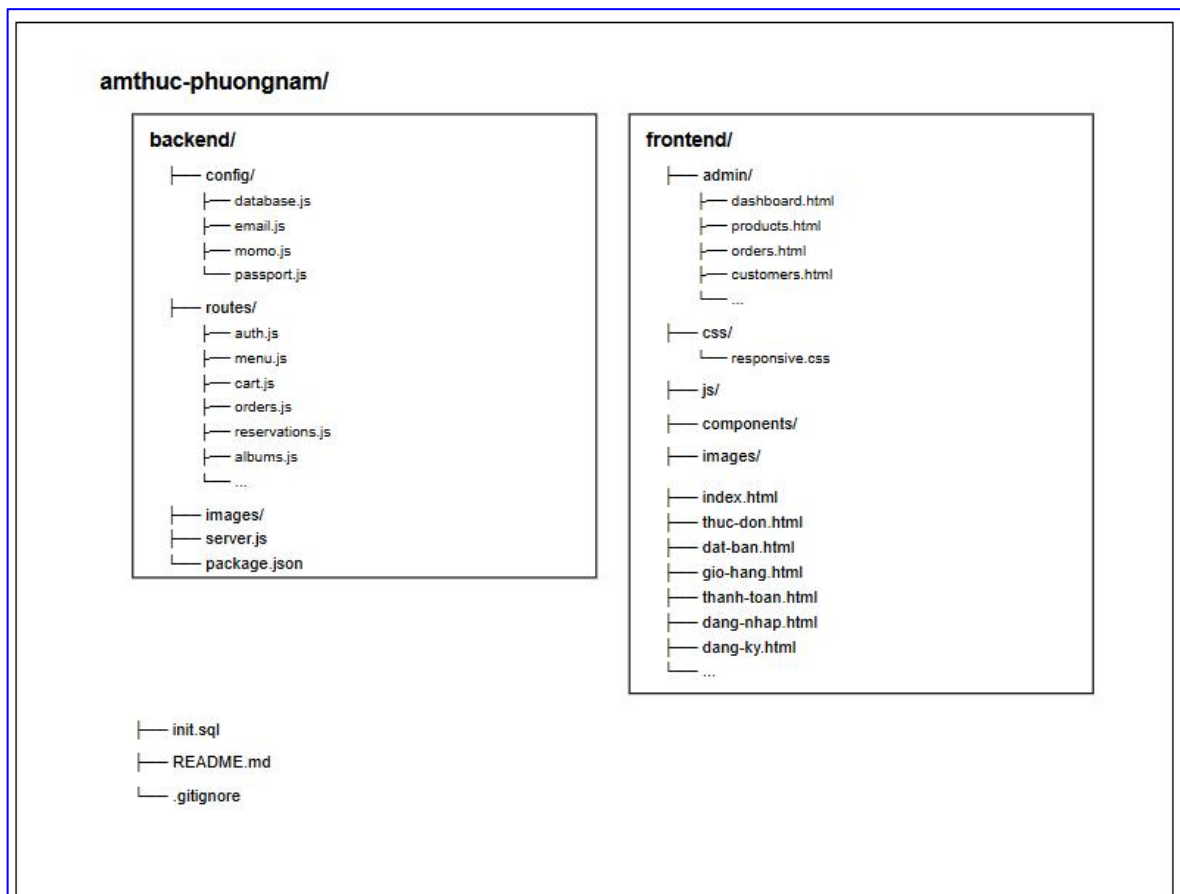
Database Server: Lưu trữ dữ liệu MySQL

3.2.2. Kiến trúc hệ thống



Hình 3.2: Kiến trúc hệ thống

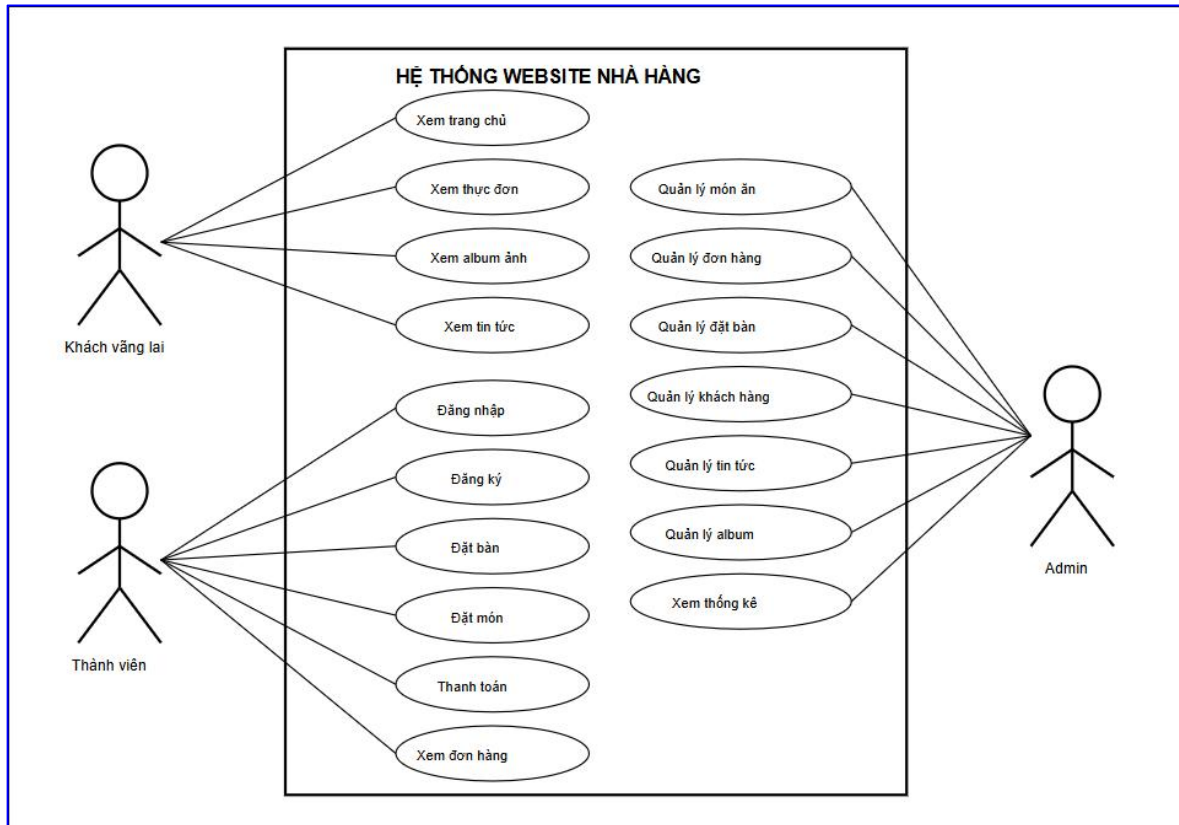
3.2.3. Cấu trúc dự án



Hình 3.3: Cấu trúc dự án

3.3. XÁC ĐỊNH CÁC YÊU CẦU CHỨC NĂNG CỦA HỆ THỐNG

3.3.1. Sơ đồ Usecase



Hình 3.4: Sơ đồ UseCase

Mô tả: Sơ đồ mô tả ba nhóm người dùng chính: Khách vãng lai, Thành viên, và Admin, mỗi nhóm có quyền truy cập các chức năng khác nhau trong hệ thống.

Khách vãng lai có thể xem các nội dung công khai như: trang chủ, thực đơn, album ảnh và tin tức.

Thành viên (người dùng đã đăng ký) được sử dụng thêm các chức năng nâng cao như: đăng nhập, đặt bàn, đặt món, thanh toán và xem đơn hàng của mình.

Admin có quyền quản trị hệ thống, bao gồm: quản lý món ăn, đơn hàng, đặt bàn, khách hàng, tin tức, album và xem báo cáo thống kê.

Sơ đồ thể hiện rõ ràng phạm vi hệ thống và mối liên hệ giữa các actor với các chức năng tương ứng.

3.4. THIẾT KẾ XỬ LÝ HỆ THỐNG

3.4.1. Thiết kế API RESTful

A. API Xác thực (Authentication)

POST /api/auth/send-verification – Gửi mã xác thực email.

POST /api/auth/verify-email – Xác thực mã và tạo tài khoản.

POST /api/auth/login – Đăng nhập lấy token.

POST /api/auth/forgot-password – Gửi yêu cầu đặt lại mật khẩu.

POST /api/auth/reset-password – Nhập mã + mật khẩu mới để reset.

GET /api/auth/me – Lấy thông tin người dùng đang đăng nhập.

PUT /api/auth/update – Cập nhật thông tin cá nhân.

POST /api/auth/change-password – Đổi mật khẩu.

B. API Món ăn (Menu)

GET /api/menu – Lấy danh sách món, hỗ trợ tìm kiếm/lọc/sắp xếp.

GET /api/menu/:id – Xem chi tiết món ăn.

GET /api/menu/category/:id – Món theo danh mục.

GET /api/menu/top-selling – Món bán chạy.

GET /api/menu/related/:id – Món liên quan.

POST /api/menu – Thêm món mới (Admin).

PUT /api/menu/:id – Cập nhật món (Admin).

DELETE /api/menu/:id – Xóa món (Admin).

PATCH /api/menu/:id/toggle-status – Ẩn/hiện món (Admin).

C. API Giỏ hàng (Cart)

GET /api/cart – Lấy giỏ hàng hiện tại.

POST /api/cart/add – Thêm món vào giỏ.

PUT /api/cart/update/:id – Thay đổi số lượng món.

DELETE /api/cart/remove/:id – Xóa một món.

DELETE /api/cart/clear – Xóa toàn bộ giỏ hàng.

D. API Đơn hàng (Orders)

POST /api/orders/create – Tạo đơn hàng mới.

GET /api/orders/my-orders – Lịch sử đơn hàng của người dùng.

GET /api/orders/:orderId – Xem chi tiết đơn hàng.

PUT /api/orders/:orderId/cancel – Hủy đơn.

GET /api/orders/all – Danh sách tất cả đơn (Admin).

PUT /api/orders/:orderId/status – Cập nhật trạng thái đơn (Admin).

GET /api/orders/stats – Thống kê doanh thu, số đơn (Admin).

E. API Đặt bàn (Reservations)

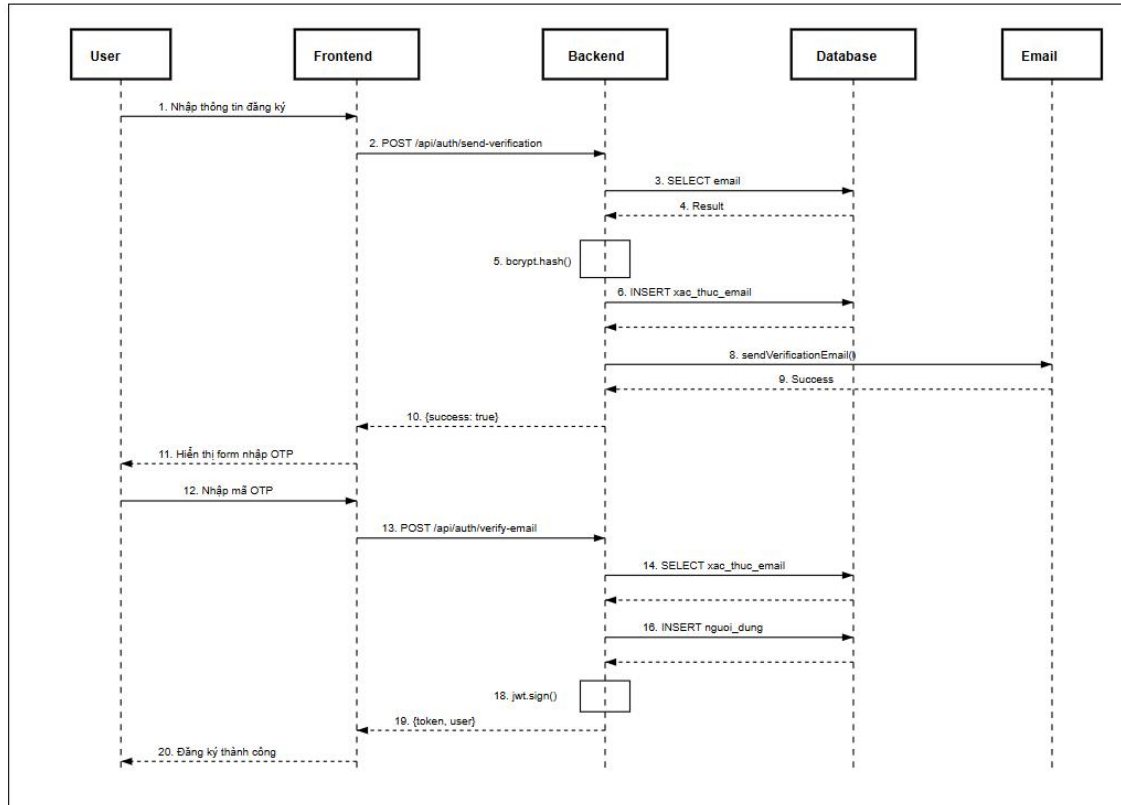
POST /api/reservations – Tạo yêu cầu đặt bàn.

GET /api/reservations – Danh sách đặt bàn (Admin).

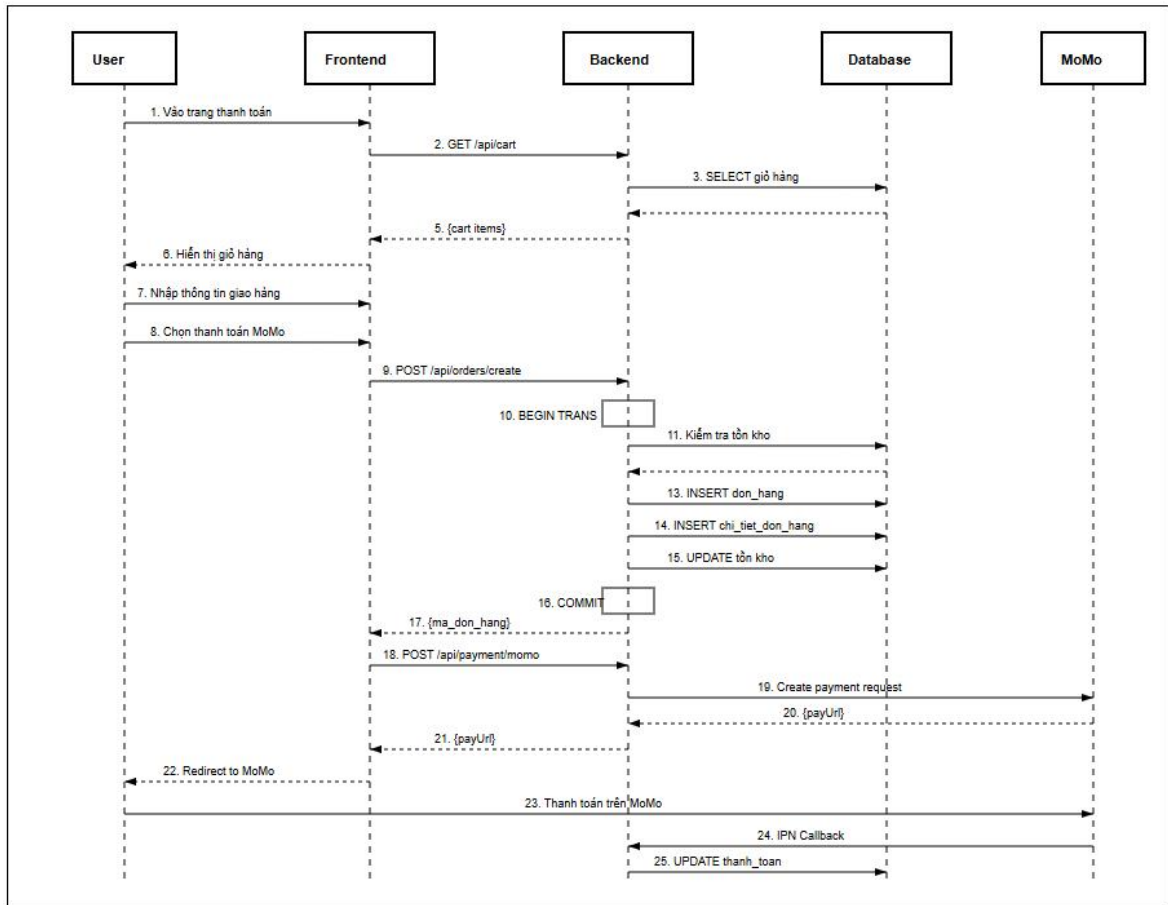
PUT /api/reservations/:id/status – Cập nhật trạng thái đặt bàn (Admin).

3.4.2. Sơ đồ tuần tự (Sequence Diagram)

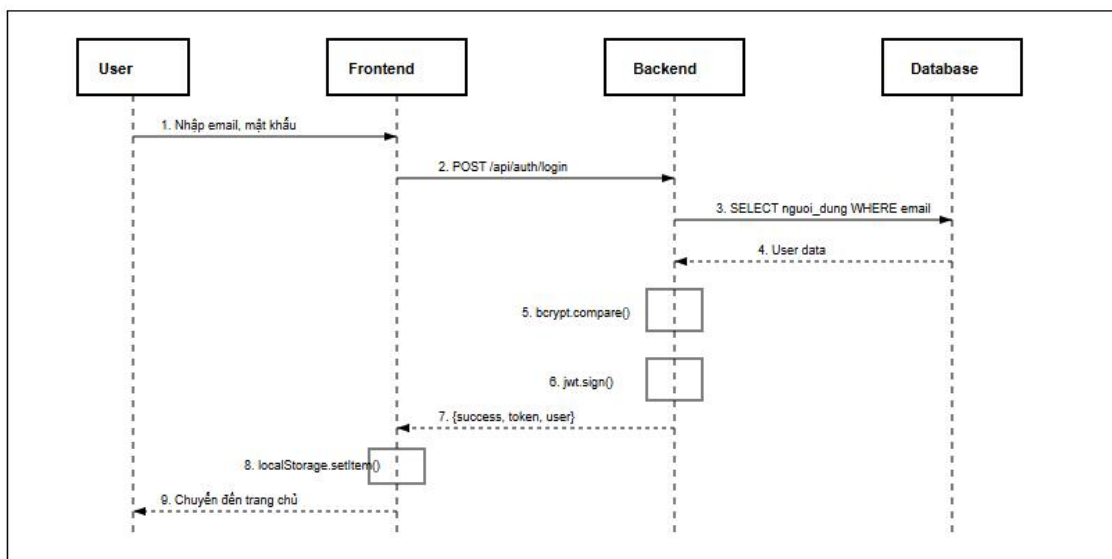
1. Quy trình Đăng ký tài khoản



2. Quy trình Đặt hàng và Thanh toán



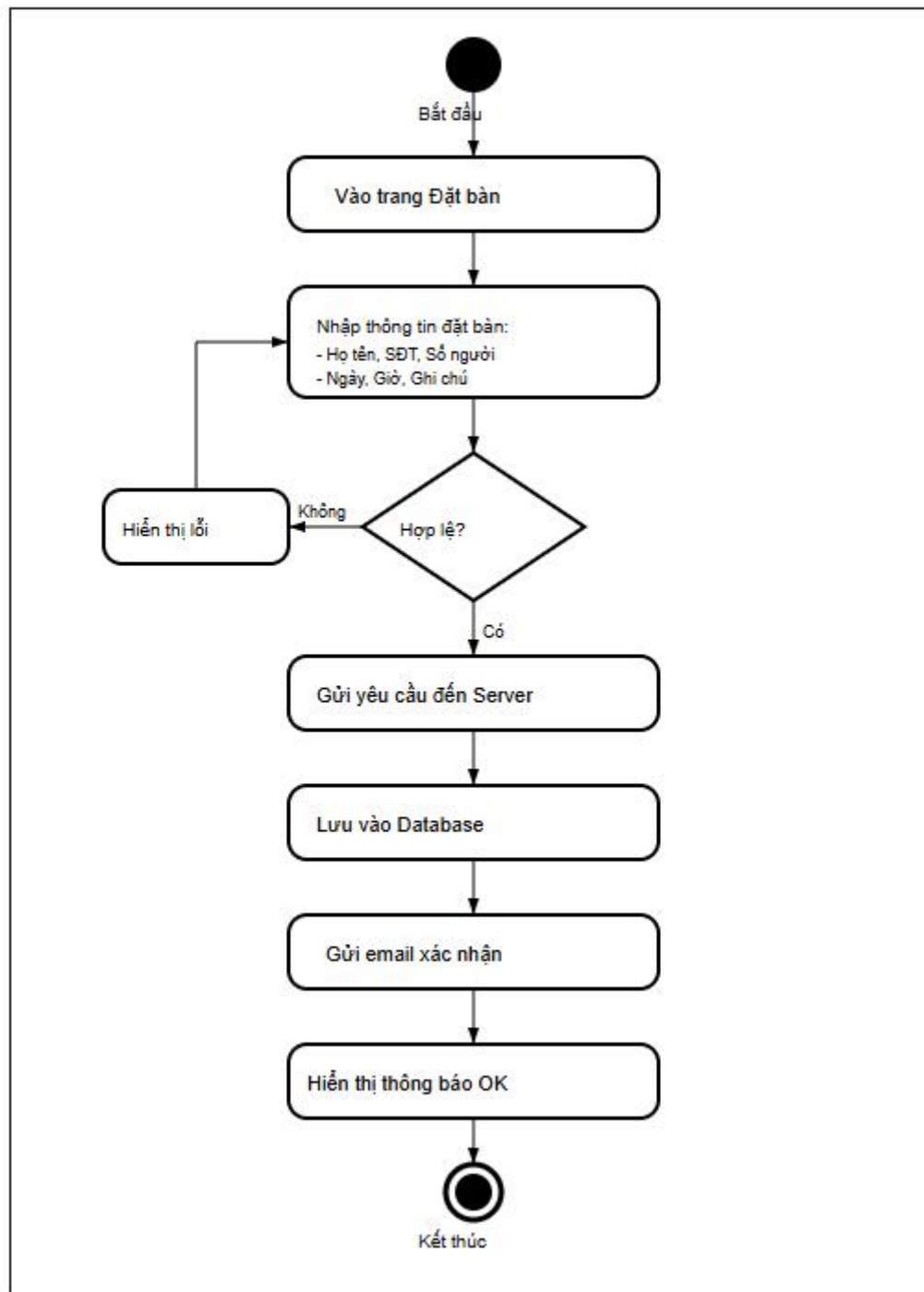
3. Quy trình Đăng nhập



Hình 3.5: Sơ đồ tuần tự

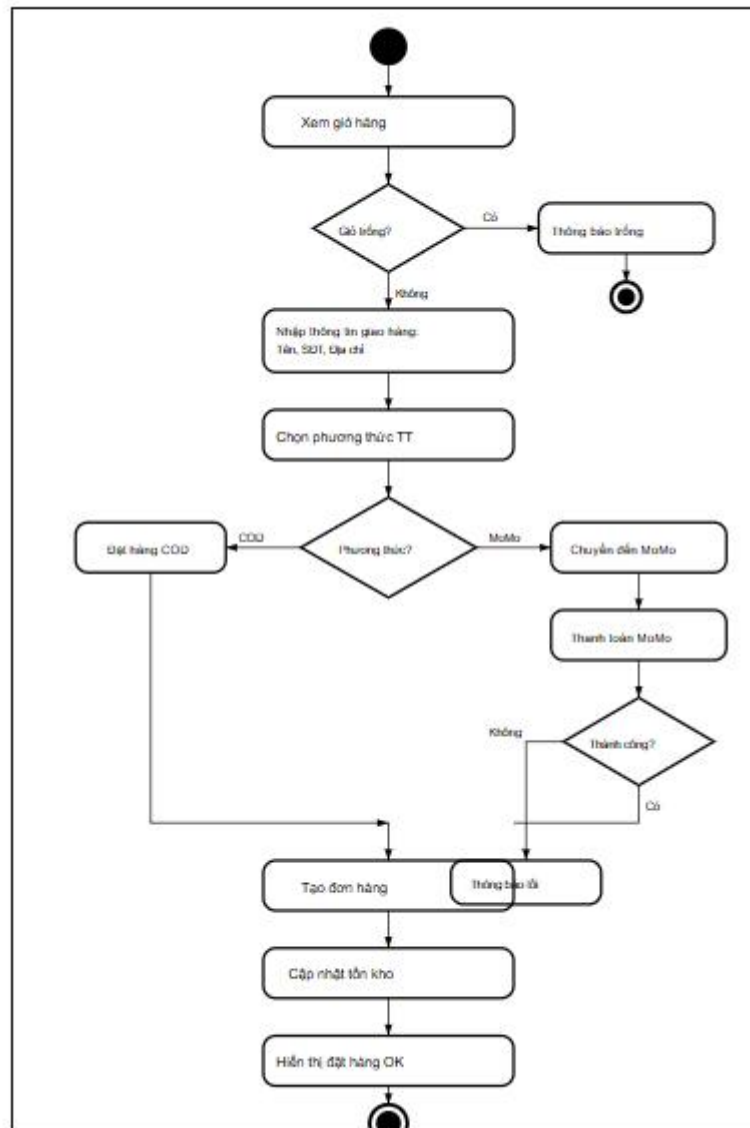
3.5. MÔ HÌNH HOẠT ĐỘNG CỦA HỆ THỐNG

1. Quy trình Đặt bàn trực tuyến



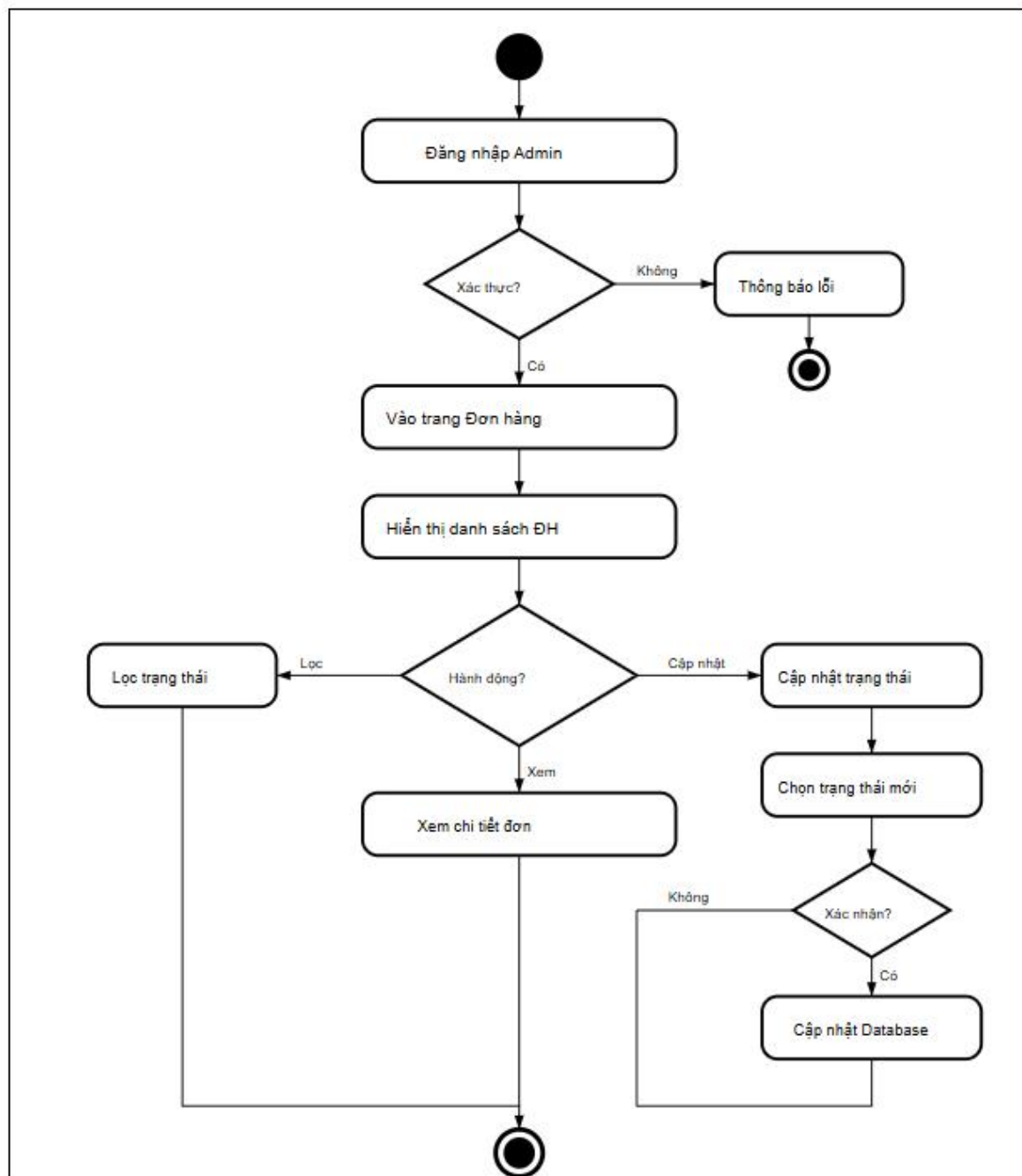
Hình 3.6: Quy trình đặt bàn trực tuyến

2. Quy trình Đặt hàng



Hình 3.7: Quy trình đặt hàng

3. Quy trình Quản lý đơn hàng (Admin)



Hình 3.8: Quy trình Quản lý đơn hàng

Kết luận Chương 3:

Chương này đã trình bày chi tiết về hiện thực hóa hệ thống website Nhà hàng Âm thực Phương Nam, bao gồm:

Mô tả hệ thống: Kiến trúc Client-Server, cấu trúc thư mục dự án

Yêu cầu chức năng: Biểu đồ Use Case cho khách hàng và admin

Thiết kế dữ liệu: Sơ đồ ERD với 15+ bảng dữ liệu quan hệ

Thiết kế xử lý: API RESTful, xử lý Transaction, JWT Authentication

Mô hình hoạt động: Sơ đồ tuần tự, sơ đồ hoạt động cho các nghiệp vụ chính