

[illegible]

(Ký tên và ghi rõ họ tên)

[illegible]

Trà Vinh, ngày tháng năm
Thành viên hội đồng
(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Trong khoảng thời gian làm đồ án, tôi đã nhận được nhiều sự giúp đỡ, đóng góp ý kiến và sự dẫn dắt chỉ bảo nhiệt tình của thầy cô, gia đình và bạn bè.

Tôi xin gửi lời cảm ơn chân thành đến giáo viên hướng dẫn Phạm Minh Dương giảng viên của Trường Đại học Trà Vinh đã tận tình hướng dẫn, chỉ bảo trong suốt quá trình làm đồ án.

Tôi cũng xin gửi cảm ơn chân thành nhất tới các thầy cô giáo trong trường Đại học Trà Vinh nói chung, các thầy cô trong Bộ môn Kỹ thuật và Công Nghệ nói riêng đã dạy dỗ cho tôi kiến thức cũng như các môn chuyên ngành, giúp tôi có được cơ sở lý thuyết vững vàng và tạo điều kiện giúp đỡ trong suốt quá trình tham gia học tập.

Với điều kiện về thời gian cũng như lượng kiến thức về đề tài rất rộng mà kinh nghiệm còn hạn chế khi làm không thể tránh được những thiếu sót. Tôi rất mong nhận được sự chỉ bảo, đóng góp ý kiến của các thầy cô để có điều kiện bổ sung, nâng cao ý thức của mình.

Tôi xin trân thành cảm ơn!

MỤC LỤC

CHƯƠNG 1: TỔNG QUAN.....	1
CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT	2
2.1. Tổng quan về NodeJS.....	2
2.1.2. NodeJS	2
2.1.2. Các thức hoạt động của NodeJS	3
2.1.3. Kiến trúc Non-blocking I/O và Event-Driven	4
2.1.4. V8 JavaScript Engine	4
2.1.5. Single-Threaded.....	4
2.1.6. Event Loop.....	4
2.1.7. Trigger Callback	4
2.1.8. NPM (Node Package Manager).....	4
2.1.9. Require-relative	5
2.2. Thành phần của Node.js	5
2.2.1. Module	5
2.2.2. Bảng điều khiển	6
2.2.3. Cluster	6
2.2.4. Đối tượng toàn cục.....	6
2.2.5. Xử lý lỗi	7
2.2.6. Streaming (Luồng).....	8
2.2.7. Buffer (Bộ nhớ đệm).....	8
2.2.8. Tên miền (Domain).....	8
2.2.9. Hệ thống phân giải tên miền Domain Name System DNS.....	8
2.2.10. Debugger (Trình gỡ lỗi).....	9
2.3. Ưu và nhược điểm của Node.JS	9
2.4. Tổng quan về MongoDB	9
2.4.1. MongoDB	10
2.4.2. Giải thích về NoSQL trong khái niệm MongoDB.....	10
2.4.3. Lịch sử phát triển của MongoDB	10
2.5. Sử dụng MongoDB mang lại những lợi ích	11
2.5.1. Truy vấn cơ sở dữ liệu đặc biệt	11

2.5.2. Cân bằng tải	11
2.5.3. Hỗ trợ đa ngôn ngữ	12
2.6. MongoDB có những ưu và nhược điểm	12
CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU	14
3.1. Mô tả hệ thống	14
3.2. Xác định các yêu cầu chức năng của hệ thống	14
3.3. Thiết kế dữ liệu hệ thống	16
3.4. Mô hình dữ liệu mức quan niệm	22
3.5. Mô hình dữ liệu mức logic	23
3.6.1. Sơ đồ User Case tổng quát	23
3.6.2. Sơ đồ User Case người dùng	24
3.6.3. Sơ đồ User Case admin	25
3.7. Các bước xây dựng website	27
CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU	30
4.1. Giao diện đăng nhập và đăng ký tài khoản	30
4.2. Giao diện tìm kiếm vé một chiều	31
4.3. Giao diện tìm kiếm vé khứ hồi	33
4.4. Giao diện xem thông tin vé	35
4.5. Giao diện thanh toán	35
4.6. Giao diện hóa đơn	36
4.7. Giao diện đánh giá	36
4.8. Giao diện thông tin liên hệ	37
4.9. Giao diện thông tin khách hàng	37
CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	39
5.1. Kết quả đạt được	39
5.2. Hạn chế	39
5.3. Hướng phát triển	39
DANH MỤC TÀI LIỆU THAM KHẢO	41

DANH MỤC HÌNH ẢNH – BẢNG BIỂU

DANH MỤC HÌNH ẢNH

Hình 1: Công cụ NodeJS.....	2
Hình 2: Cách hoạt động của NodeJS	3
Hình 3: Công cụ MongoDB	10
Hình 4: Mô hình dữ liệu mức quan niệm.....	22
Hình 5: Mô hình dữ liệu mức logic.....	23
Hình 6: Sơ đồ User Case tổng quát.....	23
Hình 7: Sơ đồ User Case người dùng.....	24
Hình 8: Sơ đồ User Case người dùng tìm kiếm chuyến xe.....	25
Hình 9: Sơ đồ User Case admin	25
Hình 10: Sơ đồ User Case admim dùng để tạo vé xe.....	26
Hình 11: Sơ đồ User Case admin sửa, xóa chuyến xe	27
Hình 12: Giao diện đăng nhập	30
Hình 13: Giao diện đăng ký	31
Hình 14: Giao diện tìm kiếm vé một chiều.....	31
Hình 15: Kết quả tìm kiếm một chiều.....	32
Hình 16: Giao diện chọn ghế	33
Hình 17: Giao diện tìm kiếm vé khứ hồi.....	34
Hình 18: Giao diện tìm kiếm và giao diện ghế	34
Hình 19: Giao diện thông tin vé.....	35
Hình 20: Giao diện thanh toán payOS	36
Hình 21: Giao diện hóa đơn	36
Hình 22: Giao diện đánh giá chuyến xe.....	37
Hình 23: Giao diện thông tin liên hệ.....	37
Hình 24: Giao diện thông tin khách hàng	38

DANH MỤC BẢNG BIỂU

Bảng 1: Các Module chính để tương tác hệ thống.....	5
Bảng 2: Biến toàn cục	7
Bảng 3: Các lỗi trong hệ thống	7
Bảng 4: Lưu thông tin tài khoản người dùng và admin	16
Bảng 5: Bảng liên hệ	17
Bảng 6: Lưu thông tin tạo vé tạo vé	18
Bảng 7: Lưu thông tin tạo vé khứ hồi	19
Bảng 8: Lưu thông tin đặt vé khứ hồi	20
Bảng 9: Lưu thông tin đặt vé.....	21

DANH MỤC TỪ VIẾT TẮT

Từ viết tắt	Diễn giải
I/O	Input/Output: Đầu vào/Đầu ra, giao tiếp dữ liệu giữa các hệ thống.
NPM	Node Package Manager: Trình quản lý gói cho Node.js.
HTTP	Hypertext Transfer Protocol: Giao thức trao đổi dữ liệu trên web.
API	Application Programming Interface: Giao diện lập trình ứng dụng.
npm	Node Package Manager: (Giống NPM) Quản lý gói cho Node.js.
util	Utility: Thư viện tiện ích trong Node.js.
fs	File System: Hệ thống tệp trong Node.js.
url	Uniform Resource Locator: Địa chỉ tài nguyên trên web.
querystring	Query String: Chuỗi truy vấn trong URL.
stream	Stream: Dòng dữ liệu, xử lý dữ liệu lớn.
zlib	Compression Library: Thư viện nén trong Node.js.
CPU	Central Processing Unit: Bộ xử lý trung tâm.
__dirname	Đường dẫn thư mục chứa file hiện tại.
__filename	Đường dẫn file hiện tại.
exports	Đối tượng xuất các module trong Node.js.
module	Đối tượng mô tả module trong Node.js.
require	Hàm nạp module trong Node.js.
DNS	Domain Name System: Hệ thống tên miền.

NoSQL	Not Only SQL: Cơ sở dữ liệu không quan hệ.
-------	--------------------------------------------

TÓM TẮT ĐỒ ÁN CHUYÊN NGÀNH

Vấn đề nghiên cứu:

Khi xây dựng hệ thống đặt vé cho công ty vận tải hành khách, cần nghiên cứu các vấn đề về nhu cầu người dùng (tìm kiếm tuyến xe, chọn ghế, thanh toán trực tuyến, lịch sử đặt vé) uy trình vận hành của công ty (quản lý chuyến đi, vé xe) và tích hợp công nghệ thanh toán trực tuyến, linh hoạt. Cuối cùng, cần chú trọng trải nghiệm người dùng, tối ưu giao diện và cá nhân hóa dịch vụ để nâng cao chất lượng hệ thống.

Các hướng tiếp cận:

Để xây dựng hệ thống đặt vé cho công ty vận tải, có thể tiếp cận từ nhiều hướng như: tập trung vào người dùng với giao diện thân thiện, dễ sử dụng và đáp ứng nhu cầu thực tế; số hóa quy trình vận hành để tự động hóa quản lý chuyến đi, vé; áp dụng công nghệ hiện đại như ứng dụng web, cơ sở dữ liệu mạnh mẽ và API thanh toán trực tuyến. Đồng thời, cần tối ưu hiệu suất bảo mật dữ liệu và tích hợp các tính năng cá nhân hóa. Ngoài ra, hướng mở rộng như theo dõi hành trình xe sẽ nâng cao trải nghiệm khách hàng và hiệu quả vận hành.

Cách giải quyết vấn đề và một số kết quả đạt được:

Để giải quyết các vấn đề trong hệ thống đặt vé, nhiều giải pháp đã được áp dụng nhằm tối ưu hóa trải nghiệm người dùng và hiệu quả vận hành. Vấn đề bảo mật được cải thiện với mã hóa dữ liệu, giúp bảo vệ thông tin và thuận tiện khi đăng nhập. Quy trình đặt vé và chọn chỗ ngồi trở nên dễ dàng hơn nhờ giao diện trực quan và cập nhật trạng thái chỗ theo thời gian thực. Hệ thống tích hợp nhiều phương thức thanh toán như thẻ tín dụng và ví điện tử, tăng tỉ lệ giao dịch thành công. Việc xác nhận đặt vé và thông báo được thực hiện nhanh chóng qua email hoặc SMS, đảm bảo người dùng luôn được cập nhật. Quy trình hủy vé và hoàn tiền được tối ưu hóa để dễ hiểu và xử lý nhanh, mang lại sự linh hoạt. Lịch trình được quản lý chặt chẽ với thông tin cập nhật và hỗ trợ khách hàng.

Kết quả:

Xây dựng thành công hệ thống đặt vé công ty vận tải hàng khách bằng Nodejs, với giao diện thân thiện phản hồi nhanh. Tăng hiệu quả hoạt động hệ thống

tự động hóa quy trình đặt vé, quản lý chuyến đi và ghế ngồi giúp giảm sai sót và tiết kiệm thời gian. Khách hàng có thể dễ dàng tìm kiếm tuyến xe, chọn ghế, và thanh toán trực tuyến mọi lúc, mọi nơi. Hệ thống gửi thông về vé đã đặt và lịch trình, giúp khách hàng yên tâm hơn. Dữ liệu cá nhân và giao dịch được mã hóa, đảm bảo an toàn, tạo niềm tin cho khách hàng.

MỞ ĐẦU

1. Lý do chọn đề tài

Trong bối cảnh hiện nay, nhu cầu di chuyển của con người ngày càng tăng cao, cùng với sự phát triển mạnh mẽ của ngành vận tải hành khách. Việc đặt vé trực tuyến không chỉ mang lại sự tiện lợi mà còn trở thành yếu tố thiết yếu giúp cải thiện trải nghiệm người dùng, đặc biệt là trong những chuyến đi cần lập kế hoạch trước hoặc khi lựa chọn chỗ ngồi theo mong muốn. Tuy nhiên, thực tế cho thấy nhiều hệ thống đặt vé hiện nay vẫn còn hạn chế, chưa tối ưu được quy trình tìm kiếm và chọn chỗ ngồi, thiếu tích hợp thanh toán linh hoạt, và chưa đảm bảo tốt về việc quản lý lịch trình cũng như hỗ trợ khách hàng kịp thời. Những vấn đề này không chỉ làm giảm sự hài lòng của hành khách mà còn ảnh hưởng đến hiệu quả vận hành của các công ty vận tải.

Với đề tài “Xây dựng hệ thống đặt vé công ty vận tải hàng khách bằng Nodejs” được chọn vì sự cần thiết của hệ thống đặt vé, nhằm bắt kịp xu hướng công nghệ và đáp ứng yêu cầu ngày càng cao của người dùng. Bên cạnh đó, sự xuất hiện của các phương thức thanh toán đa dạng và nhu cầu về thông tin cập nhật theo thời gian thực đã đặt ra yêu cầu phải xây dựng một hệ thống có khả năng tích hợp công nghệ tiên tiến, giúp tăng độ chính xác và tin cậy trong quy trình phục vụ hành khách.

Chính vì thế nên tôi chọn đề tài “xây dựng hệ thống đặt vé công ty vận tải hàng khách bằng Nodejs” còn nhằm mục đích khắc phục những sai sót của các hệ thống hiện có, từ đó cung cấp một giải pháp toàn diện, hiệu quả hơn, vừa tạo ra trải nghiệm người dùng tốt hơn, vừa nâng cao năng lực cạnh tranh cho các công ty vận tải hàng khách.

2. Mục tiêu

Đề tài nhằm xây dựng một hệ thống đặt vé trực tuyến toàn diện với các chức năng chính như quản lý tài khoản, đặt vé, chọn chỗ ngồi, và tích hợp thanh toán. Hệ thống cũng cung cấp khả năng gửi thông báo tự động, xử lý hủy vé linh hoạt và hỗ trợ khách hàng hiệu quả, hướng đến mục tiêu nâng cao sự hài lòng và tin cậy của người dùng.

3. Nội dung

Phát triển hệ thống đặt vé bao gồm các chức năng đặt vé, chọn chỗ ngồi, quản lý lịch trình và tích hợp các công cụ thanh toán, cùng giao diện người dùng thân thiện.

4. Đối tượng và phạm vi nghiên cứu

Hành khách sử dụng dịch vụ vận tải hành khách như xe buýt, tàu hỏa, và các phương tiện công cộng khác; các công ty vận tải cần một giải pháp công nghệ để cải thiện dịch vụ và quản lý vận hành.

5. Phương pháp nghiên cứu

Áp dụng phương pháp phát triển phần mềm hiện đại, tập trung vào việc tối ưu hóa quy trình đặt vé, chọn chỗ ngồi, tìm kiếm thông tin di chuyển và tích hợp các công cụ thanh toán, nhằm tạo ra giao diện người dùng đơn giản, dễ sử dụng và hiệu quả.

CHƯƠNG 1: TỔNG QUAN

Cùng với sự phát triển không ngừng của nền kinh tế và các lĩnh vực trong xã hội. Hiện nay các hoạt động du lịch được mở rộng và điều này đồng nghĩa với việc nhu cầu đi lại của mọi người ngày càng tăng cao. Mặc dù các ngành giao thông và vận tải đã nỗ lực để cải thiện dịch vụ, nhưng vấn đề về số lượng hành khách và chất lượng của mỗi chuyến đi vẫn là một thách thức lớn.

Trên thực tế, nhiều công ty vận tải hành khách vẫn sử dụng các phương pháp quản lý và bán vé truyền thống như bán vé tại quầy, dẫn đến những tình huống không mong muốn cho khách hàng. Đối diện với việc phải xếp hàng dài chờ đợi mua vé hoặc thậm chí là việc thông tin về việc hết vé không rõ ràng, khách hàng cảm thấy bức bối và khó chịu. Hơn nữa, tình huống này cũng tạo điều kiện thuận lợi cho các hoạt động không đúng luật như móc túi, cướp giật và bán vé trái phép. Bên cạnh đó nếu có một lượng khách lớn đến cùng lúc thì nhân viên không kịp đáp ứng. Điều này gặp nhiều khó khăn khi theo dõi và quản lý, cũng giảm hiệu quả kinh doanh đi rất nhiều.

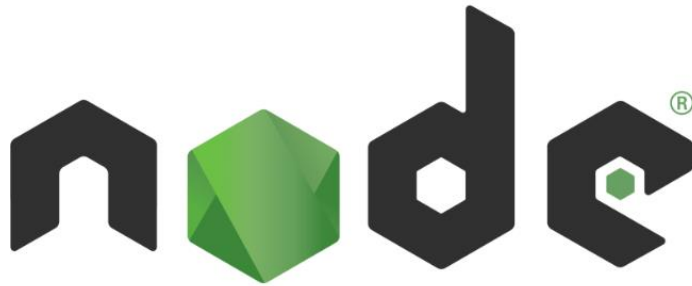
Trong tương lai, việc sử dụng hệ thống đặt vé cho công ty vận tải hàng khách sẽ là một xu hướng phát triển và cải tiến trong việc quản lý bán vé xe khách. Điều này sẽ giúp hãng xe khách tiếp cận đến nhiều khách hàng hơn và cung cấp dịch vụ tốt hơn cho khách hàng.

Biết được điều đó, tôi chọn đề tài “Xây dựng hệ thống đặt vé công ty vận tải hàng khách” là để tìm hiểu các công nghệ sử dụng như ngôn ngữ Nodejs và lưu trữ dữ liệu MongoDB. Bên cạnh đó tìm hiểu sâu hơn về cách thức hoạt động của một trang web bán hàng online. Thông qua đề tài này, giúp tôi nâng cao hiểu biết hơn về các kỹ năng, cũng như các vấn đề bán hàng online nói chung và Website quản lý bán vé xe khách nói riêng.

CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT

2.1. Tổng quan về NodeJS

JavaScript là một trong những ngôn ngữ lập trình phổ biến nhất trên thế giới hiện nay, góp phần xây dựng nên hàng triệu website khác nhau trên Internet. NodeJS là một môi trường runtime cung cấp mọi thành phần cần thiết để thực thi một chương trình viết bằng JavaScript.



Hình 1: Công cụ NodeJS

2.1.2. NodeJS

Node.js là một nền tảng mã nguồn mở được xây dựng trên V8 Javascript engine. Node.js được viết bằng c++ và Javascript. Nền tảng này được phát triển Node.js vào năm 2009 bởi Ryan Lienhart Dahl.

Node.js ra đời khi các developer đời đầu của JavaScript mở rộng nó từ một thứ chỉ chạy được trên trình duyệt thành một thứ có thể chạy trên máy của mình dưới dạng ứng dụng độc lập.

Nguồn mở (Open-source): Mã nguồn của Node.js được công bố công khai, điều này có nghĩa là bất kỳ ai cũng có thể truy cập, sử dụng và đóng góp vào mã nguồn. Node.js được duy trì bởi cộng đồng lập trình viên trên toàn thế giới và hướng dẫn đóng góp của Node.js hướng dẫn cách để có thể góp phần phát triển nó.

Đa nền tảng (Cross-platform): Node.js không phụ thuộc vào bất kỳ hệ điều hành nào cụ thể nào, nghĩa là nó có thể chạy trên Linux, macOS hoặc Windows. Điều này làm cho Node.js trở thành một lựa chọn linh hoạt cho các nhà phát triển muốn xây dựng các ứng dụng có thể hoạt động trên nhiều nền tảng khác nhau mà không cần thay đổi mã nguồn.

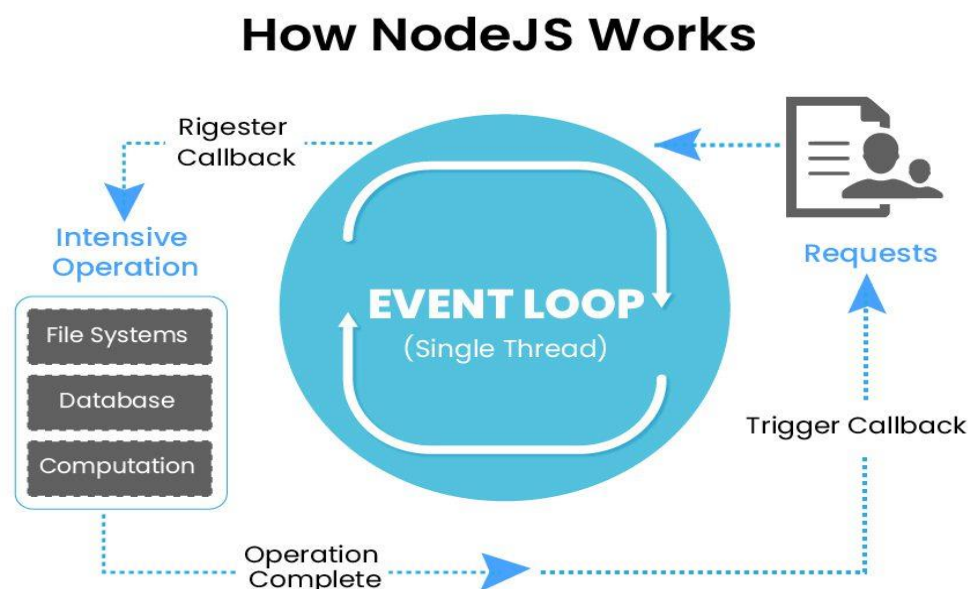
Môi trường thực thi JavaScript (JavaScript runtime environment): Để mã JavaScript có thể được thực thi, cần một môi trường chạy phù hợp. Trong khi trình duyệt như Chrome và Firefox cung cấp một môi trường thực thi cho JavaScript, Node.js mở rộng khả năng này ra ngoài trình duyệt. Node.js cho phép chạy JavaScript trên máy chủ hoặc trong bất kỳ môi trường máy tính nào khác, không chỉ trong trình duyệt.

Dựa trên V8 JavaScript Engine: Node.js được xây dựng dựa trên V8, động cơ JavaScript được phát triển bởi Google cho trình duyệt Chrome. Điều này giúp Node.js có khả năng thực thi JavaScript nhanh và hiệu quả, đồng thời hỗ trợ các tính năng mới nhất của ngôn ngữ JavaScript.

Node.js đã mở rộng khả năng của JavaScript từ việc chỉ phát triển front-end trong trình duyệt để bao gồm cả phát triển back-end. Điều này có nghĩa là các lập trình viên có thể sử dụng cùng một ngôn ngữ lập trình, JavaScript, để phát triển toàn bộ ứng dụng, từ front-end đến back-end, qua đó tạo điều kiện cho việc học tập và phát triển ứng dụng nhanh chóng và hiệu quả hơn.

2.1.2. Các thức hoạt động của NodeJS

Node.js hoạt động dựa trên một số nguyên tắc cơ bản giúp nó hiệu quả trong việc xử lý các ứng dụng có nhiều hoạt động nhập/xuất (I/O) mà không bị chặn, đồng thời giảm đáng kể sự phức tạp trong quản lý các luồng thực thi.



Hình 2: Cách hoạt động của NodeJS

2.1.3. Kiến trúc Non-blocking I/O và Event-Driven

Node.js sử dụng một mô hình non-blocking I/O (input/output) và event-driven, nghĩa là các hoạt động như đọc file, truy vấn cơ sở dữ liệu, hoặc giao tiếp mạng được thực hiện mà không chặn tiến trình chính. Điều này cho phép xử lý nhiều yêu cầu cùng lúc mà không cần tạo nhiều luồng (thread), giúp giảm bớt chi phí liên quan đến quản lý luồng và tối ưu hóa hiệu suất.

Khi một hoạt động I/O được khởi tạo, nó sẽ được gửi đến thực thi trong hệ thống hoặc cơ sở dữ liệu mà không làm chậm tiến trình chính. Sau khi hoạt động hoàn tất, một sự kiện sẽ được phát đi và xử lý bằng các hàm gọi lại (callback).

2.1.4. V8 JavaScript Engine

Node.js được xây dựng trên động cơ JavaScript V8 của Google Chrome, đây là một động cơ rất nhanh cho phép biên dịch mã JavaScript thành mã máy để thực thi trực tiếp trên phần cứng, làm tăng hiệu suất thực thi.

2.1.5. Single-Threaded

Mặc dù Node.js hoạt động trên một luồng duy nhất cho logic ứng dụng của người dùng, nó vẫn sử dụng nhiều luồng ở tầng thấp hơn thông qua thư viện libuv để xử lý các hoạt động I/O. Tuy nhiên, những chi tiết này được ẩn giấu khỏi người dùng, giúp việc lập trình đơn giản hơn mà vẫn đảm bảo hiệu suất.

2.1.6. Event Loop

Trái tim của Node.js là “event loop”. Đây là vòng lặp sự kiện mà ở đó Node.js tiếp tục lắng nghe sự kiện và thực hiện các hàm gọi lại khi một sự kiện được kích hoạt. Vòng lặp sự kiện cho phép Node.js xử lý hàng nghìn kết nối đồng thời mà không cần phải tạo ra chi phí quản lý luồng.

2.1.7. Trigger Callback

Khi thao tác I/O hoàn tất, hệ điều hành thông báo cho Node.js, và Node.js sau đó thực thi hàm callback tương ứng để xử lý kết quả hoặc tiếp tục xử lý logic.

2.1.8. NPM (Node Package Manager)

NPM là hệ thống quản lý gói cho Node.js cho phép các nhà phát triển dễ dàng chia sẻ và sử dụng mã nguồn từ nhau. NPM là một trong những kho lưu trữ mã nguồn mở lớn nhất thế giới và chứa hàng ngàn module có thể được tích hợp vào ứng dụng.

Tổng hợp lại, Node.js mang đến một mô hình hiệu quả và mạnh mẽ cho các ứng dụng web và máy chủ, nhờ khả năng xử lý đồng thời nhiều hoạt động I/O mà không bị chặn và qua đó tối ưu hóa việc sử dụng tài nguyên và cải thiện hiệu suất.

2.1.9. Require-relative

Require làm 3 thứ:

Tải module đi kèm với Node.js như hệ thống file và HTTP từ Node.js API.

Tải thư viện thứ 3 như Express và Mongoose cài đặt từ npm.

Giúp require file và mô-đun hoá project.

Require là 1 chức năng, và nó nhận tham số path tĩnh chính và trả về module.export [1].

2.2. Thành phần của Node.js

Để hiểu rõ hơn về Node.JS, chúng ta cần nắm bắt một số thuật ngữ cơ bản. Trong phần tiếp theo này, chúng ta sẽ tìm hiểu về các khái niệm như I/O, không đồng bộ, không chặn và sự kiện và lập trình hướng sự kiện.

2.2.1. Module

Module trong Node.js giống như những gói thư viện nhỏ, chứa các hàm, đối tượng và các lớp được viết sẵn để thực hiện các tác vụ cụ thể. Điều này giúp cho việc phát triển ứng dụng trở nên modun hóa và dễ dàng quản lý. Để sử dụng một module trong ứng dụng chỉ cần sử dụng hàm require(). Ví dụ:

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

Ở Node.JS có nhiều Module với nhiều chức năng khác nhau và đều cần thiết cho một ứng dụng web. Dưới đây là một số Module thường được sử dụng trong ứng dụng web:

Bảng 1: Các Module chính để tương tác hệ thống

Module chính	Mô tả
http	Tạo và quản lý các máy chủ HTTP để xây dựng các ứng dụng web
util	Cung cấp các hàm tiện ích hỗ trợ lập trình, như định dạng dữ liệu, kiểm tra kiểu dữ liệu...

fs	Tương tác với hệ thống file để đọc, ghi, xóa và quản lý các file
url	Phân tích cú pháp URL để trích xuất các thông tin như protocol, host, path...
querystring	Xử lý query string trong URL để lấy ra các tham số truyền vào
stream	Làm việc với các luồng dữ liệu một cách hiệu quả, giúp tối ưu hóa việc xử lý dữ liệu lớn
zlib	Nén và giải nén dữ liệu để giảm kích thước file và tăng tốc độ truyền tải

2.2.2. Bảng điều khiển

Bảng điều khiển là một module có chức năng hỗ trợ việc gỡ lỗi các ứng dụng JavaScript. Bảng điều khiển này cung cấp một giao diện tương tác để in ra các thông tin quan trọng như giá trị của biến, kết quả của biểu thức và các thông báo lỗi.

Một trong những phương pháp phổ biến để sử dụng bảng điều khiển là sử dụng hàm `console.log()`. Hàm này cho phép in ra bất kỳ giá trị nào lên màn hình console. Ví dụ:

```
console.log('Hello, world!');
```

2.2.3. Cluster

NodeJS, nổi tiếng với khả năng lập trình không đồng bộ, thường được xây dựng dựa trên mô hình đơn luồng. Tuy nhiên, để cải thiện hiệu năng và tận dụng tối đa tài nguyên CPU, Cluster là một giải pháp hoàn hảo.

Module Cluster trong NodeJS cho phép tạo ra nhiều tiến trình con (worker processes), mỗi tiến trình sẽ xử lý các yêu cầu đến. Điều này giúp cân bằng tải và ngăn chặn việc một tiến trình bị quá tải. Các worker process này chia sẻ cùng một công máy chủ, tạo nên một cụm các tiến trình làm việc cùng nhau.

2.2.4. Đối tượng toàn cục

Trong NodeJS các đối tượng toàn cục đóng vai trò vô cùng quan trọng. Đây là những biến hoặc hàm được sẵn có trong mọi module của NodeJS, cho phép các

lập trình viên truy cập và sử dụng chúng trực tiếp mà không cần khai báo lại. Một số đối tượng toàn cục phổ biến ở trong bảng dưới đây:

Bảng 2: Biến toàn cục

Đối tượng toàn cục	Mô tả
<code>__dirname</code>	Trả về đường dẫn tuyệt đối đến thư mục chứa file đang thực thi.
<code>__filename</code>	Trả về đường dẫn tuyệt đối đến file đang thực thi.
<code>exports</code>	Dùng để xuất các biến hoặc hàm từ module hiện tại.
<code>module</code>	Biểu diễn module hiện tại.
<code>require</code>	Dùng để nhập các module khác vào module hiện tại.

2.2.5. Xử lý lỗi

Trong ứng dụng của Node.JS thường sẽ gặp 4 lỗi phổ biến như sau:

Bảng 3: Các lỗi trong hệ thống

Lỗi JavaScript	Lỗi này xảy ra do cú pháp JavaScript không hợp lệ hoặc lỗi thời gian chạy. Ví dụ bao gồm EvalError, SyntaxError, RangeError, ReferenceError, TypeError và URIError.
Lỗi từ hệ thống	Lỗi này phát sinh từ các vấn đề cấp hệ thống, chẳng hạn như tệp không tồn tại hoặc đóng socket.
Lỗi từ người dùng	Lỗi này do logic hoặc đầu vào được xác định bởi người dùng gây ra.
Lỗi assertion	Lỗi này cho thấy vi phạm các điều kiện hoặc logic mong đợi.

2.2.6. Streaming (Luồng)

Luồng (Streaming) là một dòng dữ liệu liên tục được truyền từ nguồn đến đích. Trong lập trình, luồng được sử dụng để đọc hoặc ghi dữ liệu vào các thiết bị ngoại vi như file, mạng hoặc các bộ nhớ đệm (buffer).

Có 4 loại luồng chính:

Luồng đọc: Chỉ cho phép đọc dữ liệu từ nguồn.

Luồng ghi: Chỉ cho phép ghi dữ liệu vào đích.

Luồng duplex: Cho phép vừa đọc vừa ghi dữ liệu.

Luồng chuyển đổi: Thay đổi định dạng hoặc mã hóa dữ liệu trong quá trình truyền.

2.2.7. Buffer (Bộ nhớ đệm)

Buffer là một cấu trúc dữ liệu được sử dụng để lưu trữ và thao tác hiệu quả với dữ liệu nhị phân. Buffer cung cấp một cách linh hoạt để làm việc với dữ liệu trong bộ nhớ.

2.2.8. Tên miền (Domain)

Module tên miền giúp ngăn chặn và khắc phục các lỗi tiềm ẩn trong website. Module này sử dụng hai cơ chế chính để đảm bảo hoạt động ổn định của hệ thống:

Liên kết nội bộ: Bộ phát lỗi sẽ được tích hợp trực tiếp vào mã nguồn của website, giúp xác định và xử lý lỗi một cách nhanh chóng và hiệu quả.

Liên kết bên ngoài: Có thể thêm thủ công các bộ phát lỗi vào tên miền thông qua cấu hình hệ thống. Điều này đặc biệt hữu ích khi muốn theo dõi và quản lý các lỗi từ các nguồn bên ngoài.

2.2.9. Hệ thống phân giải tên miền Domain Name System DNS

Module DNS dùng để tương tác với các máy chủ Hệ thống tên miền (DNS). Module DNS cho phép các nhà phát triển hiệu quả phân giải tên miền thành địa chỉ IP.

Có hai phương pháp chính để phân giải DNS:

dns.resolve(): Phương pháp này thiết lập kết nối mạng với máy chủ DNS để thực hiện quá trình phân giải. Phù hợp cho các ứng dụng yêu cầu tra cứu DNS thời gian thực hoặc cần xử lý các truy vấn DNS phức tạp.

dns.lookup(): Phương pháp này thực hiện phân giải DNS mà không cần kết nối mạng. Phù hợp cho các trường hợp kết nối mạng bị hạn chế hoặc không đáng tin cậy, chẳng hạn như ứng dụng ngoại tuyến hoặc cơ chế lưu trữ cache.

2.2.10. Debugger (Trình gỡ lỗi)

Node.JS có công cụ gỡ lỗi tích hợp vô cùng tiện lợi, giúp các nhà phát triển kiểm tra và khắc phục lỗi trong mã nguồn một cách hiệu quả. Mặc dù không sở hữu nhiều tính năng phức tạp như các trình gỡ lỗi chuyên dụng, nhưng công cụ này hoàn toàn đáp ứng nhu cầu kiểm tra code đơn giản trong quá trình phát triển ứng dụng.

Để bắt đầu sử dụng công cụ gỡ lỗi, chỉ cần mở terminal và sử dụng lệnh inspect trước tên tệp JavaScript mà muốn kiểm tra. Ví dụ:

\$ node inspect myscript.js

2.3. Ưu và nhược điểm của Node.JS

Về Ưu điểm:

IO hướng sự kiện không đồng bộ, cho phép xử lý nhiều yêu cầu đồng thời.

Sử dụng JavaScript – một ngôn ngữ lập trình dễ học.

Chia sẻ cùng code ở cả phía client và server.

NPM(Node Package Manager) và module Node đang ngày càng phát triển mạnh mẽ.

Cộng đồng hỗ trợ tích cực.

Cho phép stream các file có kích thước lớn.

Về nhược điểm:

Không có khả năng mở rộng, vì vậy không thể tận dụng lợi thế mô hình đa lõi trong các phần cứng cấp server hiện nay.

Khó thao tác với cơ sở dữ liệu quan hệ.

Mỗi callback sẽ đi kèm với rất nhiều callback lồng nhau khác.

Cần có kiến thức tốt về JavaScript.

Không phù hợp với các tác vụ đòi hỏi nhiều CPU [2].

2.4. Tổng quan về MongoDB

Trước tiên, để tìm hiểu về MongoDB, cần tìm hiểu về Database. Database là một ổ chứa dữ liệu ở mức vật lý. Các collection ở mỗi Database được thiết lập lưu trữ ở một nơi trong máy tính. MongoDB có thể tạo ra nhiều dạng cơ sở dữ liệu.

2.4.1. MongoDB

MongoDB là phần mềm cơ sở dữ liệu mã nguồn mở NoSQL, được thiết kế hướng theo đối tượng và hỗ trợ trên đa nền tảng. Các bảng MongoDB có cấu trúc linh hoạt, cho phép dữ liệu không cần tuân theo bất kỳ dạng cấu trúc nào.

MongoDB hoạt động trên collection, hướng tài liệu kiểu JSON thay cho bảng để tăng tốc độ truy vấn. MongoDB có chức năng định hướng tài liệu cung cấp, hiệu suất cao, tính sẵn sàng cao và khả năng mở rộng dễ dàng. Collection trong MongoDB về bản chất thì có thể hiểu là nhóm các document, một collection sẽ chứa các tập document. Ở MongoDB, các collection không theo bản chất cũ vì các document không tuân theo cấu trúc, nghĩa là các document trong một collection không có cấu trúc cố định như nhau (không cần chia ra các cột để lưu trữ), vậy thì không cần phải định nghĩa thành phần các cột trong một collection như trong cơ sở dữ liệu quan hệ.



Hình 3: Công cụ MongoDB

2.4.2. Giải thích về NoSQL trong khái niệm MongoDB

NoSQL là cơ sở dữ liệu được xây dựng dành riêng cho các ứng dụng hiện đại, dữ liệu lưu trữ lớn và ứng dụng nền web thời gian thực. NoSQL cần đơn giản trong thiết kế, kiểm soát tính khả dụng tốt và yêu cầu database lưu trữ dữ liệu dung lượng cực lớn, tăng khả năng chịu lỗi tốt, thực hiện các truy vấn tốc độ cao không đòi hỏi năng lực phần cứng và tài nguyên hệ thống.

2.4.3. Lịch sử phát triển của MongoDB

Năm 2007, công ty phần mềm 10gen (sau này đổi thành MongoDB Inc.) đã lập kế hoạch phát triển MongoDB như một sản phẩm dịch vụ.

Năm 2009, MongoDB được phát hành mã nguồn mở, được viết bởi ngôn ngữ C++. Chính vì viết bằng C++ nên MongoDB có khả năng tính toán ở tốc độ cao, được đánh giá cao hơn các hệ quản trị cơ sở dữ liệu hiện nay.

Năm 2019, MongoDB Inc. đã hợp tác với Alibaba Cloud (công ty con thuộc tập đoàn Alibaba), cung cấp cho khách hàng giải pháp MongoDB dưới dạng dịch vụ.

Năm 2020, MongoDB được đánh giá là cơ sở dữ liệu NoSQL phổ biến nhất. Hiện nay, MongoDB nằm trong top 4 cơ sở dữ liệu được các nhà phát triển đánh giá cao trong khảo sát dành cho các nhà phát triển Stack Overflow. Với bộ tính năng đa điểm, MongoDB được sử dụng bởi nhiều thương hiệu lớn như Networks, Adobe, Google, Ebay, Facebook,...

2.5. Sử dụng MongoDB mang lại những lợi ích

Nhờ vào tính linh hoạt và sự đa dạng hóa trong cách thức chuyển hóa cơ sở dữ liệu, MongoDB trở thành giải pháp đáng tin cậy đối với nhiều doanh nghiệp. MongoDB không chỉ là một cơ sở dữ liệu dựa trên tài liệu điển hình, mà nó còn nổi bật nhờ vào một số tính năng khác như sau.

2.5.1. Truy vấn cơ sở dữ liệu đặc biệt

Thay vì sử dụng các lược đồ để xác định trước thì MongoDB lại sử dụng một trong những lợi thế của mình là khả năng xử lý dữ liệu mà không cần lược đồ xác định. Để nâng cao tính tối ưu và khả năng tiếp cận với các nhà phát triển, MongoDB đã sử dụng ngôn ngữ truy vấn tương tự như cơ sở dữ liệu SQL. Với khả năng này, MongoDB sẽ giúp chúng ta đơn giản hóa việc sắp xếp, truy vấn, cập nhật và xuất dữ liệu của mình thông qua các phương pháp phổ biến khác.

2.5.2. Cân bằng tải

Để đảm bảo tính khả dụng và độ tin cậy của dịch vụ, yêu cầu quy mô ứng dụng đám mây của doanh nghiệp phải tăng lên. MongoDB sẽ duy trì sự cân bằng thông qua sự phân phối các tập dữ liệu trên nhiều máy ảo cùng lúc, có thể thực hiện tác vụ đọc và ghi ở mức có thể chấp nhận. Đối với MongoDB dữ liệu lưu trữ được mở rộng tính qui mô theo chiều ngang, đây được gọi là Sharding. Dựa vào điều này, các tổ chức, doanh nghiệp sẽ tiết kiệm được chi phí mở rộng theo chiều dọc của phần cứng, trong khi khả năng hoạt động trên đám mây vẫn giữ nguyên.

2.5.3. Hỗ trợ đa ngôn ngữ

Đây là một trong những điều tuyệt vời khi nhắc đến MongoDB, các phiên bản được cập nhật và phát triển liên tục nhằm hỗ trợ cho quá trình điều khiển các ngôn ngữ lập trình phổ biến như Python, PHP, Ruby, C++, JavaScript,...

2.6. MongoDB có những ưu và nhược điểm

Ưu điểm

MongoDB là một cơ sở dữ liệu hướng tài liệu, sử dụng bộ nhớ nội tại, truy cập dễ dàng nhờ vào việc lập ra các chỉ mục, tăng tốc độ phản hồi truy vấn nhanh. Theo đánh giá thì tốc độ MongoDB có thể nhanh hơn 100 lần so với cơ sở dữ liệu quan hệ.

Sự linh hoạt của cơ sở dữ liệu: MongoDB sử dụng lưu trữ dữ liệu dưới dạng Document JSON nên mỗi collection sẽ có kích cỡ khác nhau và các document cũng khác nhau. Do sử dụng cơ sở dữ liệu không có lược đồ nên điều này mang lại sự linh hoạt và tự do lưu trữ dữ liệu thuộc nhiều loại khác nhau.

Khả năng mở rộng: Lợi thế về cơ sở dữ liệu theo chiều ngang, vì vậy, khi xử lý một dữ liệu lớn thì chúng ta có thể phân phối cho nhiều máy.

Đội ngũ hỗ trợ chuyên nghiệp: Khi gặp phải bất kỳ sự cố nào chúng ta có thể liên hệ trực tiếp đến hệ thống hỗ trợ để xử lý kịp thời.

Tính khả dụng cao: MongoDB không chỉ có các tính năng về sao chép mà có thể sử dụng gridFS (có thể lưu trữ và truy xuất các tệp vượt quá kích thước 16MB). Các tính năng giúp tăng tính khả dụng và đạt hiệu suất cao.

Nhược điểm

Do MongoDB không có tính ràng buộc. Do đó, người dùng phải thực sự cẩn trọng khi thao tác để tránh những sai sót không đáng có.

Các dữ liệu lớn hơn 16MB sẽ không được lưu trữ do giới hạn về kích thước lưu trữ.

MongoDB không được phép Joins như cơ sở dữ liệu quan hệ. Để sử dụng chức năng Joins, chúng ta có thể thêm Coding theo cách thủ công, vì là thủ công nên có thể làm chậm quá trình và bị ảnh hưởng đến hiệu suất.

Lồng dữ liệu trong BSON bị hạn chế, không được phép lồng những dữ liệu hơn 100 cấp.

Không có chức năng Joins nên sẽ có sự dư thừa dữ liệu, điều này là dung lượng bộ nhớ tăng không cần thiết [3].

CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU

3.1. Mô tả hệ thống

Hệ thống đặt vé cho công ty vận tải hành khách là một ứng dụng trực tuyến giúp khách hàng tìm kiếm, đặt vé và thanh toán cho các chuyến xe một cách nhanh chóng và thuận tiện. Khách hàng có thể tìm kiếm chuyến xe dựa trên các tiêu chí như điểm đi, điểm đến, ngày đi và thời gian, khách hàng có thể chọn ghế ngồi theo sở thích. Sau khi xác nhận thông tin, khách hàng có thể thanh toán trực tuyến qua các cổng thanh toán như payOS quét mã. Hệ thống cũng giúp quản lý các chuyến đi, số ghế còn trống, đồng thời tự động cập nhật tình trạng vé đã bán. Hệ thống còn gửi thông báo xác nhận đơn hàng, bảo mật dữ liệu khách hàng và giao dịch thanh toán được đảm bảo qua các biện pháp mã hóa và xác thực an toàn. Hệ thống sử dụng các công nghệ như Node.js và MongoDB để lưu trữ dữ liệu.

3.2. Xác định các yêu cầu chức năng của hệ thống

Chức năng của hệ thống bao gồm:

Chức năng đăng ký đăng nhập

Chức năng đăng ký: Yêu cầu người dùng hoặc hành khách cung cấp thông tin như họ tên, email, mật khẩu, số điện thoại, địa chỉ để tạo tài khoản.

Hệ thống kiểm tra thông tin và xác nhận tài khoản.

Chức năng đăng nhập: Yêu cầu người dùng nhập email và mật khẩu để vào hệ thống.

Quản lý thông tin cá nhân: Cho phép người dùng cập nhật thông tin cá nhân, sửa thông tin như: tên người dùng, email đăng ký, số điện thoại và mật khẩu. Cho phép người dùng xóa tài khoản sao khi không cần đến.

Chức năng đặt vé

Chọn điểm đi và điểm đến: Cung cấp các điểm đi phổ biến hoặc tìm kiếm theo từ khóa

Chọn ngày giờ khởi hành: Hiển thị lịch trình xe chạy theo ngày giờ cụ thể.

Chọn phương tiện vận tải: Gợi ý phương tiện vận tải

Xem chi tiết chuyến đi: Hiển thị thông tin vé như bảng số xe, điểm đi, điểm đến, ngày đi, số ghế còn lại, số tiền cho một vé.

Chức năng chọn chỗ ngồi

Sơ đồ chỗ ngồi: Hiển thị sơ đồ ghế ngồi chi tiết ghế trống, ghế đã đặt

Đặt chỗ: Người dùng chọn ghế dựa trên sơ đồ hiển thị, cập nhật trạng thái ghế ngay khi được đặt.

Chức năng thanh toán

Phương thức thanh toán: Hỗ trợ thanh toán qua các công thẻ tín dụng, thẻ ghi nợ, chuyển khoản ngân hàng, tích hợp ví điện tử (Momo, ZaloPay, PayPal).

Xác nhận thanh toán: Gửi thông báo thành công hoặc thất bại của giao dịch.

Chức xác nhận đặt vé

Sau khi thanh toán thành công: Hệ thống gửi vé qua email hoặc SMS.

Vé cũng có thể được lưu trữ trong tài khoản của người dùng để dễ truy cập.

Chức năng hủy/Hoàn tiền

Hủy vé: Người dùng có thể yêu cầu hủy vé trước thời gian khởi hành theo chính sách.

Hoàn tiền: Tự động xử lý hoàn tiền qua phương thức thanh toán ban đầu.

Thông báo tình trạng yêu cầu hoàn tiền qua email/SMS.

Hỗ trợ khách hàng

Cung cấp kênh liên hệ: Gửi yêu cầu hỗ trợ qua biểu mẫu trực tuyến.

Thông báo

Thông tin chuyến đi: Gửi thông báo nhắc nhở về lịch trình, thời gian khởi hành.

Thay đổi lịch trình: Gửi cảnh báo khi chuyến đi bị thay đổi hoặc hủy.

Đánh giá & Phản hồi

Đánh giá chuyến đi: Hành khách để lại đánh giá về chất lượng dịch vụ, tài xế và phương tiện.

Gửi góp ý: Cung cấp biểu mẫu để người dùng gửi ý kiến nhằm cải thiện dịch vụ.

3.3. Thiết kế dữ liệu hệ thống

Chi tiết các collection

Collection: User

Mô tả: Collection này lưu trữ thông tin của người dùng trên hệ thống, bao gồm cả thông tin cá nhân, vai trò (admin hoặc user), và thông tin đăng nhập.

Bảng 4: Lưu thông tin tài khoản người dùng và admin

SST	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	name	String	Tên người dùng, bắt buộc phải nhập.
2	email	String	Địa chỉ email của người dùng, bắt buộc phải nhập và không được trùng lặp.
3	password	String	Mật khẩu của người dùng, bắt buộc phải nhập.
4	phone	String	Số điện thoại của người dùng, bắt buộc phải nhập.
5	address	String	Địa chỉ nơi ở của người dùng, bắt buộc phải nhập.
6	role	String, Enum: ['admin', 'user'], Default: 'user'	Vai trò của người dùng trong hệ thống (có thể là 'admin' hoặc 'user'). Mặc định là 'user'.

Chi tiết các collection

Collection: lienhe

Mô tả: Collection này lưu trữ các thông tin liên hệ từ người dùng gửi tới hệ thống. Bao gồm nội dung liên hệ, email người gửi, và các phản hồi (nếu có) từ admin.

Bảng 5: Bảng liên hệ

SST	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	name	String	Tên người gửi thông điệp, bắt buộc phải nhập.
2	email	String	Địa chỉ email của người gửi, bắt buộc phải nhập.
3	message	String	Nội dung thông điệp do người dùng gửi, bắt buộc phải nhập.
4	reply	String	Nội dung phản hồi của admin, không bắt buộc.
5	repliedAt	Date	Thời gian phản hồi của admin, không bắt buộc.
6	createdAt	Date	Thời gian tạo thông điệp, mặc định là thời điểm hiện tại.

Chi tiết các collection

Collection: taove

Mô tả: Collection này lưu trữ thông tin về các vé xe được tạo trên hệ thống. Dữ liệu bao gồm thông tin chuyến đi, thời gian, giá vé, và các thông tin liên quan đến ghế ngồi.

Bảng 6: Lưu thông tin tạo vé tạo vé

SST	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	bangSoXe	String	Mã vé, tự động tăng
2	diemDi	String	Biên số xe, bắt buộc nhập, không được trùng lặp.
3	diemDen	String	Điểm khởi hành, bắt buộc nhập.
4	ngayDi	Date	Điểm đến, bắt buộc nhập.
5	ghe	Number	Ngày đi, bắt buộc nhập.
6	soTien	Number	Số lượng ghế, bắt buộc nhập.
7	gianiemyet	Number	Số tiền vé, bắt buộc nhập.
8	thoigian	String	Giá niêm yết của vé, bắt buộc nhập.
9	createdAt	Date	Thời gian khởi hành, định dạng HH:mm, phải nằm trong khoảng từ 06:00 đến 23:00, bắt buộc nhập.

Chi tiết các collection

Collection: taovekhuhoi

Mô tả: Collection này lưu trữ thông tin về các vé khứ hồi (hai chiều) trong hệ thống. Dữ liệu bao gồm thông tin điểm đi/đến, ngày đi/về, thời gian, số ghế, và giá vé.

Bảng 7: Lưu thông tin tạo vé khứ hồi

SST	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	bangSoXe	String	Biển số xe, bắt buộc và không được trùng lặp.
2	diemDi	String	Điểm xuất phát của chuyến đi, bắt buộc phải nhập.
3	diemDen	String	Điểm đến của chuyến đi, bắt buộc phải nhập.
4	ngayDi	Date	Ngày khởi hành, bắt buộc phải nhập.
5	ngayVe	Date	Ngày về của chuyến đi khứ hồi, bắt buộc phải nhập.
6	ghe	Number	Số ghế đặt cho chuyến đi, bắt buộc phải nhập.
7	ghe2	Number	Số ghế đặt cho chuyến về, bắt buộc phải nhập.
8	soTien	Number	Số tiền vé tổng cộng, bắt buộc phải nhập.
9	gianiemyet	Number	Giá niêm yết của vé, bắt buộc phải nhập.
10	thoigian	String	Thời gian khởi hành, phải nằm trong khoảng từ 06:00 đến 23:00.
11	thoigian2	String	Thời gian về, phải nằm trong khoảng từ 06:00 đến 23:00.

12	createdAt	Date	Thời gian tạo vé, mặc định là thời điểm hiện tại.
----	-----------	------	---------------------------------------------------

Chi tiết các collection

Collection: datvekuhohi

Mô tả: Collection này lưu trữ thông tin đặt vé khứ hồi của người dùng, bao gồm các chi tiết về chuyến đi, chuyến về, ghế đã chọn, thời gian, giá vé, và đánh giá từ người dùng.

Bảng 8: Lưu thông tin đặt vé khứ hồi

SST	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	bangSoXe	String	Biển số xe của chuyến đi.
2	diemDi	String	Điểm xuất phát của chuyến đi.
3	diemDen	String	Điểm đến của chuyến đi.
4	ngayDi	Date	Ngày khởi hành của chuyến đi.
5	ngayVe	Date	Ngày về của chuyến đi khứ hồi.
6	gianiemyet	Number	Giá niêm yết của vé.
7	soTien	Number	Tổng số tiền thanh toán.
8	gheDaChon	Number	Mảng các số ghế đã chọn cho chiều đi.
9	gheDaChon2	Number	Mảng các số ghế đã chọn cho chiều về.
10	userId	ObjectId, Ref: "User"	ID người dùng, tham chiếu đến bảng User.
11	thoigian	String	Thời gian khởi hành, phải nằm trong khoảng từ 06:00 đến 18:00.

12	thoigian2	String	Thời gian về, phải nằm trong khoảng từ 06:00 đến 18:00.
13	createdAt	Date	Thời gian tạo bản ghi, mặc định là thời điểm hiện tại.
14	reviews	Varchar	Mảng các đánh giá (gồm điểm đánh giá và bình luận).

Chi tiết các collection

Collection: DatVe

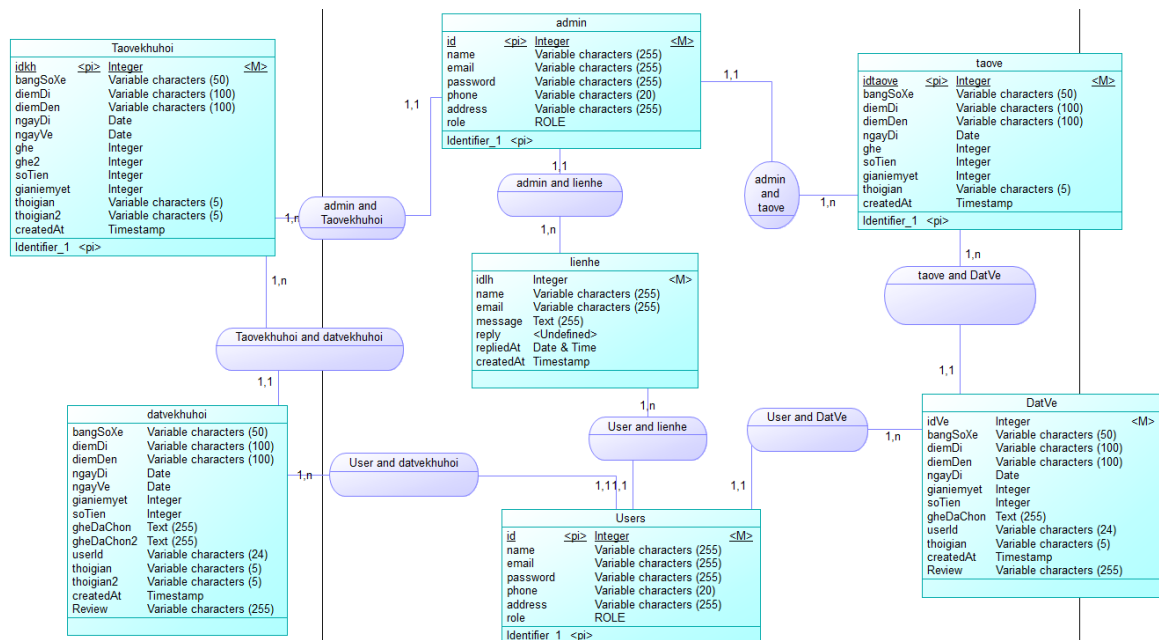
Mô tả: Collection này lưu trữ thông tin đặt vé cho các chuyến đi đơn lẻ. Mỗi vé đều có một idVe duy nhất, bao gồm các thông tin như điểm đi, điểm đến, ngày đi, ghế đã chọn, thông tin người dùng, và các đánh giá liên quan.

Bảng 9: Lưu thông tin đặt vé

SST	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	idVe	Number	ID tự động tăng, đảm bảo mỗi vé có một mã duy nhất.
2	bangSoXe	String	Biển số xe của chuyến đi.
3	diemDi	String	Điểm xuất phát.
4	diemDen	String	Điểm đến.
5	ngayDi	Date	Ngày khởi hành.
6	gianiemyet	Number	Giá niêm yết của vé.
7	soTien	Number	Tổng số tiền phải trả.
8	gheDaChon	Number	Mảng chứa các số ghế đã chọn.
9	userId	ObjectId, Ref: "User"	ID người dùng, tham chiếu đến bảng User.

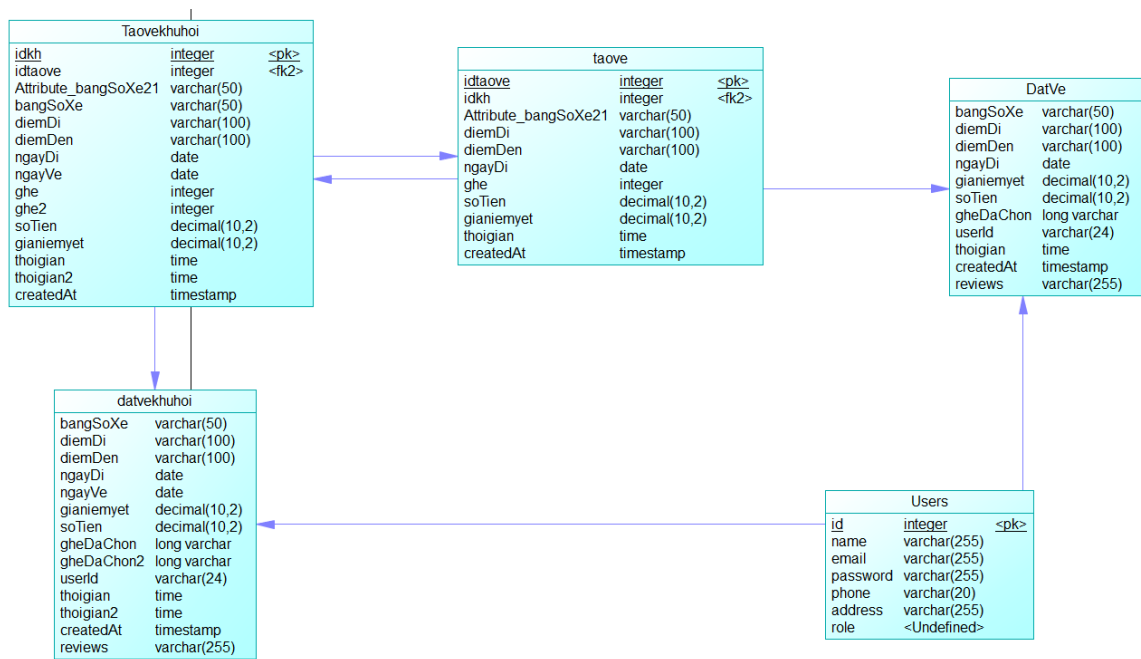
10	thoigian	String	Thời gian khởi hành (dạng HH:mm), phải nằm trong khoảng từ 06:00 đến 18:00.
11	createdAt	Date	Thời điểm tạo bản ghi, mặc định là thời gian hiện tại.
12	reviews	Varchar	Mảng chứa đánh giá từ người dùng (bao gồm rating và comment).

3.4. Mô hình dữ liệu mức quan niệm



Hình 4: Mô hình dữ liệu mức quan niệm

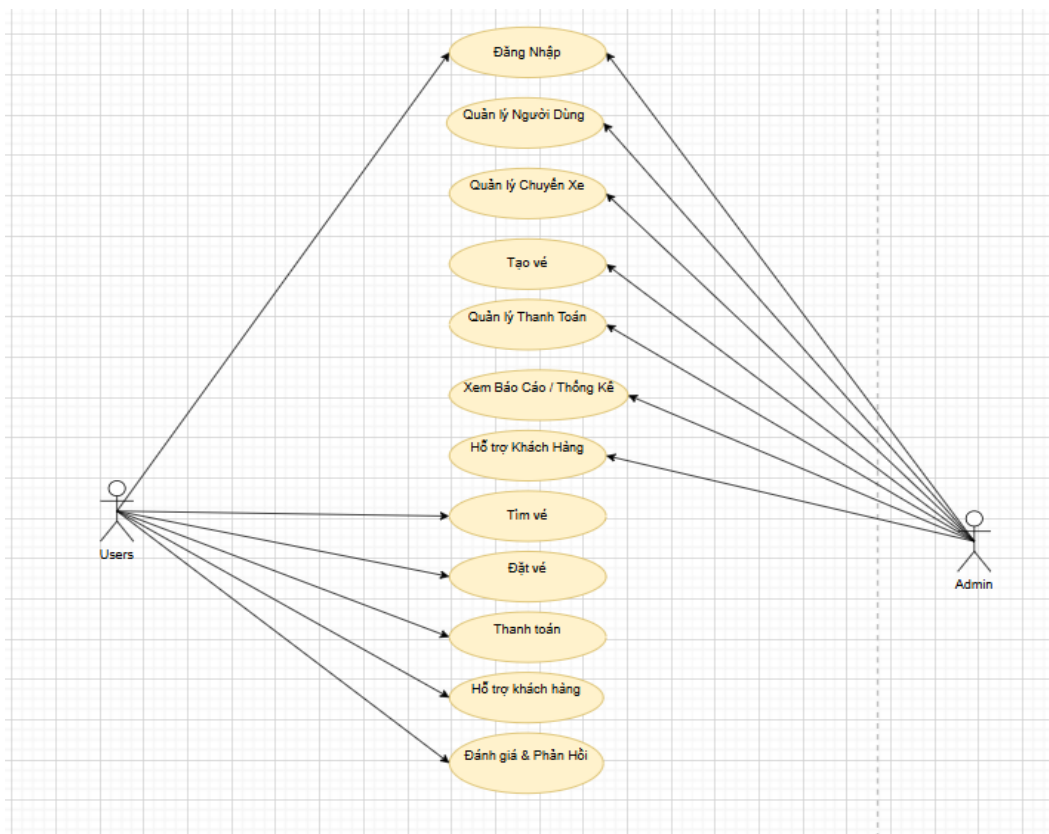
3.5. Mô hình dữ liệu mức logic



Hình 5: Mô hình dữ liệu mức logic

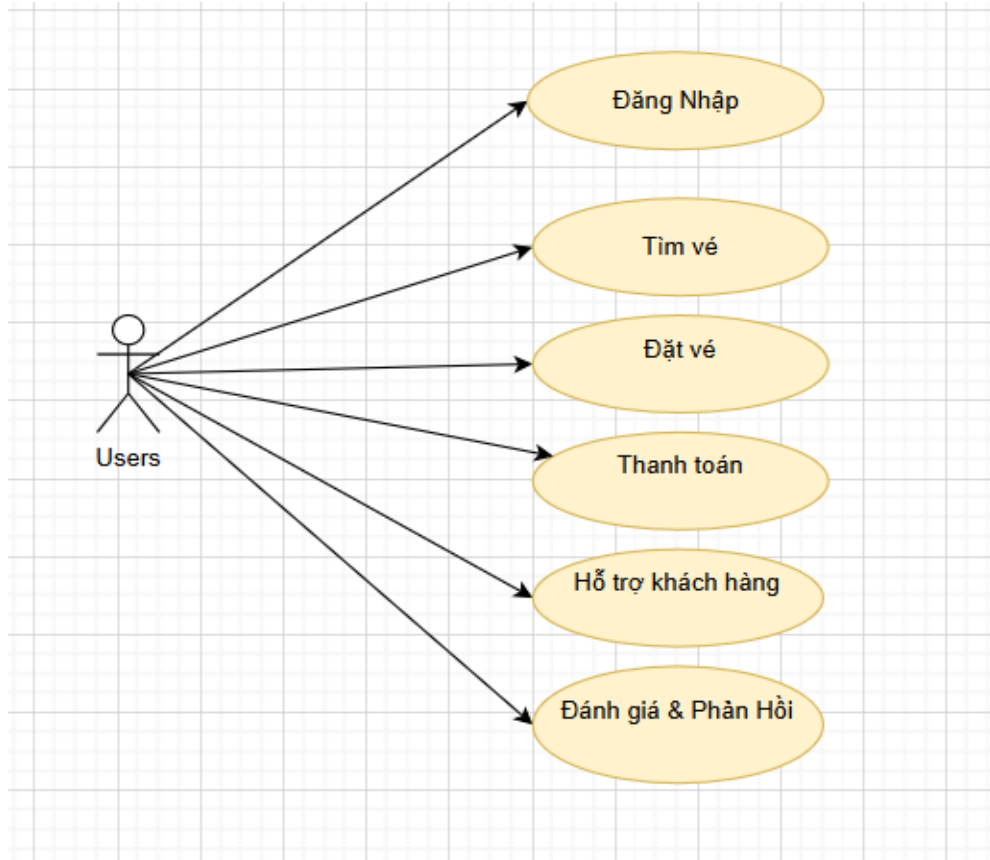
3.6. Thiết kế xử lý hệ thống

3.6.1. Sơ đồ User Case tổng quát



Hình 6: Sơ đồ User Case tổng quát

3.6.2. Sơ đồ User Case người dùng



Hình 7: Sơ đồ User Case người dùng

Sơ đồ use case mô tả một cách tổng quan về các tương tác giữa người dùng (Users) và một hệ thống đặt vé. Các use case chính được thể hiện trong sơ đồ:

Đăng nhập: Người dùng xác thực danh tính để truy cập vào các chức năng của hệ thống.

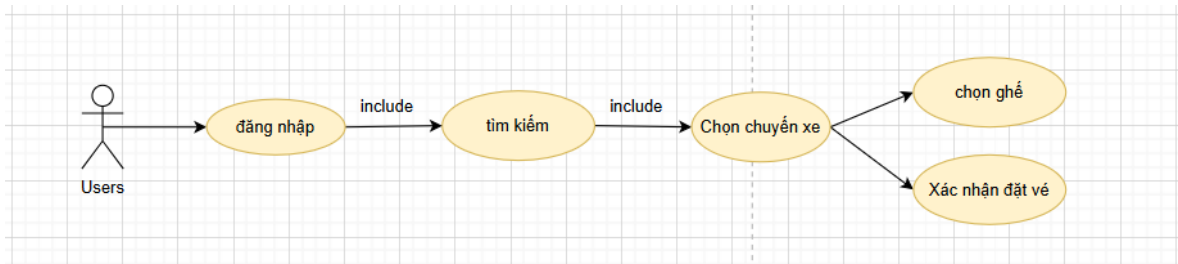
Tìm vé: Người dùng tìm kiếm các chuyến đi phù hợp với nhu cầu của mình (ví dụ: điểm đi, điểm đến, ngày đi).

Đặt vé: Người dùng chọn chuyến đi và tiến hành đặt vé.

Thanh toán: Người dùng thực hiện thanh toán cho vé đã đặt.

Hỗ trợ khách hàng: Người dùng liên hệ với hệ thống để được hỗ trợ về các vấn đề liên quan đến đặt vé, hoàn vé.

Đánh giá & Phản hồi: Người dùng đưa ra đánh giá và phản hồi về dịch vụ của hệ thống.



Hình 8: Sơ đồ User Case người dùng tìm kiếm chuyến xe

Sơ đồ use case này mô tả một quy trình đơn giản hóa của việc người dùng đặt vé xe thông qua một hệ thống trực tuyến. Các use case chính và mối quan hệ:

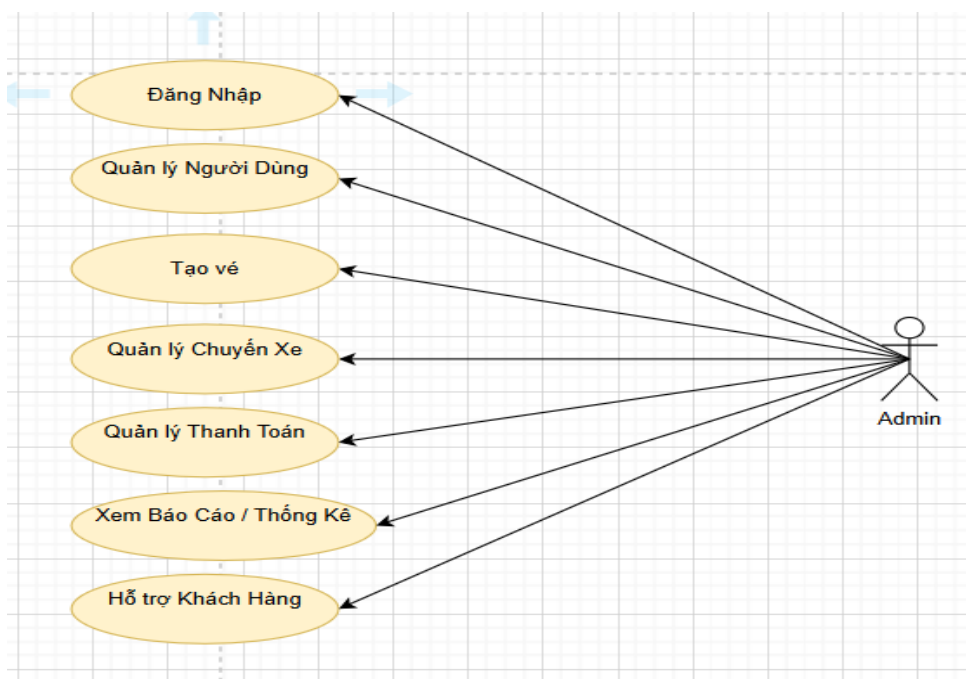
Đăng nhập: Người dùng bắt đầu bằng việc đăng nhập vào hệ thống. Đây là bước đầu tiên và bắt buộc để truy cập các chức năng khác.

Tìm kiếm: Sau khi đăng nhập, người dùng tiến hành tìm kiếm các chuyến xe phù hợp với nhu cầu của mình (điểm đi, điểm đến, ngày đi,...). Use case "Tìm kiếm" bao gồm use case "Chọn chuyến xe". Điều này có nghĩa là khi người dùng tìm kiếm, họ sẽ chọn một chuyến xe cụ thể từ kết quả tìm kiếm.

Chọn ghế: Sau khi chọn chuyến xe, người dùng sẽ tiến hành chọn ghế ngồi trên xe.

Xác nhận đặt vé: Cuối cùng, người dùng xác nhận thông tin đặt vé và hoàn tất quá trình đặt vé.

3.6.3. Sơ đồ User Case admin



Hình 9: Sơ đồ User Case admin

Sơ đồ use case cung cấp một cái nhìn tổng quan về các chức năng mà một người quản trị (Admin) có thể thực hiện trong một hệ thống quản lý vé xe. Khác với sơ đồ trước đó tập trung vào người dùng cuối, sơ đồ này thể hiện góc nhìn của người quản lý hệ thống. Các use case chính:

Đăng nhập: Admin đăng nhập vào hệ thống để truy cập các chức năng quản lý.

Quản lý người dùng: Admin thực hiện các tác vụ liên quan đến quản lý tài khoản người dùng như thêm, sửa, xóa.

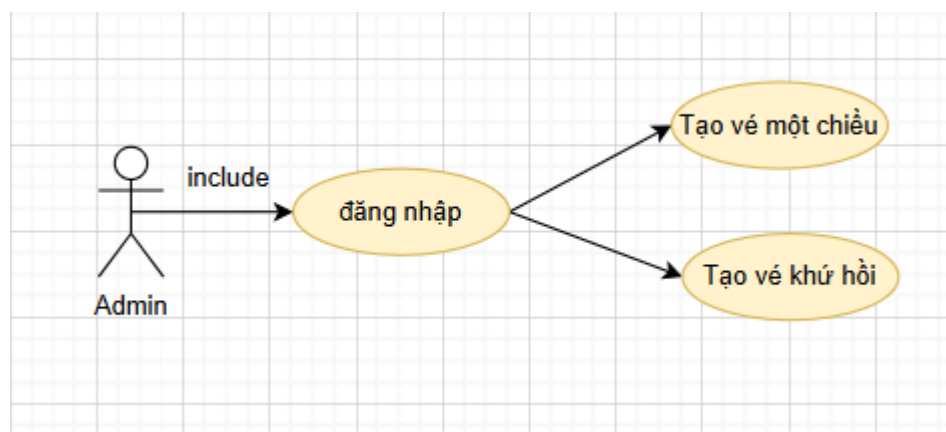
Tạo vé: Admin có thể tạo mới các loại vé, cấu hình giá vé, tuyến đường.

Quản lý chuyến xe: Admin quản lý thông tin các chuyến xe như lịch trình, xe.

Quản lý thanh toán: Admin quản lý các giao dịch thanh toán.

Xem báo cáo/thống kê: Admin xem các báo cáo thống kê về doanh thu, số lượng vé bán, để đánh giá hiệu quả hoạt động của hệ thống.

Hỗ trợ khách hàng: Admin có thể trực tiếp hỗ trợ khách hàng khi có vấn đề phát sinh.



Hình 10: Sơ đồ User Case admin dùng để tạo vé xe

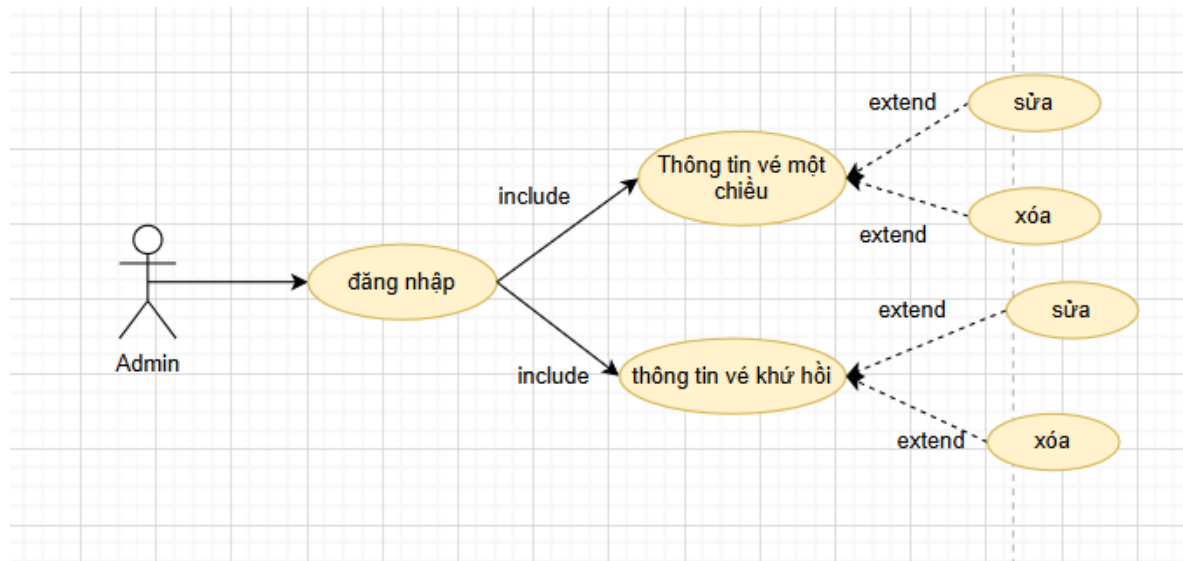
Sơ đồ use case mô tả một phần nhỏ trong hệ thống quản lý vé, tập trung vào quy trình tạo vé từ góc nhìn của người quản trị (Admin). Các use case chính và mối quan hệ:

Đăng nhập: Đây là bước bắt buộc trước khi thực hiện bất kỳ tác vụ nào khác.

Admin cần đăng nhập vào hệ thống để xác thực quyền truy cập.

Tạo vé một chiều: Admin tạo vé cho hành trình một chiều.

Tạo vé khứ hồi: Admin tạo vé cho hành trình khứ hồi.



Hình 11: Sơ đồ User Case admin sửa, xóa chuyến xe

Sơ đồ use case cung cấp một cái nhìn chi tiết hơn về hoạt động quản lý thông tin vé của người quản trị (Admin) trong hệ thống đặt vé. Các use case chính và mối quan hệ:

Đăng nhập: Như các sơ đồ trước, đây là bước bắt buộc trước khi thực hiện bất kỳ tác vụ nào khác.

Thông tin về một chiều: Use case này bao gồm các hoạt động liên quan đến việc quản lý thông tin vé một chiều, chẳng hạn như xem chi tiết vé, cập nhật thông tin vé, và xóa vé.

Thông tin về khứ hồi: Tương tự, use case này liên quan đến việc quản lý thông tin vé khứ hồi.

3.7. Các bước xây dựng website

Để xây dựng được website cần cài đặt các câu lệnh có sẵn dưới đây là một số câu lệnh đã cài đặt và sử dụng trong hệ thống đặt vé xe khách.

Bước 1: Cài đặt express

```
npm install express
```

Chức năng: Express là một framework tối giản cho Node.js dùng để xây dựng các ứng dụng web và API một cách nhanh chóng.

Tác dụng trong mã: Tạo ứng dụng Express. Sử dụng middleware và định tuyến.

Bước 2: Cài đặt mongoose

`npm install mongoose`

Chức năng: Mongoose là thư viện dùng để kết nối Node.js với MongoDB, hỗ trợ định nghĩa schema, quản lý dữ liệu.

Tác dụng trong mã: Kết nối đến cơ sở dữ liệu MongoDB. Định nghĩa và sử dụng các mô hình (models).

Bước 3: Cài đặt bcryptjs

`npm install bcryptjs`

Chức năng: Thư viện để mã hóa mật khẩu và các dữ liệu nhạy cảm.

Tác dụng trong mã: Mã hóa mật khẩu khi người dùng đăng ký. So sánh mật khẩu nhập vào với mật khẩu đã mã hóa trong cơ sở dữ liệu.

Bước 4: Cài đặt express-session

`npm install express-session`

Chức năng: Thư viện để quản lý phiên làm việc (session) trong Express.

Tác dụng trong mã: Lưu thông tin người dùng vào session sau khi đăng nhập. Quản lý các trạng thái trong phiên làm việc.

Bước 5: Cài đặt body-parser

`npm install body-parser`

Chức năng: Middleware để phân tích dữ liệu từ yêu cầu HTTP (POST, PUT) trong phần body.

Tác dụng trong mã: Phân tích dữ liệu từ form hoặc JSON để sử dụng trong mã.

Bước 6: Cài đặt moment

`npm install moment`

Chức năng: Thư viện để xử lý ngày tháng một cách dễ dàng.

Tác dụng trong mã: Định dạng ngày giờ. Tính toán khoảng thời gian.

Bước 7: Cài đặt dotenv

`npm install dotenv`

Chức năng: Thư viện để quản lý biến môi trường từ tệp `.env`.

Tác dụng trong mã: Bảo mật các thông tin nhạy cảm như URI kết nối cơ sở dữ liệu, API key.

Bước 8: Cài đặt cors

`npm install cors`

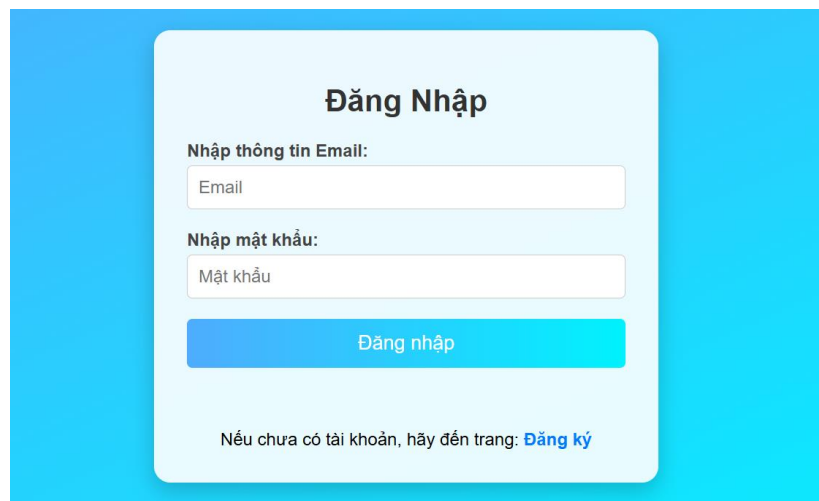
Chức năng: Thư viện để xử lý CORS (Cross-Origin Resource Sharing), cho phép các ứng dụng truy cập tài nguyên từ domain khác.

Tác dụng trong mã: Hỗ trợ yêu cầu HTTP từ các domain khác nhau (frontend/backend).

CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU

4.1. Giao diện đăng nhập và đăng ký tài khoản

Giao diện đăng nhập được thiết kế cho phép người dùng nhập địa chỉ email và mật khẩu đã đăng ký từ trước để truy cập vào hệ thống đặt vé xe. Điều này giúp đảm bảo an toàn và bảo mật thông tin của người dùng khi họ sử dụng



Đăng Nhập

Nhập thông tin Email:

Email

Nhập mật khẩu:

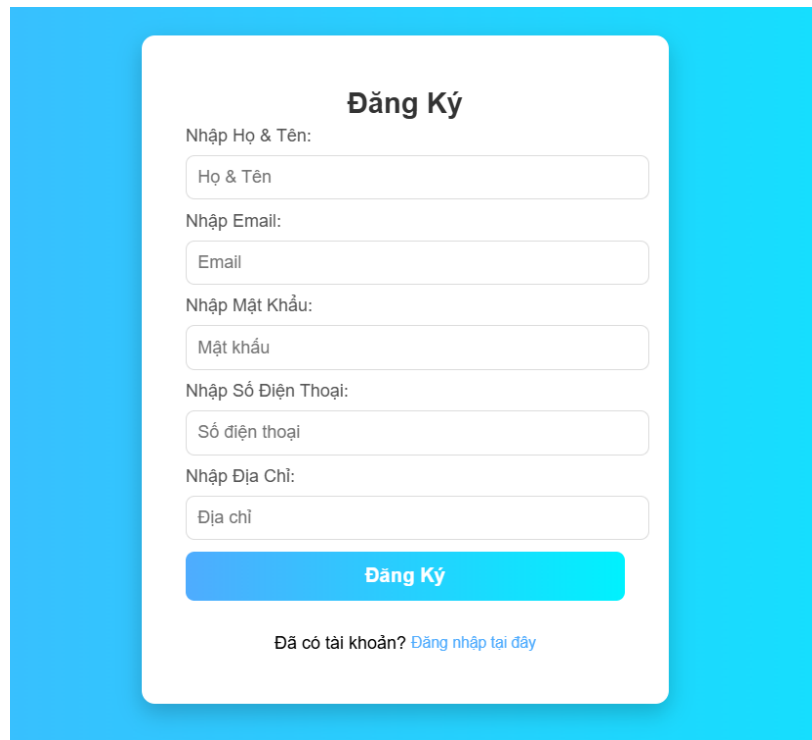
Mật khẩu

Đăng nhập

Nếu chưa có tài khoản, hãy đến trang: [Đăng ký](#)

Hình 12: Giao diện đăng nhập

Giao diện đăng ký được thiết kế để người dùng tạo tài khoản mới bằng cách nhập thông tin cá nhân bao gồm: họ và tên, địa chỉ email, mật khẩu, số điện thoại và địa chỉ. Sau khi hoàn tất và điền đầy đủ các thông tin cần thiết, người dùng có thể nhấn nút "Đăng ký" để hoàn tất quá trình. Sau đó, hệ thống sẽ tự động chuyển hướng người dùng đến trang đăng nhập, từ đó họ có thể đăng nhập vào hệ thống đặt vé xe.



Hình 13: Giao diện đăng ký

4.2. Giao diện tìm kiếm vé một chiều

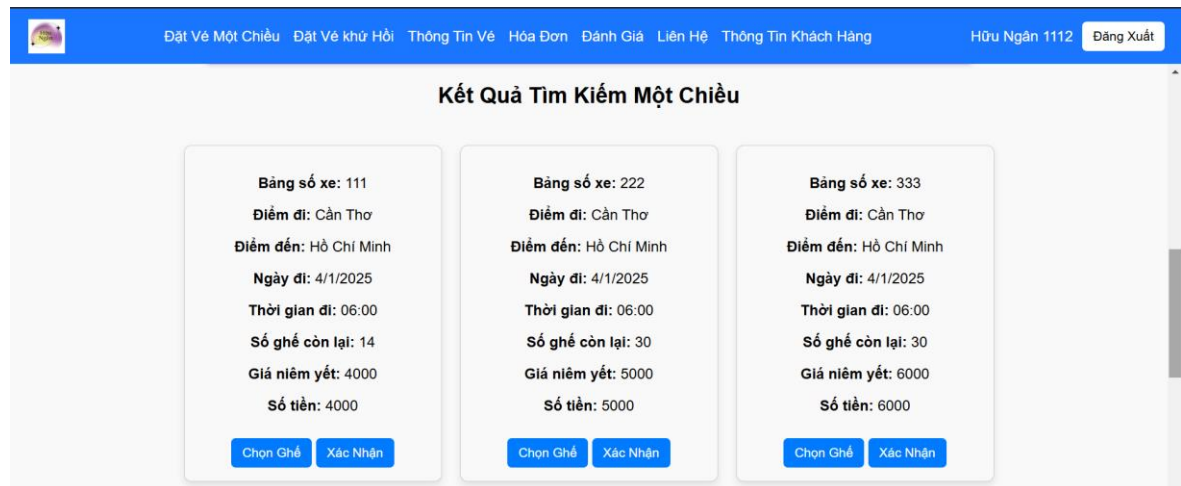
Giao diện tìm kiếm vé xe một chiều cho phép khách hàng thực hiện việc tìm kiếm vé dựa trên các tiêu chí: điểm khởi hành, điểm đến, ngày đi và thời gian khởi hành. Sau khi người dùng nhấn vào nút "Tìm chuyến xe", hệ thống sẽ hiển thị các thông tin về vé phù hợp với yêu cầu tìm kiếm của khách hàng.

Điều này giúp người dùng dễ dàng lựa chọn chuyến xe phù hợp nhất với lịch trình của mình.



Hình 14: Giao diện tìm kiếm vé một chiều

Sau khi người dùng nhấn nút "Tìm chuyến xe", hệ thống sẽ hiển thị thông tin về những chuyến xe có sẵn dựa trên các tiêu chí đã nhập vào khung tìm kiếm, bao gồm điểm khởi hành, điểm đến, ngày đi và thời gian khởi hành. Các thông tin này sẽ được hiển thị rõ ràng để khách hàng có thể dễ dàng lựa chọn chuyến xe phù hợp nhất với nhu cầu của mình.



Hình 15: Kết quả tìm kiếm một chiều

Sau khi khách hàng nhấn chọn ghế, hệ thống sẽ hiển thị danh sách các ghế còn lại để khách hàng lựa chọn. Khách hàng có thể chọn ghế dựa trên số lượng ghế trống hiển thị trên giao diện. Sau khi nhấn "Xác nhận," thông tin vé sẽ được lưu vào hệ thống. Khách hàng có thể truy cập trang thông tin vé để xem chi tiết về vé mà họ đã đặt. Giao diện này giúp việc đặt vé trở nên thuận tiện và dễ dàng hơn, đảm bảo khách hàng có thể lựa chọn được ghế ngồi ưa thích và kiểm tra lại thông tin đặt vé một cách nhanh chóng.

Bảng số xe: 222

Điểm đi: Cần Thơ

Điểm đến: Hồ Chí Minh

Ngày đi: 4/1/2025

Thời gian đi: 06:00

Số ghế còn lại: 30

Giá niêm yết: 5000

Số tiền: 5000

Chọn Ghế

<input type="checkbox"/> 1	<input type="checkbox"/> 2	<input type="checkbox"/> 16	<input type="checkbox"/> 17
<input type="checkbox"/> 3	<input type="checkbox"/> 4	<input type="checkbox"/> 18	<input type="checkbox"/> 19
<input type="checkbox"/> 5	<input type="checkbox"/> 6	<input type="checkbox"/> 20	<input type="checkbox"/> 21
<input type="checkbox"/> 7	<input type="checkbox"/> 8	<input type="checkbox"/> 22	<input type="checkbox"/> 23
<input type="checkbox"/> 9	<input type="checkbox"/> 10	<input type="checkbox"/> 24	<input type="checkbox"/> 25
<input type="checkbox"/> 11	<input type="checkbox"/> 12	<input type="checkbox"/> 26	<input type="checkbox"/> 27
<input type="checkbox"/> 13	<input type="checkbox"/> 14	<input type="checkbox"/> 28	<input type="checkbox"/> 29
<input type="checkbox"/> 15		<input type="checkbox"/> 30	

Xác Nhận

Hình 16: Giao diện chọn ghế

4.3. Giao diện tìm kiếm vé khứ hồi

Giao diện tìm kiếm vé khứ hồi cho phép khách hàng tìm kiếm vé dựa trên các tiêu chí: điểm khởi hành, điểm đến, ngày đi, ngày về và thời gian khởi hành. Sau khi người dùng nhấn vào nút "Tìm chuyến xe," hệ thống sẽ hiển thị các vé phù hợp với yêu cầu tìm kiếm của khách hàng. Thông tin về cả chuyến đi và chuyến về sẽ được hiển thị rõ ràng để khách hàng dễ dàng lựa chọn và lên kế hoạch cho hành trình của mình.

The screenshot shows a web interface for finding round-trip bus tickets. At the top, there's a navigation bar with links: 'Đặt Vé Một Chiều', 'Đặt Vé Khứ Hồi', 'Thông Tin Vé', 'Hầu Ngân', 'Đánh Giá', 'Liên Hệ', 'Thông Tin Khách Hàng', and a user profile 'Hầu Ngân 123' with a 'Đăng Xuất' (Logout) button. Below the navigation bar is a large image of a red and white bus. The main content area is titled 'Tìm Chuyến Xe Khứ hồi' and contains a form with the following fields: 'Điểm đi' (Origin) with a placeholder 'Nhập điểm đi', 'Điểm đến' (Destination) with a placeholder 'Nhập điểm đến', 'Ngày đi' (Departure Date) with a placeholder 'dd/mm/yyyy' and a calendar icon, 'Thời gian' (Time) with a placeholder '--:--:--' and a clock icon, and 'Ngày về' (Return Date) with a placeholder 'dd/mm/yyyy' and a calendar icon. A red 'Tìm chuyến xe' button is at the bottom of the form. Below the form, it says 'Kết Quả Tìm Kiếm khứ hồi'.

Hình 17: Giao diện tìm kiếm vé khứ hồi

Kết quả tìm kiếm vé khứ hồi sẽ hiển thị các thông tin chuyến đi và chuyến về dựa trên các tiêu chí mà khách hàng đã nhập, bao gồm: điểm đi, điểm đến, ngày đi, ngày về và thời gian. Sau khi hệ thống hiển thị kết quả tìm kiếm, khách hàng có thể chọn ghế cho cả lượt đi và lượt về.

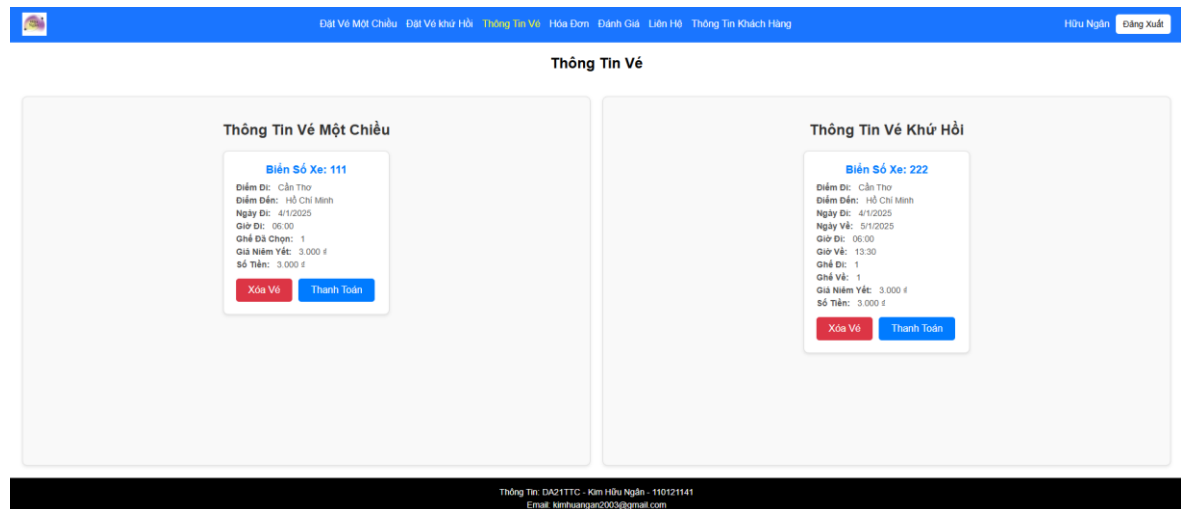
Khi đã hoàn tất việc chọn ghế, khách hàng chỉ cần nhấn nút "Xác nhận" và thông tin vé sẽ được lưu vào hệ thống. Khách hàng có thể truy cập trang thông tin vé để xem lại chi tiết về các vé mà mình đã đặt.

The screenshot displays the seat selection interface for a round-trip bus ticket. The trip details are: 'Bảng số xe: 999', 'Điểm đi: Cần Thơ', 'Điểm đến: Hà Nội', 'Ngày đi: 6/1/2025', 'Ngày về: 7/1/2025', 'Thời gian đi: 06:00', 'Thời gian về: 06:00', 'Số ghế còn lại đi: 29', 'Số ghế còn lại về: 29', 'Giá niêm yết: 3000', and 'Số tiền: 3000'. There are two columns of seat selection buttons, each with a 'Chọn Ghế Đi' (Select Departure Seat) and 'Chọn Ghế Về' (Select Return Seat) button. The seats are numbered 1 to 30. A 'Xác Nhận' (Confirm) button is at the bottom.

Hình 18: Giao diện tìm kiếm và giao diện ghế

4.4. Giao diện xem thông tin vé

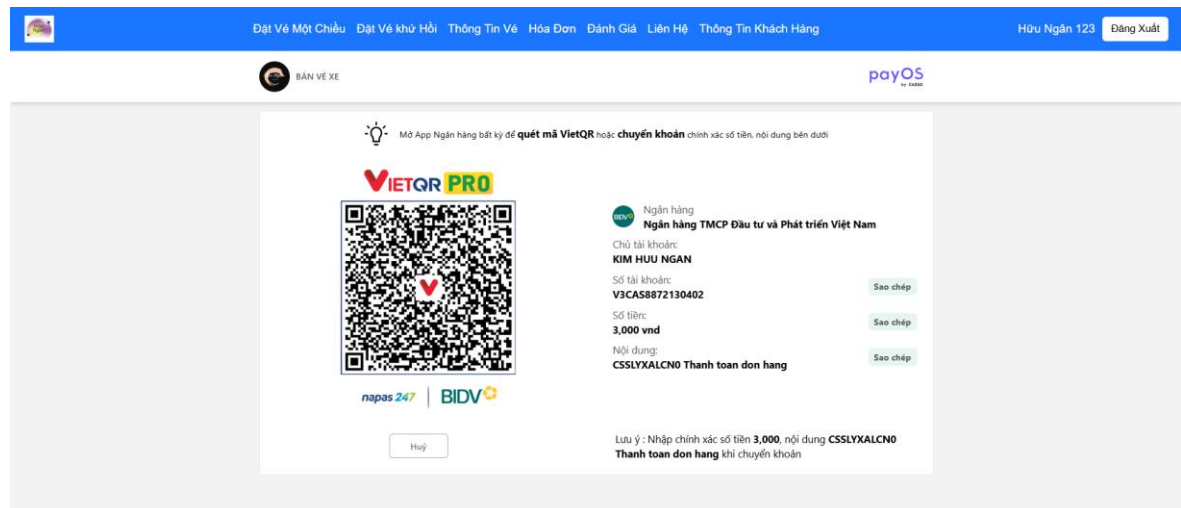
Giao diện thông tin vé được thiết kế để lưu trữ các thông tin vé mà khách hàng đã đặt trước đó. Tại đây, khách hàng có thể xem lại các thông tin chi tiết về vé, cũng như thực hiện các thao tác như xóa vé hoặc thanh toán vé sau khi đã đặt. Điều này giúp khách hàng dễ dàng quản lý và kiểm tra lại các vé đã đặt một cách thuận tiện và hiệu quả



Hình 19: Giao diện thông tin vé

4.5. Giao diện thanh toán

Giao diện thanh toán được thiết kế để hiển thị các thông tin thanh toán sau khi khách hàng nhấn nút "Thanh toán" cho vé đã đặt. Tại đây, khách hàng có thể chọn phương thức thanh toán phù hợp như quét mã QR trên hệ thống hoặc thực hiện chuyển khoản theo số tài khoản được cung cấp. Điều này giúp tối ưu hóa sự tiện lợi và linh hoạt trong quá trình thanh toán cho khách hàng. Ngoài ra, việc có nhiều tùy chọn thanh toán cũng đảm bảo rằng khách hàng có thể hoàn tất giao dịch một cách dễ dàng và an toàn.



Hình 20: Giao diện thanh toán payOS

4.6. Giao diện hóa đơn

Giao diện hóa đơn sẽ hiển thị chi tiết các hóa đơn mà khách hàng đã đặt, bao gồm cả vé một chiều và vé khứ hồi. Tại đây, khách hàng có thể kiểm tra thông tin thanh toán cho từng loại vé một cách rõ ràng và minh bạch. Hóa đơn sẽ liệt kê thông tin về các chuyến đi, ngày giờ khởi hành, giá vé, và các thông tin liên quan khác để khách hàng có thể dễ dàng theo dõi và quản lý các giao dịch của mình. Giao diện này giúp khách hàng nắm bắt thông tin về các vé đã đặt và đảm bảo rằng mọi thông tin thanh toán đều chính xác và đầy đủ.

Danh Sách Hóa Đơn Một Chiều									
Bảng Số Xe	Điểm Đi	Điểm Đến	Ngày Đi	Ghế Đã Chọn	Thời Gian	Giá Niêm Yết	Số Tiền		
333	Cần Thơ	Hồ Chí Minh	1/4/2025	1, 29, 30	06:00	6000	6000		

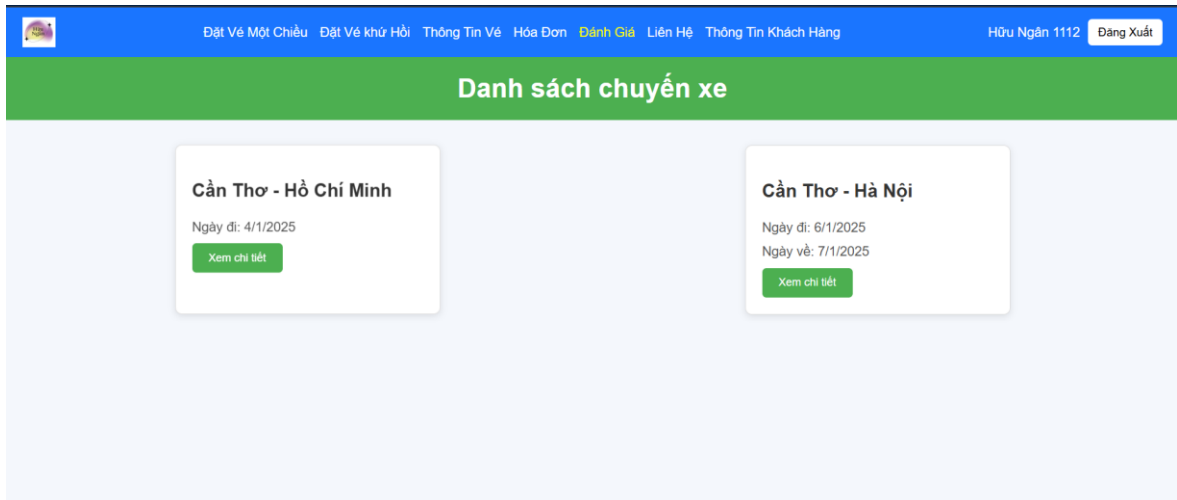
Danh Sách Hóa Đơn Khứ Chiều										
Bảng Số Xe	Điểm Đi	Điểm Đến	Ngày Đi	Ngày Về	Ghế Đi	Ghế Về	Thời Gian Đi	Thời Gian Về	Giá Niêm Yết	Số Tiền
999	Cần Thơ	Hà Nội	1/6/2025	1/7/2025	2	17	06:00	06:00	3000	3000

Hình 21: Giao diện hóa đơn

4.7. Giao diện đánh giá

Giao diện đánh giá cho phép khách hàng cung cấp ý kiến phản hồi về chuyến xe của mình sau khi đặt vé và hoàn thành hành trình. Tại đây, khách hàng có thể đánh giá các yếu tố như chất lượng dịch vụ, thái độ của nhân viên, mức độ thoải mái của xe và thời gian khởi hành/đến nơi. Việc đánh giá này không chỉ giúp cải

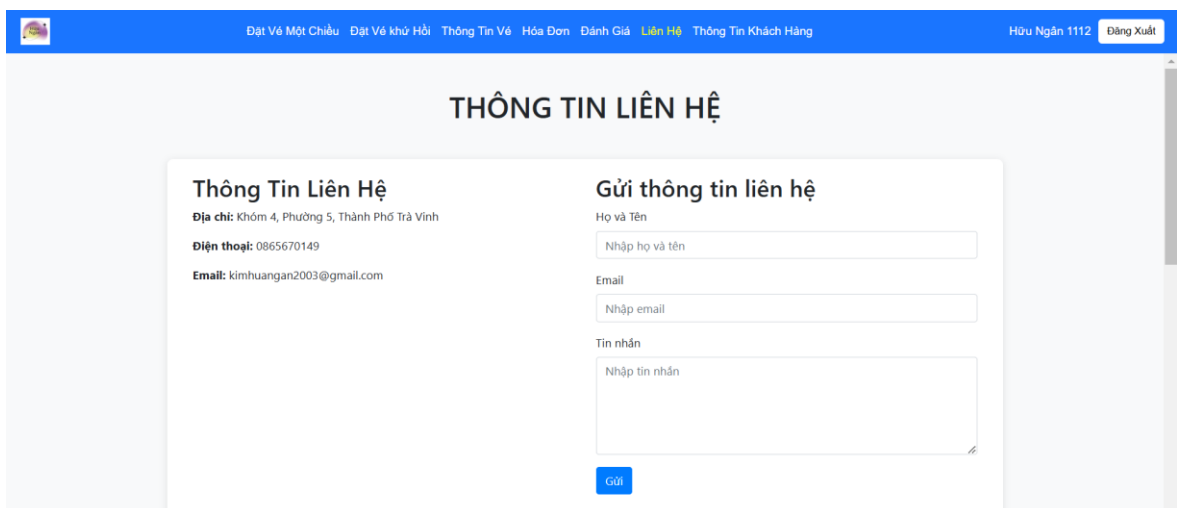
thiện chất lượng dịch vụ cho các chuyến đi sau, mà còn cung cấp thông tin hữu ích cho các khách hàng khác khi họ đang tìm kiếm và so sánh các dịch vụ xe khách.



Hình 22: Giao diện đánh giá chuyến xe

4.8. Giao diện thông tin liên hệ

Giao diện liên hệ được thiết kế để hiển thị các thông tin liên hệ của nhà cung cấp dịch vụ bán vé xe. Tại đây, khách hàng có thể dễ dàng tìm thấy thông tin liên lạc như số điện thoại, email và địa chỉ của nhà cung cấp. Ngoài ra, giao diện này cũng cho phép khách hàng nhập các thông tin cá nhân và yêu cầu cụ thể cần được giải quyết, giúp tạo điều kiện thuận lợi để nhà cung cấp phản hồi và hỗ trợ khách hàng một cách nhanh chóng và hiệu quả.



Hình 23: Giao diện thông tin liên hệ

4.9. Giao diện thông tin khách hàng

Giao diện thông tin khách hàng cho phép người dùng xem và quản lý thông tin cá nhân của mình. Tại đây, khách hàng có thể cập nhật các thông tin như tên, email,

số điện thoại và địa chỉ. Hơn nữa, khách hàng cũng có thể thay đổi mật khẩu của mình để đảm bảo tính bảo mật cho tài khoản. Ngoài ra, giao diện này còn cung cấp tùy chọn xóa tài khoản. Khi người dùng không còn cần sử dụng tài khoản của mình nữa, họ có thể lựa chọn xóa tài khoản một cách dễ dàng và nhanh chóng.

Thông Tin Cá Nhân	
Tên	Hữu Ngân
Email	kimhuangan2003@gmail.com
Số Điện Thoại	0865670149
Địa Chỉ	Trà Vinh
Mật Khẩu	*****

[Sửa Thông Tin](#) [Xóa Tài Khoản](#)

Thông Tin: DA21TTC - Kim Hữu Ngân - 110121141
Email: kimhuangan2003@gmail.com

Hình 24: Giao diện thông tin khách hàng

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết quả đạt được

Trong quá trình thu thập dữ liệu, thiết kế, xây dựng và kiểm thử một cách kỹ lưỡng “Xây dựng hệ thống đặt vé công ty vận tải hành khách bằng Nodejs” ra đời phù hợp với đặt ra yêu cầu đề ra cho việc bán vé xe khách cho các công ty. Hệ thống được phát triển trên nền tảng web. Tạo điều kiện thuận lợi và linh hoạt cho người sử dụng. Hệ thống cho thấy có thể sử dụng một cách dễ dàng và thuận tiện. Hệ thống còn cung cấp những chức năng cơ bản và cần thiết cho việc bán vé xe và quản lý vé xe hiệu quả. Qua đó, quản trị viên có thể thấy được tình hình mua bán vé xe thông qua các những số liệu đã được thống kê. Điều này mang đến một số lợi ích rất lớn trong việc nâng cao hiệu xuất bán vé xe khách, giúp có cơ sở để làm nền móng cho việc thiết lập những mục tiêu mới và cải thiện những nhược điểm trong quá trình kinh doanh.

Với giao diện màu đơn sắc, đơn giản, dễ sử dụng và thân thiện với người dùng, sẽ giúp cho khách hàng có thể sử dụng một cách thoải mái và lâu dài. Đồng thời thông qua hệ thống, khách hàng có thể lưu trữ hóa đơn vé xe, tìm kiếm các thông tin về chuyến đi một cách dễ dàng và tiết kiệm thời gian.

5.2. Hạn chế

Tư vấn đặt vé trực tuyến qua chatbox: Hệ thống chưa có tính năng tư vấn và đặt vé qua chat trực tuyến.

Lịch trình chi tiết: Chưa có hệ thống cung cấp lịch trình cụ thể của các chuyến đi bao gồm các bến xe và các tỉnh sẽ đi qua.

Hệ thống nền tảng vững chắc: Hệ thống hiện tại chưa được xây dựng nền tảng vững chắc.

Hủy vé hoàn tiền: Hệ thống chưa có hủy vé hoàn tiền cho khách hàng.

5.3. Hướng phát triển

Tư vấn đặt vé trực tuyến qua chatbox: Tích hợp một chatbot sử dụng AI để tư vấn và hỗ trợ đặt vé trực tuyến. Chatbot có thể trả lời các câu hỏi thường gặp, gợi ý lịch trình và giúp khách hàng chọn vé.

Cung cấp lịch trình chi tiết: Phát triển một hệ thống quản lý lịch trình chi tiết, bao gồm các điểm dừng và các tỉnh sẽ đi qua. Điều này giúp khách hàng nắm rõ lịch trình và chọn lựa phù hợp hơn.

Xây dựng hệ thống nền tảng vững chắc: Đầu tư vào hạ tầng công nghệ và nhân lực để xây dựng một hệ thống nền tảng vững chắc, đảm bảo khả năng mở rộng và hiệu suất cao. Điều này có thể bao gồm việc nâng cấp server, bảo mật dữ liệu và cải thiện trải nghiệm người dùng.

Tích hợp tính năng hủy vé hoàn tiền cho khách hàng đã thanh toán.

DANH MỤC TÀI LIỆU THAM KHẢO

- [1] C. L. V. Tiến, "vietnix.vn," CÔNG TY CỔ PHẦN GIẢI PHÁP VÀ CÔNG NGHỆ VIETNIX, 13 09 2024. [Online]. Available: <https://vietnix.vn/nodejs-la-gi/#nodejs-la-gi>. [Accessed 20 11 2024].
- [2] TOPDEV, "topdev.vn," CÔNG TY CỔ PHẦN XÂY DỰNG AN PHONG, 27 11 2014. [Online]. Available: <https://topdev.vn/blog/node-js-la-gi/>. [Accessed 23 11 2024].
- [3] D. Xuân, "wiki.tino.org," CÔNG TY CỔ PHẦN TẬP ĐOÀN TINO, 31 07 2023. [Online]. Available: https://wiki.tino.org/mongodb-la-gi/#Gi%E1%BB%9Bi_thi%E1%BB%87u_v%E1%BB%81_MongoDB. [Accessed 27 11 2024].