**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH**

**NGUYỄN QUỐC HƯNG**

**HỆ THỐNG QUẢN LÝ BÁN VÉ XE BUS SỬ DỤNG SPRING BOOT VÀ REACTJS**

**ĐỒ ÁN NGÀNH**

**NGÀNH KHOA HỌC MÁY TÍNH**

**TP. HỒ CHÍ MINH, 2024BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH**

**NGUYỄN QUỐC HƯNG**

**HỆ THỐNG QUẢN LÝ BÁN VÉ XE BUS SỬ DỤNG SPRING BOOT VÀ REACTJS**

**Mã số sinh viên: 2151013034**

**ĐỒ ÁN NGÀNH**

**NGÀNH KHOA HỌC MÁY TÍNH**

**Giảng viên hướng dẫn: TRƯƠNG HOÀNG VINH**

**TP. HỒ CHÍ MINH, 2024**

LỜI CẢM ƠN

Kính thưa các thầy, lời đầu tiên, em xin gửi lời cảm ơn chân thành đến các thầy đã hướng dẫn em làm dự án này trong thời gian qua. Nhờ có sự hướng dẫn tận tình và những kiến thức quý báu mà các thầy đã chia sẻ, em mới có thể hoàn thành tốt và đúng tiến độ dự án này. Đây sẽ là những kinh nghiệm, và là hành trang vững chắc để cho em tiếp tục con đường của mình sau khi rời ghế nhà trường. Một lần nữa, em xin cảm ơn và chúc các thầy thật nhiều sức khỏe để tiếp tục sự nghiệp giáo dục của mình và đạt được nhiều thành công hơn nữa trong cuộc sống.

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

TÓM TẮT ĐỐ ÁN NGÀNH

Dự án của em sẽ tập trung vào việc phát triển một hệ thống quản lý bến xe toàn diện, tích hợp quản lý nhiều nhà xe và nhiều tuyến xe trên cùng một nền tảng. Hệ thống này không chỉ giúp các công ty vận tải thành viên tự quản lý đơn vị của mình mà còn hỗ trợ nhiều mảng như quản lý phương tiện, quản lý tài xế, cung cấp dịch vụ vận chuyển hàng hóa, quản lý đánh giá từ hành khách, theo dõi doanh thu một cách minh bạch và hiệu quả.

Điểm nổi bật của dự án là các công ty vận tải có thể đăng ký khai thác các tuyến đã có sẵn từ ban quản trị bến xe, giúp giảm chi phí vận hành mà vẫn tăng cường khả năng quản lý. Tất cả các thành viên đều có thể vận hành dịch vụ của mình độc lập, đồng thời tiếp cận thông tin một cách nhanh chóng và hiệu quả hơn, từ đó tối ưu hóa hoạt động của toàn hệ thống.

Dự án cũng hướng đến việc mở rộng quy mô, nhằm quản lý được những bến xe lớn với nhiều chi tiết và quy trình phức tạp hơn. Em sẽ tiếp tục nghiên cứu và cải tiến để hệ thống này ngày càng hoàn thiện, đáp ứng được nhu cầu quản lý của các bến xe hiện đại, giúp tăng cường hiệu quả quản lý và tiết kiệm chi phí trong quá trình vận hành.

ABSTRACT

My project will focus on developing a comprehensive bus station management system that integrates the management of multiple bus operators and routes on a single platform. This system will not only allow member transport companies to independently manage their own units but also support various aspects such as vehicle management, driver management, cargo transport services, passenger reviews, and transparent and efficient revenue tracking.

One of the key features of the project is that transport companies will be able to register to operate available routes provided by the bus station management. This will help reduce operating costs while enhancing management efficiency. All members can operate their services independently while accessing information quickly and efficiently, thus optimizing the overall system's operations.

The project also aims to scale up to manage large bus stations with more complex details and processes. I will continue to research and improve the system to make it increasingly complete, meeting the management needs of modern bus stations, enhancing management efficiency, and reducing operational costs.

**MỤC LỤC**

[Chương 1. TỔNG QUAN ĐỀ TÀI 10](#_Toc179927322)

[1.1. Giới thiệu đề tài 10](#_Toc179927323)

[1.2. Mục tiêu và phạm vi đề tài 10](#_Toc179927324)

[1.3. Phương pháp nghiên cứu 11](#_Toc179927325)

[1.3.1. Xác định đề tài 11](#_Toc179927326)

[1.3.2. Phân tích đề tài 11](#_Toc179927327)

[1.3.3. Phân tích nghiệp vụ 13](#_Toc179927328)

[1.3.4. Các công nghệ sử dụng 15](#_Toc179927329)

[1.3.5. Bố cục báo cáo 16](#_Toc179927330)

[Chương 2. CƠ SỞ LÝ THUYẾT 17](#_Toc179927331)

[2.1. Spring Boot 17](#_Toc179927332)

[2.1.1. Tổng quan Spring Boot 17](#_Toc179927333)

[2.1.2. Kiến trúc Spring Boot 18](#_Toc179927334)

[2.1.3. Kiến trúc luồng hoạt động 20](#_Toc179927335)

[2.1.4. Ví dụ 21](#_Toc179927336)

[2.2. ReactJS 21](#_Toc179927337)

[2.2.1. Tổng quan ReactJS 21](#_Toc179927338)

[2.2.2. Kiến trúc ReactJS 22](#_Toc179927339)

[2.3. Cơ sở dũ liệu MySQL 24](#_Toc179927340)

[2.4. Các công cụ hỗ trợ 25](#_Toc179927341)

[2.4.1. Firebase 25](#_Toc179927342)

[2.4.2. Cloudinary 25](#_Toc179927343)

[Chương 3. HỆ THỐNG QUẢN LÝ BẾN XE KHÁCH 26](#_Toc179927344)

[3.1. Giới thiệu đề tài 26](#_Toc179927345)

[3.2. Phân tích hệ thống 27](#_Toc179927346)

[3.2.1. Lược đồ Use Case 27](#_Toc179927347)

[3.2.2. Đặc tả Use Case 31](#_Toc179927348)

[3.3. Thiết kế hệ thống 35](#_Toc179927349)

[3.3.1. Sơ đồ lớp 35](#_Toc179927350)

[3.3.2. Sơ đồ hoạt động 43](#_Toc179927351)

[3.3.3. Sơ đồ tuần tự 52](#_Toc179927352)

[3.3.4. Lược đồ cơ sở dữ liệu quan hệ 57](#_Toc179927353)

[3.4. Kiến trúc hệ thống 61](#_Toc179927354)

[3.4.1. Client (ReactJS) 62](#_Toc179927355)

[3.4.2. Webserver (Spring Boot) 62](#_Toc179927356)

[3.4.3. Database (MySQL) 62](#_Toc179927357)

[3.4.4. Firebase 63](#_Toc179927358)

[3.4.5. Cloudinary 63](#_Toc179927359)

[3.4.6. Luồng hoạt động chính của hệ thống 64](#_Toc179927360)

[3.4.7. Tổng kết 64](#_Toc179927361)

[Chương 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 64](#_Toc179927362)

[4.1. Kết luận 64](#_Toc179927363)

[4.1.1. Kết quả đề tài 64](#_Toc179927364)

[4.1.2. Hạn chế 65](#_Toc179927365)

[4.2. Hướng phát triển 65](#_Toc179927366)

DANH MỤC TỪ VIẾT TẮT

* API: Application Programming Interface
* UI: User Interface
* SPA: Single Page Application
* ORM: Object-Relational Mapping
* DOM: Document Object Model
* SQL: Structured Query Language
* OTP: One Time Password

DANH MỤC HÌNH VẼ

[Hình 2 ‑ 2‑1: Kiến trúc các tầng chính trong Spring Boot 18](#_Toc179927367)

[Hình 2 ‑ 2‑2: Sơ đồ luồng hoạt động trong Spring Boot 20](#_Toc179927368)

[Hình 2 ‑ 2‑3: Kiến trúc của ReactJS 22](#_Toc179927369)

[Hình 2 ‑ 2‑4: Tổ chức của component 23](#_Toc179927370)

[Hình 2 ‑ 2‑5: Kiến trúc của DOM 24](#_Toc179927371)

[Hình 3 ‑ 3‑1: Lược đồ use case của vai trò Admin 27](#_Toc179927372)

[Hình 3 ‑ 3‑2: Lược đồ use case của vai trò Manager\_Company 28](#_Toc179927373)

[Hình 3 ‑ 3‑3: Lược đồ use case của vai trò Driver 29](#_Toc179927374)

[Hình 3 ‑ 3‑4: Lược đồ use case của vai trò Customer 30](#_Toc179927375)

[Hình 3 ‑ 5: Lược đồ use case của User 31](#_Toc179927376)

[Hình 3 ‑ 3‑6: Sơ đồ lớp 35](#_Toc179927377)

[Hình 3 ‑ 3‑7: Sơ đồ hoạt động của UC-01 43](#_Toc179927378)

[Hình 3 ‑ 3‑8: Sơ đồ hoạt động UC-02 45](#_Toc179927379)

[Hình 3 ‑ 3‑9: Sơ đồ hoạt động UC-03 47](#_Toc179927380)

[Hình 3 ‑ 3‑10: Sơ đồ hoạt động UC-04 49](#_Toc179927381)

[Hình 3 ‑ 3‑11: Sơ đồ tuần tự UC-01 52](#_Toc179927382)

[Hình 3 ‑ 3‑12: Sơ đồ tuần tự UC-02 54](#_Toc179927383)

[Hình 3 ‑ 3‑13: Sơ đồ tuần tự UC-03 55](#_Toc179927384)

[Hình 3 ‑ 3‑14: Sơ đồ tuần tự UC-04 56](#_Toc179927385)

[Hình 3 ‑ 3‑15: Lược đồ cơ sở dữ liệu quan hệ 57](#_Toc179927386)

[Hình 3 ‑ 3‑16: Kiến trúc được sử dụng trong hệ thống 61](#_Toc179927387)

DANH MỤC BẢNG

[Bảng 3‑1: Use Case đặt và thanh toán vé 32](#_Toc179927388)

[Bảng 3‑2: Use Case đăng kí vận hành công ty vận tải 33](#_Toc179927389)

[Bảng 3‑3: Use Case tạo tuyến mới 33](#_Toc179927390)

[Bảng 3‑4: Use Case tạo chuyến mới 34](#_Toc179927391)

MỞ ĐẦU

# TỔNG QUAN ĐỀ TÀI

## Giới thiệu đề tài

Ngày nay, nhu cầu đi lại của mọi người ngày càng tăng cao. Các bến xe truyền thống bắt đầu bị quá tải, đặc biệt là tại các thành phố lớn, nơi có lưu lượng hành khách lớn và thường xuyên xảy ra ùn tắc do phương tiện tham gia giao thông đông đúc. Vì vậy, các hệ thống quản lý cũ, lạc hậu trở nên thiếu chính xác, dẫn đến nhiều sai sót và khó khăn trong công tác quản lý. Ngoài ra, vào các dịp cao điểm như lễ, tết, hành khách sẽ gặp khó khăn khi mua vé, thanh toán do hệ thống quá tải và làm ảnh hưởng đến trải nghiệm của người dân. Nhận thấy được sự cần thiết đó, em đã nghiên cứu và xây dựng hệ thống quản lý bến xe khách để giải quyết những bất cập đó. Hệ thống giúp hành khách dễ dàng theo dõi các chuyến xe, mua vé và thanh toán online dễ dàng, tiện lợi. Hệ thống còn đem đến cho khách hàng sự minh bạch và sự an toàn với các thông tin đã cung cấp khi mua vé.

## Mục tiêu và phạm vi đề tài

Nhằm giải quyết các vấn đề bất cập trong công tác quản lý bến xe theo kiểu truyền thống, hệ thống giúp nâng cao hiệu quả trong quy trình vận hành bến xe cũng như trải nghiệm người dùng thông qua việc tự động hóa quy trình bán vé, thanh toán online. Ngoài ra, hệ thống còn lưu trữ thông tin người dùng một cách an toàn, bảo mật, giúp cho thông tin không bị mất, và giảm thiểu sai sót thông tin. Hệ thống còn cho phép quản trị viên giám sát, xác thực thông tin của các công ty vận tải, đảm bảo được tính tin cậy, minh bạch về giá, chất lượng dịch vụ và nhanh chóng xử lý các khiếu nại của hành khách nhằm bảo đảm quyền lợi của khách hàng. Khách hàng sẽ trực tiếp bình luận, đánh giá ngay trên trang chính của các công ty, có thể dễ dàng tra cứu, tìm kiếm các công ty uy tín, các tuyến xe và chuyến xe phù hợp với nhu cầu di chuyển của mình, từ đó sẽ tiết kiệm được rất nhiều thời gian và công sức. Hệ thống cũng có các chức năng quản lý doanh thu, thống kê, báo cáo, hỗ trợ cho các quản trị viên cũng như các công ty có thể nắm bắt được nhu cầu của khách hàng và tình hình kinh doanh để có thể đưa ra các hướng phát triển phù hợp. Mục tiêu chính là giúp nâng cao hiệu quả trong quy trình quản lý, trải nghiệm người dùng, dễ dàng kiểm tra đảm bảo được chất lượng dịch vụ và quyền lợi của khách hàng.

## Phương pháp nghiên cứu

### Xác định đề tài

Để khắc phục các hạn chế của các hệ thống quản lý bến xe truyền thống, hệ thống mới được thiết kế nhằm tối ưu hoá quy trình vận hành và nâng cao trải nghiệm người dùng thông qua việc tự động hoá khi bán vé và thanh toán trực tuyến. Hệ thống còn có khả năng lưu trữ và bảo mật thông tin người dùng, giúp giảm thiểu rủi ro mất mát dữ liệu hoặc sai sót. Quản trị viên có thể giám sát và xác thực thông tin của các công ty vận tải, đảm bảo tính tin cậy và minh bạch về giá cả cũng như chất lượng dịch vụ, đồng thời xử lý nhanh chóng các khiếu nại để bảo vệ quyền lợi của khách hàng.

Khách hàng có thể đánh giá và bình luận trực tiếp trên trang thông tin của từng công ty, dễ dàng tra cứu các tuyến xe và chuyến xe phù hợp với nhu cầu của mình, từ đó tiết kiệm thời gian và công sức. Hệ thống cũng hỗ trợ quản lý doanh thu và cung cấp các chức năng báo cáo thống kê, giúp quản trị viên bến xe, các quản lý công ty nắm bắt được nhu cầu khách hàng và hiệu quả kinh doanh trong từng thời điểm cụ thể, từ đó xây dựng các chiến lược phát triển phù hợp. Mục tiêu của hệ thống là nâng cao hiệu quả quản lý, nâng cao trải nghiệm của người dùng, đồng thời đảm bảo chất lượng dịch vụ và quyền lợi cho khách hàng.

### Phân tích đề tài

#### Bối cảnh

Trong thời buổi hiện nay, nhu cầu di chuyển của người dân ngày càng tăng cao. Do còn sử dụng các phương thức quản lý thủ công nên các bến xe truyền thống gặp nhiều khó khăn trong việc quản lý vé, thông tin của khách hàng, giám sát quá trình hoạt động và chất lượng phục vụ của các công ty vận tải. Các cách quản lý thủ công đã lạc hậu, mang nhiều yếu tố con người nên không tránh khỏi nhiều sai sót, dẫn đến mất thời gian trong công tác vận hành cũng như khó khăn khi lưu trữ, bảo mật thông tin.

Với thời buổi công nghệ thông tin ngày càng phổ biến và phát triển, đã giúp cho các việc trong cuộc sống trở nên dễ dàng hơn. Giúp đưa ra nhiều giải pháp, cải tiến đột phá thay thế cho các cách làm truyền thống kém hiệu quả. Từ đó, giúp cho các công việc quản lý trở nên dễ dàng và thuận tiện hơn. Giúp nâng cao hiệu quả và tối ưu được chi phí vận hành.

Nhìn chung, hệ thống nào cũng sẽ có những hạn chế nhất định mà chúng ta phải luôn tìm cách giải quyết và tối ưu hóa để phù hợp với từng giai đoạn khác nhau. Đặc biệt là sự quá tải khi mà bến xe có quá nhiều xe đậu bên trong cùng một lúc do nhu cầu vận tải hành khách ngày càng tăng. Do đó, chúng ta phải có giải pháp, tính toán và sắp xếp sao cho hợp lý để có thể giảm thiểu đến mức thấp nhất tình trạng quá tải xảy ra, đảm bảo được sự ổn định của bến xe trong quá trình hoạt động.

#### Vấn đề

Các bến xe khách truyền thống thường gặp khó khăn trong công tác quản lý thông tin, xử lý vé thủ công, kiểm soát số lượng xe ra vào, quản lý giá vé, minh bạch trong doanh thu. Những vấn đề dễ nhận thấy nhất là sai sót trong việc bán vé, sai sót thông tin hành khách, và khó khăn trong việc giám sát chất lượng dịch vụ của các công ty vận tải. Ngoài ra, việc thiếu tính năng báo cáo thống kê doanh thu và giám sát hoạt động làm cho quản lý bến xe và các quản lý công ty sẽ thiếu dữ liệu, khó phân tích và đánh giá chính xác được hiệu quả của công ty cũng như bến xe trong quá trình vận hành để kịp thời đưa ra các hướng giải quyết phù hợp nhằm mục đích tối ưu chi phí vận hành và tăng lợi nhuận.

#### Giải pháp

Giải pháp cho vấn đề này là xây dựng một hệ thống quản lý bến xe khách tự động hóa các quy trình, tích hợp các chức năng như bán vé, thanh toán trực tuyến, lưu trữ và bảo mật thông tin hành khách, cũng như cung cấp các chức năng giám sát và đánh giá chất lượng dịch vụ. Cũng phải có cơ chế để kiểm soát lượng xe ra vào bến để đảm bảo cơ sở hạ tầng không bị quá tải gây ra ùn tắc giao thông. Hệ thống này cũng sẽ hỗ trợ quản lý doanh thu, báo cáo, thống kê và các chức năng quản lý, vận hành các công ty vận tải. Bên cạnh đó, khách hàng có thể dễ dàng tra cứu các chuyến đi, đánh giá nhà xe và thực hiện các giao dịch trực tuyến. Tăng tính cạnh tranh giữa các công ty vận tải đồng thời nâng cao chất lượng dịch vụ để bảo đảm quyền lợi của hành khách.

#### Ý nghĩa

Hệ thống quản lý bến xe khách sẽ giúp nâng cao chất lượng vận hành của bến xe, giảm thiểu các sai sót do yếu tố thủ công, bảo đảm tính minh bạch trong quy trình bán vé và quản lý thông tin hành khách. Đồng thời, hệ thống giúp tối ưu hóa hoạt động kinh doanh của các công ty vận tải thông qua việc cung cấp các chức năng quản lý dữ liệu, từ đó người quản lý có thể dễ dàng phân tích và điều chỉnh các hoạt động của công ty và bến xe cho phù hợp với từng thời điểm khác nhau. Đối với hành khách, hệ thống này mang lại một trải nghiệm tiện lợi và nhanh chóng, giúp họ tiết kiệm thời gian và công sức. Điều này không chỉ nâng cao hiệu suất hoạt động của bến xe mà còn góp phần hiện đại hóa và nâng cao chất lượng dịch vụ.

#### Tính khả thi

Với sự phát triển mạnh mẽ của công nghệ thông tin, việc triển khai một hệ thống quản lý bến xe tự động hoàn toàn khả thi. Các công nghệ hiện đại như cơ sở dữ liệu, hệ thống thanh toán trực tuyến có thể dễ dàng tích hợp vào hệ thống. Hơn nữa, nhu cầu cấp thiết về việc cải thiện chất lượng dịch vụ bến xe khách và nâng cao trải nghiệm người dùng là những yếu tố thúc đẩy sự phát triển và làm tăng tính khả thi của dự án này.

### Phân tích nghiệp vụ

#### Quản trị viên bến xe:

* Đăng nhập: sẽ được cấp tài khoản chứ không tự đăng kí được, có thể tự quản lý mật khẩu bằng việc đổi mật khẩu hoặc cấp lại mật khẩu khi quên.
* Quản lý thông tin các công ty vận tải, xác minh thông tin trước khi cho phép công ty vận tải đi vào hoạt động.
* Quản lý thông tin các trạm xe, có thể thêm trạm xe mới.
* Quản lý thông tin, số liệu của bến xe, có thể xem biểu đồ và xuất các báo cáo theo dạng file pdf.
* Giám sát hoạt động, giải quyết các khiếu nại liên quan đến các công ty vận tải.

#### Quản trị viên công ty:

* Đăng nhập/ đăng kí: sau khi có tài khoản và đăng kí mở công ty thì sẽ đợi quản trị viên của bến xe xác nhận và gửi thông báo về mail đã đăng kí với bến xe. Có thể tự quản lý mật khẩu bằng việc đổi mật khẩu hoặc cấp lại mật khẩu khi quên.
* Quản lý các tuyến khai thác của nhà xe, tạo tuyến mới dựa trên các trạm xe có sẵn.
* Quản lý thông tin các chuyến, có thể thêm các chuyến xe mới dựa trên các tuyến xe đã đăng kí trước.
* Quản lý xe, có thể đăng kí sử dụng xe mới, hoặc loại biên xe.
* Quản lý thông tin tài xế, có thể nhận thông tin đăng kí tài xế, xác nhận hợp tác và cũng có thể kết thúc hợp tác với các tài xế.
* Có thể xem các biểu đồ báo cáo thông tin hoạt động của công ty theo từng giai đoạn cụ thể, có thể xuất báo cáo dưới dạng file pdf.

#### Tài xế:

* Đăng nhập/ đăng kí: sau khi có tài khoản và đăng kí làm tài xế của công ty thì sẽ đợi quản trị viên của công ty xác nhận và gửi thông báo về mail đã đăng kí với nhà xe. Có thể tự quản lý mật khẩu bằng việc đổi mật khẩu hoặc cấp lại mật khẩu khi quên.
* Có thể xem lịch trình làm việc, xem các tuyến, loại xe mà mình làm việc.
* Có thể tạo vé nếu có khách dọc đường, và xử lý thanh toán bằng mã QR CODE được tạo từ đường dẫn thanh toán của VNPAY.

#### Khách hàng:

* Đăng nhập/ đăng kí: Tiến hành đăng kí nếu chưa có tài khoản. Có thể tự quản lý mật khẩu bằng việc đổi mật khẩu hoặc cấp lại mật khẩu khi quên.
* Có thể tra cứu vé theo các tuyến xe, nhà xe phù hợp với nhu cầu.
* Đặt ghế, đăng kí chuyển hàng nếu có và thanh toán online thông qua VNPAY.
* Xuất vé ra file pdf nếu đã thanh toán và hủy vé nếu không có nhu cầu mua tiếp.
* Có thể đăng kí làm công ty vận tải hành khách.
* Có thể ứng tuyển làm tài xế ở các công ty.
* Có thể nhận tư vấn từ các công ty vận tải bằng chức chat thời gian thực.
* Có thể bình luận và đánh giá trực tiếp các công ty vận tải.

### Các công nghệ sử dụng

#### Spring Boot

Spring Boot là một framework Java mã nguồn mở được phát triển dựa trên thư viện spring, nó đã được tối ưu hóa bằng việc tích hợp nhiều cấu hình sẵn giúp cho lập trình viên có thể dễ dàng khởi tạo nhanh các dự án với các cấu hình mặc định. Spring Boot hỗ trợ nhiều công cụ tiện ích như Spring Security, Spring Data, … đây là các công cụ rất cần thiết cho một dự án backend, dễ dàng triển khai các API. Tương tác với cơ sở dữ liệu bằng kỹ thuật ORM, giúp đơn giản hóa trong việc tương tác với cơ sở dữ liệu cho lập trình viên khi không phải nhớ quá nhiều về ngôn ngữ SQL thuần túy. Ngày nay, Spring Boot ngày càng trở nên phổ biến vì các tính ưu việt của nó, giúp cho người lập trình dễ dàng xây dựng các dự án và không phải tốn nhiều thời gian cho các cấu hình phức tạp.

#### ReactJS

ReactJS là một thư viện được xây dựng dựa trên ngôn ngữ JavaScript theo dạng mã nguồn mở, do Facebook phát triển, hỗ trợ xây dựng giao diện người dùng (UI) động. Ngoài ra, ReactJS được tổ chức theo khái niệm component, giúp chia nhỏ các UI nhằm tăng khả năng tái sử dụng và làm cho việc quản lý trạng thái của ứng dụng trở lên linh hoạt và rất hiệu quả. ReactJS còn được đánh giá cao và phổ biến vì tính tiện dụng, dễ học và dễ triển khai của nó, dễ dàng nâng cấp, bảo trì, tối ưu để tăng hiệu suất và phù hợp để phát triển các ứng dụng một trang (SPA) vì hạn chế tối thiểu việc tải lại toàn bộ trang mà chỉ cập nhật lại các component có thay đổi nhờ vào việc quản lý các state.

#### MySQL

MySQL là một công nghệ được phát hành dưới dạng mã nguồn mở, hỗ trợ để lưu thông tin của người dùng, các dữ liệu của hệ thống dưới dạng cơ sở dữ liệu quan hệ hoặc cấu trúc. Dễ dàng tích hợp và sử dụng chung với các công nghệ lập trình hiện đại, điển hình là Spring Boot.

#### Firebase

Firebase được phát triển bởi Google, hỗ trợ xây dựng, phát triển các dự án về web và di động, cung cấp các tiện ích như các công cụ lưu trữ dữ liệu nhờ các cơ sở dữ liệu thời gian thực, xác thực người dùng (Firebase Authentication), lưu trữ đám mây, và hosting. Firebase được sử dụng rộng rãi nhờ tính linh hoạt, và hỗ trợ truy suất cơ sở dữ liệu theo thời gian thực, từ đó tăng khả năng đồng bộ hóa dữ liệu, tính nhất quán dữ liệu. Tránh xảy ra các trường hợp xung đột dữ liệu mà không cần phải xử lý phức tạp ở backend.

### Bố cục báo cáo

#### Chương 1: Tổng quan đề tài

Mang đến một góc nhìn tổng quát về đề tài “Hệ thống quản lý bến xe”, khái quát các vấn đề, thông tin cần thiết của dự án. Giúp người đọc hiểu rõ hơn về dự án và cũng đồng thời nắm được các công nghệ và lý do tại sao lại chọn đề tài và các công nghệ hỗ trợ.

#### Chương 2: Cơ sở lý thuyết

Cung cấp thông tin về chuyên sâu về các công nghệ được sử dụng trong dự án, giải thích ưu, nhược điểm của các công nghệ và lý do tại sao lại lựa chọn các công nghệ đó để xây dựng và phát triển dự án.

#### Chương 3: Hệ thống quản lý bến xe

Giải thích lý do tại sao lại chọn đề tài để phát triển, nêu ra các vấn đề và giải pháp để xử lý các khó khăn đang tồn tại. Mô tả hệ thống bằng các mô hình, sơ đồ xử lý logic trong hệ thống và các sơ đồ tổng quan của hệ thống.

#### Chương 4: Kết luận và hướng phát triển

Đánh giá sơ lược về quá trình thực hiện đề tài, các khó khăn gặp phải, các vấn đề đã giải quyết và cả những vấn đề còn tồn tại. Nêu ra các dự định để nâng cấp, cải tiến hệ thống để phù hợp hơn trong giai đoạn phát triển tiếp theo.

# CƠ SỞ LÝ THUYẾT

## Spring Boot

### Tổng quan Spring Boot

#### Lợi ích

Thứ nhất, Spring Boot hỗ trợ phát triển các ứng dụng độc lập (Standalone application) mà không cần phải cấu hình các máy chủ bên ngoài dự án, có thể chạy trên mọi môi trường nhờ vào việc cấu hình các máy chủ Local như Tomcat, Jetty trực tiếp trong ứng dụng của mình.

Thứ hai, Spring Boot còn hỗ trợ mạnh mẽ khi cung cấp các thư viện có sẵn “starter” để dễ dàng tích hợp các chức năng vào ứng dụng. Các stater này chứa các cấu hình cần thiết cho từng chức năng cụ thể. Tiết kiệm được công sức khi không phải cấu hình thủ công như trước.

Cuối cùng, Spring Boot ra đời giúp tiết kiệm rất nhiều công sức và thơi gian cho các lập trình viên, giảm thiểu sự sai sót trong việc cấu hình, đặc biệt là các cấu hình phức tạp. Ngoài ra, còn hỗ trợ nhiều công cụ mạnh mẽ như DevTools, giúp rút gắn thời gian phát triển các phần mềm ứng dụng và nâng cao được tính tối ưu khi các lập trình viên có thể chú tâm và xử lý các logic trong ứng dụng một cách chính xác và đảm bảo chất lượng của các thuật toán.

#### Các tính năng

SpringApplication: Giúp đơn giản hóa khi khởi chạy các ứng dụng sử dụng ngôn ngữ Java thông qua việc rút gọn quy trình gọi và khởi chạy từ hàm “main”.

Externalized Configuration: Cho phép tách riêng các cấu hình ra thành từng file “.properties” hoặc các biến môi trường riêng biệt, từ đó giúp cho việc cập nhật và mở rộng ứng dụng trở nên dễ dàng và linh hoạt trong nhiều môi trường khác nhau.

Profiles: Cho phép người dùng linh hoạt cấu hình ứng dụng thay đổi cho phù hợp với nhiều loại môi trường phát triển của dự án, ví dụ như môi trường xây dựng, kiểm thử, và cài đặt sản phẩm.

Spring Boot Actuator: Hỗ trợ người lập trình với nhiều chức năng quản lý dự án mạnh mẽ với các thông số cần thiết để tối ưu sản phẩm như tình trạng ứng dụng (health check), số liệu (metrics), thông tin hệ thống (info) và rất nhiều chức năng hỗ trợ khác.

#### Ưu điểm vượt trội

Triển khai: Thuận tiện lớn nhất của Java là có thể đóng gói dưới dạng file JAR, đối với loại file này thì sẽ đạt được độ tương thích cao khi có thể chạy trên nhiều môi trường khác nhau mà không yêu cầu cấu hình phức tạp ở máy chủ.

Cấu hình: Cung cấp nhiều hình thức để cấu hình, giúp cho lập trình viên linh hoạt sử dụng theo nhu cầu mà không đụng chạm và thay đổi quá nhiều mã nguồn.

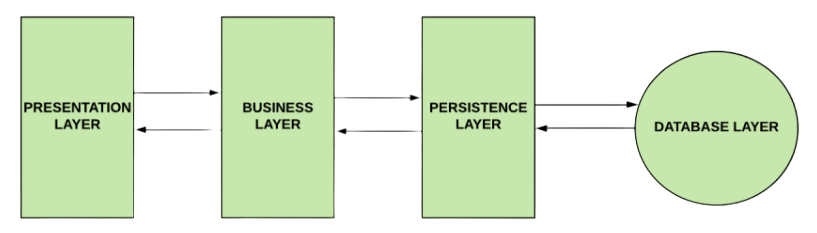
Mạnh mẽ và linh hoạt: Thư viện Spring Boot được tối ưu hóa dựa trên Spring Framework, phù hợp với việc phát triển các ứng dụng.

#### Kết luận

Spring Boot luôn là lựa chọn hàng đầu khi chọn Java là công nghệ để phát triển các ứng dụng hiện nay, phù hợp với nhiều môi trường và dễ triển khai. Có nhiều tính năng hỗ trợ như tự động cấu hình, đơn giản hơn trong việc phát triển xây dựng các ứng dụng độc lập, và giảm thiểu cấu hình thủ công, Spring Boot giúp lập trình viên tiết kiệm được thời gian và công sức bỏ ra nhưng kết quả đạt được lại cao hơn so với làm bằng công nghệ Spring Framework thuần túy.

### Kiến trúc Spring Boot

Spring Boot có kiến trúc theo kiểu phân tầng (layered architecture) với 4 lớp chính:



Hình 2 ‑ 2‑1: Kiến trúc các tầng chính trong Spring Boot

#### Presentation Layer (Lớp trình bày):

Ở đây sẽ nhận và xử lý yêu cầu được gửi đi từ người dùng (client), chứng thực (authentication), và chuyển đổi thành dữ liệu dạng JSON hoặc ngược lại.

Lớp Presentation ứng với lớp Controller, lớp này sẽ nhận và chịu trách nhiệm xử lý các yêu cầu (GET, PATCH, PUT, DELETE, POST) được gửi đến từ phía người dùng (Client).

#### Business Layer (Lớp nghiệp vụ):

Là nơi thực hiện các xử lý về logic, chứng thực (authorization) và kiểm tra dữ liệu đầu vào (validation).

Lớp này thường tương ứng với lớp Service trong Spring Boot, là nơi triển khai các nghiệp vụ của hệ thống.

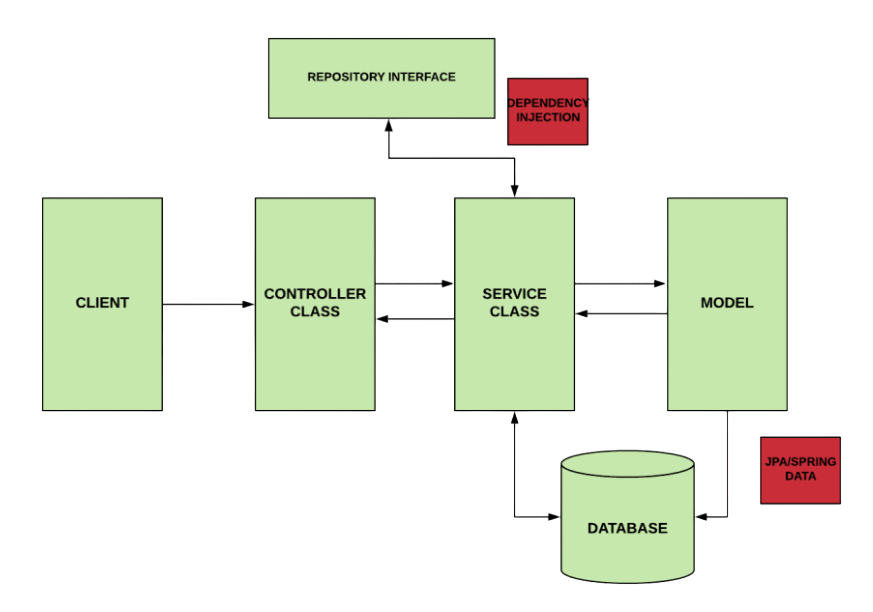
#### Persistence Layer (Lớp lưu trữ):

Là lớp trung gian giữa nghiệp vụ và cơ sở dữ liệu, chịu trách nhiệm trong làm việc với cơ sở dữ liệu. Sử dụng các lớp interface Repository để định nghĩa các nghiệp vụ và truy vấn xuống cơ sở dữ liệu, và trả về dữ liệu tương ứng.

#### Database Layer (Lớp cơ sở dữ liệu):

Đây là tầng gắn liền, trực tiếp tương tác và quản lý cơ sở dữ liệu thông qua các CRUD (Create, Delete, Update, Read).

### Kiến trúc luồng hoạt động



Hình 2 ‑ 2‑2: Sơ đồ luồng hoạt động trong Spring Boot

**- Client**: Là nơi gửi các yêu cầu (Request) đến máy chủ thông qua các CRUB (Delete, Post, Put, Patch, Get) từ giao diện người dùng hoặc từ các hệ thống khác bên ngoài.

**- Controller Class**: Lớp đầu tiên nhận yêu cầu từ phía **Client**, nhận và xử lý các dữ liệu được gửi đến để điều hướng và cung cấp cho lớp nghiệp vụ (Service). Controller cũng được hiểu như là cầu nối giữa client và các lớp nghiệp vụ xử lý phía sau.

**- Service Class**: Là lớp xử lý các yêu cầu nhận được từ lớp Controller, xử lý các logic, và làm việc với lớp Repository để tương tác với cơ sở dữ liệu.

**- Model (Pojo)**: Lớp này đại diện cho các đối tượng dữ liệu trong dự án, bao gồm cả các lớp thực thể (Entity) và được ánh xạ với cơ sở dữ liệu.

**- Repository Interface**: Là lớp trực tiếp tương tác với cơ sở dữ liệu thông qua **JPA/Spring Data**.

**- Database**: Cơ sở dữ liệu chứa các thông tin liên quan đến ứng dụng, và được quản lý thông qua **JPA/Spring Data**.

### Ví dụ

Chức năng đăng kí người dùng:

**- Client**: Người dùng nhập thông tin đăng ký trên form (tên, email, mật khẩu) và gửi yêu cầu đăng ký.

**- Controller Class**: Lớp UserController nhận yêu cầu đăng ký từ form qua phương thức POST. Ví dụ: @PostMapping("/register").

**- Service Class**: Lớp UserService kiểm tra thông tin đăng ký có hợp lệ không (như kiểm tra email đã tồn tại,…). Nếu hợp lệ, lớp này tiếp tục gửi yêu cầu đến lớp Repository để lưu thông tin người dùng vào cơ sở dữ liệu.

**- Model (Pojo)**: Lớp User là lớp đại diện cho bảng người dùng trong cơ sở dữ liệu, chứa các thuộc tính như name, email, password.

**- Repository Interface**: Lớp UserRepository lưu thông tin người dùng mới vào cơ sở dữ liệu bằng phương thức save().

**- Database**: Dữ liệu của người dùng được lưu vào bảng users trong cơ sở dữ liệu.

**- Kết quả**: Sau khi đăng ký thành công, lớp Controller trả về thông báo "Đăng ký thành công" cho người dùng.

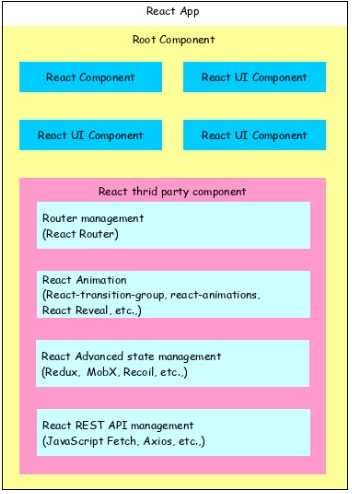
## ReactJS

### Tổng quan ReactJS

[1] React là một thư viện JavaScript declarative, hiệu quả và linh hoạt cho việc xây dựng giao diện người dùng. React cho phép bạn tạo những giao diện (UI) phức tạp từ những đoạn code nhỏ và độc lập. Những đoạn code này được gọi là “Component”. ReactJS hoạt động theo cơ chế cây DOM ảo (**Virtual DOM).** Nhờ vào cơ chế này, việc cập nhật lại giao diện được tối ưu hóa khi không tác động đến cây DOM thật, chỉ cập nhật các component thay đổi mà không cần phải cập nhật lại tất cả giao diện. Từ đó trải nghiệm của người dùng được nâng cao và hiệu suất của hệ thống được cải thiện đáng kể. React sử dụng cú pháp **JSX**, là sự kết hợp giữa JavaScript với HTML giúp cho việc phát triển trở nên dễ dàng và nhanh gọn hơn.

### Kiến trúc ReactJS

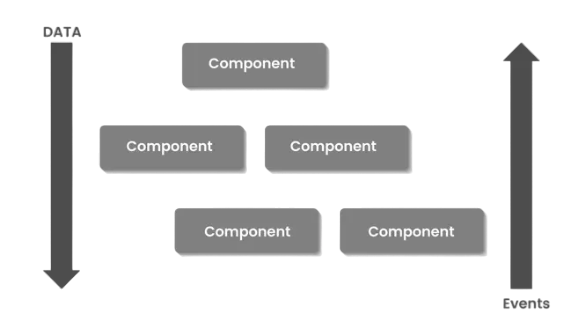
Kiến trúc của ReactJS được xây dựng theo kiểu component-based architecture. Có nghĩa là sẽ chia ra thành các component độc lập để cấu thành nên một khối. Có thể tận dụng và tái sử dụng các component, giúp cho việc quản lý giao diện trở nên dễ dàng và không còn phức tạp như khi làm thành một khối không thể tách rời.



Hình 2 ‑ 2‑3: Kiến trúc của ReactJS

#### Components

Là đơn vị chính của React, có thể là các thành phần nhỏ nhất hoặc lớn nhất của giao diện. Mỗi component có thể chứa các component khác và có thể có logic riêng hoặc chỉ để hiển thị dữ liệu. Mỗi component có thể sẽ là một đơn vị độc lập hoặc gồm nhiều component khác cấu thành.



Hình 2 ‑ 2‑4: Tổ chức của component

#### JSX (JavaScript XML)

React sử dụng JSX, là một cú pháp cho phép nhúng HTML vào file JavaScript, giúp vừa xử lý logic và cả giao diện cùng một lúc. Mang đến một cái nhìn trực quan và tiện lợi cho người lập trình viên.

#### Props (Properties)

Props là cơ chế truyền dữ liệu từ component cha sang component con trong React. Cơ chế này giúp các component con có thể nhận được các dữ liệu cần thiết để sử dụng nếu cần thiết.

#### Context

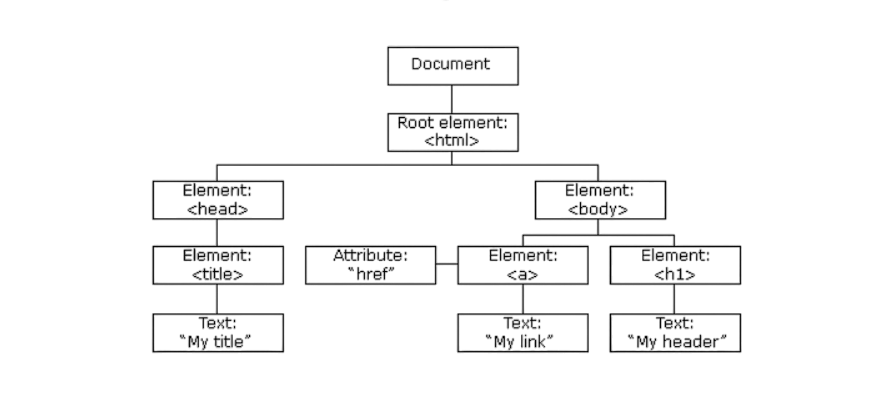
Context là một cơ chế truyền dữ liệu trong react, cho phép cung cấp dữ liệu trực tiếp từ component cha đến các component con mà không cần thông qua props ở mỗi cấp. Context gồm Provider cung cấp dữ liệu và Consumer hoặc useContext để các component con truy cập và sử dụng dữ liệu đó, giúp mã nguồn gọn gàng và dễ quản lý hơn. Phù hợp với những dữ liệu mà nhiều component cần.

#### State

Mỗi component trong React sẽ quản lý state của riêng mình. Giao diện sẽ được cập nhập lại khi state thay đổi. Dựa vào đặc tính đó, lập trình viên có thể xử lý các logic theo nhu cầu của mình một cách dễ dàng và đơn giản.

#### Virtual DOM

[2] **DOM** là tên gọi tắt của **Document Object Model** (Mô hình Đối tượng Tài liệu). Virtual DOM để chủ động quản lý giao diện, cập nhật giao diện khi cần thiết. Khi state hoặc props thay đổi giá trị, React sẽ tự động tạo một cây DOM ảo và so sánh với cây DOM thật, từ đó mà có thể nhận biết được chính xác những component nào thay đổi để cập nhật lại đúng component đó, điều này sẽ làm tăng hiệu năng khi ứng dụng không phải tải lại toàn bộ giao diện.



Hình 2 ‑ 2‑5: Kiến trúc của DOM

#### Lifecycle Methods

React chủ động quản lý các component (Lifecycle Methods) từ khi nó được tạo ra, cập nhật hoặc bị xóa khỏi giao diện. Các phương thức này giúp lập trình viên dễ dàng theo dõi và tương tác với các component trong từng giai đoạn.

## Cơ sở dũ liệu MySQL

MySQL là một hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở, phổ biến trong các dự án phát triển web và các ứng dụng di động. Nó sử dụng ngôn ngữ SQL quen thuộc để quản lý, tương tác với cơ sở dữ liệu. MySQL là một hệ quản trị dễ sử dụng, linh hoạt, hỗ trợ nhiều engine lưu trữ như InnoDB và MyISAM. Ngoài ra, MySQL còn có khả năng bảo mật mạnh mẽ. Do đó, MySQL rất được ưu chuộng và ứng dụng rộng rãi trong nhiều dự án ở nhiều lĩnh vực khác nhau.

## Các công cụ hỗ trợ

### Firebase

Firebase được cung cấp bởi Google nhằm hỗ trợ, phục vụ cho việc phát triển ứng dụng trên di động hoặc nền tảng web. Nó cung cấp đa dạng các dịch vụ như cơ sở dữ liệu thời gian thực (Realtime Database), lưu trữ đám mây (Cloud Firestore), xác thực người dùng (Authentication), lưu trữ file, gửi thông báo đẩy (Push Notifications) và các công cụ phân tích ứng dụng. Firebase giúp lập trình viên phát triển và triển khai ứng dụng nhanh chóng, không cần phải quản lý hạ tầng server, dễ dàng mở rộng khi cần thiết.

### Cloudinary

Cloudinary cung cấp dịch vụ cho phép lưu trữ các file phương tiện trên nền tảng đám mây, giúp lưu trữ an toàn và quản lý hình ảnh, video và cung cấp đường dẫn dạng CDN (Content Delivery Network) để phục vụ cho việc hiển thị tài nguyên cho các ứng dụng web và di động. Từ đó, dự án sẽ được tối ưu cách quản lý các file phương tiện, tăng hiệu suất.

# HỆ THỐNG QUẢN LÝ BẾN XE KHÁCH

## Giới thiệu đề tài

Trong thời buổi hiện nay, với sự phát triển của nền kinh tế và xã hội, nhu cầu đi lại của người dân ngày càng tăng cao để phục vụ cho nhiều mục đích khác nhau trong cuộc sống, từ công việc đến du lịch, học tập, khám chữa bệnh ở các bệnh viện lớn. Tuy nhiên, sự gia tăng đột biến về nhu cầu của hành khách, đặc biệt là tại các thành phố lớn, đã khiến cho các bến xe bắt đầu rơi vào tình trạng quá tải. Điều này không chỉ gây ra ùn tắc giao thông vào các giờ cao điểm mà còn làm cho các hệ thống quản lý hiện tại trở nên lỗi thời, kém hiệu quả và dễ dẫn đến sai sót trong việc quản lý và vận hành.

Tại các thành phố lớn với dân cư đông đúc và lưu lượng hành khách từ các nơi khác đồ về nhưng cơ sở hạ tầng không được nâng cấp, bảo trì, xuống cấp trầm trọng, tình trạng quá tải không chỉ xảy ra thường xuyên mà còn đặc biệt nghiêm trọng vào các ngày cao điểm như các lễ, Tết, giờ tan làm. Điều này khiến hành khách gặp nhiều khó khăn trong việc mua vé, thanh toán, và thậm chí phải đối mặt với tình trạng không mua được vé, không thể lên xe đúng giờ, gây ảnh hưởng nghiêm trọng đến nhu cầu đi lại của họ. Thêm vào đó, phương thức quản lý truyền thống tại các bến xe thiếu đi tính hiện đại và không còn đáp ứng được nhu cầu quản lý chính xác và minh bạch, dẫn đến sự bất tiện không đáng có.

Nhận thấy những bất cập và nhu cầu cấp thiết đó, em đã xây dựng hệ thống quản lý bến xe với mục tiêu giúp hành khách dễ dàng theo dõi thông tin về các chuyến xe, thời gian xuất phát, điểm đến, cũng như mua vé và thanh toán trực tuyến một cách tiện lợi, nhanh chóng. Hệ thống không chỉ giúp giảm thiểu tình trạng quá tải mà còn mang lại cho hành khách trải nghiệm tiện ích hơn, thông qua việc đơn giản hóa các quy trình mua vé, không phải xếp hàng và thanh toán trực tuyến dễ dàng.

Ngoài ra, hệ thống còn tạo ra sự minh bạch và rõ ràng trong việc quản lý thông tin vé, lịch trình, và quy trình thanh toán, giúp hành khách an tâm hơn khi sử dụng dịch vụ. Đồng thời, việc áp dụng công nghệ vào quản lý bến xe sẽ hỗ trợ các nhà quản lý điều hành hệ thống hiệu quả hơn, giảm thiểu sai sót, tối ưu hóa quá trình vận hành, và nâng cao chất lượng phục vụ. Đảm bảo được quyền lợi của khách hàng.

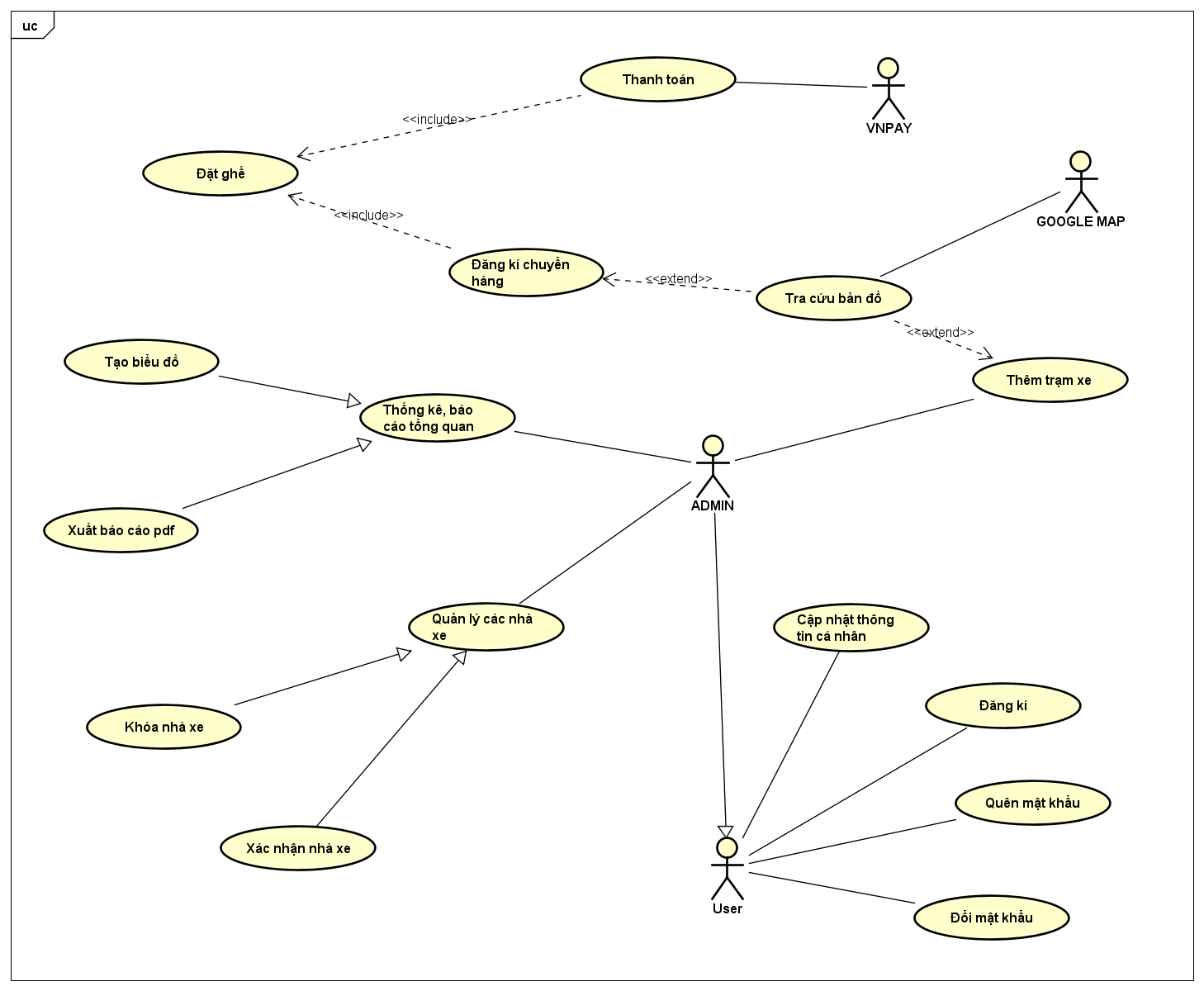
## Phân tích hệ thống

Qua thời gian phân tích và đánh giá thì em đã thiết kế được lược đồ use case, thể hiện đầy đủ các chức năng, yêu cầu cần thiết của hệ thống, được trình bày theo vai trò của từng người dùng trong hệ thống:

### Lược đồ Use Case

#### Admin (Quản trị viên bến xe):

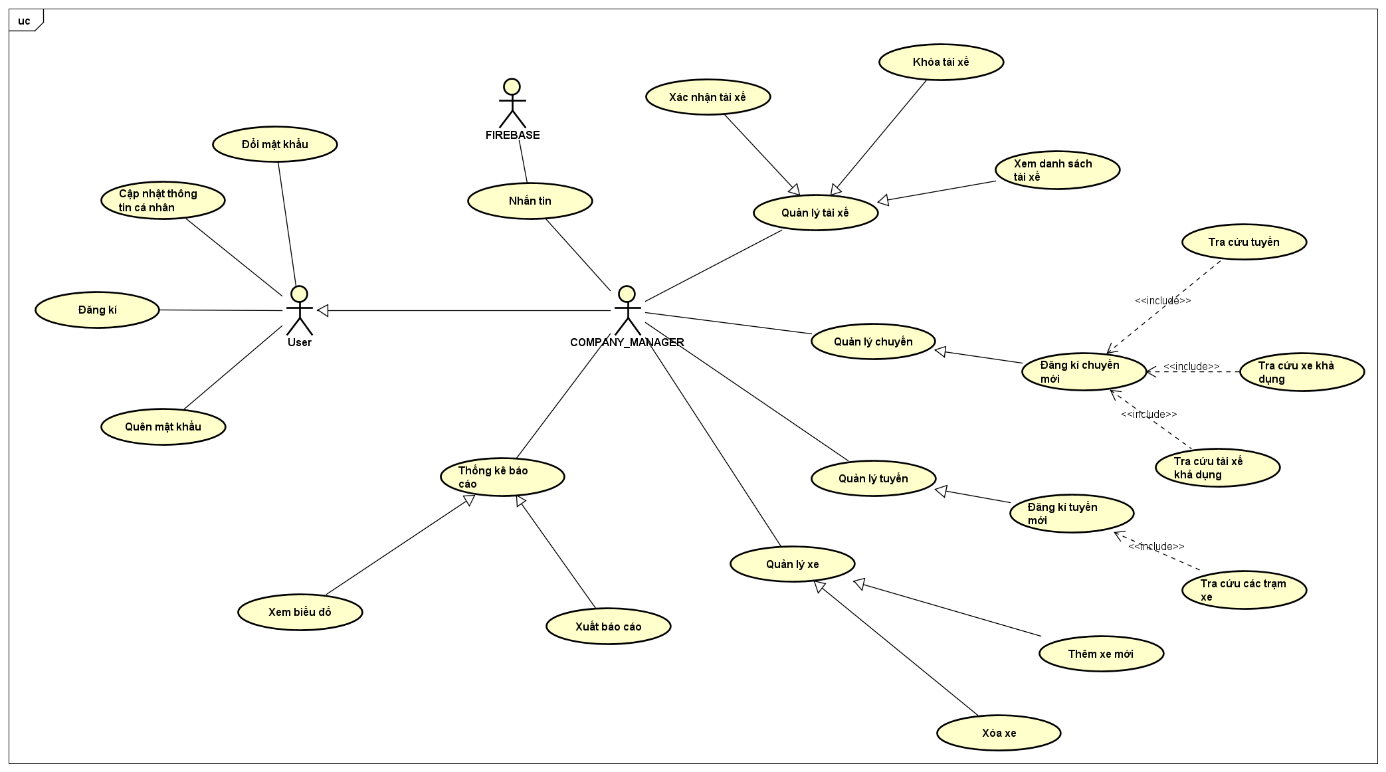
* Không thể tự đăng kí tài khoản mà phải được cấp.
* Quản lý các công ty vận tải, có quyền cấp phép hoạt động hoặc vô hiệu hóa hoạt động.
* Quản lý các trạm xe, có thể thêm mới trạm xe.
* Có thể xe báo cáo tổng quan các số liệu của bến xe, có thể xuất báo cáo thành file pdf.



Hình 3 ‑ 3‑1: Lược đồ use case của vai trò Admin

#### Company\_Manager (Quản trị viên công ty):

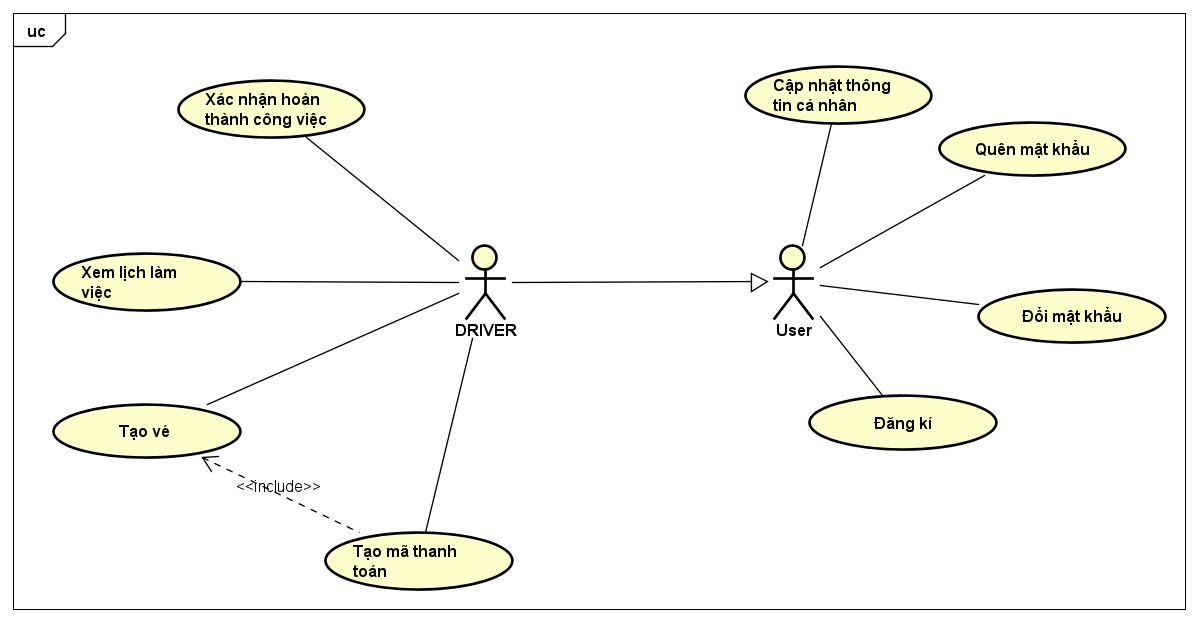
* Được quyền tạo các tuyến xe dựa trên các trạm xe có sẵn của bến.
* Tạo các chuyến xe dựa vào các tuyến xe đã đăng kí với bến xe.
* Tạo các xe mới để đưa vào phục vụ hoặc loại biên xe.
* Xác nhận hợp tác với các tài xế đăng kí, hoặc hủy hợp tác.
* Đăng kí khai thác dịch vụ vận chuyển hàng hóa.
* Xem các biểu đồ báo cáo doanh thu theo từng thời điểm bất kì, có thể xuất báo cáo dưới dạng file pdf.



Hình 3 ‑ 3‑2: Lược đồ use case của vai trò Manager\_Company

#### Driver (Tài xế):

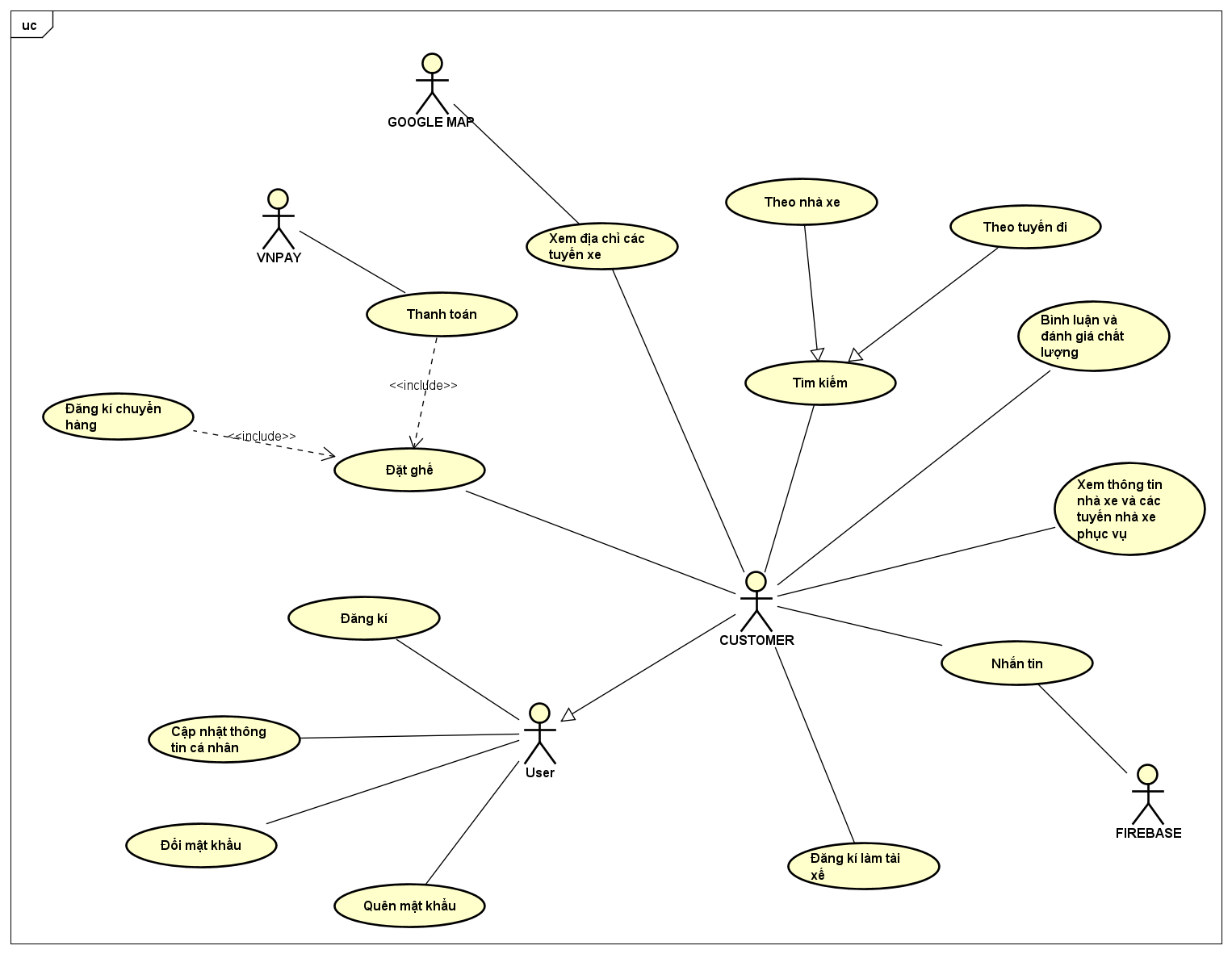
* Có thể xem được lịch trình làm việc.
* Có thể nắm được thông tin công việc, tuyến chạy, giờ xuất bến, xe cụ thể.
* Có thể tạo vé xe cho khách hàng trực tiếp khi đang thực hiện chuyến xe.
* Xử lý thông tin và cung cấp mã QR Code để hành khách có thể thanh toán qua cổng thanh toán VNPAY.



Hình 3 ‑ 3‑3: Lược đồ use case của vai trò Driver

#### Customer (Khách hàng):

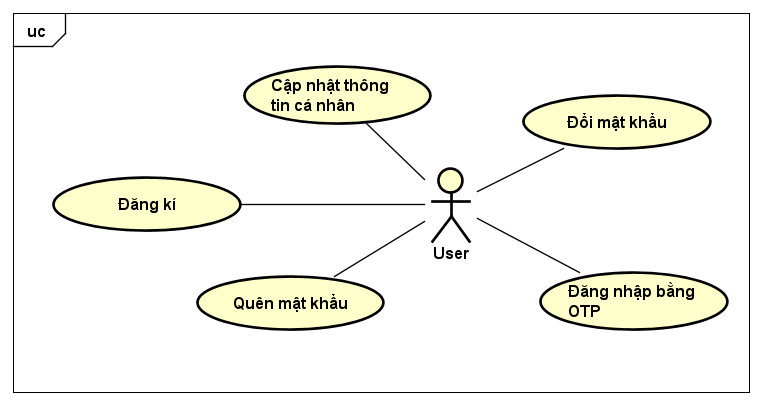
* Đăng kí tài khoản trực tiếp.
* Có thể tìm kiếm vé dễ dàng bằng các chức năng tra cứu theo tuyến xe phù hợp hoặc theo tên công ty vận tải.
* Có thể đặt ghế, đăng kí thông tin giao hàng, và thanh toán online thông qua VNPAY.
* Có thể đăng kí vận hành công ty vận tải.
* Có thể đăng kí hợp tác làm tài xế với các công ty.
* Có thể xuất vé khi đã thanh toán xong, hoặc hủy vé nếu chưa thanh toán.
* Có thể bình luận, đánh giá các công ty.
* Có thể nhắn tin và nhận tư vấn hỗ trợ trực tuyến với các nhân viên công ty bằng tính năng chat thời gian thực.



Hình 3 ‑ 3‑4: Lược đồ use case của vai trò Customer

#### User (Người dùng):

* Đăng nhập, đăng xuất.
* Yêu cầu cấp lại mật khẩu khi quên.
* Thay đổi, cập nhật thông tin cá nhân.
* Thay đổi mật khẩu.
* Đăng nhập sử dụng mã OTP.



Hình 3 ‑ 5: Lược đồ use case của User

### Đặc tả Use Case

#### Use Case đặt và thanh toán vé

|  |  |
| --- | --- |
| Use Case Id | UC - 01 |
| Tên Use Case | Use Case đặt và thanh toán vé |
| Mô tả | Cho phép khách hàng tìm kiếm, đặt vé và thanh toán online |
| Actor chính | Customer |
| Actor phụ | VNPAY |
| Tiền điều kiện | Đăng nhập, chọn ghế, đăng kí thông tin giao hàng nếu có |
| Hậu điều kiện | Thông tin vé được tạo và lưu vào cơ sở dữ liệu chờ thanh toán, khi thanh toán xong thì sẽ cập nhật thêm các thông tin cần thiết |
| Luồng hoạt động | 1. Khách hàng đăng nhập vào hệ thống. 2. Khách hàng tìm kiếm thông tin vé theo tên công ty hoặc tuyến xe phù hợp 3. Chọn chuyến xe phù hợp 4. Chọn ghế khả dụng 5. Nhập thông tin giao hàng nếu có nhu cầu giao hàng 6. Bấm đặt vé để xác nhận tạo vé 7. Vào giỏ hàng để kiểm tra và bấm thanh toán 8. Sau khi xác nhận xong thì bấm đặt vé 9. Hệ thống sẽ chuyển đến trang thanh toán của VNPAY 10. Khách hàng sẽ nhập thông tin tài khoản để hoàn tất thanh toán 11. Vào mục vé đã mua để xem và xuất file pdf nếu có nhu cầu 12. Kết thúc. |
| Luồng thay thế | * Ở bước thanh toán, nếu thanh toán không thành công thì sẽ phải vào mục vé đã mua để tiến hành hủy vé. * Nếu có nhu cầu vận chuyển hàng thì phải thêm từng vé một chứ không thể thêm một lúc nhiều vé |
| Luồng ngoại lệ | Khi chưa đăng nhập mà vẫn bấm thêm vé thì hệ thống sẽ cảnh báo |

Bảng 3‑1: Use Case đặt và thanh toán vé

#### Use Case đăng kí vận hành công ty vận tải

|  |  |
| --- | --- |
| Use Case Id | UC - 02 |
| Tên Use Case | Use Case đăng kí vận hành công ty vận tải |
| Mô tả | Cho phép người dùng có thể đăng ký mở công ty vận tải mới |
| Actor chính | Customer |
| Actor phụ | Cloudinary |
| Tiền điều kiện | Đăng nhập với vai trò customer |
| Hậu điều kiện | Hệ thống sẽ tạo và lưu thông tin công ty vào cơ sở dữ liệu với trạng thái chưa xác nhận, và đợi quản trị viên xác nhận |
| Luồng hoạt động | 1. Đăng nhập vào hệ thống với vai trò là khách hàng 2. Bấm vào chữ tham gia trên thanh navbar 3. Nhập đầy đủ thông tin của công ty theo yêu cầu 4. Bấm xác nhận 5. Đợi mail phản hồi khi quản trị viên xác thực và cấp phép hoạt động |
| Luồng thay thế | Không có |
| Luồng ngoại lệ | Không có |

Bảng 3‑2: Use Case đăng kí vận hành công ty vận tải

#### Use Case tạo tuyến mới

|  |  |
| --- | --- |
| Use Case Id | UC - 03 |
| Tên Use Case | Use Case tạo tuyến mới |
| Mô tả | Cho phép quản trị viên đăng kí khai thác tuyến mới |
| Actor chính | Company\_Manager |
| Actor phụ | Không có |
| Tiền điều kiện | Đăng nhập với vai trò là Company\_Manager |
| Hậu điều kiện | Cơ sở dữ liệu sẽ tạo và lưu tuyến xe có thông tin của công ty đăng kí |
| Luồng hoạt động | 1. Đăng nhập với vai trò Company\_Manager 2. Chọn chức năng đăng kí tuyến 3. Chọn thông tin trạm đi, trạm đến trong danh sách 4. Bấm xác nhận 5. Kết thúc |
| Luồng thay thế | Không có |
| Luồng ngoại lệ | Không có |

Bảng 3‑3: Use Case tạo tuyến mới

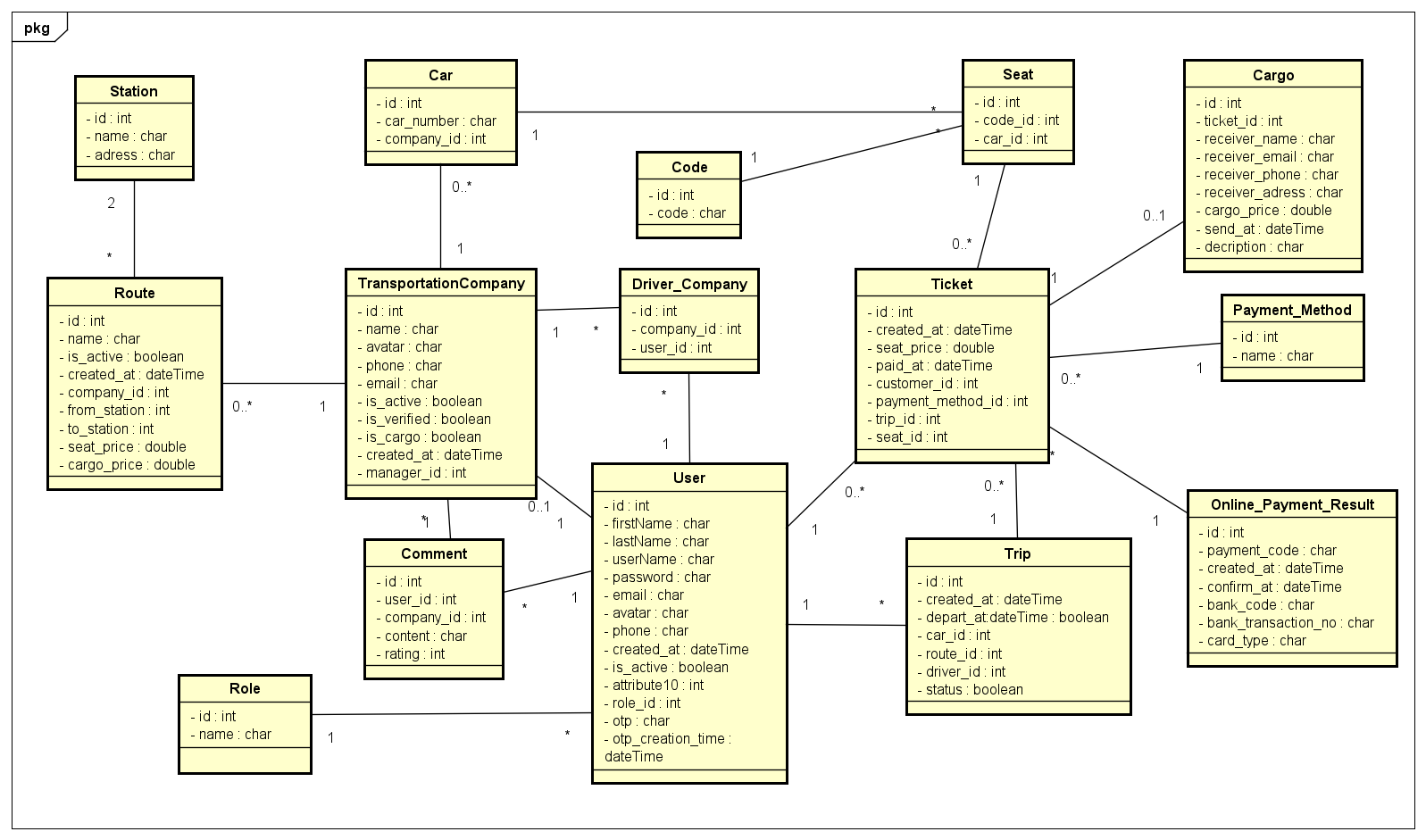
#### Use Case tạo chuyến mới

|  |  |
| --- | --- |
| Use Case Id | UC - 04 |
| Tên Use Case | Use Case tạo chuyến mới |
| Mô tả | Cho phép quản trị viên công ty đăng kí chuyến xe mới để phục vụ hành khách |
| Actor chính | Company\_Manager |
| Actor phụ | Không có |
| Tiền điều kiện | Đăng nhập với vai trò là Company\_Manager |
| Hậu điều kiện | Cơ sở dữ liệu sẽ tạo và lưu chuyến xe có thông tin của công ty và tuyến xe đã đăng kí |
| Luồng hoạt động | 1. Đăng nhập với vai trò Company\_Manager 2. Chọn chức năng tạo chuyến mới 3. Chọn tuyến xe cần đăng ký chuyến 4. Chọn ngày và giờ khởi hành 5. Chọn tài xế khả dụng 6. Chọn xe khả dụng 7. Bấm xác nhận |
| Luồng thay thế | Không có |
| Luồng ngoại lệ | Nếu như trong danh sách các tài xế khả dụng và các xe khả dụng đều trống thì sẽ không thể tạo được chuyến mới |

Bảng 3‑4: Use Case tạo chuyến mới

## Thiết kế hệ thống

### Sơ đồ lớp



Hình 3 ‑ 3‑6: Sơ đồ lớp

#### Thông tin thiết kế các lớp

1. **Station (Trạm)**

**id**: Mã định danh của trạm.

**name**: Tên trạm.

**address**: Địa chỉ của trạm.

Một trạm có thể là trạm khởi hành hoặc trạm đến cho nhiều tuyến đường (Route).

1. **Route (Tuyến đường)**

**id**: Mã định danh của tuyến đường.

**name**: Tên tuyến đường.

**is\_active**: Trạng thái hoạt động của tuyến đường.

**created\_at**: Thời gian tạo tuyến đường.

**company\_id**: Mã định danh của công ty vận tải.

**from\_station** và **to\_station**: Trạm khởi hành và trạm đến, liên kết với bảng **Station**.

**seat\_price** và **cargo\_price**: Giá vé và giá vận chuyển hàng hóa trên tuyến.

1. **Car (Xe)**

**id**: Mã định danh của xe.

**car\_number**: Biển số xe.

**company\_id**: Mã định danh của công ty sở hữu xe.

1. **Seat (Ghế)**

**id**: Mã định danh của ghế.

**code\_id**: Mã ghế (liên kết với bảng **Code**).

**car\_id**: Mã định danh của xe chứa ghế.

1. **Code (Mã)**

**id**: Mã định danh của code ghế.

**code**: Giá trị mã ghế (dùng để nhận diện ghế).

1. **Ticket (Vé)**

**id**: Mã định danh của vé.

**created\_at**: Thời gian tạo vé.

**seat\_price**: Giá vé.

**paid\_at**: Thời gian thanh toán.

**customer\_id**: Mã định danh khách hàng (liên kết với bảng **User**).

**payment\_method\_id**: Phương thức thanh toán.

**trip\_id**: Mã chuyến đi (liên kết với bảng **Trip**).

**seat\_id**: Mã ghế (liên kết với bảng **Seat**).

1. **Cargo (Hàng hóa)**

**id**: Mã định danh của hàng hóa.

**ticket\_id**: Mã vé liên quan.

**receiver\_name**, **receiver\_email**, **receiver\_phone**, **receiver\_address**: Thông tin người nhận hàng.

**cargo\_price**: Giá vận chuyển hàng hóa.

**send\_at**: Thời gian gửi hàng.

**description**: Mô tả hàng hóa.

1. **Payment\_Method (Phương thức thanh toán)**

**id**: Mã định danh phương thức thanh toán.

**name**: Tên phương thức thanh toán.

1. **Online\_Payment\_Result (Kết quả thanh toán trực tuyến)**

**id**: Mã định danh kết quả thanh toán.

**payment\_code**: Mã giao dịch thanh toán.

**created\_at**: Thời gian tạo giao dịch.

**confirm\_at**: Thời gian xác nhận thanh toán.

**bank\_code**, **bank\_transaction\_no**: Thông tin mã ngân hàng và số giao dịch.

**card\_type**: Loại thẻ thanh toán.

1. **Trip (Chuyến đi)**

**id**: Mã định danh chuyến đi.

**created\_at**: Thời gian tạo chuyến.

**depart\_at**: Thời gian khởi hành.

**car\_id**: Mã xe dùng trong chuyến đi.

**route\_id**: Mã tuyến đường (liên kết với bảng **Route**).

**driver\_id**: Mã tài xế (liên kết với bảng **Driver\_Company**).

**status**: Trạng thái chuyến đi (hoạt động hoặc không hoạt động).

1. **Driver\_Company (Tài xế của công ty)**

**id**: Mã định danh tài xế.

**company\_id**: Mã định danh của công ty tài xế làm việc.

**user\_id**: Mã định danh người dùng (liên kết với bảng **User**).

1. **TransportationCompany (Công ty vận tải)**

**id**: Mã định danh công ty.

**name**: Tên công ty.

**avatar**: Hình đại diện của công ty.

**phone**, **email**: Thông tin liên hệ.

**is\_active**, **is\_verified**, **is\_cargo**: Các trạng thái của công ty (hoạt động, xác minh, vận chuyển hàng hóa).

**created\_at**: Thời gian tạo công ty.

**manager\_id**: Mã quản lý công ty.

1. **User (Người dùng)**

**id**: Mã định danh người dùng.

**firstName**, **lastName**, **userName**: Thông tin tên và tên người dùng.

**OTP:** Chưa mã otp dùng để xác thực thay cho mật khẩu.

**OTP\_Creation\_Time**: Lưu lại thời gian tạo để xác thực tính khả dụng của OTP.

**password**, **email**, **phone**: Thông tin bảo mật và liên hệ.

**is\_active**: Trạng thái hoạt động của người dùng.

**role\_id**: Mã vai trò người dùng (liên kết với bảng **Role**).

1. **Role (Vai trò)**

**id**: Mã định danh vai trò.

**name**: Tên vai trò của người dùng.

1. **Comment (Bình luận)**

**id**: Mã định danh của bình luận.

**user\_id**: Mã người dùng đã tạo bình luận.

**company\_id**: Mã công ty mà bình luận nhắm đến.

**content**: Nội dung bình luận.

**rating**: Đánh giá của người dùng.

#### Giải thích các mối quan hệ

**1. Station (Trạm) - Route (Tuyến đường):**

**Quan hệ**: Một trạm có thể xuất hiện trong nhiều tuyến đường, vì vậy có quan hệ **một-nhiều** từ **Station** sang **Route** (từ from\_station và to\_station).

**Tác dụng**: Mỗi tuyến đường sẽ có một **trạm khởi hành** và **trạm đến**. Các trạm này liên kết với bảng **Station** để xác định chính xác tuyến xe chạy.

**2. Route (Tuyến đường) - TransportationCompany (Công ty vận tải):**

**Quan hệ**: Một công ty vận tải có thể quản lý nhiều tuyến đường, nên có quan hệ **một-nhiều** từ **TransportationCompany** sang **Route**.

**Tác dụng**: Công ty vận tải quản lý các tuyến đường mà họ cung cấp dịch vụ. Mỗi công ty vận tải có thể đăng kí khai thác nhiều tuyến đường.

**3. Car (Xe) - TransportationCompany (Công ty vận tải):**

**Quan hệ**: Một công ty vận tải sở hữu nhiều xe, nên có quan hệ **một-nhiều** từ **TransportationCompany** sang **Car**.

**Tác dụng**: Công ty vận tải quản lý các xe của mình, và mỗi xe đều thuộc sở hữu của một công ty cụ thể.

**4. Seat (Ghế) - Car (Xe):**

**Quan hệ**: Một xe có nhiều ghế, nên có quan hệ **một-nhiều** từ **Car** sang **Seat**.

**Tác dụng**: Mỗi ghế đều thuộc về một chiếc xe cụ thể, phục vụ cho mục đích đặt vé của hành khách.

**5. Seat (Ghế) - Code (Mã ghế):**

**Quan hệ**: Một mã ghế có thể áp dụng cho nhiều ghế, nên có quan hệ **một-nhiều** từ **Code** sang **Seat**.

**Tác dụng**: Mã ghế giúp phân biệt các ghế trên xe và đảm bảo xác định chính xác từng ghế.

**6. Ticket (Vé) - Seat (Ghế):**

**Quan hệ**: Một vé chỉ áp dụng cho một ghế, nhưng một ghế có thể được liên kết với nhiều vé qua các chuyến đi khác nhau, tạo ra quan hệ **một-nhiều** từ **Seat** sang **Ticket**.

**Tác dụng**: Vé được sử dụng để đặt ghế cụ thể trên các chuyến đi khác nhau. Thông tin về ghế được quản lý thông qua mã ghế và xe liên quan.

**7. Ticket (Vé) - Trip (Chuyến đi):**

**Quan hệ**: Một vé chỉ áp dụng cho một chuyến đi, nhưng một chuyến đi có thể có nhiều vé được đặt, do đó có quan hệ **một-nhiều** từ **Trip** sang **Ticket**.

**Tác dụng**: Vé cho phép hành khách tham gia một chuyến đi cụ thể. Thông tin vé gắn với chuyến đi, xác định ngày, giờ và chỗ ngồi của hành khách.

**8. Cargo (Hàng hóa) - Ticket (Vé):**

**Quan hệ**: Một vé có thể liên quan đến việc vận chuyển hàng hóa, do đó có quan hệ **một-một** từ **Ticket** sang **Cargo**.

**Tác dụng**: Vé không chỉ phục vụ hành khách mà còn quản lý việc vận chuyển hàng hóa cho khách hàng.

**9. Trip (Chuyến đi) - Route (Tuyến đường):**

**Quan hệ**: Một tuyến đường có nhiều chuyến đi, tạo ra quan hệ **một-nhiều** từ **Route** sang **Trip**.

**Tác dụng**: Mỗi chuyến đi chạy trên một tuyến đường cụ thể, và nhiều chuyến đi có thể thực hiện trên cùng một tuyến.

**10. Trip (Chuyến đi) - Car (Xe):**

**Quan hệ**: Một chuyến đi sử dụng một xe, tạo ra quan hệ **một-nhiều** từ **Car** sang **Trip**.

**Tác dụng**: Xe được sử dụng cho chuyến đi cụ thể, và nhiều chuyến đi có thể sử dụng cùng một xe tại các thời điểm khác nhau.

**11. Driver\_Company (Tài xế của công ty) - Trip (Chuyến đi):**

**Quan hệ**: Một tài xế có thể lái nhiều chuyến đi, do đó có quan hệ **một-nhiều** từ **Driver\_Company** sang **Trip**.

**Tác dụng**: Tài xế được phân công để lái xe cho các chuyến đi cụ thể, và mỗi chuyến đi chỉ có một tài xế.

**12. User (Người dùng) - Ticket (Vé):**

**Quan hệ**: Một người dùng có thể mua nhiều vé, tạo ra quan hệ **một-nhiều** từ **User** sang **Ticket**.

**Tác dụng**: Người dùng có thể đặt nhiều vé cho các chuyến đi khác nhau. Thông tin người dùng được liên kết với vé để quản lý việc đặt chỗ.

**13. User (Người dùng) - Comment (Bình luận):**

**Quan hệ**: Một người dùng có thể để lại nhiều bình luận, do đó có quan hệ **một-nhiều** từ **User** sang **Comment**.

**Tác dụng**: Người dùng có thể bình luận về dịch vụ của công ty vận tải mà họ đã sử dụng.

**14. Comment (Bình luận) - TransportationCompany (Công ty vận tải):**

**Quan hệ**: Một công ty có thể nhận được nhiều bình luận từ người dùng, tạo ra quan hệ **một-nhiều** từ **TransportationCompany** sang **Comment**.

**Tác dụng**: Công ty vận tải có thể nhận được phản hồi từ khách hàng thông qua các bình luận, giúp họ cải thiện dịch vụ.

**15. User (Người dùng) - Role (Vai trò):**

**Quan hệ**: Một người dùng có một vai trò, và một vai trò có thể được gán cho nhiều người dùng, tạo ra quan hệ **một-nhiều** từ **Role** sang **User**.

**Tác dụng**: Vai trò của người dùng xác định quyền hạn và chức năng của họ trong hệ thống.

**16. Payment\_Method (Phương thức thanh toán) - Ticket (Vé):**

**Quan hệ**: Một vé có thể thanh toán qua một phương thức thanh toán, và mỗi phương thức thanh toán có thể được sử dụng cho nhiều vé, tạo ra quan hệ **một-nhiều** từ **Payment\_Method** sang **Ticket**.

**Tác dụng**: Phương thức thanh toán của vé giúp xác định cách khách hàng thanh toán cho chuyến đi của họ.

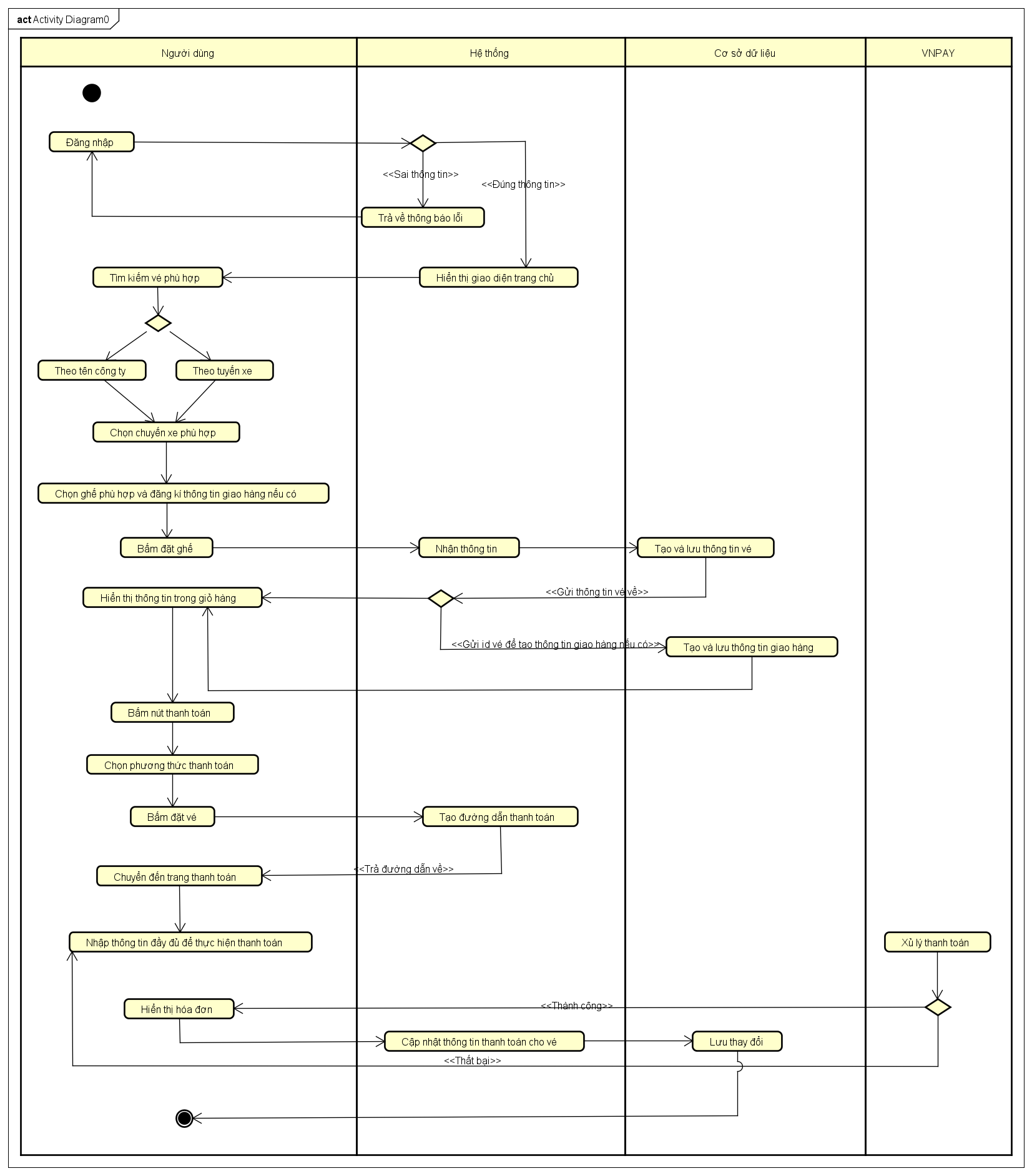
**17. Online\_Payment\_Result (Kết quả thanh toán trực tuyến) - Payment\_Method (Phương thức thanh toán):**

**Quan hệ**: Một phương thức thanh toán có thể liên quan đến nhiều giao dịch thanh toán trực tuyến, tạo ra quan hệ **một-nhiều** từ **Payment\_Method** sang **Online\_Payment\_Result**.

**Tác dụng**: Kết quả thanh toán trực tuyến lưu trữ thông tin chi tiết về các giao dịch trực tuyến thành công hoặc thất bại.

### Sơ đồ hoạt động

#### UC-01



Hình 3 ‑ 3‑7: Sơ đồ hoạt động của UC-01

Mô tả:

Bước 1: Đăng nhập

* Người dùng nhập thông tin đăng nhập.
* Hệ thống kiểm tra thông tin.
* Nếu đúng, chuyển đến Bước 2.
* Nếu sai, hệ thống trả về thông báo lỗi và quay lại Bước 1 để người dùng nhập lại thông tin.

Bước 2: Tìm kiếm vé phù hợp

* Người dùng tìm kiếm vé phù hợp.
* Theo tên công ty hoặc theo tuyến xe.
* Sau khi chọn chuyến xe phù hợp, chuyển đến Bước 3.

Bước 3: Chọn ghế và nhập thông tin giao hàng nếu có

* Người dùng chọn ghế ngồi và nhập thông tin giao hàng nếu có.
* Nhấn nút đặt ghế.
* Chuyển đến Bước 4.

Bước 4: Nhận thông tin

* Hệ thống ghi nhận thông tin đặt ghế từ người dùng và tạo vé nhưng chưa có thông tin thanh toán.
* Nếu có yêu cầu giao hàng, hệ thống sẽ tạo thông tin giao hàng và cập nhật trường ticket\_id cho thông tin giao hàng.
* Với loại vé có yêu cầu thông tin giao hàng thì mỗi lần chỉ thêm được 1 vé.
* Sau khi lưu thông tin, chuyển đến Bước 5.

Bước 5: Hiển thị thông tin trong giỏ hàng

* Hệ thống sẽ lấy thông tin vé để hiển thị thông tin giỏ hàng.
* Người dùng kiểm tra lại thông tin trong giỏ hàng.
* Nhấn nút thanh toán nếu tất cả đều đúng.
* Chuyển đến Bước 6.

Bước 6: Chọn phương thức thanh toán

* Người dùng chọn phương thức thanh toán từ các tùy chọn có sẵn.
* Nhấn nút đặt vé để xác nhận.
* Chuyển đến Bước 7.

Bước 7: Tạo đường dẫn thanh toán

* Hệ thống tạo đường dẫn thanh toán và trả lại đường dẫn cho người dùng.
* Người dùng nhập thông tin thanh toán đầy đủ và xác nhận thanh toán.
* Chuyển đến Bước 8.

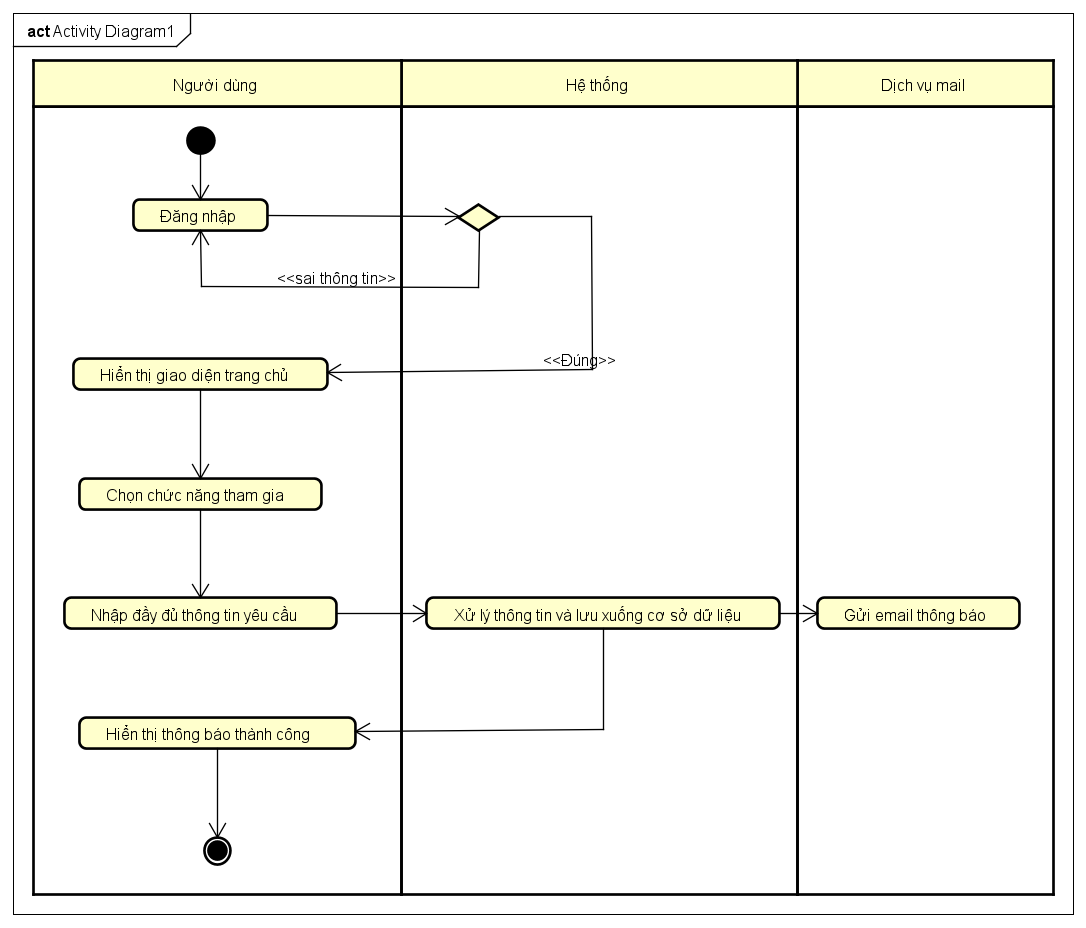
Bước 8: Xử lý thanh toán

* VNPay nhận và xử lý thông tin thanh toán.
* Nếu thành công, chuyển đến Bước 9.
* Nếu thất bại, hệ thống thông báo lỗi và quay lại Bước 6.

Bước 9: Cập nhật và lưu thông tin thanh toán

* Sau khi cập nhật xong, hiển thị hóa đơn cho người dùng.
* Hệ thống cập nhật thông tin thanh toán cho vé và lưu thay đổi vào cơ sở dữ liệu.
* Quá trình hoàn tất.

#### UC-02



Hình 3 ‑ 3‑8: Sơ đồ hoạt động UC-02

Mô tả:

Bước 1: Đăng nhập

* Người dùng nhập thông tin đăng nhập vào hệ thống.
* Hệ thống kiểm tra thông tin đăng nhập:
* Nếu sai thông tin, hệ thống thông báo lỗi và yêu cầu người dùng nhập lại, quay lại Bước 1.
* Nếu đúng thông tin, chuyển sang Bước 2.

Bước 2: Hiển thị giao diện trang chủ

* Hệ thống hiển thị giao diện trang chủ sau khi người dùng đăng nhập thành công, chuyển đến Bước 3.

Bước 3: Chọn chức năng tham gia

* Người dùng chọn chức năng mà họ muốn tham gia từ giao diện trang chủ.
* Sau khi chọn, hệ thống chuyển đến Bước 4.

Bước 4: Nhập đầy đủ thông tin yêu cầu

* Người dùng nhập các thông tin cần thiết cho chức năng mà họ đã chọn.
* Khi đã nhập xong, hệ thống chuyển đến Bước 5.

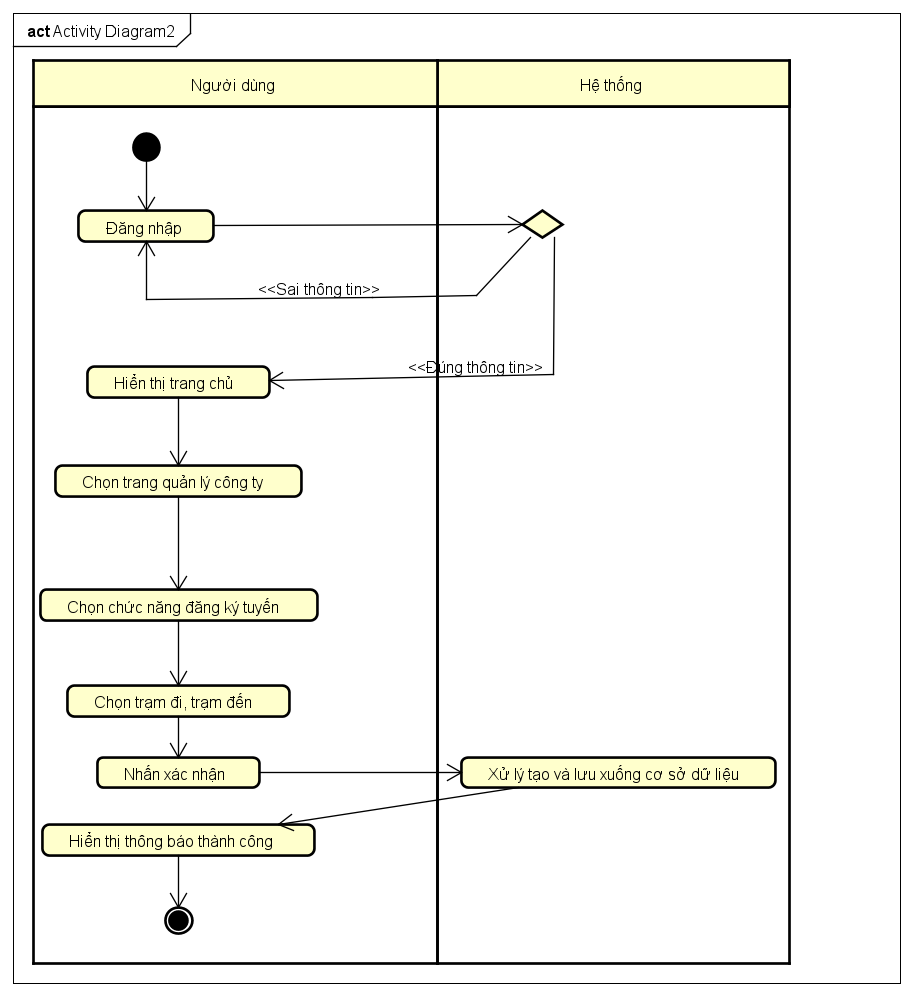
Bước 5: Xử lý thông tin và lưu vào cơ sở dữ liệu

* Hệ thống xử lý các thông tin người dùng cung cấp, kiểm tra và xác thực nếu cần.
* Sau khi xử lý, hệ thống lưu thông tin xuống cơ sở dữ liệu.
* Sau đó, hệ thống thực hiện gửi email thông báo qua Dịch vụ mail.

Bước 6: Hiển thị thông báo thành công

* Hệ thống hiển thị thông báo cho người dùng biết rằng thao tác đã thành công.
* Kết thúc quá trình.

#### UC-03



Hình 3 ‑ 3‑9: Sơ đồ hoạt động UC-03

Mô tả:

Bước 1: Đăng nhập

* Người dùng nhập thông tin đăng nhập vào hệ thống.
* Hệ thống kiểm tra thông tin đăng nhập:
* Nếu sai thông tin, hệ thống trả về thông báo lỗi và yêu cầu người dùng nhập lại, quay lại Bước 1.
* Nếu đúng thông tin, hệ thống chuyển sang Bước 2.

Bước 2: Hiển thị trang chủ

* Hệ thống hiển thị giao diện trang chủ cho người dùng sau khi đăng nhập thành công.
* Chuyển đến Bước 3 khi người dùng chọn trang quản lý công ty.

Bước 3: Chọn chức năng đăng ký tuyến

* Người dùng chọn chức năng đăng ký tuyến trên trang quản lý công ty.
* Sau khi chọn, hệ thống chuyển đến Bước 4.

Bước 4: Chọn trạm đi và trạm đến

* Người dùng chọn trạm đi và trạm đến cho tuyến mình muốn đăng ký.
* Nhấn nút xác nhận khi đã hoàn tất việc chọn trạm.
* Chuyển đến Bước 5.

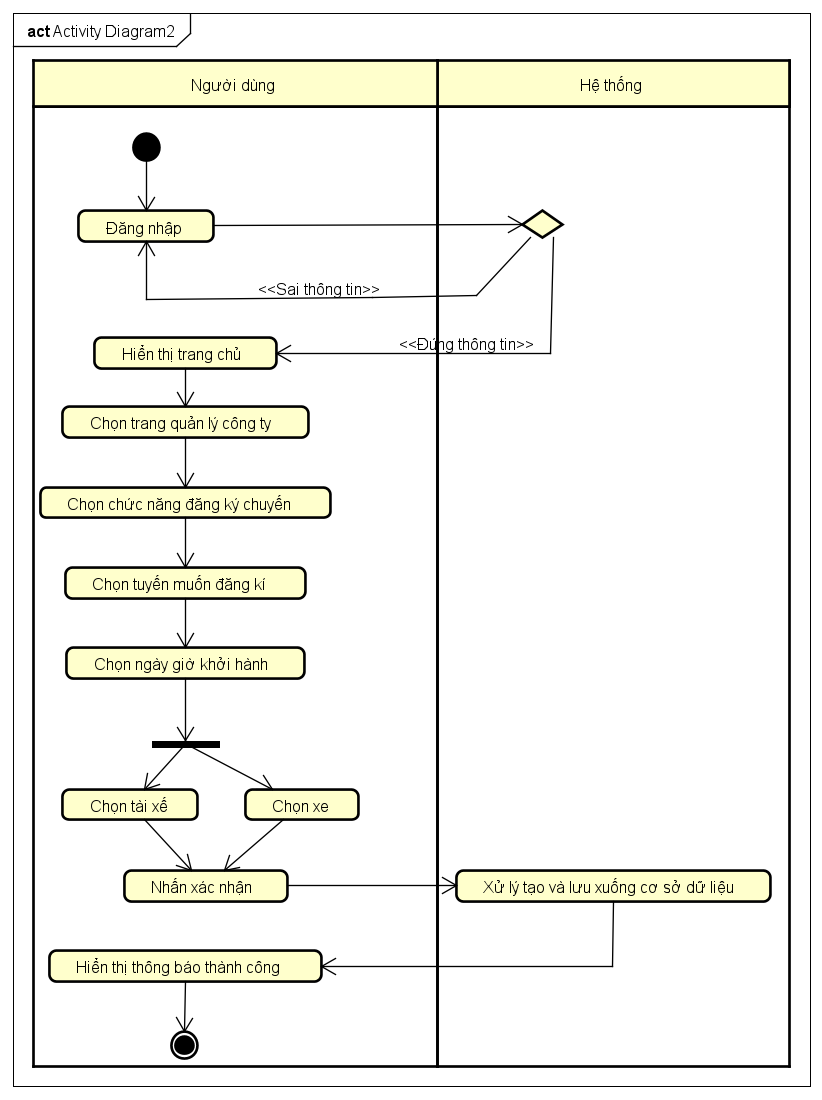
Bước 5: Xử lý tạo và lưu xuống cơ sở dữ liệu

* Hệ thống nhận thông tin từ người dùng, xử lý yêu cầu tạo và lưu thông tin xuống cơ sở dữ liệu.
* Sau khi lưu thành công, hệ thống chuyển đến Bước 6.

Bước 6: Hiển thị thông báo thành công

* Hệ thống hiển thị thông báo cho người dùng biết rằng quá trình đăng ký tuyến đã thành công.
* Kết thúc quá trình.

#### UC-04



Hình 3 ‑ 3‑10: Sơ đồ hoạt động UC-04

Mô tả:

Bước 1: Đăng nhập

* Người dùng nhập thông tin đăng nhập vào hệ thống.
* Hệ thống kiểm tra thông tin đăng nhập:
* Nếu sai thông tin, hệ thống trả về thông báo lỗi và yêu cầu người dùng nhập lại, quay lại Bước 1.
* Nếu đúng thông tin, hệ thống chuyển sang Bước 2.

Bước 2: Hiển thị trang chủ

* Hệ thống hiển thị giao diện trang chủ cho người dùng sau khi đăng nhập thành công.
* Chuyển đến Bước 3.

Bước 3: Chọn trang quản lý công ty

* Người dùng chọn trang quản lý công ty từ giao diện trang chủ.
* Sau đó, chuyển đến Bước 4.

Bước 4: Chọn chức năng đăng ký chuyến

* Người dùng chọn chức năng đăng ký chuyến trên trang quản lý công ty.
* Sau khi chọn, hệ thống chuyển đến Bước 5.

Bước 5: Chọn tuyến muốn đăng ký

* Người dùng chọn tuyến xe mà họ muốn đăng ký.
* Chuyển đến Bước 6.

Bước 6: Chọn ngày giờ khởi hành

* Người dùng chọn ngày và giờ khởi hành cho tuyến xe đã chọn.
* Sau khi chọn, hệ thống chuyển đến Bước 7.

Bước 7: Chọn tài xế và xe

* Hệ thống sẽ lấy danh sách các tài xế khả dụng, và các xe khả dụng.
* Chọn tài xế và Chọn xe khả dụng từ danh sách.
* Sau khi chọn xong cả hai, nhấn nút xác nhận để tiếp tục.
* Chuyển đến Bước 8.

Bước 8: Xử lý tạo và lưu xuống cơ sở dữ liệu

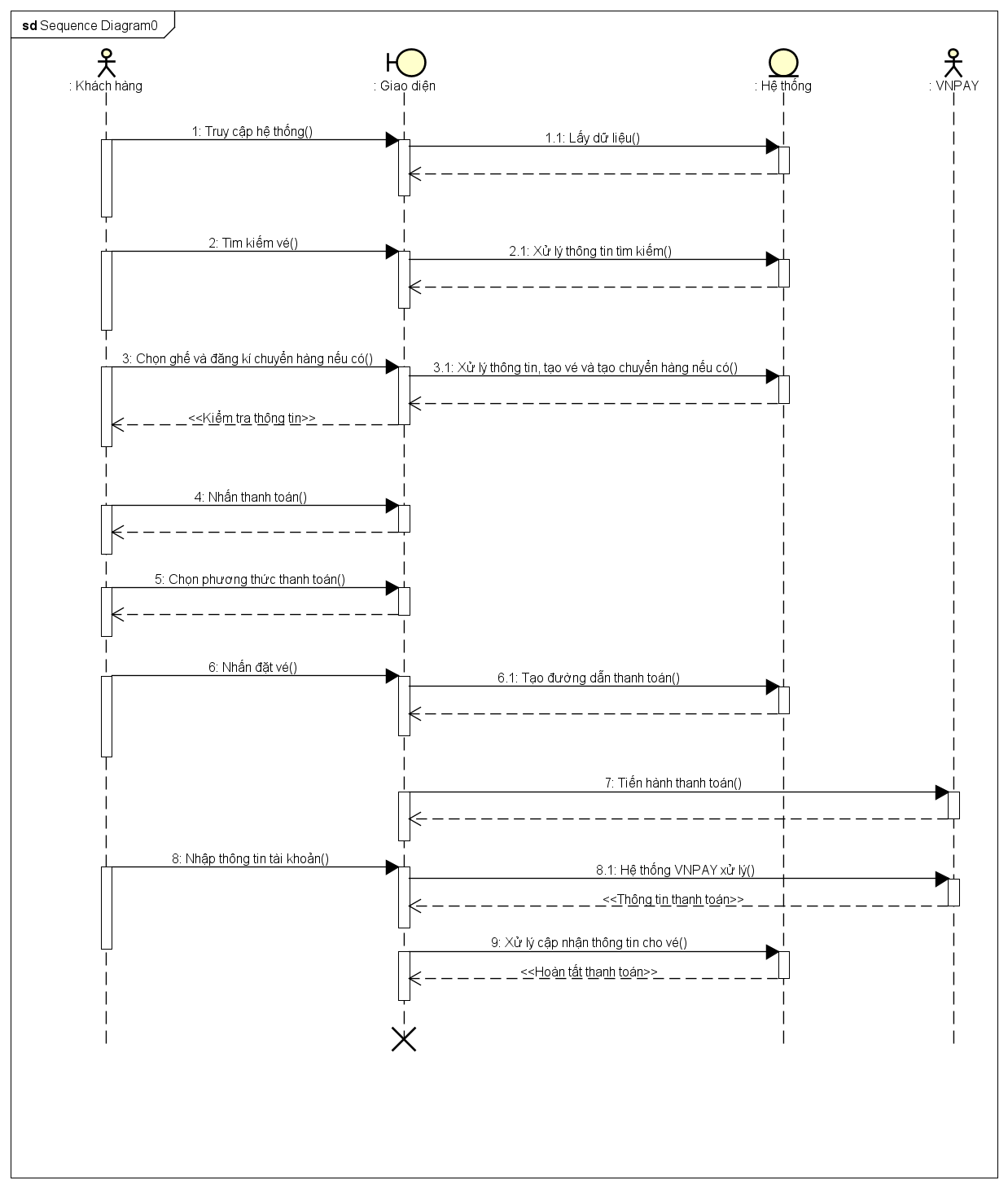
* Hệ thống nhận thông tin từ người dùng, xử lý yêu cầu và lưu thông tin xuống cơ sở dữ liệu.
* Sau khi lưu thành công, chuyển đến Bước 9.

Bước 9: Hiển thị thông báo thành công

* Hệ thống hiển thị thông báo cho người dùng biết rằng quá trình đăng ký chuyến đã thành công.
* Kết thúc quá trình.

### Sơ đồ tuần tự

#### UC-01



Hình 3 ‑ 3‑11: Sơ đồ tuần tự UC-01

Mô tả:

Bước 1: Truy cập hệ thống

* Khách hàng truy cập vào hệ thống qua giao diện.
* Giao diện gửi yêu cầu đến hệ thống để lấy dữ liệu cần thiết cho việc hiển thị giao diện.
* Hệ thống lấy dữ liệu và trả về cho giao diện để hiển thị.

Bước 2: Tìm kiếm vé

* Khách hàng thực hiện tìm kiếm vé trên giao diện.
* Giao diện gửi thông tin tìm kiếm đến hệ thống.
* Hệ thống xử lý thông tin tìm kiếm.
* Nếu tìm kiếm thành công, hệ thống trả về kết quả phù hợp để hiển thị cho khách hàng.
* Nếu không có kết quả phù hợp, hệ thống hiển thị thông báo để khách hàng có thể tìm kiếm lại.

Bước 3: Chọn ghế và đăng ký chuyến hàng (nếu có)

* Khách hàng chọn ghế phù hợp và có thể đăng ký thông tin vận chuyển hàng nếu có nhu cầu.
* Giao diện gửi yêu cầu này đến hệ thống.
* Hệ thống kiểm tra thông tin:
* Nếu thông tin hợp lệ, hệ thống tạo vé và tạo chuyến hàng nếu có yêu cầu, sau đó trả về xác nhận, vé tạo ra sẽ chưa có thông tin thanh toán.
* Nếu thông tin không hợp lệ, hệ thống trả về thông báo để khách hàng kiểm tra lại.

Bước 4: Nhấn thanh toán

* Khách hàng nhấn vào nút thanh toán trên giao diện.
* Giao diện chuyển đến bước tiếp theo cho phép khách hàng lựa chọn phương thức thanh toán.

Bước 5: Chọn phương thức thanh toán

* Khách hàng chọn phương thức thanh toán từ các tùy chọn có sẵn trên giao diện.
* Sau khi lựa chọn xong, khách hàng nhấn nút đặt vé và thực hiện thanh toán.

Bước 6: Nhấn đặt vé

* Khách hàng nhấn đặt vé để xác nhận thanh toán.
* Giao diện gửi yêu cầu đến hệ thống để tạo đường dẫn thanh toán qua VNPay.
* Hệ thống tạo đường dẫn thanh toán và trả về cho giao diện để hiển thị cho khách hàng.

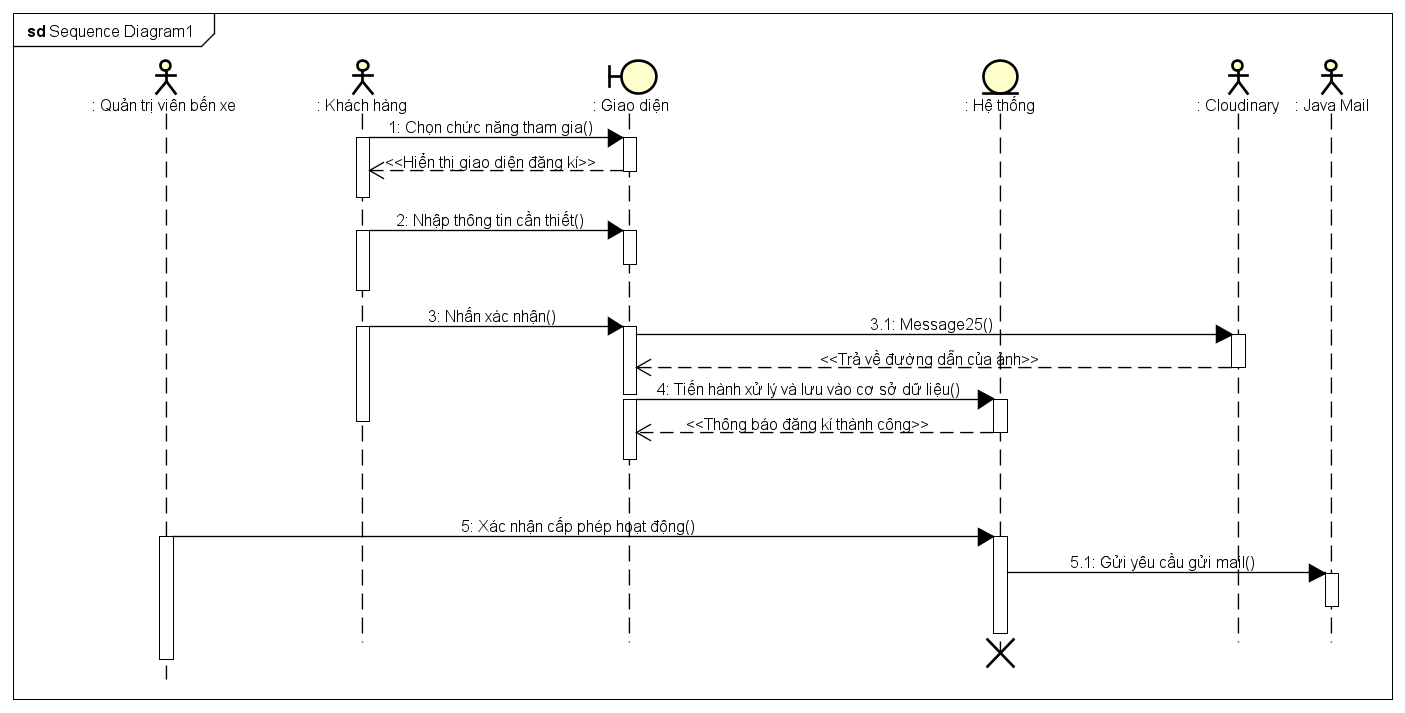
Bước 7: Tiến hành thanh toán

* Khách hàng thực hiện thanh toán trực tiếp trên server của VNPAY thông qua đường dẫn thanh toán được trả về.
* VNPay nhận yêu cầu thanh toán và xử lý.
* Nếu thanh toán thành công, VNPay gửi thông báo thành công về hệ thống, và đồng thời tạo hóa đơn online và cập nhật thông tin thanh toán cho vé đã tạo trước đó.
* Nếu thanh toán thất bại, VNPay trả về thông báo lỗi và Giao diện thông báo để Khách hàng có thể thử lại.

Bước 8: Hiển thị thông tin hóa đơn do VNPAY trả về

* Hệ thống nhận thông tin từ VNPay và hiển thị hóa đơn ra giao diện
* Khách hàng chọn về trang chủ và kết thúc.

#### UC-02



Hình 3 ‑ 3‑12: Sơ đồ tuần tự UC-02

Mô tả:

Bước 1: Chọn chức năng tham gia trên thanh Navbar

Bước 2: Điền thông tin theo yêu cầu vào giao diện

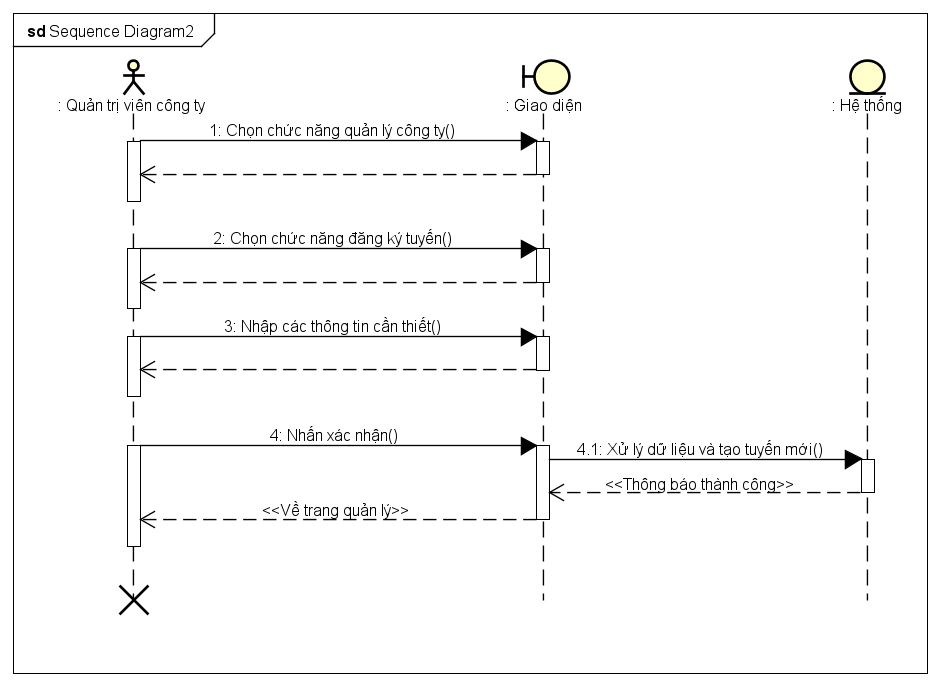
Bước 3: Hệ thống sẽ xử lý upload ảnh lên cloudinary và trả về dường dẫn ảnh rồi mới tạo công ty mới.

Bước 4: Thông báo thành công và về lại trang chủ.

Bước 5: Đợi cho quản lý bến xe xác nhận.

Bước 6: Nếu quản lý bến xe xác nhận xong thì sẽ gọi hệ thống để sử dụng JavaMail gửi thông báo và kết thúc.

#### UC-03



Hình 3 ‑ 3‑13: Sơ đồ tuần tự UC-03

Mô tả:

Bước 1: Đăng nhập và chọn chức năng quản lý công ty, hệ thống sẽ hiển thị trang quản lý công ty tương ứng.

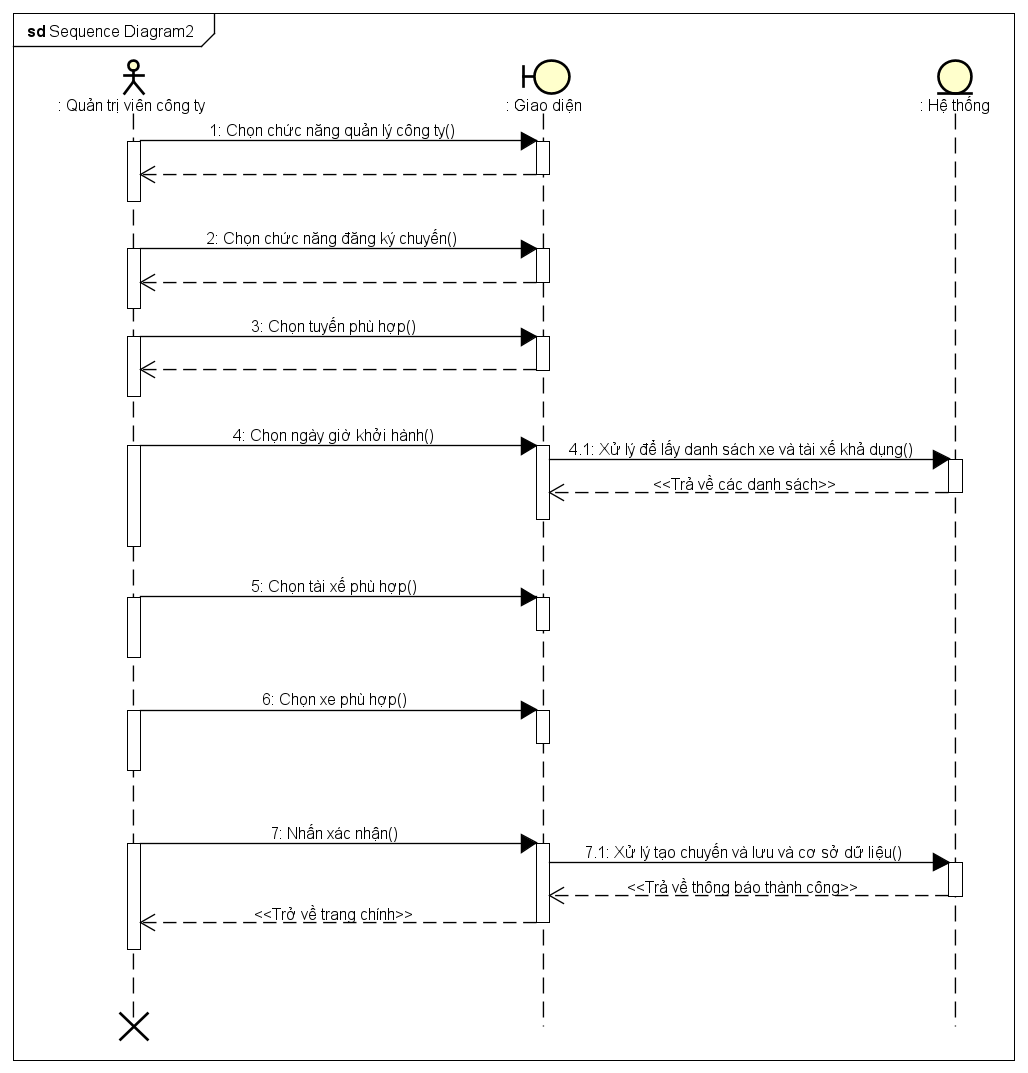
Bước 2: Chọn chức năng tạo tuyến. Hệ thống sẽ lấy dữ liệu các trạm xe mà bến xe hỗ trợ.

Bước 3: Chọn trạm xe bắt đầu.

Bước 4: Chọn trạm xe đến.

Bước 5: Nhấn đăng kí và hệ thống sẽ thông báo tạo tuyến thành công và về lại trang quản lý công ty.

#### UC-04



Hình 3 ‑ 3‑14: Sơ đồ tuần tự UC-04

Mô tả:

Bước 1: Đăng nhập và chọn chức năng quản lý công ty, hệ thống sẽ hiển thị trang quản lý công ty tương ứng.

Bước 2: Chọn chức năng tạo chuyến. Hệ thống sẽ lấy dữ liệu các tuyến xe mà nhà xe đã đăng ký trước đó.

Bước 3: Chọn tuyến xe muốn tạo chuyến.

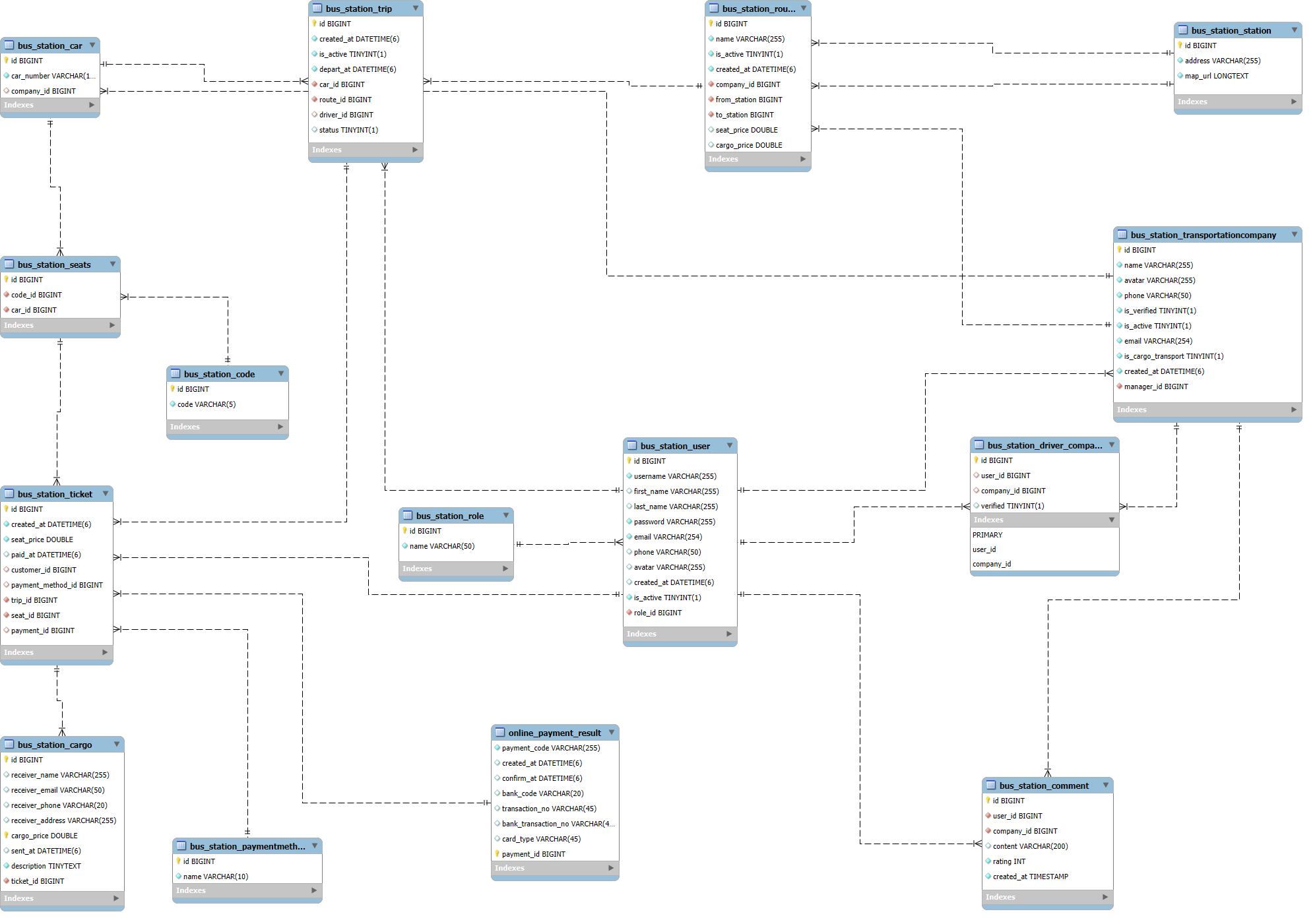
Bước 4: Chọn ngày và giờ khởi hành cho chuyến.

Bước 5: Hệ thống sẽ xử lý và lấy danh sách các tài xế và xe khả dụng.

Bước 6: Chọn tài xế và xe cho chuyến.

Bước 7: Nhấn đăng kí, hệ thống sẽ thông báo thành công và trở về trang quản lý. Kết thúc.

### Lược đồ cơ sở dữ liệu quan hệ



Hình 3 ‑ 3‑15: Lược đồ cơ sở dữ liệu quan hệ

Giải thích:

* bus\_station\_trip(*Id*, created\_at, depart\_at, #car\_id, #route\_id, #driver\_id, status)
* bus\_station\_car(*Id*, car\_number, #company\_id)
* bus\_station\_seats(*Id*, #code\_id, #car\_id)
* bus\_station\_code(*Id*, code)
* bus\_station\_ticket(*Id*, created\_at, seat\_price, paid\_at, #customer\_id, #payment\_method\_id, #seat\_id, #trip\_id, payment\_ui)
* bus\_station\_cargo(*Id*, receiver\_name, receiver\_email, receiver\_phone, receiver\_address, cargo\_price, description, #ticket\_id)
* bus\_station\_role(*Id*, name)
* bus\_station\_user(*Id*, username, password, first\_name, last\_name, email, phone, created\_at, is\_active, #role\_id, otp, otp\_creation\_time)
* bus\_station\_transportationcompany(*Id*, name, avatar, phone, email, is\_active, is\_verified, is\_cargo\_transport, created\_at, #manager\_id)
* bus\_station\_driver\_company(*Id*, #user\_id, #company\_id, verified)
* bus\_station\_comment(*Id*, #user\_id, #company\_id, content, rating, created\_at)
* bus\_station\_paymentmethod(*Id*, name)
* online\_payment\_result(*Id*, payment\_code, created\_at, confirm\_at, bank\_code, bank\_transaction\_no, card\_type, #payment\_ui)
* bus\_station\_route(*Id*, name, is\_active, created\_at, #company\_id, #from\_station, #to\_station, seat\_price, cargo\_price)
* bus\_station\_station(*Id*, name, address, map\_url)

Diễn giải các quan hệ:

1. Mối quan hệ giữa bus\_station\_trip và bus\_station\_car

* Mối quan hệ: bus\_station\_trip có khóa ngoại car\_id liên kết với bảng bus\_station\_car.
* Ý nghĩa: Mỗi chuyến đi được thực hiện bởi một xe bus cụ thể.

2. Mối quan hệ giữa bus\_station\_trip và bus\_station\_route

* Mối quan hệ: bus\_station\_trip có khóa ngoại route\_id liên kết với bảng bus\_station\_route.
* Ý nghĩa: Mỗi chuyến đi thuộc về một tuyến đường cụ thể.

3. Mối quan hệ giữa bus\_station\_trip và bus\_station\_driver\_company

* Mối quan hệ: bus\_station\_trip có khóa ngoại driver\_id liên kết với bảng bus\_station\_driver\_company.
* Ý nghĩa: Mỗi chuyến đi được lái bởi một tài xế thuộc một công ty vận tải cụ thể.

4. Mối quan hệ giữa bus\_station\_car và bus\_station\_transportationcompany

* Mối quan hệ: bus\_station\_car có khóa ngoại company\_id liên kết với bảng bus\_station\_transportationcompany.
* Ý nghĩa: Mỗi xe bus thuộc sở hữu của một công ty vận tải.

5. Mối quan hệ giữa bus\_station\_seats và bus\_station\_code

* Mối quan hệ: bus\_station\_seats có khóa ngoại code\_id liên kết với bảng bus\_station\_code.
* Ý nghĩa: Mỗi ghế trên xe bus có mã ghế riêng biệt.

6. Mối quan hệ giữa bus\_station\_seats và bus\_station\_car

* Mối quan hệ: bus\_station\_seats có khóa ngoại car\_id liên kết với bảng bus\_station\_car.
* Ý nghĩa: Mỗi ghế thuộc về một xe bus cụ thể.

7. Mối quan hệ giữa bus\_station\_ticket và bus\_station\_user

* Mối quan hệ: bus\_station\_ticket có khóa ngoại customer\_id liên kết với bảng bus\_station\_user.
* Ý nghĩa: Mỗi vé được mua bởi một người dùng cụ thể.

8. Mối quan hệ giữa bus\_station\_ticket và bus\_station\_paymentmethod

* Mối quan hệ: bus\_station\_ticket có khóa ngoại payment\_method\_id liên kết với bảng bus\_station\_paymentmethod.
* Ý nghĩa: Mỗi vé được thanh toán qua một phương thức thanh toán.

9. Mối quan hệ giữa bus\_station\_ticket và bus\_station\_seats

* Mối quan hệ: bus\_station\_ticket có khóa ngoại seat\_id liên kết với bảng bus\_station\_seats.
* Ý nghĩa: Mỗi vé chỉ định một ghế ngồi cụ thể.

10. Mối quan hệ giữa bus\_station\_ticket và bus\_station\_trip

* Mối quan hệ: bus\_station\_ticket có khóa ngoại trip\_id liên kết với bảng bus\_station\_trip.
* Ý nghĩa: Mỗi vé được bán cho một chuyến đi cụ thể.

11. Mối quan hệ giữa bus\_station\_cargo và bus\_station\_ticket

* Mối quan hệ: bus\_station\_cargo có khóa ngoại ticket\_id liên kết với bảng bus\_station\_ticket.
* Ý nghĩa: Mỗi kiện hàng hóa được liên kết với một vé cụ thể.

12. Mối quan hệ giữa bus\_station\_user và bus\_station\_role

* Mối quan hệ: bus\_station\_user có khóa ngoại role\_id liên kết với bảng bus\_station\_role.
* Ý nghĩa: Mỗi người dùng có một vai trò nhất định trong hệ thống.

13. Mối quan hệ giữa bus\_station\_transportationcompany và bus\_station\_user

* Mối quan hệ: bus\_station\_transportationcompany có khóa ngoại manager\_id liên kết với bảng bus\_station\_user.
* Ý nghĩa: Mỗi công ty vận tải được quản lý bởi một người dùng cụ thể.

14. Mối quan hệ giữa bus\_station\_driver\_company và bus\_station\_user

* Mối quan hệ: bus\_station\_driver\_company có khóa ngoại user\_id liên kết với bảng bus\_station\_user.
* Ý nghĩa: Mỗi tài xế là một người dùng trong hệ thống.

15. Mối quan hệ giữa bus\_station\_driver\_company và bus\_station\_transportationcompany

* Mối quan hệ: bus\_station\_driver\_company có khóa ngoại company\_id liên kết với bảng bus\_station\_transportationcompany.
* Ý nghĩa: Mỗi tài xế thuộc một công ty vận tải cụ thể.

16. Mối quan hệ giữa bus\_station\_comment và bus\_station\_user

* Mối quan hệ: bus\_station\_comment có khóa ngoại user\_id liên kết với bảng bus\_station\_user.
* Ý nghĩa: Mỗi bình luận được tạo bởi một người dùng.

17. Mối quan hệ giữa bus\_station\_comment và bus\_station\_transportationcompany

* Mối quan hệ: bus\_station\_comment có khóa ngoại company\_id liên kết với bảng bus\_station\_transportationcompany.
* Ý nghĩa: Mỗi bình luận được viết về một công ty vận tải.

18. Mối quan hệ giữa online\_payment\_result và bus\_station\_ticket

* Mối quan hệ: online\_payment\_result có khóa ngoại payment\_ui liên kết với bảng bus\_station\_ticket.
* Ý nghĩa: Mỗi kết quả thanh toán trực tuyến liên quan đến một vé cụ thể.

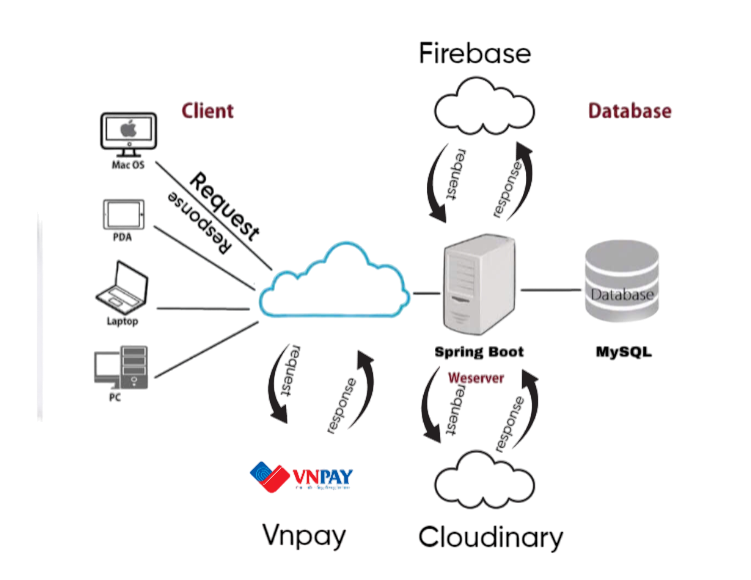
19. Mối quan hệ giữa bus\_station\_route và bus\_station\_transportationcompany

* Mối quan hệ: bus\_station\_route có khóa ngoại company\_id liên kết với bảng bus\_station\_transportationcompany.
* Ý nghĩa: Mỗi tuyến đường được quản lý bởi một công ty vận tải.

20. Mối quan hệ giữa bus\_station\_route và bus\_station\_station (từ trường from\_station và to\_station)

* Mối quan hệ: bus\_station\_route có các khóa ngoại from\_station và to\_station liên kết với bảng bus\_station\_station.
* Ý nghĩa: Mỗi tuyến đường có một trạm khởi hành và một trạm đến, đều liên kết với bảng bus\_station\_station.

## Kiến trúc hệ thống



Hình 3 ‑ 3‑16: Kiến trúc được sử dụng trong hệ thống

Hệ thống được phát triển chủ yếu trên nền tảng Spring Boot, sử dụng cơ sở dữ liệu MySQL để quản lý dữ liệu, ReactJS để xây dựng giao diện người dùng. Ngoài ra còn kết hợp đồng với nhiều nền tảng khách để khai thác các tiện ích sẵn có như Firebase (Chat thời gian thực), VNPAY (Thanh toán online), Cloudinary (Lưu trữ ảnh). Chi tiết kiến trúc sẽ được trình bày ở dưới:

### Client (ReactJS)

**Gửi yêu cầu**: Người dùng sẽ tương tác trực tiếp với giao diện, sau đó sẽ gửi yêu cầu (request) đến server bằng các phương thức như (GET, POST, PUT, PATCH, DELETE).

**Nhận phản hồi**: Sau khi server xử lý trong thì sẽ trả lại thông tin cần thiết (response), giao diện sẽ hứng dữ liệu trả về đó để phục vụ cho các nhu cầu ở phía người dùng.

### Webserver (Spring Boot)

**Framework Spring Boot**: Là nền tảng của hệ thống, nơi tiếp nhập và xử lý hầu hết các yêu cầu từ phía người dùng. Spring Boot sẽ tiếp nhận, phân tích ở lớp controller và sử dụng đúng nghiệp vụ đã được định nghĩa ở lớp service để xử lý yêu cầu đó. Và trả lại cho người dùng thông tin sau khi xử lý xong với một định dạng phù hợp như JSON.

**Luồng hoạt động của Webserver**:

Khi nhận được yêu cầu từ phía người dùng (Client), Spring Boot phân tích yêu cầu để xác định đúng nghiệp vụ cần thực hiện.

Sau đó sẽ sử dụng đúng dịch vụ đã được định nghĩa ở tầng service để xử lý, ở service sẽ gọi các phương thức tương ứng ở tầng repository để tương tác với cơ sở dữ liệu. Và trả ra các dữ liệu cần thiết để gửi về cho phía người dùng nếu có.

### Database (MySQL)

**Lưu trữ dữ liệu**: Là nơi lưu trữ và quản lý thông tin cả hệ thống trong suốt quá trình hoạt động.

**Tương tác với Webserver**: Lớp repository sẽ làm việc trực tiếp với cơ sở dữ liệu, quản lý thông quá các lệnh (SELECT, INSERT, UPDATE, và DELETE).

### Firebase

**Dịch vụ đám mây**: Firebase được cung cấp bởi Google. Cho phép người dùng sử dụng nhiều dịch vụ lưu trữ. Trong hệ thống của em sử dụng dịch vụ Realtime Databasse để lưu trữ và quản lý phục vụ cho tính năng chat thời gian thực.

**Luồng dữ liệu thời gian thực**: Được xử lý trực tiếp ở bên ReactJS, cho phép người dùng có thể tạo tin nhắn, lấy tin nhắn cho từng cuộc hội thoại với user khác.

### Cloudinary

**Quản lý đa phương tiện**: Cloudinary được sử dụng để lưu trữ và xử lý các file hình ảnh. Khi người dùng tải lên một tệp hình ảnh, Webserver sẽ gửi file đó đến Cloudinary để lưu trữ.

**Tương tác với Webserver**: Sau khi nhận yêu cầu upload hình ảnh từ client, server sẽ sử dụng dịch vụ đã được định nghĩa để gửi file ảnh đó lên cho Cloudinary xử lý upload, sau đó Cloudinary sẽ trả về một đường dẫn lưu ảnh, sau đó server sẽ trả đường dẫn đó về cho client.

**6. VNPay**

**Xử lý thanh toán**: VNPay là dịch vụ trung gian thanh toán điện tử. Spring Boot chỉ đóng vai trò là nơi tạo ra đường dẫn thanh toán (PaymentUrl) và trả về cho client để thực hiện thanh toán trực tiếp trên server của VNPAY.

**Luồng xử lý thanh toán**:

Sau khi client tính toán được tổng giá, thì sẽ gửi yêu cầu tạo đường dẫn thanh toán đến Spring Boot.

Spring Boot sẽ tạo paymentUrl dựa trên các thông số đã cài đặt sẵn và thông số từ client gửi đến và trả đường dẫn đó về phía client.

Client chuyển đến đường dẫn đó để thực hiện thanh toán trực tiếp trên server của VNPAY.

VNPay xử lý và trả về trạng thái của giao dịch (thành công hoặc thất bại).

### Luồng hoạt động chính của hệ thống

**Bước 1**: Client gửi yêu cầu đến Webserver thông qua một thiết bị như máy tính hoặc điện thoại.

**Bước 2**: Webserver (Spring Boot) nhận yêu cầu và phân tích nó ở tầng controller.

Sau đó controller sẽ gọi đúng nghiệp vụ để xử lý.

Service sẽ gọi phương thức tương ứng ở repository để tương tác với cơ sở dữ liệu.

Controller sẽ xử lý dữ liệu nếu có trước khi trả về cho phía client.

**Bước 3:** Webserver gửi phản hồi về người dùng (Client), kết thúc quá trình.

### Tổng kết

Hệ thống được xây dựng dựa trên nền tảng Sring Boot, ReactJS và MySQL để quản lý dữ liệu. Ngoài ra còn kết hợp các dịch vụ sẵn có khác như Cloudinary, Firebase, VNPAy để xử lý các nghiệp vụ phức tạp. Điều này mang lại sự tiện lợi và linh hoạt khi xây dựng, phát triển và hoàn thiện hệ thống. Luồng dữ liệu được xác định một cách rõ rang, logic dễ dàng sử dụng, bảo trì, và mở rộng hệ thống sau này.

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

### Kết quả đề tài

Qua đề tài, em đã thiết kế hệ thống khá đầy đủ với như cầu quản lý một bến xe. Có đầy đủ các vai trò người dùng, và các chức năng hỗ trợ cần thiết cho từng vai trò cụ thể.

Thứ nhất, hệ thống cung cấp các chức năng quản lý bến xe cho quản trị viên bễn xe, có thể quản lý các công ty, các trạm xe. Có thể xem thống kê các số liệu tổng quát và xuất báo cáo ra file PDF nếu có nhu cầu.

Thứ hai, quản lý công ty cũng được hỗ trợ rất nhiều, các công việc quan trọng, cần thiết để quản lý và vận hành nhà xe cũng được cung cấp khá đầy đủ. Quản trị viên nhà xe có thể quản lý tuyến xe, chuyến xe, xe phục vụ, tà xế và có thể đăng kí vận chuyển hàng hoặc hủy vận chuyển hàng. Ngoài ra, người quản trị còn có thể xem các biểu đồ số liệu cần thiết để quản lý công ty như doanh theo từng thời điểm cụ thể và cũng có thể xuất biểu đồ ra file PDF.

Thứ ba, Tài sẽ có thể xem được lịch trình làm việc của mình, có thể tại vé cho khách vãng lai và xử lý thanh toán bằng mã QR Code thông qua cổng thanh toán VNPAY.

Thứ tư, khách hàng có thể tìm kiếm vé bằng các công cụ hỗ trợ như tìm theo tên công ty hoặc theo tuyết phù hợp. Có thể chọn tùy chọn đăng kí gửi hàng và điền thông tin nếu muốn, và thực hiện thanh toán online thông qua VNPAY. Ngoài ra, khách hàng còn có thể bình luận, đánh giá chất lượng dịch vụ của các công ty và nhận tư vấn từ các quản trị viên công ty thông qua tính năng chat trực tuyến.

### Hạn chế

Bên cạnh đó, hệ thống vẫn còn các hạn chế như chưa thể kiểm soát được lượng xe ra vào bến cho nên có thể sẽ dẫn đến một vấn đề là có thể sẽ xảy qua quá tải khi mà lượng xe vào lớn hơn lượng xe xuất bến. Điều này sẽ làm cho bến xe trở nên ùn tắc. Em sẽ phải tìm ra một phương án nào đó để giải quyết vấn đề này.

## Hướng phát triển

Em sẽ nghiên cứu cải thiện các nghiệp vụ quản lý để nâng cao hiệu quả và tính ổn định của hệ thống. Về vấn đề giải quyết hạn chế còn tồn tại của hệ thống, em sẽ thiết kế một giải pháp để quản lý được số lượng xe ở trong bến xe vào cùng một thời điểm, đảm bảo bến xe không bị quá tải. Ví dụ như tại một thời điểm, bến xe có đủ chỗ chứa cho 100 xe và cho phép đậu tối đa 2 tiếng, thì em sẽ có cơ chế để kiểm soát vấn đề này. Giải pháp của em có thể là cho nhà xe chọn giờ vào bến trong lúc đăng kí tuyến và xếp chuyến đó vào một bảng băm với các key là các khung giờ cố định. Nếu có xe đăng kí vào khung giờ có key là N thì số lượng xe tối đa ở khung giờ N và N + 1 sẽ giảm đi 1 và khi xe xuất bến thì sẽ cập nhật lại số lượng tối đa.

TÀI LIỆU THAM KHẢO

[1] "Hướng dẫn: Giới thiệu về React," ReactJS Documentation, Truy cập: <https://vi.legacy.reactjs.org/tutorial/tutorial.html#overview>, Tháng 9, 2024.

[2] "Spring Boot Documentation," Spring, [Truy cập: Tháng 9, 2024]. [Online]. Có thể tìm thấy tại: [https://docs.spring.io/spring-boot/docs/current/reference/html/documentation-overview.html&#8203;:contentReference[oaicite:0]{index=0}&#8203;:contentReference[oaicite:1]{index=1}](https://docs.spring.io/spring-boot/docs/current/reference/html/documentation-overview.html&#8203;:contentReference%5Boaicite:0%5D%7Bindex=0%7D&#8203;:contentReference%5Boaicite:1%5D%7Bindex=1%7D).

[3] "Cloudinary Documentation," Cloudinary, [Truy cập: Tháng 9, 2024]. [Online]. Có thể tìm thấy tại: https://cloudinary.com/documentation&#8203;:contentReference[oaicite:2]{index=2}.

[4] "Firebase Documentation," Firebase, [Truy cập: Tháng 9, 2024]. [Online]. Có thể tìm thấy tại: https://firebase.google.com/docs&#8203;:contentReference[oaicite:3]{index=3}.

PHỤ LỤC