

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO MÔN HỌC PHÁT TRIỂN PHẦN MỀM HƯỚNG DỊCH VỤ

**ĐỀ TÀI: NỀN TẢNG TƯ VẤN THÔNG SỐ KỸ THUẬT Ô TÔ THEO KIẾN
TRÚC HƯỚNG DỊCH VỤ**

GVDH: Hoàng Quang Huy

Nhóm: 16

Họ tên sinh viên: Nguyễn Đình Hội - 2022603030

Lê Thị Ngọc Lan - 2022602329

Lê Văn Hải - 2022603180

Đào Xuân Giang – 2022603320

Lớp: 20241IT6151001

Hà Nội, tháng 12 năm 2024

LỜI MỞ ĐẦU

Ngày nay, với sự phát triển mạnh mẽ của khoa học công nghệ, các ngành công nghiệp liên quan đến công nghệ thông tin đang giữ vai trò tiên phong trong việc thay đổi cách thức con người tiếp cận và giải quyết vấn đề. Trong số đó, công nghệ web và các nền tảng dịch vụ trực tuyến đang ngày càng chứng minh tầm quan trọng qua việc mang lại sự tiện lợi, nhanh chóng và hiệu quả cho người dùng trong mọi lĩnh vực của đời sống.

Một trong những lĩnh vực nhận được sự quan tâm lớn hiện nay là ngành công nghiệp ô tô. Với sự gia tăng không ngừng về nhu cầu sở hữu xe hơi, việc tìm kiếm thông tin và so sánh các thông số kỹ thuật trở thành một nhu cầu tất yếu. Người dùng mong muốn có thể tiếp cận thông tin một cách chính xác, nhanh chóng và dễ dàng mà không cần tốn quá nhiều thời gian nghiên cứu từ nhiều nguồn khác nhau.

“Nền tảng tư vấn thông số kỹ thuật ô tô theo kiến trúc hướng dịch vụ” được xây dựng nhằm giải quyết bài toán này. Nền tảng không chỉ cung cấp một cơ sở dữ liệu chi tiết về thông số kỹ thuật, giá cả của các mẫu xe mà còn giúp người dùng so sánh, đánh giá và đưa ra quyết định phù hợp với nhu cầu cá nhân. Nền tảng hướng đến việc mang lại trải nghiệm tốt nhất cho người dùng trong việc chọn lựa và mua sắm xe ô tô. Kỳ vọng góp phần vào sự phát triển bền vững cho ngành công nghiệp ô tô trong thời đại số hóa. Nội dung chính của bài báo cáo gồm:

Chương 1: Tổng quan về đề tài

Chương 2: Tìm hiểu về kiến trúc hướng dịch vụ

Chương 3: Phân tích thiết kế hệ thống

Chương 4: Cài đặt chương trình

Chương 5: Kiểm thử

Chương 6: Kết luận và bài học kinh nghiệm

Trong quá trình thực hiện bài tập lớn này, chúng em xin gửi lời cảm ơn sâu sắc đến thầy Hoàng Quang Huy - giảng viên môn Phát Triển Phần Mềm Hướng Dịch Vụ đã nhiệt tình hướng dẫn và chỉ bảo chúng em trong suốt quá trình thực hiện đề tài, giúp cho đề tài của chúng em được hoàn thiện hơn, tốt hơn. Trong quá trình hoàn thành bài tập lớn chúng em sẽ không tránh khỏi những sai sót, rất mong sự thông cảm và đóng góp ý kiến bổ sung của các thầy cô giáo và của tất cả các bạn sinh viên.

Chúng em xin chân thành cảm ơn!

MỤC LỤC

LỜI MỞ ĐẦU	2
CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI.....	5
1.1. Lý do chọn đề tài.....	5
1.2. Mục tiêu đề tài.....	5
1.3. Yêu cầu chung và phương hướng phát triển.....	5
1.3.1. Yêu cầu chung.....	5
1.3.2. Phương hướng phát triển	6
1.4. Các chức năng chính	6
1.5. Đối tượng và phạm vi nghiên cứu	6
CHƯƠNG 2. TÌM HIỂU VỀ KIẾN TRÚC HƯỚNG DỊCH VỤ	7
2.1. Kiến trúc hướng dịch vụ là gì?	7
2.2. Kiến trúc hướng dịch vụ mang lại những lợi ích gì?	10
2.3. Những nguyên tắc cơ bản của kiến trúc hướng dịch vụ là gì?	10
2.4. Kiến trúc hướng dịch vụ gồm những thành phần nào?	10
2.5. Kiến trúc hướng dịch vụ hoạt động như thế nào?	11
2.6. So sánh và đánh giá kiến trúc SOA và kiến trúc Monolithic.	12
2.7. Mô hình triển khai SOA cho ứng dụng	14
CHƯƠNG 3. PHÂN TÍCH THIẾT KẾ HỆ THỐNG.....	18
3.1. Mô hình hoá dữ liệu.....	18
3.1.1. Biểu đồ lớp	18
3.1.2. Biểu đồ thực thể liên kết	18
3.2. Mô hình hoá chức năng.....	19
3.2.1. Các yêu cầu của hệ thống.....	19
3.2.2. Biểu đồ usecase	19
3.2.3. Đặc tả usecase	20
CHƯƠNG 4. CÀI ĐẶT CHƯƠNG TRÌNH	29
4.1. Kỹ thuật lấy dữ liệu demo cho chương trình	29
4.2. Cài đặt chương trình phía backend	30
4.2.1. Các kiến trúc và công nghệ sử dụng.....	30
4.2.2. Kết quả đạt được	31
4.3. Cài đặt chương trình phía frontend.....	37
4.3.1. Công nghệ sử dụng	37

4.3.2. Kết quả đạt được	37
CHƯƠNG 5. KIỂM THỬ	43
5.1. Cơ sở lý thuyết.....	43
5.1.1. Khái niệm kiểm thử.....	43
5.1.2. Các phương pháp kiểm thử.....	43
5.2. Các công cụ sử dụng	44
5.2.1. Postman	44
5.2.2. Jmeter	44
5.3. Kiểm thử phần mềm	45
5.3.1. Lập kế hoạch kiểm thử	45
5.3.2. Phân tích thiết kế kiểm thử	46
5.3.3. Thực hiện kiểm thử	47
5.3.4. Báo cáo kiểm thử	48
CHƯƠNG 6. KẾT LUẬN VÀ BÀI HỌC KINH NGHIỆM	66
6.1. Kết luận.....	66
6.2. Bài học kinh nghiệm	66
6.3. Đề xuất cho dự án tương lai.....	66
TÀI LIỆU THAM KHẢO	67

CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI

1.1. Lý do chọn đề tài

Trong bối cảnh công nghiệp ô tô không ngừng phát triển, việc lựa chọn một chiếc xe phù hợp với nhu cầu cá nhân đang trở thành một vấn đề quan trọng đối với người tiêu dùng. Tuy nhiên, sự đa dạng về mẫu mã, tính năng và mức giá trên thị trường hiện nay lại khiến người dùng gặp nhiều khó khăn trong việc tìm kiếm và so sánh các thông số kỹ thuật giữa các dòng xe.

Hơn nữa, tại Việt Nam, xu hướng sử dụng các nền tảng trực tuyến để tra cứu thông tin và mua sắm ngày càng phổ biến. Tuy nhiên, phần lớn các nền tảng hiện nay chủ yếu tập trung vào bán hàng, mà chưa có một hệ thống nào thực sự mạnh mẽ trong việc cung cấp thông tin kỹ thuật chi tiết và so sánh giữa các dòng xe. Điều này đã tạo ra một khoảng trống trong việc hỗ trợ người dùng đưa ra các quyết định chính xác và tối ưu nhất khi lựa chọn xe hơi.

Việc xây dựng một nền tảng tư vấn thông số kỹ thuật ô tô theo hướng dịch vụ không chỉ đáp ứng nhu cầu cấp thiết của người tiêu dùng, mà còn giúp nâng cao trải nghiệm mua sắm trực tuyến trong lĩnh vực này. Với tầm quan trọng của công nghệ trong việc hỗ trợ đời sống con người, đề tài này mang ý nghĩa thực tiễn cao, đồng thời tạo cơ hội để áp dụng các kiến thức đã học vào việc giải quyết một bài toán cụ thể trong thực tế.

1.2. Mục tiêu đề tài

- Xây dựng một nền tảng tư vấn thông số kỹ thuật ô tô phục vụ nhu cầu, tiết kiệm thời gian, chi phí cho người tiêu dùng.
- Học hỏi thêm được nhiều kiến thức, tăng thêm sự hiểu biết về các phương pháp cũng như cách để xây dựng một nền tảng theo kiến trúc hướng dịch vụ.
- Xây dựng nền tảng mang lại tính tiện lợi cho người dùng
- Xây dựng nền tảng giúp việc quản lý trở nên dễ dàng, thuận tiện và nhanh chóng.

1.3. Yêu cầu chung và phương hướng phát triển

1.3.1. Yêu cầu chung

- Tìm hiểu các phương pháp xây dựng một nền tảng, hệ thống theo kiến trúc hướng dịch vụ.
- Cung cấp thông tin chính xác và đầy đủ về các thông số kỹ thuật xe ô tô.
- Giao diện người dùng thân thiện, dễ sử dụng, hỗ trợ tìm kiếm và so sánh xe nhanh chóng.
- Hệ thống cần bảo mật, ổn định và có khả năng mở rộng dễ dàng.

1.3.2. Phương hướng phát triển

- Triển khai ý tưởng, tạo dựng một nền tảng tư vấn thông số kỹ thuật ô tô với một số chức năng cơ bản của tài khoản khách hàng và tài khoản admin như đăng ký, đăng nhập, tìm kiếm, xem chi tiết xe, so sánh xe, quản lý người dùng, quản lý xe.
- Hướng đến phục vụ nhu cầu của người dùng trong lĩnh vực công nghiệp ô tô.
- Xây dựng một nền tảng với nhiều thông tin đa dạng, phong phú, chính xác.
- Trình bày giao diện nền tảng có logic, khoa học, dễ nhìn, màu sắc hài hoà.
- Có thể sử dụng cho mục đích thương mại điện tử trong tương lai.

1.4. Các chức năng chính

- Đối với khách hàng :
 - + Tìm kiếm xe
 - + Xem xe
 - + Xem chi tiết 1 xe
 - + So sánh xe
- Đối với Admin:
 - + Đăng ký/ Đăng nhập
 - + Quản lý User
 - + Quản lý xe

1.5. Đối tượng và phạm vi nghiên cứu

- Đối tượng:
 - + Các thông số kỹ thuật ô tô
 - + Các nền tảng trực tuyến tư vấn và so sánh xe ô tô
 - + Người tiêu dùng ô tô
- Phạm vi nghiên cứu:
 - + Xây dựng các chức năng cho khách hàng
 - + Xây dựng các chức năng cho Admin
 - + Tìm hiểu về kiến trúc hướng dịch vụ

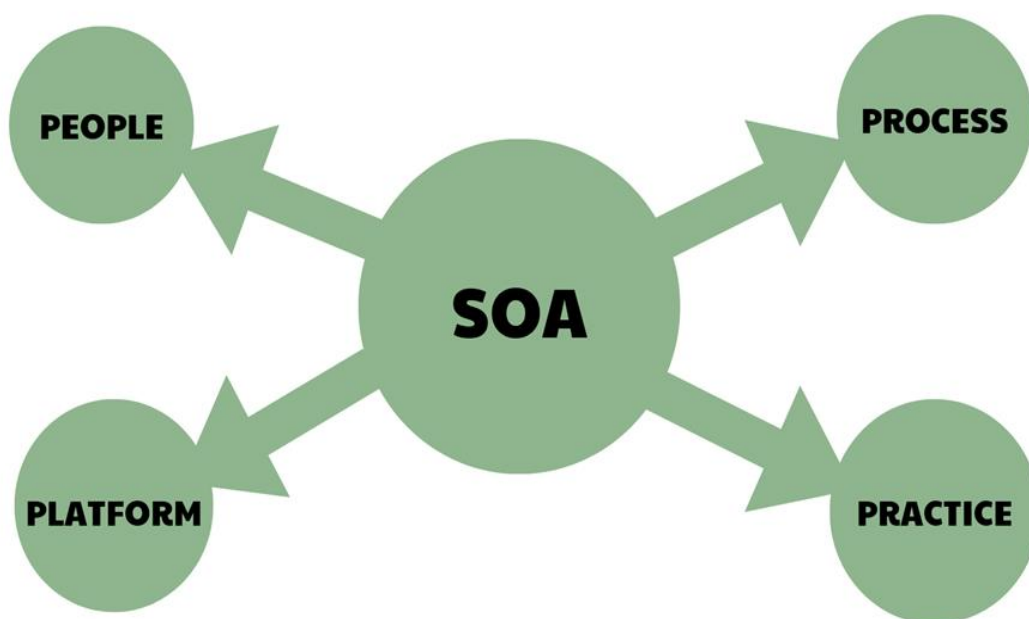
CHƯƠNG 2. TÌM HIỂU VỀ KIẾN TRÚC HƯỚNG DỊCH VỤ

2.1. Kiến trúc hướng dịch vụ là gì?

— Kiến trúc hướng dịch vụ (SOA) là một phương pháp phát triển phần mềm sử dụng các thành phần của phần mềm được gọi là dịch vụ để tạo ra các ứng dụng dành cho doanh nghiệp. Mỗi dịch vụ cung cấp một tính năng doanh nghiệp, đồng thời các dịch vụ cũng có thể giao tiếp với nhau giữa nhiều nền tảng và ngôn ngữ. Nhà phát triển tận dụng SOA để tái sử dụng các dịch vụ trong nhiều hệ thống khác nhau hoặc kết hợp một số dịch vụ độc lập để thực hiện các tác vụ phức tạp.

Ví dụ: Một ví dụ về kiến trúc hướng dịch vụ cho người tiêu dùng là ứng dụng dành cho người chạy bộ. Thay vì xây dựng các công cụ tùy chỉnh để theo dõi khoảng cách và bản đồ, ứng dụng có thể tích hợp các dịch vụ cho thiết bị GPS của điện thoại và dịch vụ bản đồ hiện có, chẳng hạn như Google Maps.

Có 4 yếu tố quan trọng cấu thành nên một hệ thống SOA:



— **Platform (Nền tảng):** Nền tảng của SOA bao gồm các công nghệ và hạ tầng cần thiết cho việc phát triển, triển khai, và vận hành các dịch vụ:

- + **Middleware:** Là phần mềm trung gian hỗ trợ giao tiếp giữa các dịch vụ khác nhau. Middleware giúp kết nối các dịch vụ lại với nhau và quản lý các giao thức giao tiếp, truyền tải dữ liệu.
- + **Enterprise Service Bus (ESB):** ESB cung cấp khả năng điều phối dữ liệu và giao tiếp giữa các dịch vụ khác nhau. ESB có thể chuyển đổi dữ

liệu từ định dạng của dịch vụ này sang định dạng của dịch vụ khác, cho phép các dịch vụ tương tác mà không cần phụ thuộc chặt chẽ vào nhau.

+ **Web Services:** Đây là các dịch vụ dựa trên giao thức web như SOAP (Simple Object Access Protocol) và REST (Representational State Transfer) để các dịch vụ có thể giao tiếp với nhau qua mạng. SOAP thường được dùng cho các giao tiếp cần tính bảo mật và tiêu chuẩn cao, trong khi REST linh hoạt và nhẹ hơn, phổ biến trong các ứng dụng web hiện đại.

+ **API Gateway:** Là công quản lý truy cập đến các dịch vụ. API Gateway điều phối lưu lượng truy cập, kiểm soát bảo mật, quản lý các phiên bản API, và cung cấp khả năng quản lý tập trung, giúp các dịch vụ bảo mật hơn và quản lý dễ dàng hơn.

— **People (Yếu tố con người):** Yếu tố con người trong SOA đóng vai trò quan trọng trong việc phát triển, vận hành, và quản lý các dịch vụ.

+ **Developers (Nhà phát triển):** Các nhà phát triển chịu trách nhiệm thiết kế và xây dựng các dịch vụ độc lập theo mô hình SOA.

+ **Architects (Kiến trúc sư):** Là người đảm bảo các dịch vụ được thiết kế tuân thủ các nguyên tắc SOA. Họ định hình cấu trúc tổng thể của hệ thống để các dịch vụ có thể tương tác hiệu quả và linh hoạt.

+ **Operators (Người vận hành):** Là người chịu trách nhiệm giám sát và quản lý các dịch vụ khi chúng được triển khai trong môi trường sản xuất. Các operator phải đảm bảo các dịch vụ hoạt động ổn định, giám sát các lỗi phát sinh, xử lý các sự cố và duy trì tính liên tục của hệ thống.

+ **Business Analysts (Chuyên viên phân tích nghiệp vụ):** Những người này định nghĩa các yêu cầu nghiệp vụ, đảm bảo các dịch vụ được phát triển đáp ứng mục tiêu của tổ chức.

— **Process (Quy trình):** SOA yêu cầu quy trình quản lý và vận hành để đảm bảo các dịch vụ hoạt động có hệ thống, nhất quán, và an toàn:

+ **Service Lifecycle Management (Quản lý vòng đời dịch vụ):** Đây là quy trình quản lý các dịch vụ từ giai đoạn thiết kế, phát triển, kiểm thử, triển khai, và bảo trì.

+ **Governance (Quản trị):** Quản trị SOA liên quan đến việc thiết lập các quy tắc, tiêu chuẩn, và chính sách cho các dịch vụ. Nó bao gồm quản lý quyền truy cập, bảo mật, đảm bảo tính nhất quán và tương thích giữa các dịch vụ.

+ **Change Management (Quản lý thay đổi):** Đây là quy trình để kiểm soát và quản lý các thay đổi trong dịch vụ. Điều này giúp các dịch vụ cập

nhật mà không làm gián đoạn các dịch vụ đang hoạt động, từ đó hạn chế ảnh hưởng tiêu cực đến hệ thống.

+ **Monitoring and Optimization (Giám sát và tối ưu hóa):** Giám sát dịch vụ nhằm phát hiện sớm các vấn đề, từ đó tối ưu hóa hiệu suất và sử dụng tài nguyên hiệu quả. Việc giám sát và tối ưu hóa giúp duy trì tính ổn định và nâng cao hiệu quả của hệ thống.

– **Practice (Thực hành):**

+ **Designing for Reusability (Thiết kế để tái sử dụng):** Các dịch vụ nên được xây dựng để có thể tái sử dụng trong nhiều ứng dụng mà không cần chỉnh sửa nhiều. Điều này tiết kiệm tài nguyên và giảm thời gian phát triển.

+ **Loose Coupling (Liên kết lỏng lẻo):** Các dịch vụ liên kết lỏng lẻo, không phụ thuộc quá nhiều vào nhau. Nhờ vậy, khi cập nhật hoặc thay thế dịch vụ, hệ thống vẫn hoạt động mà không cần thay đổi các dịch vụ khác.

+ **Service Contracts (Hợp đồng dịch vụ):** Mỗi dịch vụ có "hợp đồng" xác định đầu vào, đầu ra và cách thức hoạt động. Điều này giúp các bên liên quan hiểu và tích hợp dịch vụ dễ dàng hơn.

+ **Security (Bảo mật):** Bảo mật là yếu tố quan trọng trong SOA. Xác thực, mã hóa dữ liệu và quản lý truy cập giúp đảm bảo chỉ người được ủy quyền mới có thể sử dụng dịch vụ.

+ **Monitoring and Management (Giám sát và Quản lý):** Giám sát dịch vụ giúp phát hiện các vấn đề kịp thời, theo dõi hiệu suất, và tối ưu hóa dịch vụ để hoạt động hiệu quả hơn.

+ **Versioning (Quản lý phiên bản):** Khi dịch vụ có thay đổi hoặc nâng cấp, cần duy trì nhiều phiên bản dịch vụ để không ảnh hưởng đến các ứng dụng sử dụng phiên bản cũ.

+ **Testing (Kiểm thử):** Kiểm thử đơn vị, tích hợp và hiệu năng giúp đảm bảo các dịch vụ hoạt động chính xác, đặc biệt khi kết hợp với các dịch vụ khác trong hệ thống SOA.

2.2. Kiến trúc hướng dịch vụ mang lại những lợi ích gì?

- Rút ngắn thời gian đưa ra thị trường : Nhà phát triển tái sử dụng các dịch vụ trên những quy trình kinh doanh khác nhau để tiết kiệm thời gian và chi phí. Họ có thể hợp dịch các ứng dụng nhanh hơn bằng SOA so với việc lập trình và thực hiện tích hợp từ đầu.
- Bảo trì hiệu quả : Các dịch vụ nhỏ dễ tạo dựng, cập nhật và khắc phục lỗi hơn những đoạn mã lớn trong ứng dụng đơn khối. Việc sửa đổi bất kỳ dịch vụ nào trong SOA cũng không làm ảnh hưởng đến chức năng tổng thể của quy trình kinh doanh.
- Khả năng thích ứng cao hơn: SOA dễ thích ứng hơn với những cải tiến về công nghệ. Bạn có thể hiện đại hóa các ứng dụng của mình một cách hiệu quả và tiết kiệm.

2.3. Những nguyên tắc cơ bản của kiến trúc hướng dịch vụ là gì?

- Khả năng tương tác: Mỗi dịch vụ trong SOA bao gồm các tài liệu mô tả chỉ rõ chức năng của dịch vụ cùng các điều khoản và điều kiện liên quan. Mọi hệ thống máy khách đều có thể vận hành dịch vụ, dù dùng nền tảng cơ sở hay ngôn ngữ lập trình gì.
- Liên kết ít phụ thuộc: Các dịch vụ trong SOA cần được liên kết ít phụ thuộc, có càng ít sự phụ thuộc vào các tài nguyên bên ngoài như mô hình dữ liệu hay hệ thống thông tin thì càng tốt. Các dịch vụ này cũng nên ở tình trạng vô trạng thái mà không giữ lại bất kỳ thông tin nào từ các phiên làm việc hay giao dịch trước. Bằng cách này, khi bạn sửa đổi một dịch vụ, các ứng dụng máy khách và dịch vụ khác đang sử dụng dịch vụ này sẽ không bị ảnh hưởng đáng kể.
- Tính trừu tượng: Khách hàng hoặc người sử dụng dịch vụ trong SOA không cần biết logic lập trình hay chi tiết triển khai của dịch vụ. Đối với họ, dịch vụ nên như một chiếc hộp đen. Khách hàng nhận được thông tin cần thiết về tính năng và cách sử dụng dịch vụ thông qua hợp đồng dịch vụ và các tài liệu mô tả dịch vụ khác.
- Độ chi tiết: Các dịch vụ trong SOA nên có kích thước và phạm vi phù hợp, lý tưởng là tính đóng gói một chức năng kinh doanh riêng biệt mỗi dịch vụ. Sau đó, nhà phát triển có thể sử dụng nhiều dịch vụ để tạo ra một dịch vụ tổng hợp phục vụ việc thực hiện những thao tác phức tạp.

2.4. Kiến trúc hướng dịch vụ gồm những thành phần nào?

— Service (Dịch vụ)

Dịch vụ là những khối dựng cơ bản của SOA. Chúng có thể là dịch vụ tư nhân – chỉ dành cho người dùng nội bộ của một tổ chức – hoặc công cộng – tất cả mọi người đều có thể truy cập dịch vụ đó qua Internet. Cụ thể, mỗi dịch vụ có ba đặc điểm chính:

- + Triển khai dịch vụ: Triển khai dịch vụ là phần mã xây dựng logic để thực hiện chức năng dịch vụ cụ thể.
- + Hợp đồng dịch vụ: Hợp đồng dịch vụ xác định bản chất của dịch vụ cùng các điều khoản và điều kiện liên quan, chẳng hạn như những điều kiện tiên quyết để sử dụng dịch vụ, chi phí dịch vụ, và chất lượng của dịch vụ được cung cấp.
- + Giao diện dịch vụ: Trong SOA, các dịch vụ hoặc hệ thống khác giao tiếp với một dịch vụ thông qua giao diện của dịch vụ đó. Giao diện xác định cách bạn có thể gọi dịch vụ để thực hiện các hoạt động hoặc trao đổi dữ liệu. Giao diện giúp làm giảm sự phụ thuộc giữa dịch vụ và trình yêu cầu dịch vụ.

— **Service provider (Nhà cung cấp dịch vụ)**

Nhà cung cấp dịch vụ tạo dựng, duy trì và cung cấp một hoặc nhiều dịch vụ mà người khác có thể sử dụng. Các tổ chức có thể tự tạo dịch vụ của riêng hoặc mua từ một nhà cung cấp dịch vụ bên thứ ba.

— **Service customer (Người tiêu dùng dịch vụ)**

Người sử dụng dịch vụ yêu cầu nhà cung cấp dịch vụ vận hành một dịch vụ cụ thể. Đó có thể là cả một hệ thống, một ứng dụng, hay một dịch vụ khác. Hợp đồng dịch vụ nêu rõ các quy tắc nhà cung cấp dịch vụ và người sử dụng dịch vụ phải tuân theo khi tương tác với nhau. Nhà cung cấp dịch vụ và người sử dụng dịch vụ có thể đến từ các bộ phận, tổ chức và thậm chí là các ngành khác nhau.

— **Service registry (Sổ đăng ký dịch vụ)**

Sổ đăng ký dịch vụ, hay kho dịch vụ, là một danh mục các dịch vụ có sẵn, có thể truy cập qua mạng. Sổ đăng ký dịch vụ này lưu trữ các tài liệu mô tả dịch vụ từ các nhà cung cấp dịch vụ. Tài liệu mô tả dịch vụ chứa thông tin và cách giao tiếp với dịch vụ. Người sử dụng dịch vụ có thể dễ dàng tìm được những dịch vụ họ cần bằng cách dùng sổ đăng ký dịch vụ.

2.5. Kiến trúc hướng dịch vụ hoạt động như thế nào?

Các dịch vụ giao tiếp bằng những quy tắc được thiết lập sẵn, quyết định việc trao đổi thông tin qua một mạng lưới. Những quy tắc này được gọi là giao thức giao tiếp. Sau đây là một vài giao thức tiêu chuẩn để triển khai SOA:

- + Giao thức truy cập đối tượng đơn giản(SOAP)
- + HTTP RESTful
- + Apache Thrift
- + Apache ActiveMQ

- + Dịch vụ thông báo Java (JMS)

2.6. So sánh và đánh giá kiến trúc SOA và kiến trúc Monolithic.

— Định nghĩa

+ Kiến trúc Monolithic

- Là kiến trúc nguyên khối, nơi toàn bộ ứng dụng được phát triển, triển khai và chạy dưới dạng một thực thể duy nhất.
- Gồm các module chặt chẽ như UI, logic xử lý, và cơ sở dữ liệu.

+ Kiến trúc SOA

- Là kiến trúc phân tách ứng dụng thành các dịch vụ độc lập, giao tiếp với nhau thông qua các giao thức chuẩn như HTTP, SOAP, hoặc REST.
- Mỗi dịch vụ đại diện cho một chức năng cụ thể trong hệ thống.

— Đặc điểm chi tiết

Tiêu chí	Monolithic	SOA
Quản lý và triển khai	Tất cả các phần của ứng dụng nằm trong một thực thể duy nhất. Dễ quản lý nhưng triển khai khó khi phát triển lớn hơn.	Dịch vụ độc lập, triển khai từng phần dễ dàng mà không ảnh hưởng toàn hệ thống.
Hiệu suất	Hiệu suất cao hơn khi ứng dụng nhỏ vì không có độ trễ mạng giữa các module.	Hiệu suất giảm nếu có nhiều dịch vụ giao tiếp qua mạng, nhưng linh hoạt hơn với ứng dụng lớn.
Mở rộng	Chỉ mở rộng toàn bộ ứng dụng, khó mở rộng riêng lẻ từng module.	Dễ dàng mở rộng từng dịch vụ mà không cần thay đổi hệ thống khác.
Khả năng tái sử dụng	Mức độ tái sử dụng thấp, mỗi ứng dụng thường phải xây dựng lại từ đầu.	Dịch vụ có thể được tái sử dụng bởi nhiều hệ thống hoặc ứng dụng khác.
Độ phức tạp	Ít phức tạp hơn trong giai đoạn đầu, nhưng khó bảo trì khi ứng dụng lớn.	Phức tạp hơn, cần quản lý giao tiếp, bảo mật, và đồng bộ dữ liệu giữa các dịch vụ.
Khả năng chịu lỗi	Một lỗi nhỏ có thể làm hỏng toàn bộ ứng dụng.	Lỗi ở một dịch vụ thường không ảnh hưởng lớn đến toàn bộ hệ thống.
Chi phí	Chi phí thấp hơn trong giai đoạn đầu phát triển.	Chi phí cao hơn khi xây dựng ban đầu do cần thiết kế và tích hợp các dịch vụ.
Tính linh hoạt	Khó thay đổi hoặc nâng cấp mà không làm gián đoạn hệ thống.	Linh hoạt, cho phép thay đổi hoặc nâng cấp từng dịch vụ mà không ảnh hưởng hệ thống chính.

— **Ưu điểm và nhược điểm**

+ Kiến trúc Monolithic

• Ưu điểm:

- ✓ Đơn giản: Dễ thiết lập và triển khai khi ứng dụng nhỏ.
- ✓ Hiệu suất cao: Không cần giao tiếp mạng giữa các module.
- ✓ Chi phí thấp: Không cần quản lý nhiều dịch vụ riêng lẻ.

• Nhược điểm:

- ✓ Khó bảo trì: Ứng dụng phức tạp khi phát triển lớn.
- ✓ Khó mở rộng: Chỉ có thể mở rộng theo chiều dọc (tăng tài nguyên cho server).
- ✓ Độ phụ thuộc cao: Một lỗi nhỏ có thể ảnh hưởng toàn bộ hệ thống.

+ Kiến trúc SOA

• Ưu điểm:

- ✓ Mở rộng dễ dàng: Mở rộng dịch vụ cụ thể mà không cần thay đổi toàn bộ hệ thống.
- ✓ Tái sử dụng cao: Các dịch vụ có thể tái sử dụng bởi nhiều ứng dụng khác nhau.
- ✓ Bảo trì tốt hơn: Thay đổi hoặc sửa lỗi ở một dịch vụ không ảnh hưởng đến dịch vụ khác.
- ✓ Tính chịu lỗi cao: Hệ thống ít bị ảnh hưởng bởi lỗi của một dịch vụ.

• Nhược điểm:

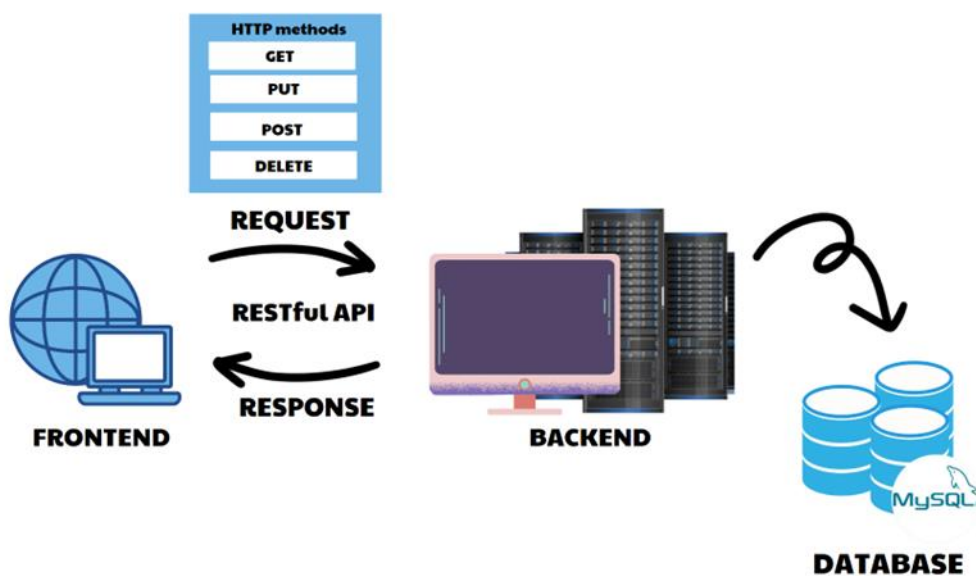
- ✓ Độ phức tạp cao: Cần quản lý giao tiếp giữa các dịch vụ, bảo mật, và dữ liệu phân tán.
- ✓ Hiệu suất: Có thể gặp vấn đề về độ trễ mạng
- ✓ Chi phí cao: Cần đầu tư vào công nghệ và nhân lực để thiết kế, triển khai.

— **Đánh giá kiến trúc SOA và kiến trúc Monolithic**

Tiêu chí	Monolithic	SOA
Phù hợp khi	Ứng dụng nhỏ, đơn giản, ngân sách hạn chế.	Hệ thống lớn, phức tạp, yêu cầu khả năng mở rộng cao.
Ngắn hạn vs Dài hạn	Tốt trong ngắn hạn, không hiệu quả khi mở rộng dài hạn.	Tốt trong dài hạn, đặc biệt với các tổ chức lớn.
Quy mô đội ngũ	Thích hợp cho đội ngũ nhỏ đến trung bình	Đội ngũ lớn với kỹ năng cao để phát triển và vận hành.

2.7. Mô hình triển khai SOA cho ứng dụng

Triển khai SOA theo RESTful API



— RESTful API là gì?

RESTful API là giao diện mà hai hệ thống máy tính sử dụng để trao đổi thông tin một cách an toàn qua internet. Hầu hết các ứng dụng kinh doanh phải giao tiếp với các ứng dụng nội bộ và bên thứ ba khác để thực hiện nhiều tác vụ khác nhau. Nó chú trọng vào tài nguyên hệ thống (tệp văn bản, ảnh, âm thanh, video, hoặc dữ liệu động...), bao gồm các trạng thái tài nguyên được định dạng và được truyền tải qua HTTP.

— Lý do lựa chọn RESTful API

- + Khả năng mở rộng

Các hệ thống triển khai REST API có thể mở rộng hiệu quả vì REST tối ưu hóa tương tác giữa máy khách và máy chủ. Tính không trạng thái loại bỏ tải máy chủ vì máy chủ không phải lưu giữ thông tin yêu cầu của máy khách trước đó. Bộ nhớ đệm được quản lý tốt loại bỏ một phần hoặc toàn bộ một số tương tác giữa máy khách và máy chủ. Tất cả các tính năng này hỗ trợ khả năng mở rộng mà không gây ra tình trạng tắc nghẽn giao tiếp làm giảm hiệu suất.

- + **Tính linh hoạt**

Dịch vụ web RESTful hỗ trợ tách biệt hoàn toàn giữa máy khách và máy chủ. Chúng đơn giản hóa và tách rời các thành phần máy chủ khác nhau để mỗi phần có thể phát triển độc lập. Các thay đổi về nền tảng hoặc công nghệ tại ứng dụng máy chủ không ảnh hưởng đến ứng dụng máy khách. Khả năng phân lớp các chức năng ứng dụng làm tăng tính linh hoạt hơn nữa. Ví dụ, các nhà phát triển có thể thực hiện các thay đổi đối với lớp cơ sở dữ liệu mà không cần viết lại logic ứng dụng.

- + **Độc lập**

REST API độc lập với công nghệ được sử dụng. Bạn có thể viết cả ứng dụng máy khách và máy chủ bằng nhiều ngôn ngữ lập trình khác nhau mà không ảnh hưởng đến thiết kế API. Bạn cũng có thể thay đổi công nghệ cơ bản ở cả hai bên mà không ảnh hưởng đến giao tiếp.

- **RESTful API hoạt động như thế nào**

- + Máy khách liên hệ với máy chủ bằng cách sử dụng API khi cần một tài nguyên.
- + Máy khách gửi yêu cầu đến máy chủ. Máy khách làm theo tài liệu API để định dạng yêu cầu theo cách mà máy chủ có thể hiểu được.
- + Máy chủ xác thực máy khách và xác nhận rằng máy khách có quyền thực hiện yêu cầu đó.
- + Máy chủ nhận được yêu cầu và xử lý nội bộ.
- + Máy chủ trả về phản hồi cho máy khách. Phản hồi chứa thông tin cho máy khách biết yêu cầu có thành công hay không. Phản hồi cũng bao gồm bất kỳ thông tin nào mà máy khách yêu cầu.

- **Thành phần của máy khách RESTful API trong hệ thống ứng dụng bài tập lớn**

- + **Mã định danh tài nguyên duy nhất**

Máy chủ xác định từng tài nguyên bằng các mã định danh tài nguyên duy nhất. Máy chủ thực hiện xác định tài nguyên bằng cách sử dụng Uniform Resource Locator (URL). URL chỉ định đường dẫn đến tài nguyên.

+ Methods

Các nhà phát triển thường triển khai API RESTful bằng cách sử dụng Giao thức truyền siêu văn bản (HTTP). Một phương thức HTTP cho máy chủ biết những gì cần làm với tài nguyên:

- GET

GET để truy cập tài nguyên nằm tại URL được chỉ định trên máy chủ. Có thể lưu trữ các yêu cầu GET và gửi tham số trong yêu cầu API RESTful để hướng dẫn máy chủ lọc dữ liệu trước khi gửi.

- POST

Máy khách sử dụng POST để gửi dữ liệu đến máy chủ. Chúng bao gồm biểu diễn dữ liệu với yêu cầu. Gửi cùng một yêu cầu POST nhiều lần có tác dụng phụ là tạo cùng một tài nguyên nhiều lần.

- PUT

Khách hàng sử dụng PUT để cập nhật tài nguyên hiện có trên máy chủ. Không giống như POST, việc gửi cùng một yêu cầu PUT nhiều lần trong dịch vụ web RESTful sẽ mang lại cùng một kết quả.

- DELETE

Khách hàng sử dụng yêu cầu DELETE để xóa tài nguyên. Yêu cầu DELETE có thể thay đổi trạng thái máy chủ. Tuy nhiên, nếu người dùng không có xác thực phù hợp, yêu cầu sẽ không thành công.

+ HTTP headers

- Dữ liệu

Các yêu cầu API REST có thể bao gồm dữ liệu cho POST, PUT và các phương thức HTTP khác để hoạt động thành công.

- Các tham số

Yêu cầu API RESTful có thể bao gồm các tham số cung cấp cho máy chủ nhiều thông tin chi tiết hơn về những gì cần thực hiện.

Tham số đường dẫn chỉ định chi tiết URL.

Tham số truy vấn yêu cầu thêm thông tin về tài nguyên.

Các tham số cookie xác thực khách hàng một cách nhanh chóng.

— Phương pháp xác thực RESTful API

Khóa API là một tùy chọn khác để xác thực REST API. Trong phương pháp này, máy chủ gán một giá trị được tạo duy nhất cho máy khách lần đầu. Bất cứ khi nào máy khách cố gắng truy cập tài nguyên, nó sẽ sử dụng khóa API duy nhất để xác minh chính nó.

— **Phản hồi của máy chủ**

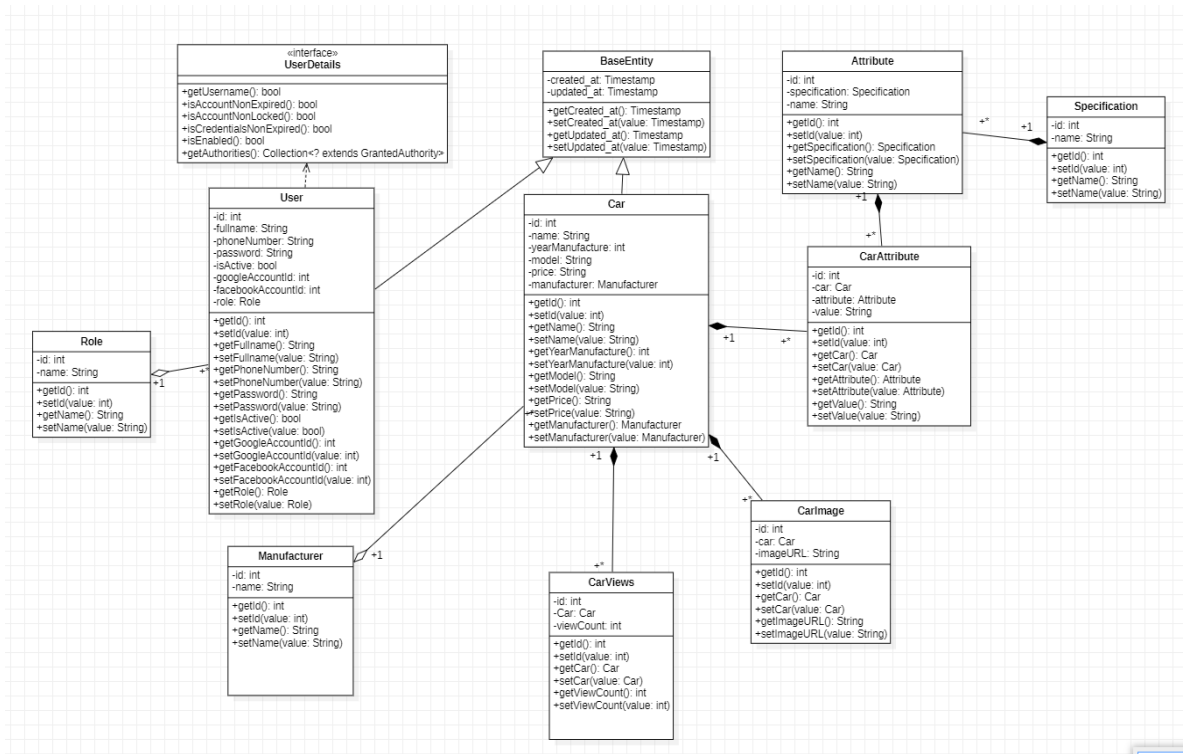
Dòng trạng thái chứa mã trạng thái ba chữ số thông báo yêu cầu thành công hay thất bại.

- + 200: Phản hồi thành công chung
- + 201: Phản hồi thành công của phương thức POST
- + 400: Yêu cầu không đúng mà máy chủ không thể xử lý
- + 404: Không tìm thấy tài nguyên

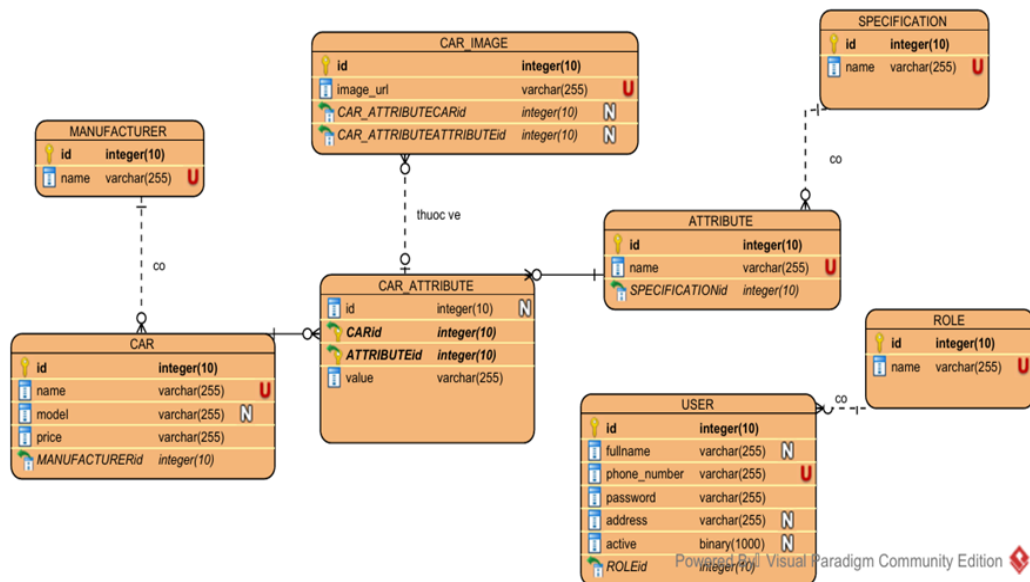
CHƯƠNG 3. PHÂN TÍCH THIẾT KẾ HỆ THỐNG

3.1. Mô hình hoá dữ liệu

3.1.1. Biểu đồ lớp



3.1.2. Biểu đồ thực thể liên kết



3.2. Mô hình hoá chức năng

3.2.1. Các yêu cầu của hệ thống

3.2.1.1. Yêu cầu chức năng

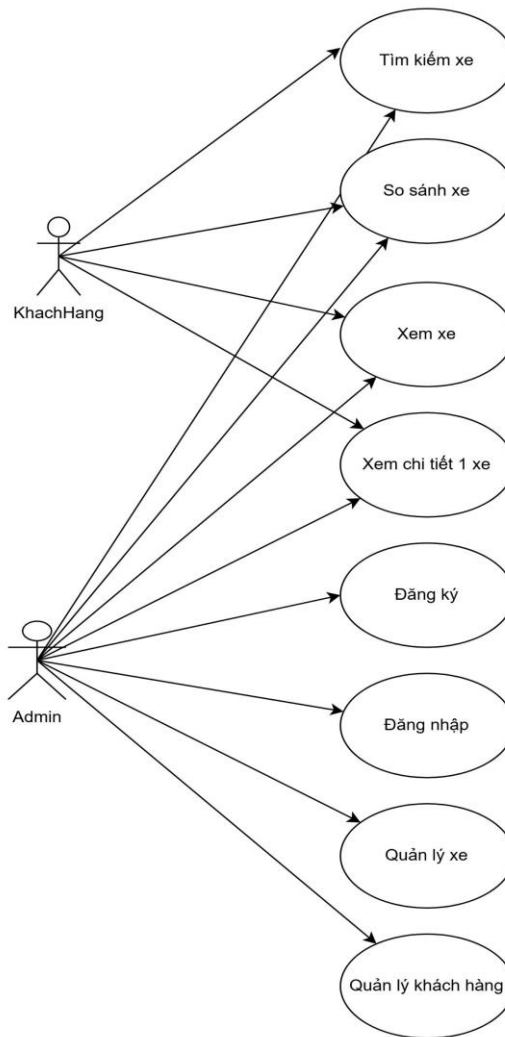
- Đăng ký: Người dùng có thể tạo tài khoản bằng cách cung cấp thông tin như số điện, mật khẩu, tên, địa chỉ....
- Đăng nhập: Người dùng có thể đăng nhập bằng email và mật khẩu đã đăng ký
- Phân quyền người dùng: Người dùng bình thường có quyền truy cập thông tin và tìm kiếm, so sánh xe, trong khi admin sẽ có quyền quản lý dữ liệu xe, dữ liệu người dùng, chỉnh sửa thông tin.
- Tìm kiếm xe: Cho phép người dùng tìm kiếm xe theo các tiêu chí như tên xe, nhà sản xuất.
- So sánh thông số kỹ thuật xe: Cung cấp chức năng so sánh thông số giữa các dòng xe để giúp người dùng đưa ra quyết định mua sắm phù hợp.
- Xem xe: Cung cấp thông tin cơ bản như tên và giá về các mẫu xe
- Xem chi tiết 1 xe: Cung cấp các thông tin chi tiết về từng mẫu xe bao gồm thông số kỹ thuật, giá bán...

3.2.1.2. Yêu cầu phi chức năng

- Hiệu suất và tốc độ: Việc xử lý các yêu cầu cần nhanh chóng, đảm bảo trải nghiệm người dùng không bị gián đoạn
- Bảo mật: Đảm bảo dữ liệu người dùng và thông tin được bảo mật, sử dụng token giúp quản lý phiên làm việc và phân quyền người dùng
- Khả năng mở rộng: thiết kế dễ dàng mở rộng để bổ sung thêm các mẫu xe mới.
- Tính tương thích: tương thích với các trình duyệt phổ biến và thiết bị di động
- Dễ sử dụng và thân thiện với người dùng: Giao diện trực quan, dễ sử dụng, dễ tìm kiếm và so sánh các mẫu xe
- Tính linh hoạt trong cập nhật dữ liệu: Cho phép admin dễ dàng cập nhật và chỉnh sửa thông tin xe.

3.2.2. Biểu đồ usecase

3.2.2.1. Usecase tổng quát



3.2.3. Đặc tả usecase

3.2.3.1. Mô tả Usecase “Đăng ký”

Cho phép khách hàng đăng ký tài khoản.

– Luồng sự kiện:

+ Luồng cơ bản:

1. Use-case bắt đầu khi người dùng click vào biểu tượng User và nhấn vào “ĐĂNG KÝ”. Hệ thống hiển thị form đăng ký gồm các trường: Họ và Tên, Số điện thoại, E-mail, Địa chỉ, Mật khẩu, Xác nhận mật khẩu, Ngày sinh
2. Người dùng nhập: Họ và Tên, Số điện thoại, E-mail, Địa chỉ, Mật khẩu, Xác nhận mật khẩu, Ngày sinh rồi click vào nút “ĐĂNG KÝ”. Hệ thống xác thực các trường người dùng nhập vào, nếu đúng thì thêm tài khoản vào bảng USERS, ROLES trong Cơ sở dữ liệu,

với is_active(Trạng thái tài khoản) được set giá trị mặc định là 1 (được hoạt động) và cho phép người dùng truy cập vào hệ thống. Use-case kết thúc.

+ Luồng rẽ nhánh:

1. Tại bước 2 trong luồng cơ bản, khi người dùng nhập số điện thoại hoặc E-mail đã có trong Cơ sở dữ liệu hoặc nhập sai số điện thoại so với quy cách. Hệ thống quay lại bước 2 trong luồng cơ bản với thông báo lỗi tương ứng.
2. Tại bước 2 trong luồng cơ bản, khi người dùng nhập tại trường “Xác nhận mật khẩu” không trùng khớp với trường “Mật khẩu” ở trên. Hệ thống quay lại bước 2 trong luồng cơ bản với thông báo lỗi tương ứng.
3. Tại bất kỳ bước nào trong luồng cơ bản nếu không kết nối được với cơ sở dữ liệu thì hệ thống sẽ hiển thị một thông báo lỗi và use case kết thúc.

– Các yêu cầu đặc biệt:

Không có.

– Tiền điều kiện:

Không có.

– Hậu điều kiện:

Một bản ghi mới được tạo ra trong bảng USERS và ROLES

– Điểm mở rộng:

Đăng ký/đăng nhập qua bên thứ 3 (Google)

3.2.3.2. Mô tả Usecase “Đăng nhập”

Cho phép khách hàng đăng nhập tài khoản.

– Luồng sự kiện:

+ Luồng cơ bản:

1. Use-case bắt đầu khi người dùng click vào biểu tượng User rồi nhấn vào “ĐĂNG NHẬP”. Hệ thống hiển thị form đăng nhập gồm các trường: Số điện thoại, Mật khẩu, Checkbox “Nhớ Tài Khoản”.
2. Người dùng nhập Số điện thoại, Mật khẩu, rồi click vào nút “ĐĂNG NHẬP”. Hệ thống truy cập vào CSDL so sánh phone_number và password trong bảng USERS. Nếu đúng thì hệ thống sẽ cho phép người dùng truy cập vào hệ thống. Use-case kết thúc.

- + Luồng rẽ nhánh:
 1. Tại bước 2 trong luồng cơ bản, khi người dùng nhập sai Số điện thoại hoặc mật khẩu. Hệ thống quay lại bước 2 trong luồng cơ bản cùng với thông báo lỗi.
 2. Tại bước 2 trong luồng cơ bản, khi người dùng nhập đúng Số điện thoại hoặc mật khẩu. Hệ thống kiểm tra is_active, nếu = 0 (vô hiệu hoá) thì sẽ hiển thị một cửa sổ thông báo ở góc trên bên phải màn hình và quay lại trang chủ của trang web. Use-case kết thúc.
 3. Tại bất kỳ bước nào trong luồng cơ bản nếu không kết nối được với cơ sở dữ liệu thì hệ thống sẽ hiển thị một thông báo lỗi và use case kết thúc.
- Các yêu cầu đặc biệt:
Không có.
- Tiền điều kiện:
Không có.
- Hậu điều kiện:
Nếu đăng nhập thành công thì hệ thống sẽ thay đổi trạng thái người dùng.
- Điểm mở rộng:
 - + Quên mật khẩu
 - + Đăng ký
 - + Đăng nhập/đăng ký qua bên thứ ba (Google)

3.2.3.3. Mô tả Usecase “Xem xe”

Use case này cho phép khách hàng xem một số thông tin cơ bản của xe

- Luồng sự kiện:
 - + Luồng cơ bản:

Use-case bắt đầu khi khách hàng click vào “Xem xe” trên thanh menu. Hệ thống sẽ truy cập vào cơ sở dữ liệu lấy một số thông tin về các xe bao gồm tenXe, nhaSX, namSX, dongXe, giaTien từ bảng CARS và hiển thị danh sách lên màn hình .
 - + Luồng rẽ nhánh:

Tại bất kỳ bước nào trong luồng cơ bản, nếu không kết nối được với CSDL thì hệ thống sẽ hiển thị một thông báo lỗi và use case kết thúc.
- Các yêu cầu đặc biệt:

Không có.

— Tiền điều kiện:

Không có.

— Hậu điều kiện:

Không có

— Điểm mở rộng:

Không có

3.2.3.4. Mô tả Usecase “Xem chi tiết 1 xe”

Use-case này cho phép khách hàng khách hàng xem chi tiết 1 xe.

— Luồng sự kiện:

+ Luồng cơ bản:

1. Use case này bắt đầu khi khách hàng click vào tên nhà sản xuất. Hệ thống sẽ lấy thông tin về các xe gồm tenXe, namSX, giaTien từ bảng CARS và hìnhAnhXe từ bảng CAR_IMAGES của nhà sản xuất đó và hiển thị lên màn hình.

2. Khách hàng click vào tên xe hoặc hình ảnh xe. Hệ thống sẽ truy cập cơ sở dữ liệu lấy thông tin chi tiết của xe đó từ bảng CARS, CAR_IMAGES, ATTRIBUTES, CAR_ATTRIBUTE, MANUFACTURERS, SPECIFICATIONS và hiển thị lên màn hình. Use case kết thúc.

+ Luồng rẽ nhánh:

Tại bất kỳ bước nào trong luồng cơ bản, nếu không kết nối được với CSDL thì hệ thống sẽ hiển thị một thông báo lỗi và use case kết thúc.

— Các yêu cầu đặc biệt:

Không có.

— Tiền điều kiện:

Không có.

— Hậu điều kiện:

Không có

— Điểm mở rộng:

Không có

3.2.3.5. Mô tả Usecase “So sánh xe”

Use case này cho phép người dùng so sánh thông tin những chiếc xe để chọn ra lựa chọn phù hợp với họ.

- Luồng sự kiện
 - + Luồng cơ bản:
 1. Use case bắt đầu khi khách hàng click vào nút “So sánh” bên cạnh xe.
 2. Người dùng chọn các xe muốn so sánh (ví dụ: tích chọn từng xe để thêm vào bảng so sánh).
 3. Người dùng nhấn vào nút "So sánh" để thực hiện thao tác so sánh các xe đã chọn.
 4. Hệ thống truy xuất thông tin của các xe được chọn từ cơ sở dữ liệu, được lấy từ các bảng CARS, CAR_IMAGES, ATTRIBUTES, CAR_ATTRIBUTE, MANUFACTURERS, SPECIFICATIONS.
 5. Hệ thống hiển thị bảng so sánh với các thông tin chi tiết theo từng tiêu chí cho mỗi xe.
 6. Người dùng có thể tiếp tục chọn thêm xe để so sánh hoặc loại bỏ xe ra khỏi bảng so sánh nếu muốn.
 - + Luồng rẽ nhánh.
 1. Bảng so sánh giới hạn xe: Nếu người dùng chọn vượt quá số lượng xe cho phép trong bảng so sánh (ví dụ: tối đa 4 xe), hệ thống sẽ hiện thông báo và yêu cầu người dùng bỏ bớt một xe.
 2. Không có thông tin xe: Nếu một xe không có dữ liệu về một tiêu chí nhất định, hệ thống sẽ hiển thị "Không có thông tin" hoặc ô trống cho tiêu chí đó.
 3. Lỗi kết nối cơ sở dữ liệu: Nếu hệ thống không thể truy xuất dữ liệu xe từ cơ sở dữ liệu, sẽ hiện thông báo lỗi và yêu cầu người dùng thử lại sau.
- Các yêu cầu đặc biệt:
 - Không có
- Tiền điều kiện:
 - Không có
- Hậu điều kiện:
 - Không có
- Các điều kiện mở rộng: Không có

3.2.3.6. Mô tả Usecase “Tìm kiếm xe”

Use case này cho phép khách hàng tìm kiếm xe liên quan đến dữ liệu nhập vào

- Luồng sự kiện:

+ Luồng cơ bản:

Use-case bắt đầu khi khách hàng nhập dữ liệu vào ô tìm kiếm và ấn vào nút tìm kiếm trên thanh menu. Hệ thống sẽ truy cập vào cơ sở dữ liệu lấy một số thông tin về các xe bao gồm tenXe, nhaSX, namSX, dongXe, giaTien từ bảng CARS và hiển thị danh sách xe có liên quan như tên xe, nhà sản xuất xe lên màn hình

+ Luồng rẽ nhánh:

Tại bất kỳ bước nào trong luồng cơ bản, nếu không kết nối được với CSDL thì hệ thống sẽ hiển thị một thông báo lỗi và use case kết thúc.

— Các yêu cầu đặc biệt:

Không có.

— Tiền điều kiện:

Không có.

— Hậu điều kiện:

Không có

— Điểm mở rộng:

Không có

3.2.3.7. Mô tả Usecase “Quản lý người dùng”

Cho phép quản trị viên quản lý người dùng.

— Luồng sự kiện:

+ Luồng cơ bản:

+ Use case này bắt đầu khi người quản trị bấm vào nút “Tài khoản” trên menu quản trị. Hệ thống lấy thông tin từ bảng users, roles gồm roles_id, name, phoneNumber, address, creatorday, date_of_birth rồi hiển thị danh sách tài khoản lên màn hình.

+ Thêm người dùng:

Người quản trị nhấn vào nút “Thêm tài khoản”: Hệ thống sẽ đưa ra một biểu mẫu gồm các thông tin như họ, tên, email, số điện thoại, mật khẩu, lựa chọn vai trò.

Người quản trị điền vào các trường và nhấn vào nút “Thêm” để tạo một tài khoản mới. Hệ thống sẽ thêm tài khoản này vào bảng NguoiDung và cập nhật lại danh sách người dùng lên màn hình.

+ Sửa tài khoản người dùng:

Người quản trị nhấn vào nút “Sửa” của một tài khoản. Hệ thống sẽ hiển thị biểu mẫu sửa thông tin tài khoản gồm các thông tin như họ, tên, email, số điện thoại, mật khẩu, lựa chọn vai trò.

Người quản trị nhập thông tin mới vào biểu mẫu và nhấn “Cập nhật”. Hệ thống sẽ sửa thông tin của tài khoản và hiển thị danh sách người dùng đã được cập nhật lên màn hình.

+ Xoá tài khoản:

Người quản trị nhấn vào nút “Xóa”. Hệ thống sẽ thông báo “Bạn có chắc muốn xóa tài khoản này ?” hiện lên.

Người quản trị nhấn “Đồng ý” trên thông báo hệ thống. Hệ thống sẽ xóa tài khoản này khỏi cơ sở dữ liệu và hiển thị danh sách tài khoản sau khi đã xóa.

+ Luồng rẽ nhánh:

Tại bất kì bước nào trong luồng cơ bản nếu không kết nối được với CSDL thì hệ thống sẽ hiển thị một thông báo lỗi và use case kết thúc

– Các yêu cầu đặc biệt:

Phải đăng nhập để thực hiện thao tác

– Tiền điều kiện:

Không có.

– Hậu điều kiện:

Không có

– Điểm mở rộng:

không có

3.2.3.8. Mô tả Usecase “Quản lý xe”

Cho phép quản trị viên quản lý xe

– Luồng sự kiện:

+ Luồng cơ bản:

Use case này bắt đầu khi người quản trị bấm vào nút “Tên Xe” trên menu quản trị. Hệ thống lấy thông tin từ bảng cars, manufactures, car_attribute, attributes, car_images, specifications gồm car_id, car_name, year_manufacture, price, model rồi hiển thị danh sách tài khoản lên màn hình.

1. Thêm xe:

Người quản trị nhấn vào nút “Thêm xe”: Hệ thống sẽ đưa ra một biểu mẫu gồm các thông tin như tên xe, năm sản xuất, giá, hãng sản xuất.

Người quản trị điền vào các trường và nhấn vào nút “Thêm” để tạo một tài khoản mới. Hệ thống sẽ thêm xe này vào bảng Car_database Cars và cập nhật lại danh sách người dùng lên màn hình.

2. Sửa xe:

Người quản trị nhấn vào nút “Sửa” của một tài khoản, Hệ thống sẽ hiển thị biểu mẫu sửa thông tin tài khoản gồm các thông tin tên xe, năm sản xuất, giá, hãng sản xuất.

Người quản trị nhập thông tin mới vào biểu mẫu và nhấn “Cập nhật”. Hệ thống sẽ sửa thông tin của tài khoản và hiển thị danh sách người dùng đã được cập nhật lên màn hình.

3. Xóa xe:

Người quản trị nhấn vào nút “Xóa”. Hệ thống sẽ thông báo “Bạn có chắc muốn xóa tài khoản này ?” hiện lên.

Người quản trị nhấn “Đồng ý” trên thông báo hệ thống. Hệ thống sẽ xóa tài khoản này khỏi cơ sở dữ liệu và hiển thị danh sách tài khoản sau khi đã xóa.

Use case kết thúc.

+ Luồng rẽ nhánh:

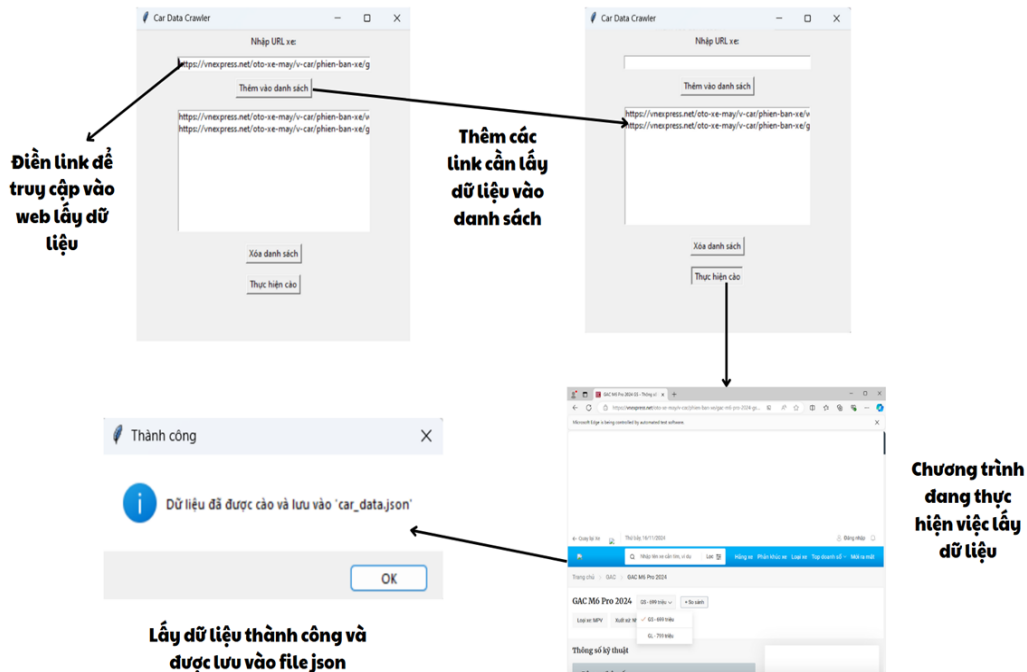
Tại bất kỳ bước nào trong luồng cơ bản, nếu không kết nối được với CSDL thì hệ thống sẽ hiển thị một thông báo lỗi và use case kết thúc.

- Các yêu cầu đặc biệt:
 - Phải đăng nhập để thực hiện thao tác
- Tiền điều kiện:
 - Không có.
- Hậu điều kiện:
 - Không có
- Điểm mở rộng:
 - không có

CHƯƠNG 4. CÀI ĐẶT CHƯƠNG TRÌNH

4.1. Kỹ thuật lấy dữ liệu demo cho chương trình

- Sử dụng Python, VS Code tạo một ứng dụng đơn giản để thực hiện việc cào dữ liệu.
- Demo việc cào dữ liệu:



- Dữ liệu từ file json vừa thực hiện thành công:

```
car_data.json
D: > PTHDV-16 > {} car_data.json > ...
1
2 {
3   "name": "Wuling Bingo",
4   "model": null,
5   "price": "",
6   "thumbnail": null,
7   "manufacturer": {
8     "name": "VinFast"
9   },
10  "year_manufacture": "2024",
11  "specifications": [
12    {
13      "name": "Động cơ/hộp số",
14      "attributes": [
15        {
16          "name": "Kiểu động cơ",
17          "value": "Mô-tơ điện"
18        },
19        {
20          "name": "Công suất mô-tơ điện (mã lực)",
21          "value": "67"
22        },
23        {
24          "name": "Mô-men xoắn mô-tơ điện (Nm)",
25          "value": "150"
26        },
27        {
28          "name": "Loại pin",
29          "value": "LFP"
30        },
31        {
32          "name": "Dung lượng pin (kWh)",
33          "value": "31,9"
34        },
35        {
36          "name": "Tầm hoạt động (km)",
37          "value": "333"
38        }
39      ]
40    }
41  ]
42 }
```

4.2. Cài đặt chương trình phía backend

4.2.1. Các kiến trúc và công nghệ sử dụng

— Docker

Docker là một nền tảng phần mềm giúp đóng gói các ứng dụng cùng với tất cả các phụ thuộc của chúng vào trong một container. Các container này có thể chạy trên bất kỳ hệ điều hành nào có Docker, giúp đảm bảo tính nhất quán và dễ dàng triển khai ứng dụng vào bất kỳ môi trường nào từ máy tính cá nhân đến server hoặc môi trường cloud. Docker giúp giải quyết vấn đề "works on my machine" bằng cách tạo ra các môi trường cô lập, dễ dàng quản lý và triển khai.

— Cloud run

Cloud Run là một dịch vụ của Google Cloud giúp triển khai ứng dụng container lên cloud mà không cần lo lắng về việc quản lý cơ sở hạ tầng. Cho phép chạy các ứng dụng hoặc API trong môi trường container mà không cần phải tạo và quản lý máy chủ, giúp việc triển khai trở nên đơn giản và tiết kiệm chi phí. Cloud Run sẽ tự động mở rộng hoặc thu nhỏ lượng tài nguyên tùy theo lưu lượng truy cập.

— Redis

Redis là một hệ thống lưu trữ dữ liệu dạng key-value trong bộ nhớ, được sử dụng rộng rãi trong việc xây dựng các ứng dụng web hiệu suất cao, hỗ trợ tính năng như caching, message brokering, và quản lý phiên (sessions). Redis nổi bật nhờ vào tốc độ đọc/ghi cực kỳ nhanh chóng nhờ vào việc lưu trữ dữ liệu trong bộ nhớ RAM thay vì trên đĩa cứng.

— Mysql

MySQL là một hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở, được sử dụng rộng rãi trong các ứng dụng web để lưu trữ và quản lý dữ liệu. MySQL hỗ trợ SQL để truy vấn và quản lý dữ liệu. Đây là một hệ cơ sở dữ liệu phổ biến vì tính ổn định, hiệu suất cao, và dễ sử dụng.

— Phpmyadmin

phpMyAdmin là một ứng dụng web mã nguồn mở viết bằng PHP, được sử dụng để quản lý cơ sở dữ liệu MySQL hoặc MariaDB thông qua giao diện đồ họa. Nó cung cấp các tính năng như tạo, xóa, sửa bảng, nhập và xuất dữ liệu, chạy truy vấn SQL, và nhiều tính năng khác, giúp người dùng dễ dàng quản lý cơ sở dữ liệu mà không cần phải sử dụng dòng lệnh.

— Java spring boot

Spring Boot là một framework mã nguồn mở của Java được thiết kế để phát triển các ứng dụng Java một cách nhanh chóng và dễ dàng. Spring Boot giúp cấu hình tự động, giảm thiểu mã nguồn cần phải viết và dễ dàng triển khai các ứng dụng vào môi trường

sản xuất. Nó hỗ trợ các ứng dụng RESTful API, microservices, và các ứng dụng web, đồng thời tích hợp tốt với các công nghệ khác như Docker, Redis và MySQL.

— IntelliJ IDEA Community Edition 2024.3

IntelliJ IDEA là một IDE (Integrated Development Environment) mạnh mẽ, được phát triển bởi JetBrains, dành cho các lập trình viên Java và nhiều ngôn ngữ khác. Phiên bản Community Edition là miễn phí và bao gồm các tính năng cơ bản để phát triển Java và các ngôn ngữ như Kotlin, Groovy, và Scala. IntelliJ IDEA hỗ trợ tích hợp với các công cụ khác như Docker và Spring Boot, cung cấp môi trường phát triển tối ưu cho các ứng dụng Java.

4.2.2. Kết quả đạt được

— API hiển thị tất cả ô tô ra màn hình

@GetMapping("")

```
public ResponseEntity < CarListResponseDTO > getAllCars(
    @RequestParam("page") int page,
    @RequestParam("limit") int limit
) {
    PageRequest pageRequest = PageRequest.of(
        page, limit,
        Sort.by("id").ascending()
    );
    Page < CarResponseDTO > carPage = iCarService.getAllCars(pageRequest);
    int totalPages = carPage.getTotalPages();
    List < CarResponseDTO > cars = carPage.getContent();
    return ResponseEntity.ok(CarListResponseDTO.builder()
        .cars(cars)
        .totalPage(totalPages)
        .build()
    );
}
```

— **API tìm kiếm ô tô theo tên hoặc model**

```
@GetMapping("searchCar")
public ResponseEntity < ? > searchCarByNameOrManufacturer(
    @RequestParam("keyword") String keyword,
    @RequestParam("page") int page,
    @RequestParam("limit") int limit,
    @RequestParam(defaultValue = "name") String sort,
    @RequestParam(defaultValue = "ASC") String direction
) {
    Sort.Direction sortDirection = direction.equalsIgnoreCase("DESC") ?
Sort.Direction.DESC : Sort.Direction.ASC;
    Sort sortBy = Sort.by(sortDirection, sort);
    Pageable pageable = PageRequest.of(
        page, limit, sortBy
    );
    return ResponseEntity.ok(iCarService.searchCars(keyword, pageable));
}
```

— **API thống kê lượt xem ô tô nhiều nhất**

```
@GetMapping("/top")
public ResponseEntity < ? > getTopCar() {
    return ResponseEntity.ok(iCarService.getTopCar());
}
```

— **API thêm, sửa, xóa ô tô**

```
@GetMapping("/{id}")
public ResponseEntity < CarDetailResponseDTO >
getCarDetail(@PathVariable("id") int id) {
    return ResponseEntity.ok(iCarService.getCarDetail(id));
}

@GetMapping("/list")
public ResponseEntity < List < CarDetailResponseDTO >>
getCarDetailList(@RequestBody List < Integer > ids) {
    return ResponseEntity.ok(iCarService.getCarDetailList(ids));
}
```



```

    @PostMapping("/add")
    public ResponseEntity < ? > addCar(@RequestBody AddCarRequestDTO
addCarRequestDTO) {
        return ResponseEntity.ok(iCarService.createCar(addCarRequestDTO));
    }

```

```

    @PostMapping("/addDetail")
    public ResponseEntity < ? > addCarDetail(@RequestBody
AddCarDetailsRequestDTO addCarDetailsRequestDTO) {
        return
ResponseEntity.ok(iCarService.createCarDetail(addCarDetailsRequestDTO));
    }

```

```

    @PutMapping("/update")
    public ResponseEntity < ? > updateCar(UpdateCarRequestDTO car) {
        return ResponseEntity.ok(iCarService.updateCar(car));
    }

```

```

    @DeleteMapping("/delete")
    public ResponseEntity < ? > deleteCar(int id) {
        iCarService.deleteCar(id);
        return ResponseEntity.ok("Deleted");
    }

```

— API đăng ký

```

    @PostMapping("/register")
    public ResponseEntity < ? > createUser(@Valid @RequestBody UserRequestDTO
userRequestDTO, BindingResult result) {
        try {
            if (result.hasErrors()) {
                List < String > errorMessage =
result.getAllErrors().stream().map(ObjectError::getDefaultMessage).toList();
                return ResponseEntity.badRequest().body(errorMessage.toString());
            }

```

```

if
(!userRequestDTO.password().equalsIgnoreCase(userRequestDTO.retypePassword
())) {
    return ResponseEntity.badRequest().body("Not match Password");
}
userService.createUser(userRequestDTO);
return ResponseEntity.ok("Register ok!");
} catch (Exception e) {
    return ResponseEntity.badRequest().body(e.getMessage());
}
}

```

— API đăng nhập

```

@PostMapping("/login")
public ResponseEntity < UserLoginResponseDTO > login(@Valid
@RequestBody UserLoginDTO userLoginDTO

) throws Exception {
    try {
        String token = userService.login(userLoginDTO.phoneNumber(),
userLoginDTO.password());
        UserLoginResponseDTO userLoginResponseDTO = new
UserLoginResponseDTO("Login Success", token);
        return ResponseEntity.ok().body(userLoginResponseDTO);

    } catch (Exception e) {
        return ResponseEntity.badRequest().body(new
UserLoginResponseDTO("Login Fail", e.getMessage()));
    }
}

```

— **API lấy tất cả danh sách người dùng**

@GetMapping("getAllUsers")

```
public ResponseEntity < ? > getAllUsers(HttpServletRequest request) {  
    try {  
        String token = request.getHeader("Authorization");  
        List < UserResponseDTO > userResponseDTOList =  
userService.getAllUsers(token);  
        return ResponseEntity.ok().body(userResponseDTOList);  
    } catch (Exception e) {  
        return ResponseEntity.badRequest().body(e.getMessage());  
    }  
}
```

— **API thêm, sửa, xóa người dùng**

@PostMapping("addUser")

```
public ResponseEntity < ? > addUser(@RequestBody UserRequestDTO  
userRequestDTO, HttpServletRequest request) throws Exception {  
    String token = request.getHeader("Authorization");  
    User user = userService.addUser(userRequestDTO, token);  
    UserResponseDTO userResponseDTO =  
UserMapper.INSTANCE.toUserResponseDTO(user);  
    if (user != null) {  
        return ResponseEntity.ok(userResponseDTO);  
    } else {  
        return ResponseEntity.badRequest().body("Loi");  
    }  
}
```

@PutMapping("updateUser")

```
public ResponseEntity < ? > updateUser(@RequestBody UpdateUserRequestDTO  
updateUserRequestDTO, HttpServletRequest request) throws Exception {  
    String token = request.getHeader("Authorization");  
    User user = userService.updateUser(updateUserRequestDTO, token);  
    if (user != null) {
```

```

        return ResponseEntity.ok(user);
    } else {
        return ResponseEntity.badRequest().body("Loi");
    }
}

@DeleteMapping("/delete")
public ResponseEntity < ? > deleteUser(@RequestParam String phoneNumber,
    HttpServletRequest request) {
    try {
        String token = request.getHeader("Authorization");
        userService.deleteByPhoneNumber(phoneNumber, token);
        return ResponseEntity.ok("Delete user success");
    } catch (Exception e) {
        return ResponseEntity.badRequest().body(e.getMessage());
    }
}
}
}

```

— API lấy hình ảnh mẫu xe

```

@GetMapping("/images/{imageName}")
public ResponseEntity < ? > getImage(@PathVariable("imageName") String
imageName) {
    try {
        java.nio.file.Path imagePath = Paths.get("uploads/" + imageName);
        UrlResource resource = new UrlResource(imagePath.toUri());
        if (resource.exists()) {
            return ResponseEntity.ok()
                .contentType(MediaType.IMAGE_JPEG)
                .body(resource);
        } else {
            return ResponseEntity.notFound().build();
        }
    } catch (Exception e) {
        return ResponseEntity.notFound().build();
    }
}
}

```

— **API cập nhật hình ảnh xe**

```
@PostMapping("/thumbnail/{id}")
public ResponseEntity < ? > updateThumbnail(
    @PathVariable("id") int id,
    @RequestParam("file") MultipartFile file
) {
    return ResponseEntity.ok(iCarService.updateThumbnail(id, file));
}
```

4.3. Cài đặt chương trình phía frontend

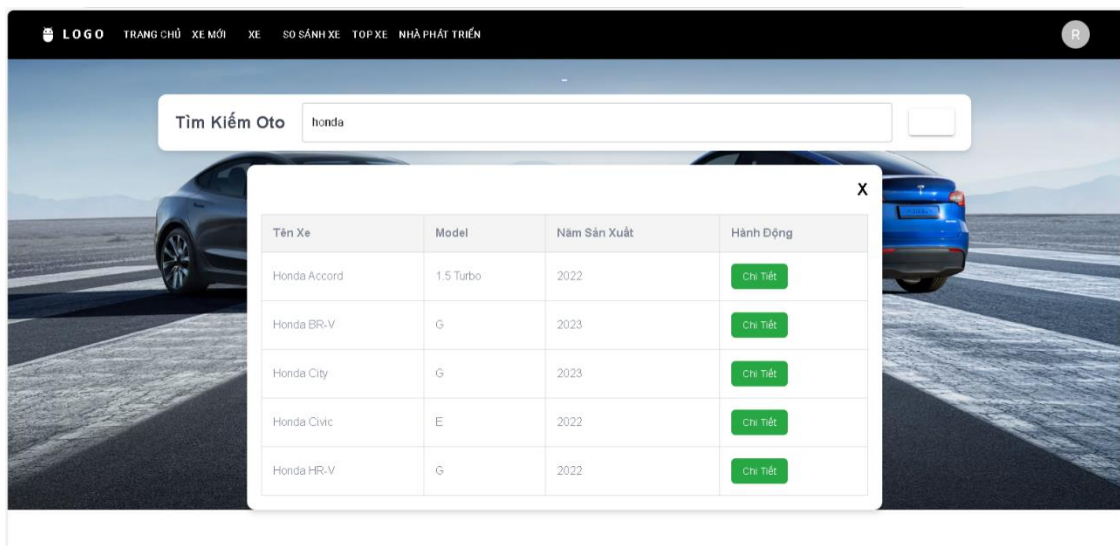
4.3.1. Công nghệ sử dụng

- **ReactJS:** Là thư viện JavaScript được sử dụng để xây dựng giao diện người dùng. ReactJS cho phép tạo ra các thành phần (components) dễ dàng tái sử dụng, giúp phát triển giao diện mượt mà và hiệu quả.
- **React Router:** Sử dụng để điều hướng giữa các trang khác nhau trong ứng dụng.
- **Axios:** Được sử dụng để gửi yêu cầu HTTP đến API để lấy dữ liệu (nếu bạn cần lấy dữ liệu từ một server hoặc từ một file JSON).
- **Styled-components, SCSS, CSS Modules:** Để tạo kiểu cho các thành phần giao diện.
- **Chart.js** là một thư viện JavaScript đơn giản và mạnh mẽ giúp tạo các biểu đồ động và tương tác trong các ứng dụng web. Nó hỗ trợ nhiều loại biểu đồ như đường, cột, tròn, radar, và phân tán. Chart.js dễ sử dụng, có thể tùy chỉnh cao và hoạt động tốt trên mọi trình duyệt hiện đại.

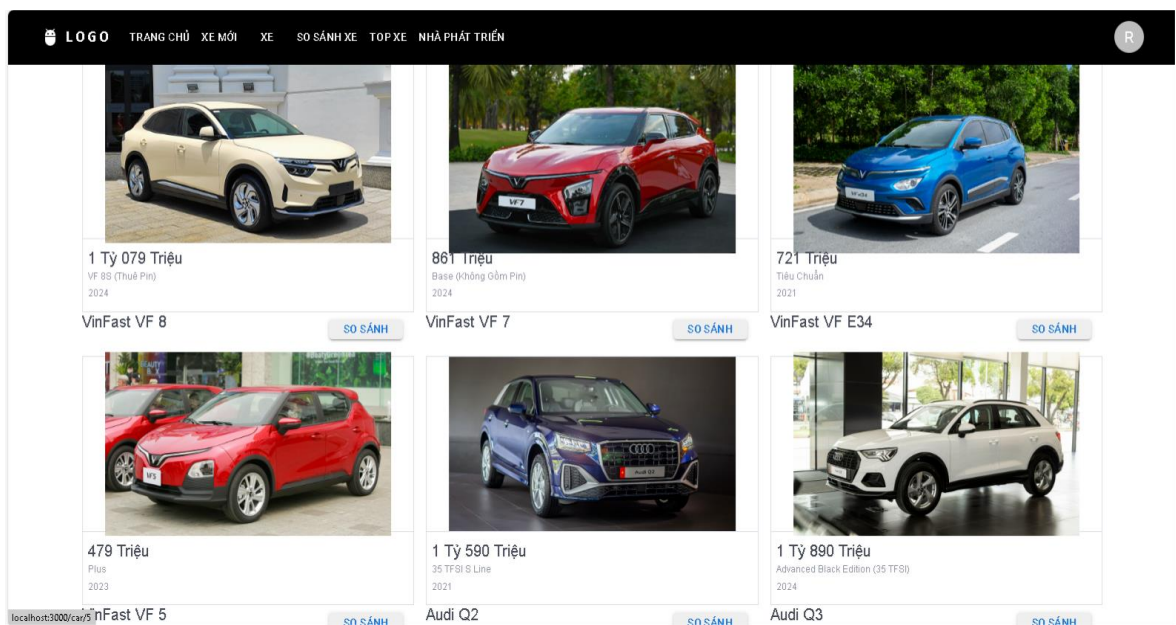
4.3.2. Kết quả đạt được

- **Giao diện người dùng**

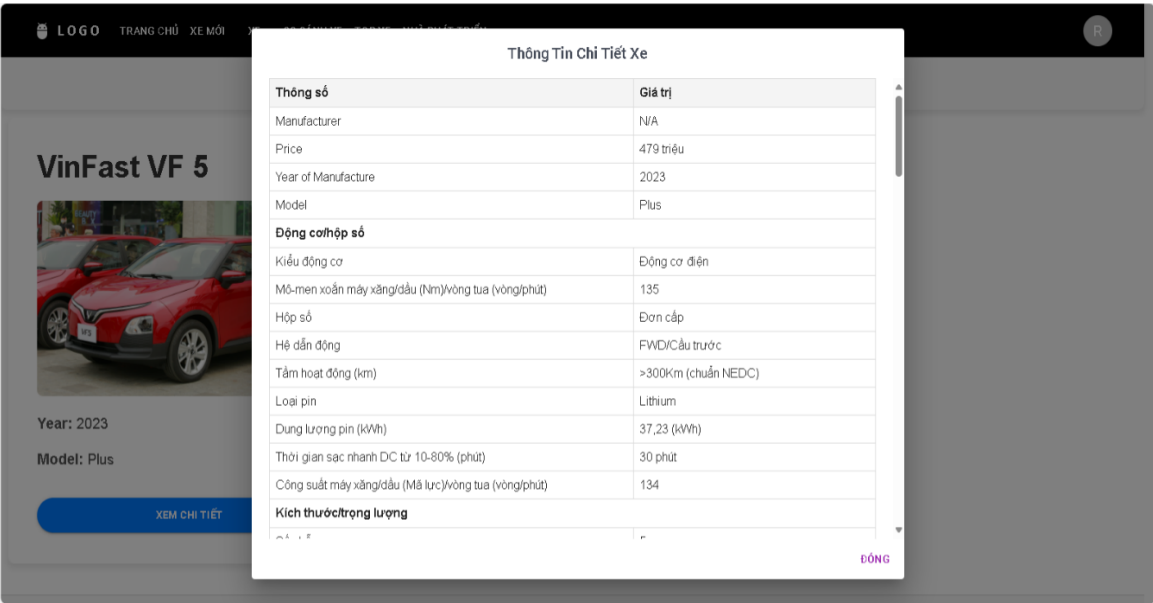
+ Giao diện tìm kiếm ô tô



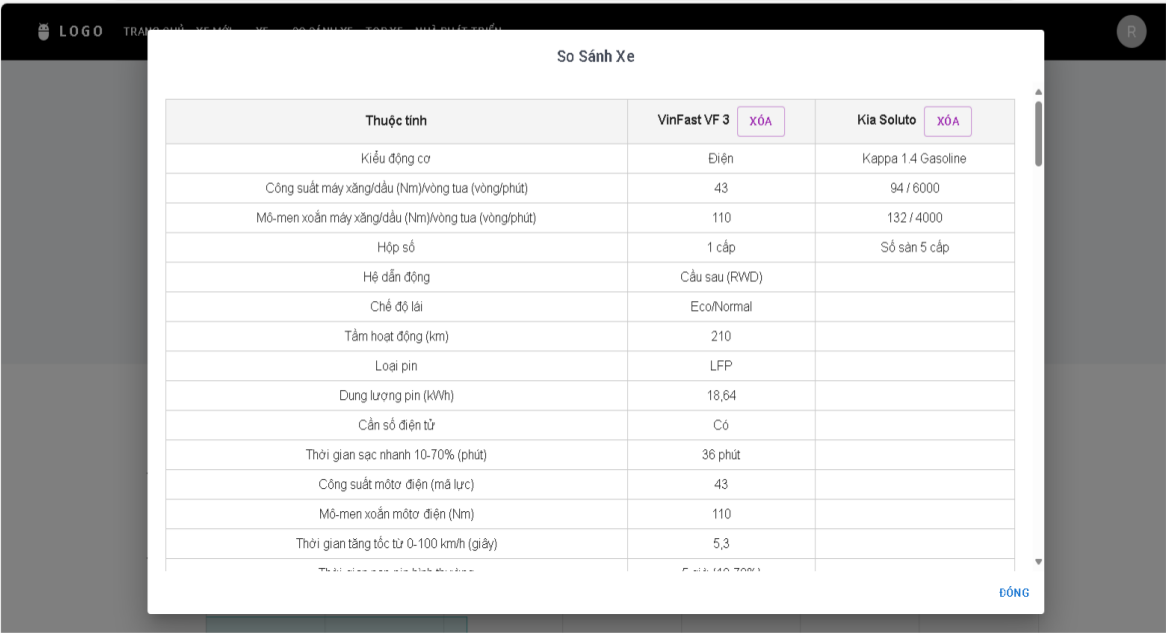
+ Giao diện hiện thị các ô tô



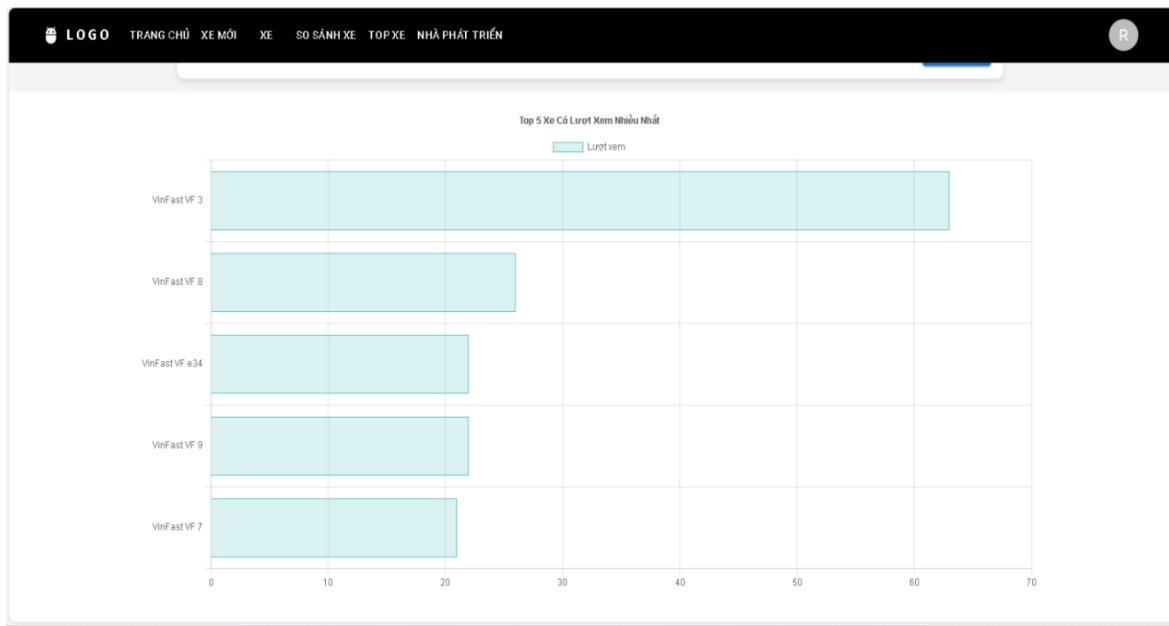
+ Giao diện xem chi tiết ô tô



+ Giao diện so sánh ô tô



+ Giao diện thống kê ô tô có lượt xem cao nhất



— **Giao diện nhà phát triển**

+ Giao diện đăng ký vào dashboard

Đăng Nhập

Số điện thoại

Mật khẩu

Đăng nhập

[Quên mật khẩu?](#)

+ Giao diện quản lý người dùng

The screenshot shows a web dashboard with a sidebar on the left containing navigation links: 'Bảng điều khiển' (selected), 'Người dùng', 'Ô tô', 'Tài liệu thêm', and 'Trang cá nhân'. The main content area is titled 'Danh sách người dùng' (User List) and includes a search bar at the top right with the text 'Tìm kiếm...'. Below the title is a table of users with columns for ID, Name, Status, Role, and Action buttons. The table contains 6 rows of user data. At the bottom right of the table is a pagination control showing pages 1, 2, 3, and 4.

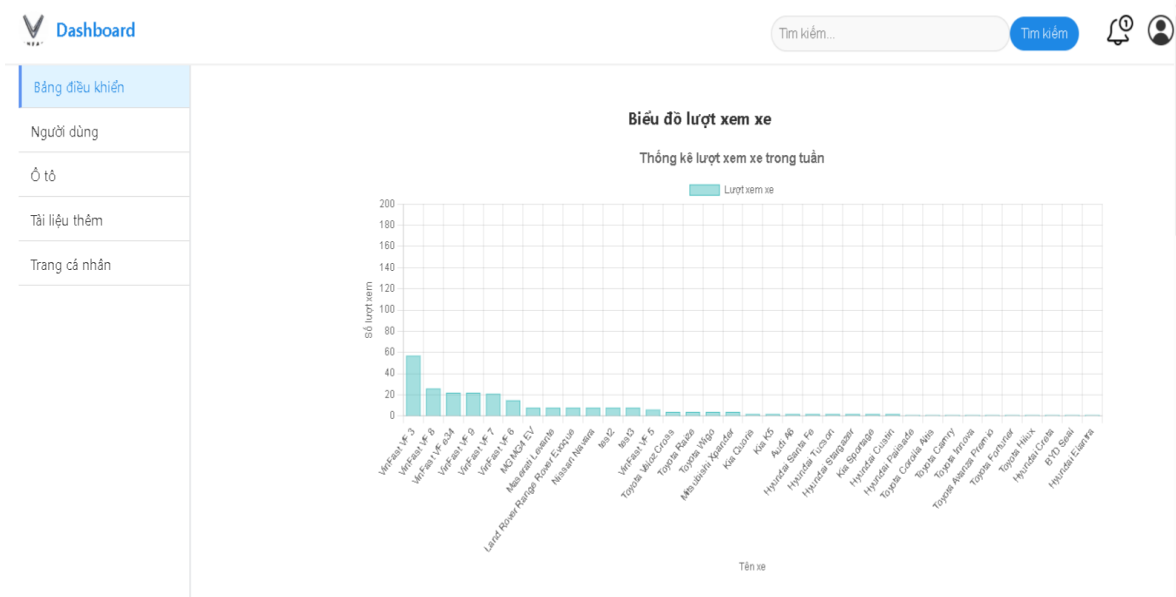
ID	Tên	Trạng thái	Vai trò	Hành động
1.	nguyen hoi - 123456	Online	USER	Sửa, Xóa
2.	Đào Xuân Giang - 12345678	Offline	USER	Sửa, Xóa
3.	Đào Xuân Giang - 1234567	Offline	USER	Sửa, Xóa
4.	Nguyễn lan - abc123 @#	Offline	ADMIN	Sửa, Xóa
5.	Nguyễn Đình Hội - 08124718	Offline	ADMIN	Sửa, Xóa
6.	Nguyễn Đình Hội1 - 0812471	Offline	ADMIN	Sửa, Xóa

+ Giao diện quản lý ô tô

The screenshot shows a web dashboard with a sidebar on the left containing navigation links: 'Bảng điều khiển', 'Người dùng', 'Ô tô' (selected), 'Tài liệu thêm', and 'Trang cá nhân'. The main content area is titled 'Danh sách ô tô' (Car List) and includes a search bar at the top right with the text 'Tìm kiếm...'. Below the title is a table of cars with columns for ID, Name, and Action buttons. The table contains 6 rows of car data. At the bottom right of the table is a pagination control showing pages 1, 2, 3, and 4.



ID	Tên	Hành động
1.	VinFast VF 5 - Plus	Xóa, Sửa, Chi Tiết
2.	Audi Q2 - 35 TFSI S Line	Xóa, Sửa, Chi Tiết
3.	Audi Q3 - Advanced Black Ed	Xóa, Sửa, Chi Tiết
4.	Audi A4 - 40 TFSI Advanced F	Xóa, Sửa, Chi Tiết
5.	Audi A6 - 45 TFSI	Xóa, Sửa, Chi Tiết
6.	Audi Q5 - 45 TFSI quattro S Li	Xóa, Sửa, Chi Tiết

+ Giao diện thống kê lượt xem



+ Giao diện các câu hỏi thường gặp

Dashboard

Tìm kiếm... [Tìm kiếm](#)  

Bảng điều khiển

- Người dùng
- Ô tô
- Tài liệu thêm**
- Trang cá nhân

Tài liệu tham khảo

Hướng dẫn và tài liệu liên quan đến hệ thống quản trị.

Tìm kiếm tài liệu...

[Tất cả](#) [Hướng dẫn sử dụng](#) [Quản lý ô tô](#) [Câu hỏi thường gặp](#) [Bảo trì hệ thống](#)

Hướng dẫn sử dụng hệ thống quản trị

Chỉ biết hướng dẫn sử dụng các chức năng cơ bản của hệ thống.

Cập nhật thông tin ô tô

Hướng dẫn chi tiết cách cập nhật thông tin ô tô trong hệ thống.

Câu hỏi thường gặp (FAQ)

Những câu hỏi thường gặp trong việc sử dụng hệ thống.

CHƯƠNG 5. KIỂM THỬ

5.1. Cơ sở lý thuyết

5.1.1. Khái niệm kiểm thử

- Theo Glenford Myers: Kiểm thử là quá trình vận hành chương trình để tìm ra lỗi
- Theo IEEE: Kiểm thử là:
 - + Là quá trình vận hành hệ thống hoặc thành phần dưới những điều kiện xác định, quan sát hoặc ghi nhận kết quả và đưa ra đánh giá về hệ thống hoặc thành phần đó.
 - + Là quá trình phân tích phần mềm để tìm ra sự khác biệt giữa điều kiện thực tế và điều kiện yêu cầu và dựa vào điểm khác biệt đó để đánh giá tính năng phần mềm

5.1.2. Các phương pháp kiểm thử

Có nhiều cách phân loại kiểm thử, thông thường các nhà phát triển phân loại kiểm thử dựa trên các yếu tố như: mục đích kiểm thử (test purpose), chiến lược kiểm thử (test strategy), phương pháp kiểm thử (test method), kỹ thuật kiểm thử (test technique) và loại kiểm thử (test type).

- Phân loại kiểm thử theo cấp độ (levels): đơn vị; kiểm thử tích hợp; hệ thống; chấp nhận.
- Kiểm thử chức năng: Kiểm thử chức năng theo các cấp độ ở các mức đơn vị; tích hợp; hệ thống; chấp nhận.
- Kiểm thử hồi quy.
- Kiểm thử hệ thống (chức năng/phi chức năng): Kiểm thử sơ lược (smoke testing), kiểm thử tải dữ liệu (load testing), kiểm thử tải trọng (stress testing), kiểm thử hiệu suất (performance testing), kiểm thử chấp nhận (User Acceptance Testing), kiểm thử bảo mật (security testing), kiểm thử cấu hình.
- Phân loại theo hình thức tiến hành: Kiểm thử thủ công và tự động. Kiểm thử thủ công - thực hiện kiểm thử mọi thứ bằng tay, từ viết TC đến thực hiện kiểm thử. Kiểm thử tự động thực hiện một cách tự động các bước trong kịch bản kiểm thử bằng cách dùng một công cụ trợ giúp.
- Phân loại dựa vào phương pháp tiến hành kiểm thử chia ra kiểm thử tĩnh và kiểm thử động.
 - + Kiểm thử tĩnh: Một hình thức của kiểm thử mà phần mềm không được sử dụng thực sự. Kiểm thử tĩnh thường không kiểm thử chi tiết mà chủ yếu kiểm tra tính đúng đắn của mã nguồn, thuật toán hoặc tài liệu. Các hoạt động trong kỹ thuật này như : Đi xuyên suốt (walk through), thanh tra (inspection)

- + Kiểm thử động: Một hình thức kiểm thử phần mềm chạy mã lập trình thực tế trong các tình huống, diễn ra khi bản thân chương trình đó đang được sử dụng. Kiểm thử động có thể bắt đầu trước khi chương trình đã hoàn tất.
- Phân loại dựa vào kỹ thuật kiểm thử chia ra thành kiểm thử hộp trắng và kiểm thử hộp đen.
 - + Kiểm thử hộp trắng: Kiểm thử theo góc nhìn thực hiện. Người thực hiện kiểm thử cần có kiến thức về chi tiết thiết kế và thực hiện bên trong chương trình. Kiểm tra phủ lệnh (phủ nhánh, phủ các điều kiện con).
 - + Kiểm thử hộp đen: Đây là phương pháp kiểm thử theo góc nhìn sử dụng. Kiểm thử dựa trên các yêu cầu và đặc tả phần phần mềm và không đòi hỏi kiến thức về chi tiết thiết kế và thực hiện ở bên trong chương trình.
- Phân loại theo loại kiểm thử (Type testing): kiểm thử chức năng, phi chức năng, cấu trúc và kiểm thử liên quan tới sự thay đổi.
 - + Phân loại kiểm thử dựa trên chức năng: kiểm thử đơn vị, smoke, giao diện, tích hợp, hệ thống, hồi quy, chấp nhận.
 - + Kiểm thử phi chức năng: kiểm thử hiệu năng (performance), chịu tải, áp lực, khả dụng, bảo trì, tin cậy.
 - + Kiểm thử cấu trúc: kiểm thử cấu trúc dòng lệnh.
 - + Kiểm thử liên quan đến thay đổi: kiểm thử xác nhận (alpha, beta); kiểm thử hồi quy

5.2. Các công cụ sử dụng

5.2.1. Postman

Một nền tảng cộng tác để phát triển API. Công cụ này là một ứng dụng khách API phổ biến; cho phép chúng ta thiết kế, xây dựng, chia sẻ, kiểm thử và ghi lại các API. Postman trước đây chỉ là một plugin của Chrome, bây giờ đã được mở rộng giải pháp với các phiên bản của cả Windows & Mac.

5.2.2. Jmeter

Được sử dụng để phân tích và đo lường hiệu năng của một dải các dịch vụ. JMeter chủ yếu được sử dụng trong các ứng dụng dịch vụ web và trực tuyến, phát triển trên nền tảng Java, là công cụ kiểm thử tải nguồn mở.

5.3. Kiểm thử phần mềm

5.3.1. Lập kế hoạch kiểm thử

Tên mục tiêu	Nội dung chi tiết
Kiểm tra tính đầy đủ của tất cả các chức năng	Trên nền tảng tư vấn thông số kỹ thuật ô tô, đảm bảo rằng mỗi trang và chức năng của hoạt động chính xác như yêu cầu của người dùng.
Kiểm thử giao diện người dùng (Frontend)	Đảm bảo tính tương thích giữa các trang, thao tác người dùng và hiển thị dữ liệu trên các thiết bị khác nhau.
Kiểm thử các API (Backend)	Đảm bảo các chức năng như đăng nhập, tìm kiếm xe, so sánh xehoạt động đúng với dữ liệu từ cơ sở dữ liệu và các yêu cầu hệ thống.
Kiểm tra sự tương tác giữa Frontend và Backend	Xác nhận rằng các API trả về kết quả chính xác và có thể xử lý yêu cầu của người dùng.
Kiểm thử hiệu năng	Kiểm tra khả năng hệ thống đáp ứng các yêu cầu về hiệu suất khi có tải lớn. Đảm bảo các API và giao diện người dùng có thể xử lý số lượng lớn người dùng đồng thời mà không gặp sự cố.

— Phương pháp kiểm thử

Phương pháp kiểm thử	Mục tiêu	Các thao tác kiểm thử/Công cụ sử dụng
Kiểm Thử Thủ Công (Manual Testing)	Đảm bảo rằng các chức năng cơ bản hoạt động đúng như kỳ vọng khi người dùng thực hiện thao tác trực tiếp.	Tìm kiếm xe, xem chi tiết xe, so sánh xe, quản lý người dùng, quản lý xe....
Kiểm Thử Hiệu Năng (Performance Testing)	Đảm bảo hệ thống có thể chịu được tải lớn và hoạt động mượt mà khi có nhiều người dùng đồng thời.	Sử dụng công cụ JMeter để kiểm thử hiệu suất của hệ thống

5.3.2. Phân tích thiết kế kiểm thử

Phân tích thiết kế kiểm thử là quá trình xác định các yêu cầu và phương pháp kiểm thử chi tiết cho từng chức năng của hệ thống.

— **Mục tiêu phân tích thiết kế kiểm thử:**

- + Xác định các phương thức kiểm thử cho từng chức năng.
- + Lập kế hoạch cho kịch bản kiểm thử.
- + Chỉ ra các yếu tố cần kiểm tra như các tương tác giữa frontend và backend, giao diện người dùng, các API, bảo mật

— **Giao diện và các chức năng chính:**

Để kiểm thử hệ thống, chúng ta sẽ phân chia các chức năng chính theo từng trang web và các API backend.

— **Phương pháp kiểm thử nhóm thực hiện**

Tên phương pháp	Mục tiêu
Kiểm thử hộp đen (Black-box testing)	Đánh giá tính đúng đắn của chức năng mà không cần biết mã nguồn.
Kiểm thử giao diện người dùng (UI testing)	Đảm bảo giao diện người dùng dễ sử dụng, các nút, form nhập liệu hoạt động chính xác.
Kiểm thử tích hợp (Integration testing)	Kiểm tra sự tương tác giữa các module hệ thống, ví dụ: frontend với backend.
Kiểm thử bảo mật (Security testing)	Đảm bảo rằng thông tin người dùng như mật khẩu được bảo mật
Kiểm thử hiệu năng (Performance Testing)	Đảm bảo hệ thống hoạt động ổn định, nhanh và có thể chịu tải cao mà không gặp vấn đề về hiệu suất.

5.3.3. Thực hiện kiểm thử

a. Kiểm thử thủ công

— Kiểm thử giao diện người dùng (UI)

+ Mục tiêu:

Kiểm tra tất cả các tính năng giao diện người dùng, bao gồm các button, form, thông báo lỗi,... để đảm bảo rằng giao diện hoạt động như yêu cầu và đáp ứng đúng với kỳ vọng của người dùng.

+ Các bước kiểm thử thủ công:

- Thực hiện các thao tác người dùng: Tương tác với tất cả các tính năng giao diện của ứng dụng, như nhập dữ liệu vào form, nhấn các button, thay xem các thông báo lỗi, và kiểm tra sự hiển thị thông tin chính xác.
- Thao tác chính: Tìm kiếm xe, so sánh xe, xem chi tiết 1 xe, xem xe...
- Kiểm tra tính năng: Kiểm tra tất cả các tính năng có đúng như yêu cầu hay không (ví dụ, thông báo lỗi xuất hiện khi người dùng nhập sai thông tin).
- Tạo bảng thống kê lỗi: Ghi lại tất cả các vấn đề hoặc lỗi phát hiện trong quá trình kiểm thử. Cung cấp các chi tiết như mô tả lỗi, ảnh chụp màn hình, các bước tái tạo lỗi....

— Kiểm thử API

+ Mục tiêu Kiểm thử API:

- Đảm bảo API hoạt động đúng theo yêu cầu, xử lý đúng các dữ liệu và trả về kết quả chính xác.
- Kiểm tra khả năng bảo mật, hiệu suất, và khả năng xử lý lỗi của API.

+ Quy trình Kiểm thử API:

- Chuẩn bị kiểm thử:

Xác định API cần kiểm thử và chuẩn bị công cụ (Postman, cURL, v.v.).

Đảm bảo môi trường kiểm thử sẵn sàng.

- Thực hiện kiểm thử:

Gửi yêu cầu: Sử dụng các phương thức HTTP (GET, POST, PUT, DELETE) để gửi yêu cầu.

Kiểm tra phản hồi: Xác minh mã trạng thái, định dạng dữ liệu, và nội dung trả về.

Kiểm thử các trường hợp lỗi: Gửi yêu cầu với dữ liệu không hợp lệ hoặc thiếu thông tin.

Kiểm tra bảo mật và phân quyền: Đảm bảo API chỉ cho phép truy cập hợp lệ.

- Ghi nhận và báo cáo:

Ghi lại các lỗi phát hiện, mô tả lỗi và các bước tái tạo.

Tạo bảng thống kê lỗi và báo cáo kết quả kiểm thử.

b. Kiểm thử hiệu năng

— Mục tiêu kiểm thử hiệu năng:

Mục tiêu của kiểm thử hiệu năng là xác định khả năng chịu tải và độ ổn định của hệ thống khi có nhiều người dùng cùng lúc, và đánh giá hiệu suất của hệ thống dưới các điều kiện khác nhau.

— Quy trình kiểm thử hiệu năng:

- + Lập kế hoạch kiểm thử hiệu năng: Xác định các chỉ tiêu hiệu năng cần kiểm tra (thời gian phản hồi tối đa, số lượng người dùng đồng thời tối đa,...).
- + Chạy thử nghiệm: Sử dụng công cụ JMeter để thực hiện các kiểm thử hiệu năng.
- + Phân tích kết quả: Đo lường và phân tích các chỉ số hiệu năng như thời gian phản hồi, tỷ lệ thành công, độ trễ,... để đánh giá khả năng chịu tải của hệ thống.

5.3.4. Báo cáo kiểm thử

5.3.4.1. Tổng quan kiểm thử

— Mục tiêu và phạm vi nghiên cứu

+ Mục tiêu kiểm thử nhằm đảm bảo rằng hệ thống hoạt động ổn định, các chức năng chính được triển khai đúng theo yêu cầu và không xảy ra lỗi nghiêm trọng ảnh hưởng đến trải nghiệm người dùng. Đảm bảo hệ thống hoạt động hiệu quả và đáp ứng các yêu cầu về tính năng và hiệu suất.

+ Các chức năng được kiểm thử bao gồm: đăng ký, đăng nhập, tìm kiếm xe, xem chi tiết 1 xe, so sánh xe, thêm xe, sửa xe, xoá xe, thêm người dùng, sửa thông tin người dùng, xoá người dùng.

— Các loại kiểm thử đã thực hiện:

+ Kiểm thử giao diện người dùng (UI): Đảm bảo các chức năng trên giao diện hoạt động đúng và giao diện hiển thị thân thiện với người dùng.

+ Kiểm thử API: Kiểm tra tính chính xác của các API

+ Kiểm thử hiệu năng: Đảm bảo hệ thống có thể xử lý lượng người dùng lớn và duy trì hiệu suất ổn định dưới tải cao. Kiểm tra thời gian phản hồi, khả năng chịu tải và khả năng mở rộng.

5.3.4.2. Chi tiết kết quả kiểm thử

+ Kết quả kiểm thử API

+ Chức năng đăng ký

+ Test Case	
Test Priority: High	Test Designed by: Lê Thị Ngọc Lan
Module Name: Test User	Test Designed date: 10/12/2024
Test Title: Đăng ký	Test Executed by: Lê Thị Ngọc Lan
Description: Đăng ký vào dashboard	Test Executed date: 10/12/2024

STT	Tên Test Case	Mô tả	Dữ liệu gửi	Kết quả mong đợi	Kết quả thực tế	Kết quả test Pass/ Fail
1	Đăng ký thành công	Người dùng nhập đúng các thông tin để đăng ký	"fullname": "Lanle165", "phone_number": "0379998771", "address": "admin", "password": "123456", "retype_password": "123456", "date_of_birth": "2004-05-16", "facebook_account_id": "0", "google_account_id": "0", "role_id": 1	200OK: Trả về thông báo "register ok!"	Như kết quả mong đợi	Pass
2	Mật khẩu và mật khẩu xác nhận không trùng khớp	Người dùng nhập mật khẩu không trùng khớp	"fullname": "Lanle165", "phone_number": "0379998771", "address": "admin", "password": "123456", "retype_password": "123456789", "date_of_birth": "2004-05-16", "facebook_account_id": "0", "google_account_id": "0", "role_id": 1	400 Bad request	400 Bad request	fail

3	Số điện thoại đã tồn tại	Người dùng đăng ký tài khoản với sdt đã tồn tại	"fullname": "Lanle165", "phone_number": "0379998771", "address": "admin", "password": "123456", "retype_password": "123456", "date_of_birth": "2004-05-16", "facebook_account_id": "0", "google_account_id": "0", "role_id": 1	400 Bad request: Thông báo "Phone number already exists"	Như kết quả mong đợi	Pass
4	Sai định dạng ngày sinh	Người dùng nhập sai định dạng ngày sinh	"fullname": "Lanle165", "phone_number": "0379998771", "address": "admin", "password": "123456", "retype_password": "123456", "date_of_birth": "20-05-16", "facebook_account_id": "0", "google_account_id": "0", "role_id": 1	400 Bad request	Như kết quả mong đợi	Pass

+ Chức năng đăng nhập

+ Test Case	
Test Priority: High	Test Designed by: Lê Thị Ngọc Lan
Module Name: Test User	Test Designed date: 10/12/2024
Test Title: Đăng nhập	Test Executed by: Lê Thị Ngọc Lan
Description: Đăng nhập vào trang web	Test Executed date: 10/12/2024

STT	Tên Test Case	Mô tả	Dữ liệu gửi	Kết quả mong đợi	Kết quả thực tế	Kết quả test Pass/ Fail
-----	---------------	-------	-------------	------------------	-----------------	-------------------------------

1	Đăng nhập thành công	Người dùng nhập đúng số điện thoại và mật khẩu	{ "phone_number": "abc123!!@#", "password": "abc123!!@#" }	200 OK: Trả về token phiên làm việc và thông báo thành công.	Như kết quả mong đợi	pass
2	Sai mật khẩu	Người dùng nhập số điện thoại đúng nhưng mật khẩu sai	{ "phone_number": "abc123!!@#", "password": "abc123" }	400 Bad Request: Thông báo "Login Fail".	Như kết quả mong đợi	pass
3	Số điện thoại không tồn tại	Người dùng nhập số điện thoại không tồn tại trong hệ thống	{ "phone_number": "0123456", "password": "abc123" }	400 Bad request Thông báo "Login fail"	Như kết quả mong đợi	pass
4	Thiếu trường số điện thoại	Người dùng không gửi trường số điện thoại trong request body	{ "password": "123" }	400 Bad request Thông báo "Login fail"	Như kết quả mong đợi	pass
5	Thiếu trường password	Người dùng không gửi trường password trong request body	{ "phone_number": "0123456", "password": "" }	400 Bad request Thông báo "Login fail"	Như kết quả mong đợi	pass
6	Thiếu cả 2 trường số điện thoại và password	Người dùng không gửi trường số điện thoại và password trong request body	{ "password": "" }	400 Bad request Thông báo "Login fail"	400 Bad request Thông báo "Login fail"	pass

+ Chức năng tìm kiếm xe

+ Test Case	
Test Priority: High	Test Designed by: Lê Thị Ngọc Lan

Module Name: Test Car	Test Designed date: 10/12/2024
Test Title: Tìm kiếm xe	Test Executed by: Lê Thị Ngọc Lan
Description: Tìm kiếm xe	Test Executed date: 10/12/2024

STT	Tên Test Case	Mô tả	Dữ liệu gửi	Kết quả mong đợi	Kết quả thực tế	Kết quả test Pass/ Fail
1	Tìm kiếm xe thành công	Người dùng nhập tên các key và value cần thiết	Params với các key và value: “page-0”, “limit-10”, “keyword-Vinfast”	200OK: trả về file json gồm các xe có tên như đã nhập	Như kết quả mong đợi	Pass
2	Tìm kiếm xe không thành công	Người dùng nhập thiếu các key và value	Params không chứa dữ liệu nào	400 Bad request: Thông báo “tìm kiếm thất bại”	400 bad request	Fail

+ Chức năng lấy thông tin các xe

+ Test Case	
Test Priority: High	Test Designed by: Lê Thị Ngọc Lan
Module Name: Test Car	Test Designed date: 10/12/2024
Test Title: Lấy thông tin các xe	Test Executed by: Lê Thị Ngọc Lan
Description: Lấy thông tin các mẫu xe	Test Executed date: 10/12/2024

ST T	Tên Test Case	Mô tả	Dữ liệu gửi	Kết quả mong đợi	Kết quả thực tế	Kết quả test Pass / Fail
1	Lấy danh sách các mẫu xe thành công	Gửi request với tham số page=0 và limit=10 để lấy thông tin các mẫu xe	Method: GET URL: {{ API_PREFIX }}/car?page=0&limit=10	Mã trạng thái: 200 OK Trả về JSON danh sách các mẫu xe	Như kết quả mong đợi	Pass
2	Lỗi do tham số không hợp lệ	Lỗi do tham số không hợp lệ	Method: GET URL: {{ API_PREFIX }}/car?page=0&limit=abc	Mã trạng thái: 400 Bad Request	Như kết quả mong đợi	Pass
3	Lỗi khi không cung cấp tham số bắt buộc	Không gửi tham số page hoặc limit	Method: GET URL: {{ API_PREFIX }}/car	Mã trạng thái: 400 Bad Request Trả về JSON thông báo lỗi: - error: "Missing required parameters: page or limit."	Mã trạng thái: 400 Bad Request	Fail

+ Chức năng xem chi tiết xe

+ Test Case	
Test Priority: High	Test Designed by: Lê Thị Ngọc Lan
Module Name: Test Car	Test Designed date: 10/12/2024
Test Title: Xem chi tiết xe	Test Executed by: Lê Thị Ngọc Lan
Description: Xem thông tin chi tiết của 1 xe	Test Executed date: 10/12/2024

STT	Tên Test Case	Mô tả	Dữ liệu gửi	Kết quả mong đợi	Kết quả thực tế	Kết quả test Pass/ Fail
1	Xem chi tiết 1 xe thành công	Người dùng nhập id của để thực hiện việc xem thông tin chi tiết xe	“id=1”	200OK: trả về file JSON gồm các thông tin chi tiết của xe như thông số kỹ thuật, giá cả....	Như kết quả mong đợi	Pass
2	Xem chi tiết 1 xe không thành công	Người dùng không nhập vào id xe	“id=”	400 Bad request	400 Bad request	Pass

+ Chức năng thêm xe

+ Test Case	
Test Priority: High	Test Designed by: Lê Thị Ngọc Lan
Module Name: Test Car	Test Designed date: 10/12/2024
Test Title: Thêm xe	Test Executed by: Lê Thị Ngọc Lan

Description: Thêm xe	Test Executed date: 10/12/2024
----------------------	--------------------------------

ST T	Tên Test Case	Mô tả	Dữ liệu gửi	Kết quả mong đợi	Kết quả thực tế	Kết quả test Pass / Fail
1	Thêm xe thành công	Thêm xe thành công vào trong danh sách	Token và các thông tin của xe : { "name" : "KiemThu1", "year_manufacture" : 2022, "model" : "Ban tai", "price" : "100 triệu", "manufacturer" : { "id" : 1, "name" : "VinFast" } }	200OK : Trả về một file JSON với thông tin của xe mới thêm	Như kết quả mong đợi	Pass
2	Xoá xe không thành công do token	Token không hợp lệ	1 token không hợp lệ: eyJhbGciOiJIUzI1NiJ9.eyJwaG9uZU51bWJlI6ImFiYzEyMyFAIyIsInN1YiI6ImFiYzEyMyFAIyIsImV4cCI6MTczNzc4NjQyMX0.0u svqqILPXOJBxhVbnu7_0eZUThDgR	400 Bad request	Như kết quả mong đợi	Pass

+ Chức năng xoá xe

+ Test Case	
Test Priority: High	Test Designed by: Lê Thị Ngọc Lan
Module Name: Test Car	Test Designed date: 10/12/2024
Test Title: Xoá xe	Test Executed by: Lê Thị Ngọc Lan

Description: Xoá xe	Test Executed date: 10/12/2024
---------------------	--------------------------------

STT	Tên Test Case	Mô tả	Dữ liệu gửi	Kết quả mong đợi	Kết quả thực tế	Kết quả test Pass/ Fail
1	Xoá xe thành công	Xoá xe đã chọn ra khỏi ds	1 token xác nhận phân quyền	200OK: Thông báo “Xoá xe thành công”	200OK: “Deleted”	Fail
2	Xoá xe không thành công	Token không hợp lệ	1 token không hợp lệ	400 Bad request: Thông báo “Token không hợp lệ”	400 Bad request	Fail

+ Chức năng lấy top xe có lượt xem nhiều nhất

+ Test Case	
Test Priority: High	Test Designed by: Lê Thị Ngọc Lan
Module Name: Test Car	Test Designed date: 10/12/2024
Test Title: Top xe	Test Executed by: Lê Thị Ngọc Lan
Description: Lấy ra top xe có lượt xem nhiều nhất	Test Executed date: 10/12/2024

STT	Tên Test Case	Mô tả	Dữ liệu gửi	Kết quả mong đợi	Kết quả thực tế	Kết quả test Pass/ Fail
-----	---------------	-------	-------------	------------------	-----------------	-------------------------------

1	Hiển thị top xe thành công	Hiển thị top xe có lượt xem nhiều nhất thành công	Không yêu cầu đầu vào	200OK: trả về file JSON danh sách các mẫu xe có lượt xem nhiều nhất	Như kết quả mong đợi	Pass
2	Hiển thị top xe thất bại	Hiển thị top xe thất bại do sai đường dẫn API	Không yêu cầu	400 Bad request	Như kết quả mong đợi	Pass

+ Chức năng thêm người dùng

+ Test Case	
Test Priority: High	Test Designed by: Lê Thị Ngọc Lan
Module Name: Test User	Test Designed date: 10/12/2024
Test Title: Thêm người dùng	Test Executed by: Lê Thị Ngọc Lan
Description: Thêm thông tin người dùng vào trong danh sách	Test Executed date: 10/12/2024

STT	Tên Test Case	Mô tả	Dữ liệu gửi	Kết quả mong đợi	Kết quả thực tế	Kết quả test Pass/ Fail
1	Thêm người dùng thành công	Thêm thông tin người dùng thành công	"fullname": "Lanle160504", "phone_number": "037909156", "address": "admin", "password": "123456", "retype_password": "123456", "date_of_birth": "2004-05-16", "facebook_account_id": "0", "google_account_id": "0", "role_id": 1	200OK: “Người dùng được thêm thành công”	200OK: Hiển thị thông tin người dùng mới	Fail

2	Thêm người dùng không thành công	Thêm thông tin người dùng không thành công do số điện thoại đã tồn tại	"fullname": "Lanle160504", "phone_number": "037909156", "address": "admin", "password": "123456", "retype_password": "123456", "date_of_birth": "2004-05-16", "facebook_account_id": "0", "google_account_id": "0", "role_id": 1	400 Bad request	Như kết quả mong đợi	pass
---	----------------------------------	--	--	-----------------	----------------------	------

+ Chức năng xoá người dùng

+ Test Case	
Test Priority: High	Test Designed by: Lê Thị Ngọc Lan
Module Name: Test User	Test Designed date: 10/12/2024
Test Title: Xoá người dùng	Test Executed by: Lê Thị Ngọc Lan
Description: Xoá thông tin người dùng ra khỏi danh sách	Test Executed date: 10/12/2024

STT	Tên Test Case	Mô tả	Dữ liệu gửi	Kết quả mong đợi	Kết quả thực tế	Kết quả test Pass/ Fail
1	Xoá người dùng thành công	Xoá thông tin người dùng thành công ra khỏi danh sách	"phone_number=12341234"	200OK: Thông báo "Delete user success"	Như kết quả mong đợi	Pass

2	Xoá người dùng không thành công	Xoá thông tin người dùng không thành công do thiếu sdt	“phone_number=”	400 Bad request	Như kết quả mong đợi	Pass
---	---------------------------------	--	-----------------	-----------------	----------------------	------

+ Chức năng sửa thông tin người dùng

+ Test Case	
Test Priority: High	Test Designed by: Lê Thị Ngọc Lan
Module Name: Test User	Test Designed date: 10/12/2024
Test Title: Sửa người dùng	Test Executed by: Lê Thị Ngọc Lan
Description: Sửa thông tin người dùng	Test Executed date: 10/12/2024

STT	Tên Test Case	Mô tả	Dữ liệu gửi	Kết quả mong đợi	Kết quả thực tế	Kết quả test Pass/ Fail
1	Sửa thông tin người dùng thành công	Sửa thông tin người dùng với các dữ liệu nhập vào thành công	"fullname": "Lanle165", "phone_number": "0379998771", "address": "admin", "password": "Lanle2004", "retype_password": "Lanle2004", "date_of_birth": "2004-05-16", "facebook_account_id": "0", "google_account_id": "0", "role_id": 1	200 OK: Sửa thành công và trả về file JSON với thông tin người dùng đã sửa	Như kết quả mong đợi	Pass
2	Sửa thông tin không	Sửa thông tin không thành công do	"fullname": "Lanle165", "phone_number": "0379998771", "address": "admin", "password": "Lanle2004", "retype_password": "123456", "date_of_birth": "2004-05-16",	400 Bad request	Như kết quả mong đợi	Pass

	thành công	xác nhận mật khẩu không trùng khớp	"facebook_account_id": "0", "google_account_id": "0", "role_id": 1			
--	------------	------------------------------------	--	--	--	--

+ Chức năng hiển thị ảnh

+ Test Case	
Test Priority: High	Test Designed by: Lê Thị Ngọc Lan
Module Name: Test Car	Test Designed date: 10/12/2024
Test Title: Hiển thị ảnh xe	Test Executed by: Lê Thị Ngọc Lan
Description: Hiển thị ảnh xe ra	Test Executed date: 10/12/2024

STT	Tên Test Case	Mô tả	Dữ liệu gửi	Kết quả mong đợi	Kết quả thực tế	Kết quả test Pass/ Fail
1	Hiển thị ảnh xe thành công	Hiển thị ra màn hình hình ảnh xe	Đường dẫn của ảnh xe	200OK: Hiển thị ảnh xe đúng	Như kết quả mong đợi	Pass
2	Hiển thị ảnh xe không thành công	Hiển thị ảnh xe không thành công do sai đường dẫn	Đường dẫn sai	400 Bad request	Như kết quả mong đợi	Pass

+ Chức năng cập nhật hình ảnh xe

+ Test Case	
Test Priority: High	Test Designed by: Lê Thị Ngọc Lan
Module Name: Test Car	Test Designed date: 10/12/2024
Test Title: Hiển thị ảnh xe	Test Executed by: Lê Thị Ngọc Lan
Description: Hiển thị ảnh xe ra	Test Executed date: 10/12/2024

STT	Tên Test Case	Mô tả	Dữ liệu gửi	Kết quả mong đợi	Kết quả thực tế	Kết quả test Pass/ Fail
1	Cập nhật hình ảnh xe thành công	Kiểm tra chức năng cập nhật hình ảnh xe với dữ liệu hợp lệ.	Endpoint: /car/thumbnail/1 Tập hình ảnh: VinFast-VF-6-2023-2933.jpg Token hợp lệ	200OK và trả về file JSON thông tin ảnh mới cập nhật	Như kết quả mong đợi	Pass
2	Cập nhật hình ảnh với token không hợp lệ	Kiểm tra chức năng cập nhật hình ảnh xe khi sử dụng token không hợp lệ.	Endpoint: /car/thumbnail/1 Tập hình ảnh: VinFast-VF-6-2023-2933.jpg Token không hợp lệ	Trả về mã lỗi 401 Unauthorized	Như kết quả mong đợi	Pass
3	Cập nhật hình ảnh với tập không hợp lệ	Kiểm tra chức năng cập nhật hình ảnh xe khi gửi tập không phải hình ảnh.	Endpoint: /car/thumbnail/1 Tập đính kèm: test.txt	Trả về mã lỗi 400 Bad Request.	Như kết quả mong đợi	Pass

— **Kết quả kiểm thử giao diện người dùng**

STT	Mô tả kiểm thử	Yêu cầu kiểm thử	Thực hiện	Kết quả	Ghi chú
1	Kiểm tra nút "Tìm xe" trên trang chủ	Nút tìm kiếm phải hoạt động chính xác, cho phép tìm xe theo yêu cầu của người dùng	Kiểm tra thủ công trên trang chủ	Hiện thị trang tìm kiếm xe tốt	Không phát sinh lỗi
2	Kiểm tra tính năng "Xem chi tiết xe"	Khi người click vào “xem chi tiết xe” hệ thống phải hiển thị ra màn hình các thông số kỹ thuật, chi tiết của xe đó	Kiểm tra thủ công trên trang xem chi tiết xe	Hiện thị chi tiết xe thành công	Không phát sinh lỗi
3	Kiểm tra tính tương thích giao diện trên các thiết bị di động	Giao diện phải hiển thị chính xác trên các màn hình di động (kích thước màn hình khác nhau)	Kiểm tra trên các thiết bị di động (iPhone, Android)	Chưa tích hợp giao diện responsive nên hiển thị lỗi, chưa được tốt	
4	Kiểm tra khả năng hiển thị các thông báo lỗi	Khi người dùng nhập sai thông tin, hệ thống phải hiển thị thông báo lỗi rõ ràng và dễ hiểu	Kiểm tra thủ công khi nhập sai thông tin	Thành công, thông báo lỗi hiển thị đúng, dễ hiểu	Không phát sinh lỗi
5	Kiểm tra chức năng “So sánh xe”	Giao diện phải hiển thị chính xác thông tin của các	Kiểm tra thủ công bằng cách chọn 2-3 xe để so sánh	Thực hiện thành công	Không phát sinh lỗi

		xe được chọn để so sánh, bao gồm thông số kỹ thuật chi tiết			
6	Kiểm tra chức năng “Top xe”	Giao diện phải hiển thị cho người dùng xem top xe có lượt xem nhiều nhất	Kiểm tra thủ công bằng cách truy cập "Top xe"	Hiển thị chính xác biểu đồ top xe	Không phát sinh lỗi
7	Kiểm tra chức năng “Thêm xe”	Khi nhập đầy đủ và hợp lệ thông tin xe, giao diện phải hiển thị thông báo thêm thành công	Thực hiện thủ công bằng cách nhập dữ liệu vào form thêm xe	Thêm thành công với 1 xe	Yêu cầu đăng nhập với quyền admin
8	Kiểm tra chức năng “Sửa xe”	Khi chỉnh sửa thông tin xe hợp lệ, giao diện phải hiển thị thông báo cập nhật thành công	Thực hiện thủ công bằng cách chỉnh sửa xe trong danh sách xe	Thành công	Yêu cầu đăng nhập với quyền admin
9	Kiểm tra chức năng “Xóa xe”	Khi chọn "Xóa xe", giao diện phải hiển thị thông báo và danh sách xe phải cập nhật sau khi xóa thành công	Thực hiện thủ công bằng cách xóa xe trong danh sách xe	Thành công	Yêu cầu đăng nhập với quyền admin
10	Kiểm tra chức năng “Thêm	Khi nhập đầy đủ và hợp lệ	Thực hiện thủ công bằng cách	Thành công	Yêu cầu đăng nhập

	người dùng”	thông tin người dùng, giao diện phải hiển thị thông báo thêm người dùng thành công	nhập dữ liệu vào form thêm người dùng		với quyền admin
11	Kiểm tra chức năng “Sửa người dùng”	Khi chỉnh sửa thông tin người dùng hợp lệ, giao diện phải hiển thị thông báo cập nhật thành công	Thực hiện thủ công bằng cách chỉnh sửa người dùng trong danh sách	Thành công	Yêu cầu đăng nhập với quyền admin
12	Kiểm tra chức năng “Xóa người dùng”	Khi chọn "Xóa người dùng", giao diện phải hiển thị popup xác nhận và danh sách người dùng phải cập nhật sau khi xóa thành công	Thực hiện thủ công bằng cách xóa người dùng trong danh sách	Thành công	Yêu cầu đăng nhập với quyền admin

— **Kết quả kiểm thử hiệu năng**

STT	Mô tả kiểm thử	Yêu cầu kiểm thử	Thực hiện	Kết quả	Ghi chú
1	Kiểm tra thông lượng	Đảm bảo thông lượng đạt ít nhất 1000 yêu cầu/giây	Thực hiện bằng Jmeter Gửi 1000 yêu cầu HTTP đồng thời	Thông lượng đạt 1331.6 yêu cầu/giây	Đạt yêu cầu về thông lượng. Hệ thống đáp ứng tốt về số lượng yêu cầu trong một giây.

2	Kiểm tra thời gian phản hồi	Thời gian phản hồi trung bình không vượt quá 500ms	Thực hiện bằng Jmeter Gửi 1000 yêu cầu HTTP đồng thời	Thời gian phản hồi trung bình 134ms	Đạt yêu cầu. Thời gian phản hồi trung bình nằm trong giới hạn chấp nhận được.
3	Kiểm tra độ ổn định	90% yêu cầu phải hoàn thành trong vòng 500ms	Phân tích 90% Line từ báo cáo	90% yêu cầu trong 524ms	Không đạt yêu cầu. Một số yêu cầu bị chậm, dẫn đến thời gian phản hồi cho 90% yêu cầu vượt ngưỡng.
4	Kiểm tra lỗi	Tỷ lệ lỗi phải dưới 5%	Gửi 1000 yêu cầu HTTP đồng thời	Tỷ lệ lỗi 66.80%	Không đạt yêu cầu. Tỷ lệ lỗi cao bất thường.
5	Kiểm tra khả năng xử lý dữ liệu	Lượng dữ liệu nhận và gửi không thấp hơn 80% giá trị dự kiến	Đánh giá lượng dữ liệu trao đổi	2817.68 KB/giây nhận, 50.08 KB/giây gửi	Đạt yêu cầu. Hệ thống xử lý dữ liệu tương đối phù hợp với kịch bản tải kiểm thử.

CHƯƠNG 6. KẾT LUẬN VÀ BÀI HỌC KINH NGHIỆM

6.1. Kết luận

- Tính khả thi và hiệu quả: Sau khi thực hiện các bước phân tích, thiết kế và triển khai, hệ thống đã hoạt động đúng như yêu cầu ban đầu. Các tính năng cơ bản đã được xây dựng và thử nghiệm thành công, đáp ứng một số mục tiêu của dự án.
- Tính năng chính của hệ thống: Dự án đã hoàn thành các tính năng chính như quản lý xe, quản lý người dùng, tìm kiếm xe, xem chi tiết xe, so sánh xe,... Tất cả các tính năng này đều được triển khai và kiểm thử để đảm bảo tính ổn định.
- Chất lượng và hiệu suất: Các test case đã được thực hiện để kiểm tra sự hoạt động của hệ thống, bao gồm cả các trường hợp thành công và thất bại. Hệ thống đã đáp ứng được các yêu cầu về hiệu suất và bảo mật, nhưng vẫn còn một số vấn đề cần cải thiện trong tương lai.

6.2. Bài học kinh nghiệm

- Kế hoạch triển khai rõ ràng: Cần lập kế hoạch triển khai chi tiết. Việc xác định các yêu cầu và chia nhỏ các công việc giúp phát triển dễ dàng, theo dõi tiến độ và giảm thiểu rủi ro.
- Kiểm thử là yếu tố không thể thiếu: Kiểm thử hệ thống là một phần rất quan trọng trong quá trình phát triển phần mềm. Việc thực hiện các test case chi tiết, bao gồm các tình huống thành công và lỗi, giúp phát hiện sớm các vấn đề và cải thiện chất lượng phần mềm.
- Tính linh hoạt và điều chỉnh kịp thời: Trong suốt quá trình phát triển, nhóm đã gặp phải một số tình huống không lường trước được. Tuy nhiên, việc duy trì một cách tiếp cận linh hoạt và sẵn sàng điều chỉnh đã giúp giải quyết nhanh chóng các vấn đề phát sinh và đảm bảo tiến độ.
- Tầm quan trọng của giao tiếp nhóm: Một yếu tố then chốt là sự giao tiếp tốt giữa các thành viên trong nhóm. Việc trao đổi và thảo luận thường xuyên giúp đảm bảo rằng mọi người đều có cùng hướng đi và giải quyết các vấn đề một cách hiệu quả.
- Xử lý lỗi và cải tiến liên tục: Việc ghi nhận và xử lý các lỗi trong suốt quá trình phát triển là rất quan trọng.

6.3. Đề xuất cho dự án tương lai

- Đảm bảo tính bảo mật: Bảo mật dữ liệu người dùng là ưu tiên hàng đầu trong mọi dự án phần mềm. Các biện pháp bảo mật cần phải được triển khai ngay từ đầu và kiểm thử kỹ lưỡng trong suốt quá trình phát triển.
- Tăng cường kiểm thử tự động: Việc sử dụng các công cụ kiểm thử tự động sẽ giúp giảm thiểu sai sót và tiết kiệm thời gian kiểm thử thủ công. Điều này giúp đảm bảo chất lượng hệ thống một cách hiệu quả hơn.
- Chú trọng đến trải nghiệm người dùng: Hệ thống cần phải dễ sử dụng, thân thiện và đáp ứng được nhu cầu của người dùng. Việc nghiên cứu và cải tiến liên tục về giao diện người dùng (UI/UX) sẽ giúp nâng cao sự hài lòng của người dùng cuối.

TÀI LIỆU THAM KHẢO

- [1] **Disadvantages of SOA.** Exforsys. [Online]. Available: <https://www.exforsys.com/tutorials/soa/soa-disadvantages.html>. [Accessed: 15-Nov-2024].
- [2] **How to Build an API.** Postman Blog. [Online]. Available: <https://blog.postman.com/how-to-build-an-api/>. [Accessed: 15-Nov-2024].
- [3] **Kiểm thử phần mềm.** Testing.vn. [Online]. Available: <https://www.testing.vn/kiem-thu-phan-mem/#:~:text=%E2%9C%93%20Ki%E1%BB%83m%20th%E1%BB%AD%20ph%E1%BA%A7n%20m%E1%BB%81m,v%C3%A0o%20s%E1%BB%AD%20d%E1%BB%A5ng%20th%E1%BB%B1c%20t%E1%BA%BF>. [Accessed: 15-Nov-2024].
- [4] **Monolith, SOA, Microservices, and Serverless.** RubyGarage. [Online]. Available: <https://rubygarage.org/blog/monolith-soa-microservices-serverless>. [Accessed: 15-Nov-2024].
- [5] **What is an API?** Amazon Web Services. [Online]. Available: <https://aws.amazon.com/vi/what-is/api>. [Accessed: 15-Nov-2024].
- [6] **What is a RESTful API?** Amazon Web Services. [Online]. Available: <https://aws.amazon.com/what-is/restful-api/>. [Accessed: 15-Nov-2024].