ĐẠI HỌC BÁCH KHOA HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP

Hệ thống đặt lịch sửa chữa và bảo dưỡng ô tô

ĐINH HỮU HẢI
hai.dh204544@sis.hust.edu.vn

Ngành Khoa học máy tính

Giảng viên hướng dẫn: TS. Nguyễn Khánh Phương _____

Chữ kí GVHD

Khoa: Khoa học máy tính

Trường: Công nghệ Thông tin và Truyền thông

HÀ NỘI, 06/2024

LỜI CẢM ƠN

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

Em xin gửi lời cảm ơn chân thành đến giảng viên hướng dẫn - Tiến sĩ Nguyễn Khánh Phương người đã tận tình hướng dẫn, chỉ bảo em trong suốt quá trình làm đồ án. Cũng xin cảm ơn đến gia đình, bạn bè những người luôn ở bên cạnh đã luôn tạo điều kiện, quan tâm, giúp đỡ, động viên em trong suốt quá trình học tập và hoàn thành đồ án tốt nghiệp.

Cuối cùng xin cảm ơn chính bản thân đã nỗ lực hết sức mình để hoàn thành đồ án một cách tốt nhất.

Với điều kiện về thời gian cũng như lượng kiến thức còn đang hạn chế, đồ án này không thể tránh được những thiếu sót. Em rất mong nhận được sự chỉ bảo, đóng góp ý kiến của các thầy cô để em có điều kiện bổ sung, nâng cao ý thức của mình, phục vụ tốt hơn công việc thực tế sau này.

Em xin trân thành cảm ơn!

LỜI CAM KẾT

Họ và tên sinh viên: Đinh Hữu Hải .....
Điện thoại liên lạc: 0346572346 .....
Email: hai.dh204544@sis.hust.edu.vn .....
Lớp: IT1-02-k65 .....
Hệ đào tạo: Cử nhân .....

Tôi – *Đinh Hữu Hải* – cam kết Đồ án Tốt nghiệp (ĐATN) là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của *TS.Nguyễn Khánh Phương*. Các kết quả nêu trong ĐATN là trung thực, là thành quả của riêng tôi, không sao chép theo bất kỳ công trình nào khác. Tất cả những tham khảo trong ĐATN – bao gồm hình ảnh, bảng biểu, số liệu, và các câu từ trích dẫn – đều được ghi rõ ràng và đầy đủ nguồn gốc trong danh mục tài liệu tham khảo. Tôi xin hoàn toàn chịu trách nhiệm với dù chỉ một sao chép vi phạm quy chế của nhà trường.

Hà Nội, ngày tháng năm

Tác giả ĐATN

TÓM TẮT NỘI DUNG ĐỒ ÁN

Hệ thống đặt lịch sửa chữa bảo dưỡng ô tô giành cho khách hàng hiện tại chủ yếu là các website của chính garage đó tạo nên và còn một số hạn chế. Việc quản lý garage thì phải dùng thêm phần mềm quản lý khác. Các hướng tiếp cận đó đều chủ yếu phục vụ các vấn đề của garage, còn việc theo dõi tiến trình sửa chữa của khách hàng chưa được đáp ứng. Hướng tiếp cận của em là tạo ra hệ thống phục vụ cho cả khách hàng và cả cho việc quản lý garage. Chọn hướng tiếp cận này là vì nó mang lại sự tiện lợi toàn diện, giúp khách hàng dễ dàng theo dõi tiến trình sửa chữa xe và đồng thời tối ưu hóa quy trình quản lý cho garage.

Đóng góp chính của đồ án là phát triển một hệ thống quản lý garage toàn diện, thân thiện với người dùng, giúp tối ưu hóa quá trình vận hành và cải thiện hiệu suất công việc.

Sinh viên thực hiện

(Ký và ghi rõ họ tên)

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

DANH MỤC THUẬT NGỮ VÀ TỪ VIẾT TẮT

Thuật ngữ	**Ý nghĩa**
API	Giao diện lập trình ứng dụng (Application Programming Interface)
EUD	Phát triển ứng dụng người dùng cuối(End-User Development)
GWT	Công cụ lập trình Javascript bằng Java của Google (Google Web Toolkit)
HTML	Ngôn ngữ đánh dấu siêu văn bản (HyperText Markup Language)
IaaS	Dịch vụ hạ tầng

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1 Đặt vấn đề

Trong những năm gần đây, thị trường ô tô toàn cầu đã chứng kiến sự tăng trưởng mạnh mẽ, đặc biệt là tại các quốc gia đang phát triển. Sự gia tăng nhanh chóng của số lượng xe ô tô đã tạo ra nhu cầu lớn về các dịch vụ sửa chữa và bảo dưỡng. Theo các thống kê gần đây, số lượng ô tô đăng ký mới và ô tô cũ đang tăng lên hàng năm, dẫn đến áp lực lớn cho các garage sửa chữa để đáp ứng nhu cầu này.

Việc quản lý và điều hành một garage sửa chữa và bảo dưỡng xe hơi hiện nay gặp nhiều thách thức do lượng công việc lớn và sự đa dạng trong các yêu cầu dịch vụ. Nhiều garage vẫn sử dụng các phương pháp thủ công để quản lý lịch hẹn và thông tin khách hàng, dẫn đến hiệu suất thấp và dễ gây ra sai sót. Các phương pháp này không chỉ tốn thời gian mà còn có nguy cơ cao dẫn đến nhầm lẫn, mất thông tin, và ảnh hưởng đến chất lượng dịch vụ. Trong bối cảnh này, các garage cần một giải pháp công nghệ tiên tiến để tối ưu hóa quy trình làm việc, nâng cao hiệu quả và giảm thiểu sai sót.

Việc phát triển một hệ thống quản lý đặt lịch sửa chữa và bảo dưỡng tích hợp toàn diện không chỉ giúp cải thiện hiệu quả vận hành của garage mà còn nâng cao chất lượng dịch vụ và trải nghiệm khách hàng. Điều này không chỉ đáp ứng nhu cầu ngày càng cao của thị trường ô tô mà còn giúp các garage cạnh tranh và phát triển bền vững trong môi trường kinh doanh đầy thách thức hiện nay.

1.2 Mục tiêu và phạm vi đề tài

Một số giải pháp công nghệ đã được áp dụng như các phần mềm quản lý khách hàng (CRM) hoặc hệ thống quản lý dịch vụ (SMS). Tuy nhiên, những giải pháp này thường không được tùy chỉnh riêng cho ngành dịch vụ sửa chữa xe, thiếu tính linh hoạt và không tích hợp đầy đủ các chức năng cần thiết như quản lý lịch hẹn, thông tin xe, theo dõi tiến độ công việc và quản lý kho phụ tùng. Sự thiếu hụt này dẫn đến việc các garage phải sử dụng nhiều phần mềm khác nhau để quản lý từng khía cạnh của công việc, gây ra sự phức tạp và khó khăn trong việc đồng bộ thông tin.

Trước những hạn chế trên, em đã chọn phát triển một hệ thống quản lý đặt lịch sửa chữa và bảo dưỡng ô tô với tính năng tích hợp toàn diện, nhằm tối ưu hóa quá trình vận hành của garage. Lý do chọn hướng tiếp cận này là vì nó không chỉ giải quyết được vấn đề quản lý lịch hẹn một cách hiệu quả mà còn cung cấp một nền tảng toàn diện giúp quản lý thông tin khách hàng, xe cộ và các quy trình bảo dưỡng,

từ đó tăng cường hiệu quả làm việc và cải thiện trải nghiệm khách hàng.

Mục tiêu là xây dựng hệ thống đặt lịch sửa chữa bảo dưỡng ô tô tích hợp toàn diện. Các mục tiêu cụ thể bao gồm:

- Xây dựng ứng dụng web cho khách hàng: (i) Cho phép khách hàng đặt lịch hẹn trực tuyến, (ii) Quản lý thông tin cá nhân và lịch sử sửa chữa, bảo dưỡng xe.
- Cung cấp công cụ quản lý cho garage: (i) Theo dõi và quản lý lịch hẹn, (ii) Phân công công việc, (iii) Theo dõi tiến độ công việc.
- Tích hợp cơ chế thông báo tự động: (i) Nhắc nhở khách hàng về các cuộc hẹn, (ii) Thông báo bảo trì định kỳ.

1.3 Định hướng giải pháp

Để giải quyết vấn đề quản lý lịch hẹn và bảo dưỡng garage, hệ thống sẽ được phát triển dưới dạng một ứng dụng web tích hợp, sử dụng các công nghệ và phương pháp hiện đại.

- React: Được chọn để xây dựng giao diện người dùng (UI) vì React cho phép tạo các thành phần giao diện tái sử dụng và dễ dàng quản lý trạng thái ứng dụng, giúp phát triển giao diện người dùng một cách hiệu quả và linh hoạt.
- Express: Sử dụng Express để xây dựng máy chủ phía sau (backend) vì nó là một framework nhẹ của Node.js, hỗ trợ tốt cho việc phát triển các API cần thiết cho việc quản lý dữ liệu và tương tác với cơ sở dữ liệu.
- Firebase Cloud Messaging (FCM): FCM được chọn để cung cấp các thông báo tự động cho người dùng vì nó hỗ trợ gửi thông báo thời gian thực đến các thiết bị và trình duyệt, giúp nhắc nhở khách hàng về các cuộc hẹn và bảo trì định kỳ.
- Socket: Tích hợp Socket để cung cấp tính năng nhắn tin thời gian thực giữa khách hàng và nhân viên garage, giúp tăng cường sự tương tác và hỗ trợ ngay lập tức.
- Hệ quản trị cơ sở dữ liệu MySql: MySql được chọn để lưu trữ và quản lý dữ liệu vì nó cung cấp khả năng quản lý cơ sở dữ liệu quan hệ mạnh mẽ, đảm bảo tính nhất quán và toàn vẹn của dữ liệu. Ngoài ra SQL còn hỗ trợ các truy vấn phức tạp, giúp dễ dàng lấy và phân tích dữ liệu.

Đóng góp chính của đồ án này là phát triển một hệ thống đặt lịch sửa chữa, bảo dưỡng ô tô tại nhà, theo đó là việc quản lý garage toàn diện, giúp tối ưu hóa quá trình vận hành và cải thiện hiệu suất công việc.

1.4 Bố cục đồ án

Phần còn lại của báo cáo đồ án tốt nghiệp này được tổ chức như sau.

Chương 2 trình bày về khảo sát và phân tích yêu cầu của hệ thống, đưa ra những thông số thực tế về các hệ thống tương đương sau đó phân tích và đưa ra những yêu cầu chức năng và phi chức năng của hệ thống đặt lịch, quản lý. Kết quả của chương này sẽ là một bản phân tích chi tiết về các yêu cầu cần thiết để phát triển hệ thống.

Trong chương 3, em giới thiệu những công nghệ sử dụng để triển khai hệ thống như ReactJs, NodeJs,...Mỗi công nghệ sẽ được mô tả về cách thức hoạt động, ưu điểm và hạn chế. Chương này sẽ cung cấp một cái nhìn tổng quan về kiến trúc và công nghệ của hệ thống.

Chương 4 sẽ mô tả chi tiết quá trình phân tích hệ thống, từ việc thiết kế cơ sở dữ liệu, xây dựng các module chức năng, đến việc tích hợp các công nghệ đã chọn.

Chương 5 trình bày về những vấn đề gặp phải sau đó sẽ đưa ra giải pháp và đóng góp nổi bật trong đồ án.

Chương 6 đưa ra kết luận tổng quát về những gì đã đạt được qua quá trình nghiên cứu và phát triển hệ thống quản lý garage.

CHƯƠNG 2. KHẢO SÁT VÀ PHÂN TÍCH YÊU CẦU

Trong chương này, chúng ta sẽ tìm hiểu quá trình khảo sát và phân tích yêu cầu cho hệ thống quản lý garage. Chương này bắt đầu bằng việc tìm hiểu thị trường ô tô cũng như nhu cầu sửa chữa. Sau đó sử dụng biểu đồ use case theo hướng dẫn của template để mô tả chi tiết các yêu cầu chức năng của hệ thống. Các yêu cầu này sẽ bao gồm những chức năng cần thiết để hệ thống hoạt động hiệu quả cũng như các yêu cầu phi chức năng nhằm đảm bảo tính ổn định của hệ thống.

2.1 Khảo sát hiện trạng

Ở Việt Nam, cứ 1.000 người thì có 55 người sở hữu ô tô. Đáng chú ý, nhờ sự gia tăng mạnh mẽ của tầng lớp trung lưu, Việt Nam có tỷ lệ sở hữu ô tô tăng nhanh nhất thế giới. Trong giai đoạn từ năm 2015-2020, tỷ lệ này của Việt Nam tăng tới 17%/năm. Theo sau là Trung Quốc và Ấn Độ với tốc độ lần lượt là 14% và 10%/năm trong giai đoạn này. Vì vây nhu cầu về 1 hệ thống kết nối giữa người sở hữu ô tô và các hệ thống garage là rất lớn.

Trên thị trường đã có những hệ thống, ứng dụng lớn như:

1. AutoLeap

Hình 2.1: AutoLeap - Modern Auto Repair Shop Software

Hình 2.1 biểu thị tên thương hiệu Autoleap - phần mềm sửa chữa ô tô hiện đại với:

- Tính năng chính: AutoLeap cung cấp một giải pháp quản lý toàn diện dành cho các cửa hàng sửa chữa xe hơi, bao gồm quản lý lịch hẹn, khách hàng, thống kê, đưa ra chiến lược kinh doanh,....
- Ưu điểm: Sử dụng các công nghệ tiên tiến như AI và học máy để tối ưu hóa hoạt động.
- Nhược điểm: Khả năng tùy chỉnh hạn chế cho các mẫu tiếp thị và chi phí cao hơn so với một số nền tảng khác. Chủ yếu phục vụ cho thị trường Mỹ và Canada.

2. Agara

Hình 2.2: Agara

Hình 2.2 hiển thị bộ nhận diện thương hiệu của Agara - phần mềm quản trị garage ô tô:

- Agara là hệ thống quản lý Gara từ A tới Z: Báo giá, lập lệnh sửa chữa - Quản lý kho phụ tùng - Quản lý tài chính kế toán - Lập lịch và chăm sóc khách hàng.
- Nhược điểm: Không hỗ trợ khách hàng đặt lịch, thanh toán trên hệ thống.

3. oto hà thành

Hình 2.3: Website hathanhoto

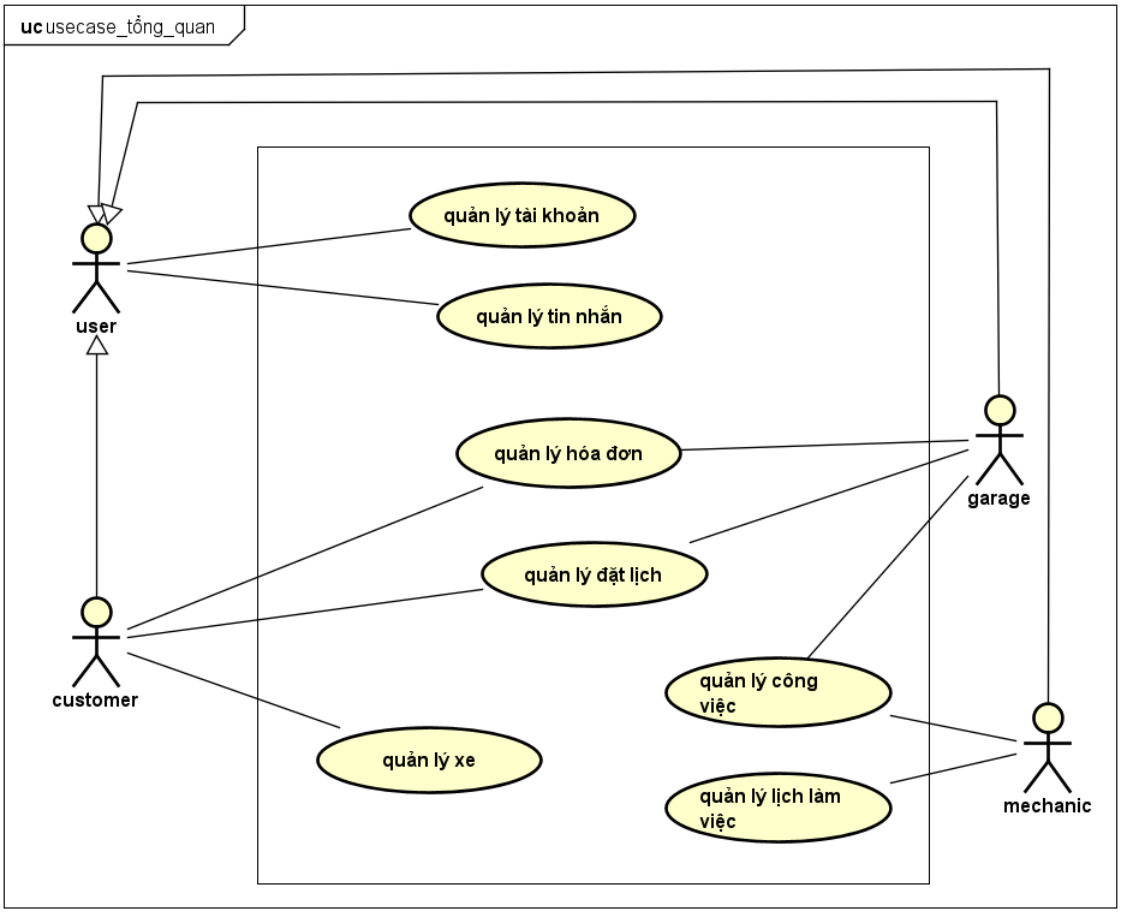
Hình 2.3 trình bày slogan thương hiệu Hà Thành Garage - Hệ thống website sửa chữa và chăm sóc ô tô cao cấp.

- Ưu điểm: website tập trung vào cung cấp thông tin sản phẩm, dịch vụ và tư vấn cho khách hàng về các vấn đề liên quan đến xe ô tô.
- Nhược điểm: Không hỗ trợ nhắc lịch bảo dưỡng với khách hàng.

Với những khảo sát đạt được, hệ thống đặt lịch sửa chữa bảo dưỡng ô to hướng tới việc tạo ra ứng dụng cho phép khách hàng đặt lịch, theo dõi tiến độ và thanh toán trực tiếp trên ứng dụng, ngoài ra còn xây dựng hệ thống quản lý garage nhằm phục vụ cho khách hàng nhiều tiện ích nâng cao trải nghiệm người dùng.

2.2 Tổng quan chức năng

2.2.1 Biểu đồ use case tổng quát

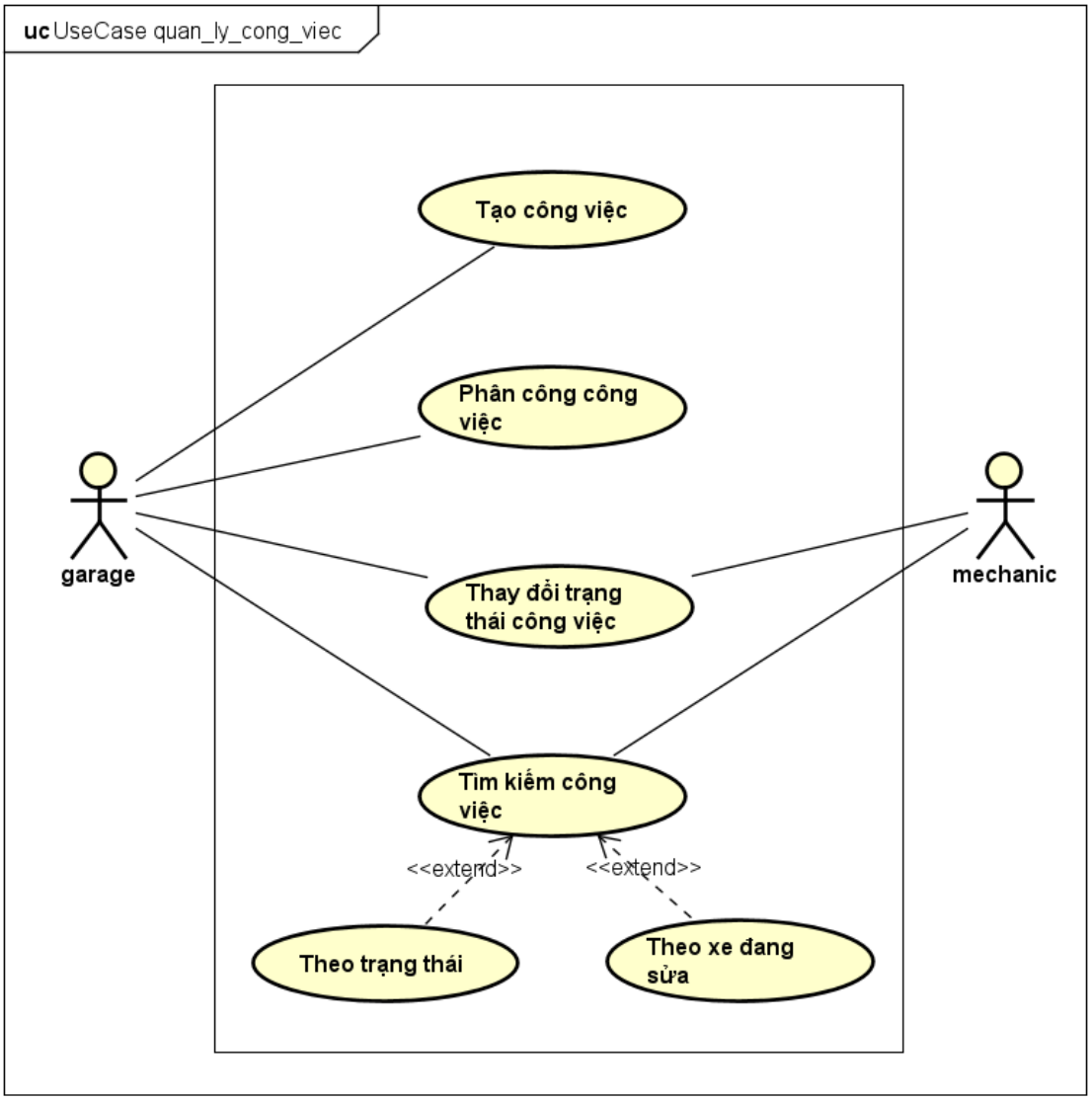


Hình 2.4: Biểu đồ usecase tổng quát

Hình 2.4 trình bày biểu đồ usecase tổng quát của hệ thống. Hệ thống đặt lịch sửa chữa, bảo dưỡng ô tô có các tác nhân chính tham gia: khách hàng (customer), garage ô tô (garage) và thợ sửa chữa (mechanic) cụ thể:

- User thông thường có thể tạo và quản lý tài khoản của mình, nhắn tin với các garage đang sửa chữa.
- Khách hàng là người sử dụng dịch vụ của garage, họ có thể đặt lịch hẹn, thanh toán cho dịch vụ và quản lý thông tin ô tô của mình.
- Garage là đơn vị cung cấp dịch vụ, chịu trách nhiệm quản lý lịch hẹn, tạo hóa đơn và quản lý phân công công việc liên quan đến dịch vụ ô tô.
- Thợ sửa chữa chữa là người thực hiện các công việc sửa chữa và bảo dưỡng ô tô theo lịch phân công từ garage và quản lý lịch làm việc của mình.

2.2.2 Biểu đồ use case phân rã quản lý công việc



Hình 2.5: Biểu đồ usecase quản lý công việc

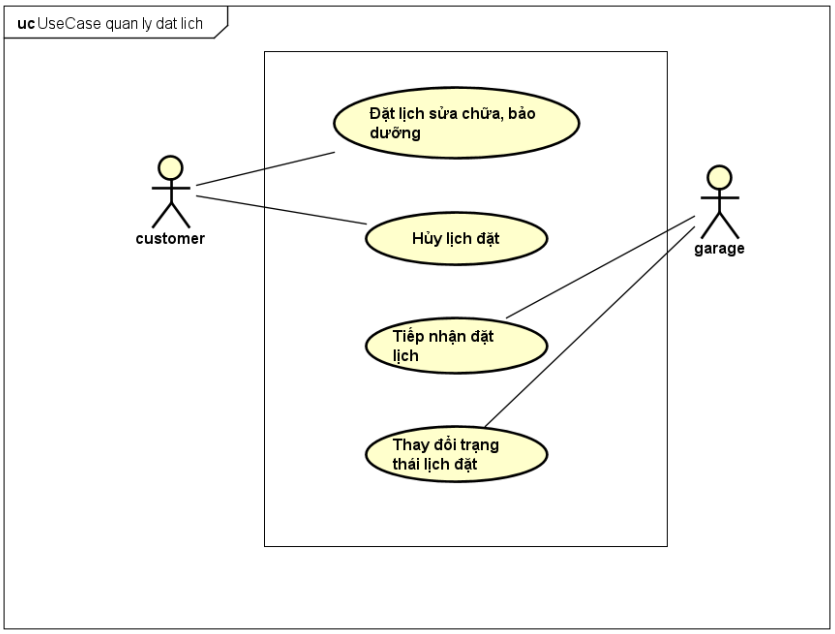
Hình 2.5 biểu diễn biểu đồ usecase quản lý công việc. Cả người quản lý garage và thợ sửa chữa đều có thể thay đổi trạng thái công việc như là thời gian bắt đầu, thời gian kết thúc, có thể tìm kiếm theo trạng thái hoặc theo tên ô tô đang sửa chữa trong garage. Ngoài ra, người quản lý có thể thực hiện các quyền:

- Tạo công việc: sau khi nhận lịch đặt thì garage sẽ tạo các công việc liên quan đến việc sửa chữa ô tô đó.
- Phân công công việc: Garage sẽ phân công các công việc cho thợ sửa chữa.

2.2.3 Biểu đồ use case phân rã quản lý đặt lịch

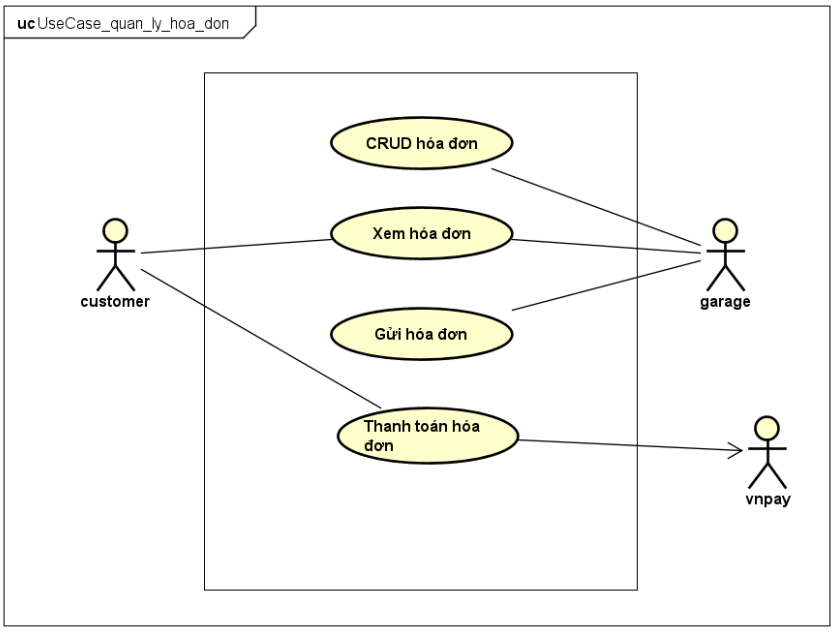
Hình 2.6 trình bày về biểu đồ usecase quản lý đặt lịch. Khách hàng (customer) sẽ đặt lịch sửa chữa hoặc bảo dưỡng, sau đó garage sẽ tiếp nhận và quản lý lịch

đặt đó. Sau khi đặt lịch thì khách hàng có thể theo dõi tiến độ làm việc của garage, khách hàng có thể hủy lịch đặt khi chưa có garage nào tiếp nhận.



Hình 2.6: Biểu đồ usecase quản lý đặt lịch

2.2.4 Biểu đồ use case phân rã quản lý hóa đơn



Hình 2.7: Biểu đồ usecase quản lý hóa đơn

Hình 2.7 trình bày về biểu đồ usecase phân rã quản hóa đơn. Phía quản lý garage sẽ có những chức năng sau:

- CRUD hóa đơn: Sau khi sửa chữa xong, garage sẽ tạo hóa đơn và thêm, sửa,

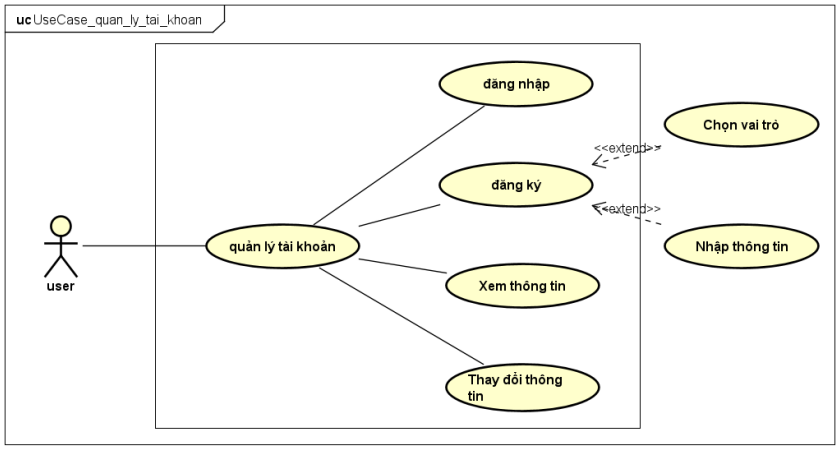
xóa các mặt hàng, dịch vụ trong hóa đơn.

- Gửi hóa đơn: Sau khi tạo xong hóa đơn garage sẽ gửi cho khách hàng.

Khách hàng sau khi nhận được hóa đơn thì sẽ có các chức năng:

- Xem hóa đơn: khách hàng xem hóa đơn trước và sau khi thanh toán, kết quả thanh toán.
- Thanh toán hóa đơn: Khách hàng tiến hành thanh toán qua nền tảng ví vnpay.

2.2.5 Biểu đồ use case phân rã quản tài khoản



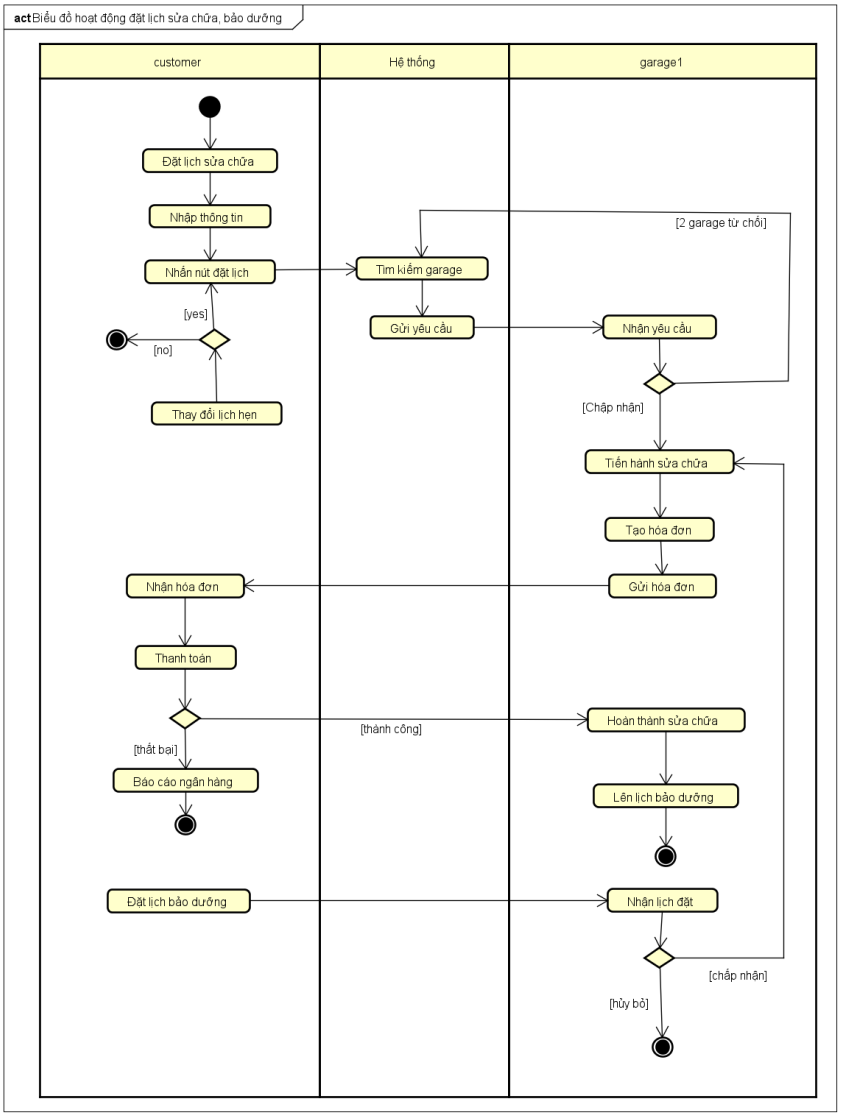
Hình 2.8: Biểu đồ usecase quản lý tài khoản

Hình 2.8 minh họa biểu đồ usecase quản lý tài khoản. Để quản lý tài khoản cá nhân, người dùng có thể thực hiện các hành động sau:

- Đăng ký: Người dùng chọn vai trò và nhập thông tin cá nhân tương ứng với vai trò đã chọn.
- Đăng nhập: Có thể đăng nhập sau khi tạo tài khoản.
- Xem thông tin, thay đổi thông tin: Sau khi đăng nhập thành công, khách hàng có thể xem và thay đổi thông tin cá nhân.

2.2.6 Quy trình nghiệp vụ

Hệ thống đặt lịch sửa chữa bảo dưỡng có quy trình nghiệp vụ gồm lần lượt các bước cơ bản sau: (i) Người dùng chọn dịch vụ sửa chữa, (ii) Hệ thống sẽ gửi yêu cầu tới 2 garage gần nhất, (iii) Garage xác nhận và garage sẽ tiến hành sửa chữa, (iv) Khi sửa chữa xong thì garage sẽ tạo hóa đơn và gửi cho khách hàng, (v) khách hàng tiến hành thanh toán, (vi) Garage xác nhận hoàn thành và lên lịch bảo dưỡng cho xe, (vii) Khách hàng đặt lịch bảo dưỡng với garage, (viii) Garage chập nhận hoặc từ chối lịch bảo dưỡng. **Hình 2.9** mô tả hoạt động đặt lịch sửa chữa bảo dưỡng.



Hình 2.9: Biểu đồ hoạt động đặt lịch sửa chữa, bảo dưỡng

2.3 Đặc tả chức năng

2.3.1 Đặc tả use case đăng ký tài khoản

Bảng 2.1 đặc tả usecase đăng ký tài khoản.

Mã use case	UC001	Tên use case	Quản lý tài khoản
Tác nhân	User		
Mô tả	Đăng ký tài khoản với vai trò là khách hàng, garage, thợ sửa chữa		
Tiền điều kiện	Không		
Luồng sự kiện chính	STT	Thực hiện	Hành động
	1	Người dùng	Nhấn nút đăng ký
	2	Hệ thống	Hiển thị form đăng ký
	3	Người dùng	Nhập thông tin

	4	Người dùng	Chọn vai trò
	5	Người dùng	Ấn nút đăng ký
	6	Hệ thống	Kiểm tra các thông tin
	7	Hệ thống	Thông báo đăng ký thành công và chuyển hướng người dùng sang màn hình chính
Luồng sự kiện thay thế	STT	Thực hiện	Hành động
	4a	Người dùng	Chọn vai trò garage
	4b	Người dùng	Nhập thông tin garage
	6a	Hệ thống	Thông báo không đúng yêu cầu
	7a	Hệ thống	Thông báo thông tin đã được đăng ký
Hậu điều kiện	Không		

Bảng 2.1: Bảng đặc tả use case quản lý tài khoản

2.3.2 Đặc tả use case quản lý đặt lịch

Bảng 2.2 đặc tả usecase quản lý đặt lịch.

Mã use case	UC002	Tên use case	Quản lý đặt lịch
Mục đích sử dụng	Khách hàng tiến hành đặt lịch sửa chữa, bảo dưỡng, garage thì tiếp nhận và quản lý tiến độ		
Tác nhân	Khách hàng, garage		
Điều kiện tiên quyết	User đăng nhập thành công		
Luồng sự kiện chính	STT	Thực hiện	Hành động
	1	Khách hàng	Chọn dịch vụ sửa chữa, bảo dưỡng
	2	Khách hàng	Nhập thông tin địa chỉ, mô tả, thời gian,...
	3	Khách hàng	Nhấn nút gửi
	4	Hệ thống	Kiểm tra các trường thông tin nhập vào
	5	Hệ thống	Chuyển người dùng sang trang theo dõi lịch đặt
	6	Hệ thống	Gửi yêu cầu, thông báo tới garage
	7	Garage	Chấp nhận yêu cầu
	8	Hệ thống	Gửi thông báo tới khách hàng

	9	Garage	Thay đổi trạng thái đặt lịch sang đang sửa chữa
	10	Garage	Thay đổi trạng thái đặt lịch sang hoàn thành
Luồng sự kiện thay thế	STT	Thực hiện	Hành động
	5a	Hệ thống	Thông báo các trường nhập vào không hợp lệ
	8a	Garage	Từ chối lịch đặt của khách hàng
Hậu điều kiện	Không		

Bảng 2.2: Bảng đặc tả use case quản lý đặt lịch

2.3.3 Đặc tả use case quản lý công việc

Bảng 2.3 đặc tả usecase quản lý công việc sẽ bao gồm (i) tạo công việc, (ii) cập nhật tiến độ công việc.

Mã use case	UC003	Tên use case	Quản lý công việc
Mục đích sử dụng	Garage tạo và phân chia, quản lý công việc, thợ sửa chữa nhận, quản lý công việc được giao		
Tác nhân	Garage, thợ sửa chữa		
Điều kiện tiên quyết	User đăng nhập thành công và đang ở màn hình danh sách công việc		
Luồng sự kiện chính	STT	Thực hiện	Hành động
	1	Garage	Chọn tạo công việc mới
	2	Hệ thống	Hiển thị form tạo công việc
	3	Garage	Nhấn nút tạo
	4	Hệ thống	Kiểm tra thông tin nhập vào
	5	Garage	Nhấn nút cập nhật
	6	Hệ thống	Hiển thị công việc cần cập nhật
	7	Garage	Chọn thợ để giao nhiệm vụ
	8	Garage	Nhấn nút cập nhật
	9	Hệ thống	Kiểm tra và cập nhật thợ sửa chữa cho nhiệm vụ
	10	Garage, Thợ sửa chữa	Nhấn nút cập nhật
	11	Hệ thống	Hiển thị công việc cần cập nhật

	12	Garage, Thợ sửa chữa	Cập nhật tiến độ công việc (đang tiến hành, hoàn thành)
	13	Hệ thống	Thông báo cập nhật thành công
Luồng sự kiện thay thế	STT	Thực hiện	Hành động
	4a	Garage	Nhấn nút hủy
	9a	Hệ thống	Thông báo các trường nhập vào không hợp lệ
	13a	Hệ thống	Thông báo các trường nhập vào không hợp lệ
Hậu điều kiện	Không		

Bảng 2.3: Bảng đặc tả use case quản lý công việc

2.3.4 Đặc tả use case quản lý hóa đơn

Bảng 2.4 đặc tả usecase quản lý hóa đơn. Phía Garage gồm các chức năng (i) tạo hóa đơn, (ii) thêm dịch vụ vào hóa đơn, (iii) xem hóa đơn. Khách hàng sẽ có chức năng (i) xem hóa đơn, (ii) thanh toán hóa đơn.

Mã use case	UC004	Tên use case	Quản lý hóa đơn
Mục đích sử dụng	Garage tạo hóa đơn và người dùng xem và thanh toán hóa đơn đó		
Tác nhân	Garage, Khách hàng		
Điều kiện tiên quyết	User đăng nhập thành công và công việc đang ở trạng thái “in-progress”		
Luồng sự kiện chính	STT	Thực hiện	Hành động
	1	Garage	Chọn tạo hóa đơn
	2	Hệ thống	Tạo hóa đơn rỗng và hiển thị
	3	Garage	Nhấn nút thêm bảng giá
	4	Hệ thống	Hiển thị form nhập giá dịch vụ
	5	Garage	Nhập các thông tin dịch vụ vào form
	6	Garage	Nhấn nút thêm
	7	Hệ thống	Kiểm tra các thông tin và thêm vào hóa đơn
	8	Hệ thống	Hiển thị hóa đơn với dịch vụ vừa cập nhật
	9	Garage	Nhấn vào dịch vụ

	10	Hệ thống	Hiển thị thông tin bảng giá dịch vụ
	11	Garage	Chỉnh sửa thông tin
	12	Garage	Nhấn nút cập nhật
	13	Hệ thống	Thông báo cập nhật thành công
	14	Hệ thống	Cập nhật hóa đơn và hiển thị
	15	Garage, Khách hàng	Chọn xem hóa đơn
	16	Hệ thống	Hiển thị hóa đơn
	17	Khách hàng	Chọn phương thức thanh toán và bấm vào nút thanh toán
	18	1l|Hệ thống	Chuyển hướng người dùng sang trang thanh toán
	19	Khách hàng	Tiến hành thanh toán thành công
	20	Hệ thống	Chuyển hướng người dùng quay lại và cập nhật trạng thái thanh toán
Luồng sự kiện thay thế	STT	Thực hiện	Hành động
	7a	Hệ thống	Thông báo các trường nhập vào không hợp lệ
	12a	Garage	Nhấn nút xóa dịch vụ
	12b	Garage	Nhấn nút cancel
	19a	Khách hàng	Thanh toán thất bại
Hậu điều kiện	Không		

Bảng 2.4: Bảng đặc tả use case quản lý hóa đơn

2.4 Yêu cầu phi chức năng

Các yêu cầu phi chức năng cho một trang web xác định các tiêu chí mô tả cách thức một trang web, trang web phải hoạt động, đảm bảo rằng trang web đáp ứng các thuộc tính chất lượng nhất định và sự mong đợi của người dùng. Dưới đây là một số yêu cầu phi chức năng cho ứng dụng này:

1. Hiệu suất
    - Thời gian phản hồi: Hệ thống phải có khả năng phản hồi nhanh chóng với thời gian tải trang dưới 3 giây cho tất cả các chức năng.
    - Khả năng xử lý: Hệ thống cần hỗ trợ ít nhất 1000 người dùng đồng thời mà không ảnh hưởng đến hiệu năng.

2. Độ tin cậy
    - Hệ thống phải đạt mức khả dụng ít nhất 99.9, với thời gian ngừng hoạt động tối đa không quá 8.76 giờ mỗi năm.
    - Khôi phục sau sự cố: Hệ thống phải có khả năng khôi phục sau sự cố trong vòng 1 giờ.
3. Tính dễ dùng
    - Giao diện người dùng thân thiện: Giao diện phải trực quan và dễ sử dụng, phù hợp với người dùng không có nhiều kinh nghiệm về công nghệ.
4. Tính dễ bảo trì
    - Mã nguồn: Mã nguồn phải được viết rõ ràng, có cấu trúc tốt, dễ hiểu và dễ bảo trì.
    - Tài liệu: Mã nguồn, kiến trúc và chức năng của trang web được ghi lại đầy đủ để dễ bảo trì và phát triển trong tương lai.
5. Yêu cầu kỹ thuật
    - Cơ sở dữ liệu MySQL.
    - Công nghệ sử dụng: Frontend sử dụng ReactJs, backend sử dụng Nodejs, deploy bằng docker, sử dụng Firebase Cloud Messaging (FCM) để gửi thông báo đẩy, sử dụng Socket.io để hỗ trợ giao tiếp thời gian thực.

CHƯƠNG 3. CÔNG NGHỆ SỬ DỤNG

3.1 Framework front-end: ReactJs

Hình 3.1: ReactJs

3.1.1 Giới thiệu về ReactJs

ReactJS (**hình 3.1**) là một thư viện JavaScript mã nguồn mở được phát triển bởi Facebook nhằm tạo ra các ứng dụng web nhanh và hiệu quả với mã nguồn.

Mục đích chính của ReactJS là khiến cho website hoạt động mượt mà, khả năng mở rộng cao và đơn giản.

Thay vì làm việc trên toàn ứng dụng web, ReactJS cho phép các nhà phát triển có thể phá vỡ giao diện người dùng phức tạp một cách thuận lợi thành các thành phần đơn giản.

ReactJS cho phép tạo ra các ứng dụng web có UI tốt hơn, qua đó nâng cao trải nghiệm của người dùng như lượt tương tác, tỷ lệ click, lượt chuyển đổi.

3.1.2 Các khái niệm về REACTJS

a, Virtual Dom

Một trong những đặc điểm nổi bật của React là Virtual DOM (DOM ảo). Khi bạn cập nhật trạng thái (state) của một thành phần, React không cập nhật DOM trực tiếp mà thay vào đó, nó tạo ra một bản sao của DOM gọi là Virtual DOM. Sau đó, React so sánh Virtual DOM cũ và mới, và chỉ cập nhật những phần thay đổi vào DOM thực sự, điều này giúp cải thiện hiệu suất ứng dụng bằng cách giảm thiểu số lượng thao tác trên DOM.

b, JSX

JSX (JavaScript XML) là một phần mở rộng của JavaScript, cho phép bạn viết HTML trong JavaScript một cách dễ dàng và thú vị hơn. JSX giúp tạo ra các thành phần UI trong React một cách rõ ràng và dễ hiểu hơn so với việc sử dụng các phương pháp tạo DOM truyền thống. Với JSX, bạn có thể tạo ra cấu trúc UI phức tạp bằng cách kết hợp các thành phần và các biểu thức JavaScript trong cùng một nơi.

c, Components

Trong React, chúng ta xây dựng các ứng dụng web bằng cách sử dụng các thành phần nhỏ. Các thành phần này có thể tái sử dụng ở nhiều nơi khác nhau và có thể có các trạng thái và thuộc tính khác nhau. Mỗi thành phần trong React có thể có một trạng thái riêng biệt, có thể thay đổi, và React sẽ tự động cập nhật giao diện người dùng dựa trên các thay đổi này. React làm cho việc bảo trì mã code trở nên dễ dàng, đặc biệt là khi làm việc với các dự án lớn. Một thành phần React đơn giản cần phải có ít nhất một phương thức là render để định nghĩa cách thức hiển thị của nó. Ngoài ra, React cũng cung cấp nhiều phương thức khác để quản lý trạng thái và vòng đời của các thành phần.

d, Props và State

Props là viết tắt của Properties, đại diện cho các thuộc tính mà một thành phần cha có thể truyền cho một thành phần con. Props là bất biến (immutable), nghĩa là sau khi được truyền vào một thành phần, chúng không thể thay đổi bởi thành phần con đó.

State thể hiện trạng thái nội tại của một thành phần, mà có thể thay đổi khi người dùng tương tác với ứng dụng. Mỗi khi state thay đổi, React sẽ tự động cập nhật lại giao diện người dùng (UI) để phản ánh trạng thái mới này.

3.1.3 Lý do sử dụng

- Tái sử dụng component: React cho phép xây dựng các component có thể tái sử dụng nhiều lần trong các phần khác nhau của ứng dụng, giúp giảm thiểu mã lặp và cải thiện khả năng bảo trì.
- Chuyển trang nhanh: SPA giúp người dùng có trải nghiệm mượt mà hơn do không cần tải lại toàn bộ trang khi điều hướng giữa các trang khác nhau.
- Quản lý trạng thái dễ dàng: React kết hợp với các thư viện như Redux hoặc MobX để quản lý trạng thái ứng dụng một cách hiệu quả.
- Khả năng chuyển đổi và mở rộng sang ứng dụng di động: React Native cho phép chuyển đổi mã ReactJS thành ứng dụng di động, giúp tiết kiệm thời gian

và nguồn lực trong việc phát triển ứng dụng đa nền tảng.

- Cộng đồng lớn và hệ sinh thái phong phú: React có tài liệu rất chi tiết và dễ hiểu, cùng với nhiều tutorial và hướng dẫn từ cộng đồng.Có nhiều thư viện và công cụ bổ trợ phát triển React, như React Router, Redux, Next.js,...

3.2 Framework back-end: Express.Js

Hình 3.2: ExpressJs

3.2.1 Giới thiệu về ExpressJs

Expressjs (**hình 3.2**) hay còn được viết là Express js, Express.js. Đây là một framework mã nguồn mở miễn phí cho Node.js. Express.js được sử dụng trong thiết kế và xây dựng các ứng dụng web một cách đơn giản và nhanh chóng.

Vì Express js chỉ yêu cầu ngôn ngữ lập trình Javascript nên việc xây dựng các ứng dụng web và API trở nên đơn giản hơn với các lập trình viên và nhà phát triển. Expressjs cũng là một khuôn khổ của Node.js do đó hầu hết các mã code đã được viết sẵn cho các lập trình viên có thể làm việc.

Nhờ có Expressjs mà các nhà lập trình có thể dễ dàng tạo các ứng dụng 1 web, nhiều web hoặc kết hợp. Do có dung lượng khá nhẹ, Expressjs giúp cho việc tổ chức các ứng dụng web thành một kiến trúc MVC có tổ chức hơn. Để có thể sử dụng được mã nguồn này, chúng ta cần phải biết về Javascript và HTML.

Expressjs cũng là một phần của công nghệ giúp quản lý các ứng dụng web một cách dễ dàng hơn hay còn được gọi là ngăn xếp phần mềm MEAN. Nhờ có thư viện Javascript của Express js đã giúp cho các nhà lập trình xây dựng nên các ứng dụng web hiệu quả và nhanh chóng hơn. Expressjs cũng được sử dụng để nâng cao các chức năng của Node.js.

3.2.2 Lý do sử dụng

- ExpressJS là một công nghệ rất dễ học và sử dụng nếu đã biết JavaScript. Điều này giúp cho việc phát triển back-end dễ dàng hơn rất nhiều. Bởi vì mã

JavaScript được diễn giải thông qua Google V8 JavaScript Engine của Nodejs, ExpressJS cho phép mã được thực hiện một cách nhanh chóng và hiệu quả.

- ExpressJS có thể giúp phát triển máy chủ nhanh hơn bằng việc cung cấp các tính năng phổ biến của Nodejs dưới dạng hàm có thể tái sử dụng. Ngoài ra, ExpressJS cũng đóng vai trò là phần mềm trung gian, giúp tổ chức các chức năng khác nhau của ứng dụng.
- ExpressJS cũng cung cấp một cơ chế định tuyến nâng cao có thể giúp duy trì vững trạng thái của trang web. Nó cũng cung cấp các công cụ tạo khuôn mẫu cho phép các nhà phát triển tạo được nội dung động trên các trang web này bằng việc bắt đầu xây dựng các mẫu HTML ở phía máy chủ.
- ExpressJS hỗ trợ phát triển ứng dụng theo mô hình MVC, một mô hình phổ biến cho việc lập trình web hiện nay. Điều này giúp cho việc phát triển ứng dụng được tổ chức và dễ bảo trì hơn.

3.3 Firebase Cloud Messaging (FCM)

Hình 3.3: Firebase Cloud Messaging

3.3.1 Giới thiệu về FCM

Hình 3.3 mô tả biểu tượng của fierbase cloud messaging.

Giải pháp gửi thông báo qua đám mây của Firebase (FCM) là một giải pháp gửi thông báo trên nhiều nền tảng, cho phép bạn gửi tin nhắn một cách đáng tin cậy mà không mất phí.

Khi sử dụng FCM, bạn có thể thông báo cho ứng dụng khách rằng email mới hoặc dữ liệu khác hiện có thể đồng bộ hoá. Bạn có thể gửi tin nhắn thông báo để tăng mức độ tương tác lại và tỷ lệ giữ chân người dùng. Đối với các trường hợp sử dụng như nhắn tin nhanh, một tin nhắn có thể chuyển tải trọng lên tới 4096 byte tới một ứng dụng khách.

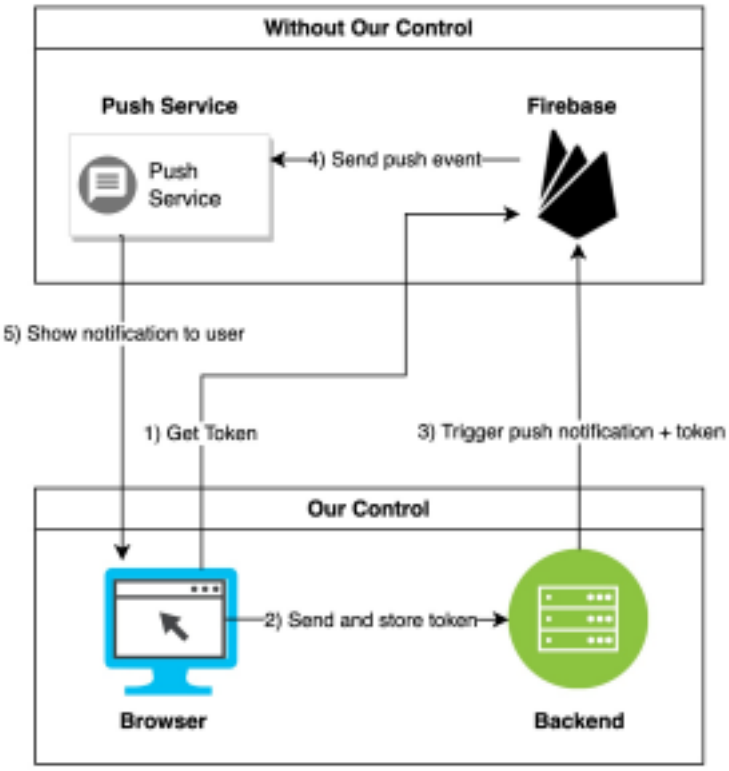
Các khả năng chính:

- Gửi nội dung thông báo được hiển thị cho người dùng của bạn. Hoặc gửi thông

báo dữ liệu và xác định hoàn toàn những gì xảy ra trong mã xử lý ứng dụng của bạn.

- Phân phối thông báo cho ứng dụng khách của bạn theo 1 trong 3 cách – tới thiết bị đơn lẻ, đến nhóm thiết bị hoặc tới các thiết bị đã đăng ký chủ đề.
- Gửi lời cảm ơn, tin nhắn trò chuyện và các tin nhắn khác từ các thiết bị về máy chủ của bạn qua kênh kết nối tiết kiệm pin và đáng tin cậy của FCM.

3.3.2 Cách hoạt động



Hình 3.4: Cấu trúc hoạt động của FCM

Quá trình triển khai FCM bao gồm hai thành phần chính để gửi và nhận:

- Một môi trường đáng tin cậy, chẳng hạn như Cloud Functions cho Firebase hoặc một máy chủ ứng dụng để tạo, nhắm mục tiêu và gửi thông báo.
- Một ứng dụng khách của Apple, Android hoặc web (JavaScript) nhận thông báo qua dịch vụ truyền tải dành riêng cho nền tảng tương ứng.

Cấu trúc hoạt động của FCM như sau (minh họa ở **hình 3.4**):

1. Thiết bị di động đăng ký device token: Khi ứng dụng được cài đặt và khởi chạy lần đầu tiên, Firebase SDK sẽ tự động đăng ký device token với FCM. Device token này được lưu trữ trong backend để sử dụng sau này.

2. Backend gửi thông báo: Khi muốn gửi thông báo đẩy đến người dùng, hệ thống sẽ sử dụng FCM API từ backend. API này cho phép xác định nội dung thông báo, đối tượng nhận và các tùy chọn khác.
3. FCM nhận thông báo từ backend và sử dụng device token để xác định thiết bị cần nhận thông báo. Dịch vụ đẩy sau đó sẽ gửi thông báo đến thiết bị đó qua mạng internet.
4. Khi thiết bị nhận được thông báo từ FCM, nó sẽ hiển thị thông báo cho người dùng. Người dùng có thể tương tác với thông báo bằng cách bấm vào nút hành động hoặc tắt thông báo.

3.4 Redux

Hình 3.5: Redux

3.4.1 Giới thiệu về Redux

Redux (**hình 3.5**) là một thư viện quản lý trạng thái (state management) tương thích với các ứng dụng web, phổ biến trong việc phát triển ứng dụng front-end sử dụng JavaScript và ReactJS.

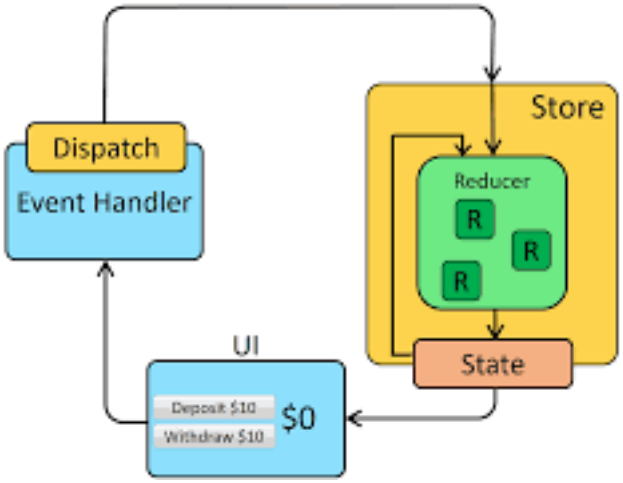
Redux có nhiệm vụ quản lý trạng thái phức tạp trong ứng dụng web, giúp tách biệt logic và giao diện người dùng. Với tiếp cận dễ hiểu và theo dõi, Redux giúp theo dõi và cập nhật trạng thái một cách hiệu quả, đồng thời đảm bảo tính nhất quán cho dữ liệu trong ứng dụng, từ đó dễ dàng debug khi gặp lỗi.

3.4.2 Thành phần chính và cách hoạt động

Hình 3.6 giải thích luồng hoạt động của redux bao gồm các thành phần chính:

1. Actions

   Action đơn giản là những Event được tạo ra bằng việc sử funcion và send data từ app đến Redux store. Data có thể được gửi bằng nhiều cách như gọi API,



Hình 3.6: Luồng hoạt động của Redux

hay thao tác của User lên App, hoặc submit form.

Mỗi action của Redux là một object chứa type của action và data payload, trong đó type dùng để miêu tả loại action, còn payload thì chứa data được gửi lên store.

2. Reducers

   Sau khi dispatch một action nào đó, Reducers trong Redux có nhiệm vụ lấy state hiện tại của app, thêm dữ liệu nhận được từ việc dispatch action, và trả về một state mới. Những states này được lưu như những objects và chúng định rõ cách state của một ứng dụng thay đổi trong việc phản hồi một action được gửi đến store.

   Ngoài ra, khi build một ứng dụng lớn cần sử dụng nhiều reducer, chúng ta sẽ sử dụng method combineReducers() của Redux để kết hợp tất cả các reducer lại thành một list các reducer, mỗi một reducer sẽ xử lí một state riêng nhỏ.

3. Store

   Store chính là "single source of truth" nắm giữ toàn bộ states của app chúng ta và đồng thời cung cấp những method để có thể thao tác với state, dispatch action, và sau khi reducer xử lí các action nhận được thì nó trả về state mới cho store và các component sẽ được rerender khi state của nó có thay đổi.

CHƯƠNG 4. THIẾT KẾ, TRIỂN KHAI VÀ ĐÁNH GIÁ HỆ THỐNG

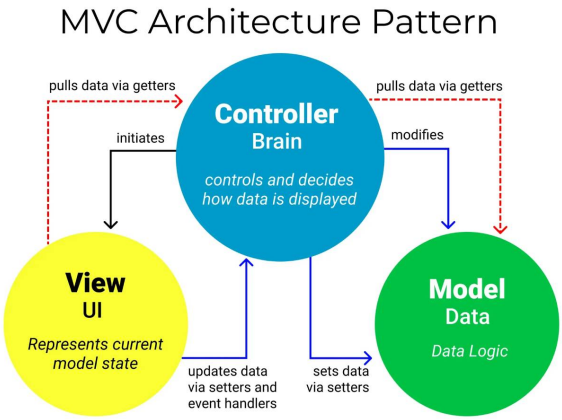
4.1 Thiết kế kiến trúc

4.1.1 Lựa chọn kiến trúc phần mềm

Đối với hệ thống này, em sử dụng kiến trúc MVC (Model-View-Controller) - hình 4.1.

Mô hình MVC, hay Model-View-Controller, là một kiến trúc phần mềm được sử dụng để tổ chức và quản lý mã nguồn trong quá trình phát triển ứng dụng. Kiến trúc này tách biệt ba thành phần chính: Model, View, và Controller, giúp giảm độ phức tạp của hệ thống và làm cho mã nguồn dễ quản lý hơn.

- Model (M - Model): Đây là thành phần chịu trách nhiệm cho xử lý dữ liệu và logic kinh doanh của ứng dụng. Model là nơi lưu trữ thông tin, thực hiện các thao tác cập nhật và truy vấn dữ liệu, mà không quan tâm đến cách dữ liệu được hiển thị hoặc tương tác với người dùng.
- View (V - View): View là thành phần hiển thị giao diện người dùng và đảm nhận trách nhiệm hiển thị thông tin từ Model. View không có logic kinh doanh và chỉ chịu trách nhiệm về việc hiển thị dữ liệu một cách đẹp mắt và dễ hiểu cho người dùng.
- Controller (C - Controller): Controller là thành phần điều phối và xử lý sự kiện từ người dùng. Nó nhận lệnh từ người dùng thông qua View, sau đó cập nhật Model dựa trên những thay đổi này và điều hướng hiển thị trở lại View. Controller giữ vai trò quan trọng trong việc duy trì sự đồng bộ giữa Model và View.



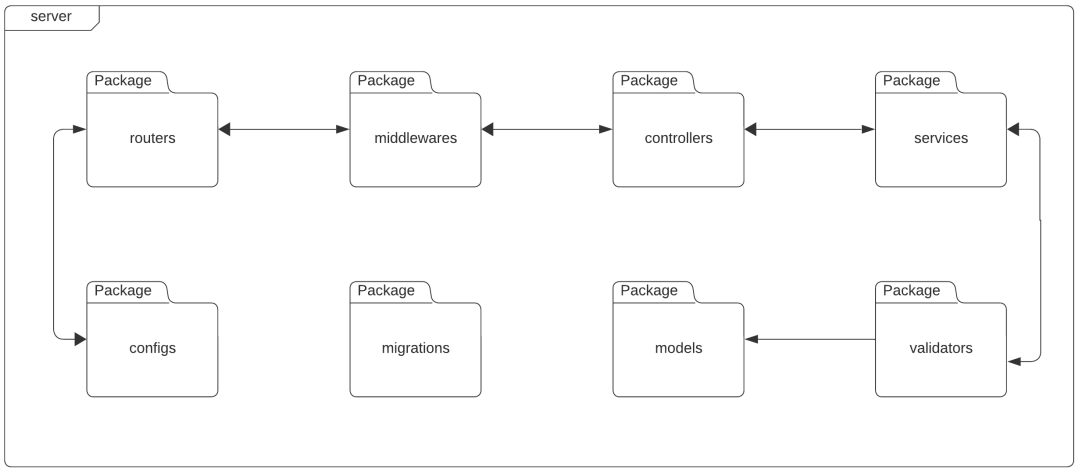
Hình 4.1: Mô hình mvc

Trong hệ thống đặt lịch này, mô hình mvc được áp dụng như sau:

- Phía client là phần View được phát triển bằng React, sử dụng Redux để quản lý trạng thái của ứng dụng.
- Phía server chịu trách nhiệm xử lý Controller và Model thông qua nền tảng Node.js và cơ sở dữ liệu MySQL.
- Giao tiếp giữa client và server được thực hiện thông qua giao thức HTTP, tuân theo các nguyên tắc của RESTful API. Bao gồm việc sử dụng các phương thức HTTP như GET, POST, PUT và DELETE để thực hiện các hoạt động CRUD (Create, Read, Update, Delete) trên dữ liệu.

4.1.2 Thiết kế tổng quan

a, Thiết kế tổng quan phía backend



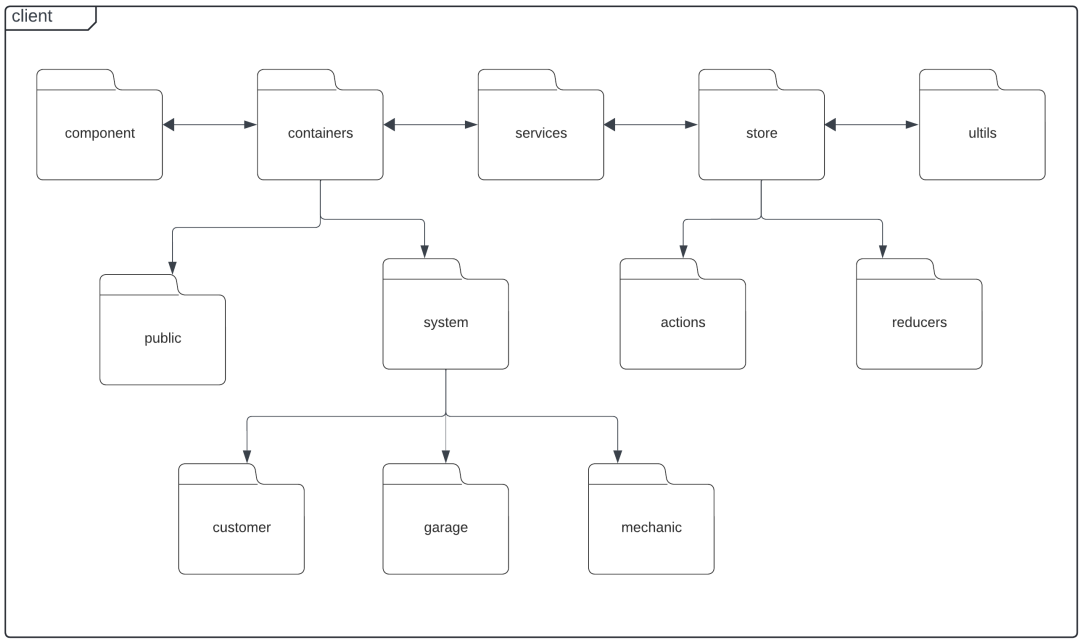
Hình 4.2: UML Package diagram phía backend

Thiết kế tổng quan các gói ở phía backend (hình 4.2) được mô tả như sau:

- Configs package: Chứa các cấu hình như môi trường, cấu hình kết nối cơ sở dữ liệu.
- Routers package: Chứa các tệp định tuyến, xác định các endpoint của API và ánh xạ chúng tới các controller tương ứng.
- Middleware package: Xử lý các request trước khi chúng đến controller, thực hiện nhiệm vụ xác thực.
- Controller package: Chứa các tệp controller, quản lý logic xử lý của các yêu cầu API, nhận yêu cầu từ router và tương tác với service hoặc model.
- Services package: Chứa các tệp service, xử lý logic nghiệp vụ của ứng dụng, thường là các phương thức mà controllers sẽ gọi đến.

- Validators package: Chứa các tệp kiểm tra và xác thực dữ liệu đầu vào cho các yêu cầu, đảm bảo dữ liệu đầu vào hợp lệ trước khi xử lý.
- Models package: Chứa các định nghĩa của các mô hình dữ liệu (database models) sử dụng với ORM.
- Migrations package: Chứa các tệp quản lý di chuyển cơ sở dữ liệu, giúp tạo và cập nhật cấu trúc cơ sở dữ liệu.

b, Thiết kế tổng quan phía frontend



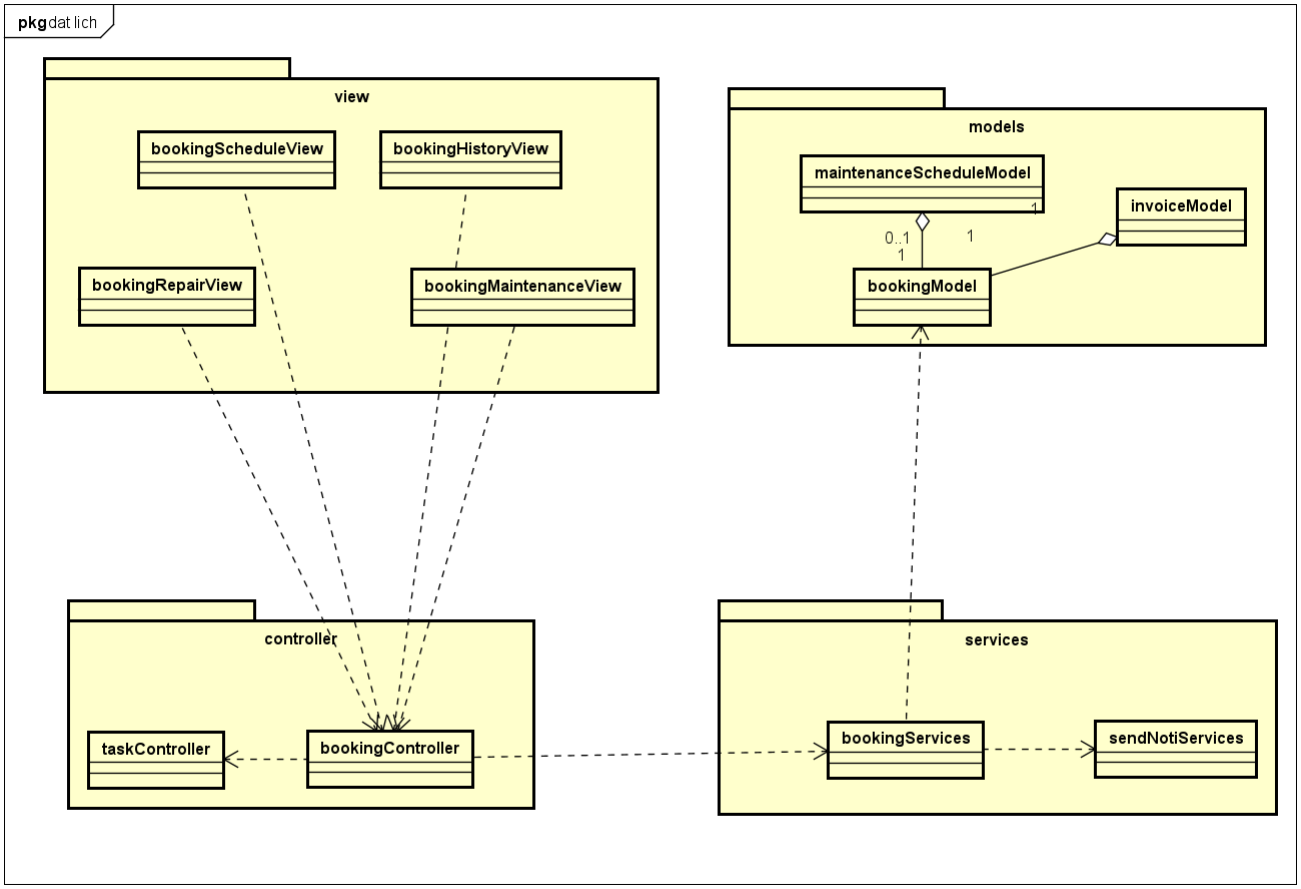
Hình 4.3: UML Package diagram phía frontend

Thiết kế tổng quan các gói ở phía frontend (**hình 4.3**) được mô tả như sau:

- Component package: Chứa các component dùng chung.
- Containers package: Chứa view hiển thị phía client chia ra 2 phần public chứa giao diện đăng nhập, đăng ký, phần system chứa các giao diện của từng actor sau khi đăng nhập.
- Services package: Chứa các file liên quan đến việc gọi API.
- Store package: Chứa các file liên quan đến quản lý state thông qua Redux, bao gồm actions và reducers.

4.1.3 Thiết kế chi tiết gói

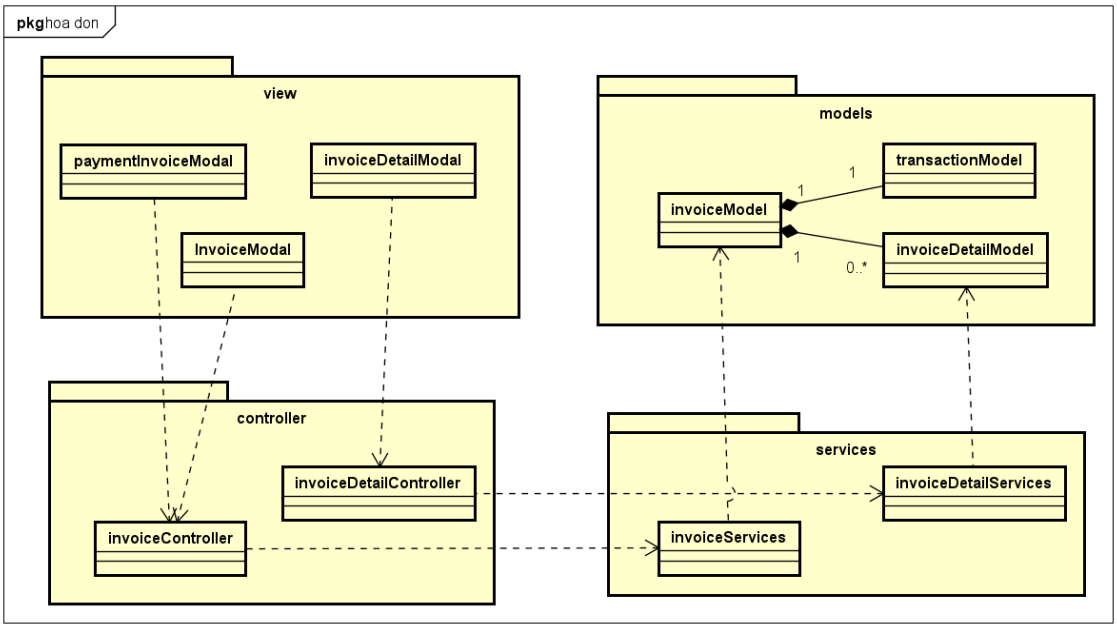
a, Class Diagram cho chức năng đặt lịch



Hình 4.4: Class Diagram cho chức năng đặt lịch

Hình 4.4 thể hiện class diagram cho chức năng đặt lịch. Gói view chứa các giao diện tương tác với hệ thống. Gói controller chứa bookingController, taskController được gọi khi đặt lịch bảo dưỡng. Khi có sự thay đổi về lịch đặt thì sẽ gửi thông báo thông qua sendNotiServices. Sau khi kết thúc quá trình sửa chữa thì invoiceModel, maintenanceScheduleModal cũng được tạo theo bookingModel.

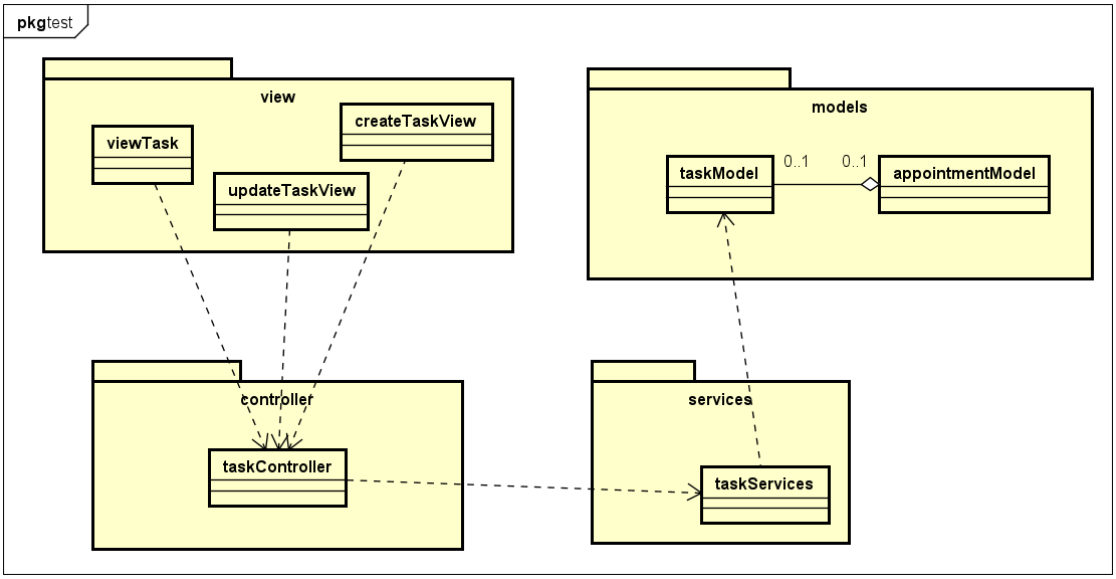
b, Class diagram cho hóa đơn



Hình 4.5: Class Diagram hóa đơn

Thông tin đơn giá của các mặt hàng, dịch vụ trong hóa đơn được tạo đi kèm với việc tạo hóa đơn. **Hình 4.5** thể hiện class diagram hóa đơn.

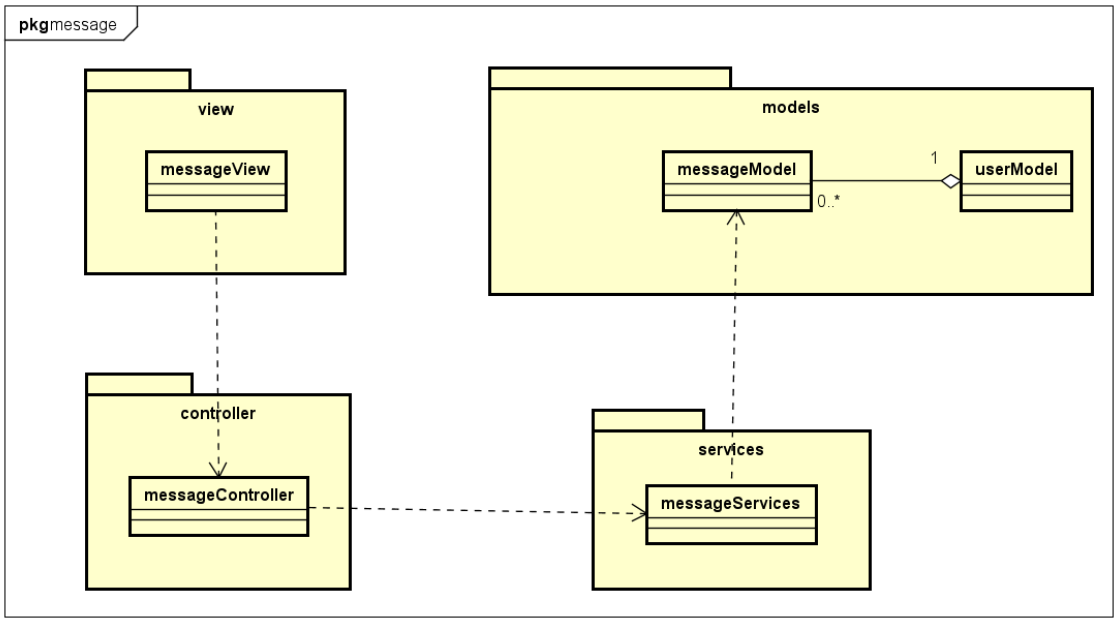
c, Class diagram cho chức năng giao nhiệm vụ



Hình 4.6: Class Diagram giao nhiệm vụ

Hình 4.6 thể hiện class diagram chức năng giao nhiệm vụ cụ thể: Nhiệm vụ sẽ do garage và thợ sửa chữa quản lý, khi nhiệm vụ được triển khai thì lịch làm việc của thợ sửa chữa cũng được cập nhật theo.

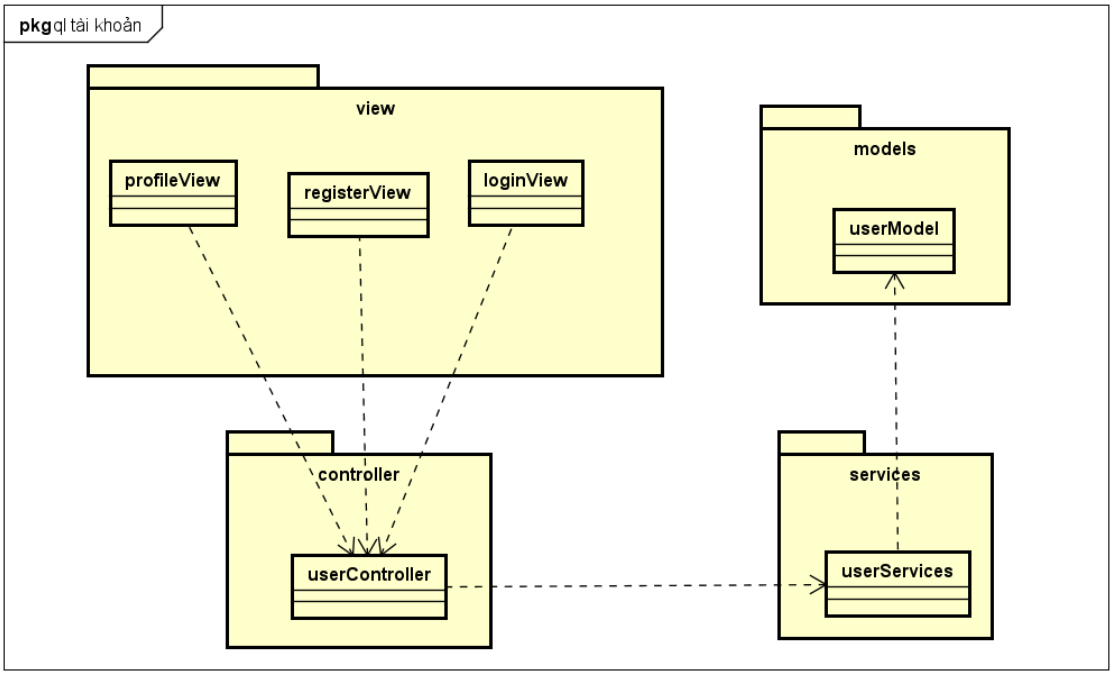
d, Class diagram cho chức năng nhắn tin



Hình 4.7: Class Diagram nhắn tin

Tin nhắn sẽ gắn với từng user giúp cho việc trao đổi giữa người dùng thuận tiện hơn. **Hình 4.7** thể hiện class diagram chức năng nhắn tin.

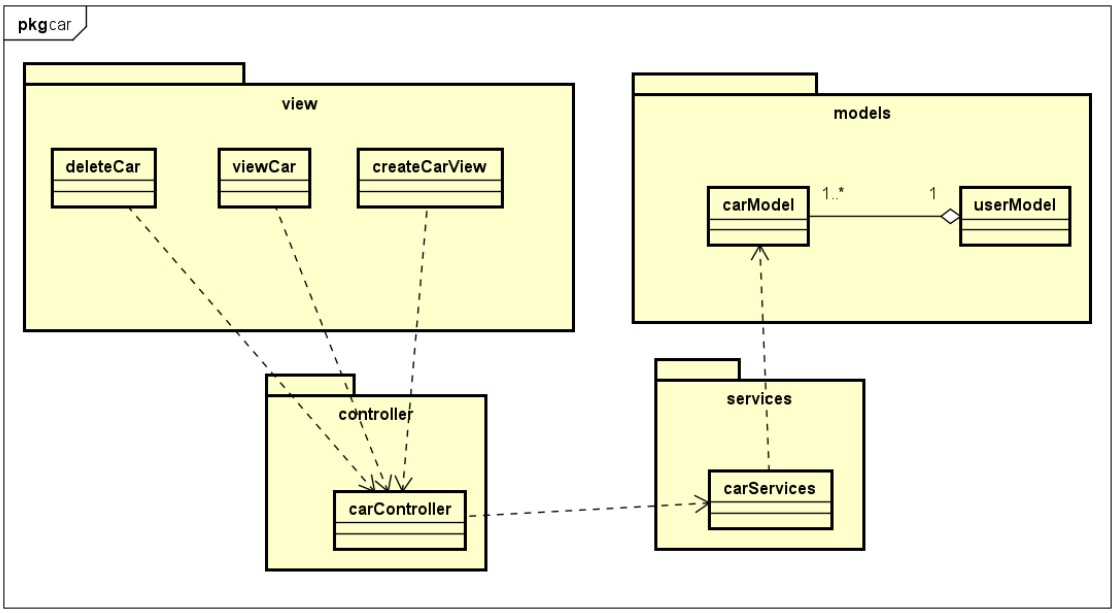
e, Class diagram cho chức năng quản lý tài khoản



Hình 4.8: Class Diagram quản lý tài khoản

Hình 4.8 trình bày class diagram cho chức năng quản lý tài khoản. Mỗi người dùng sẽ quản lý tài khoản cá nhân của mình sau khi đăng ký.

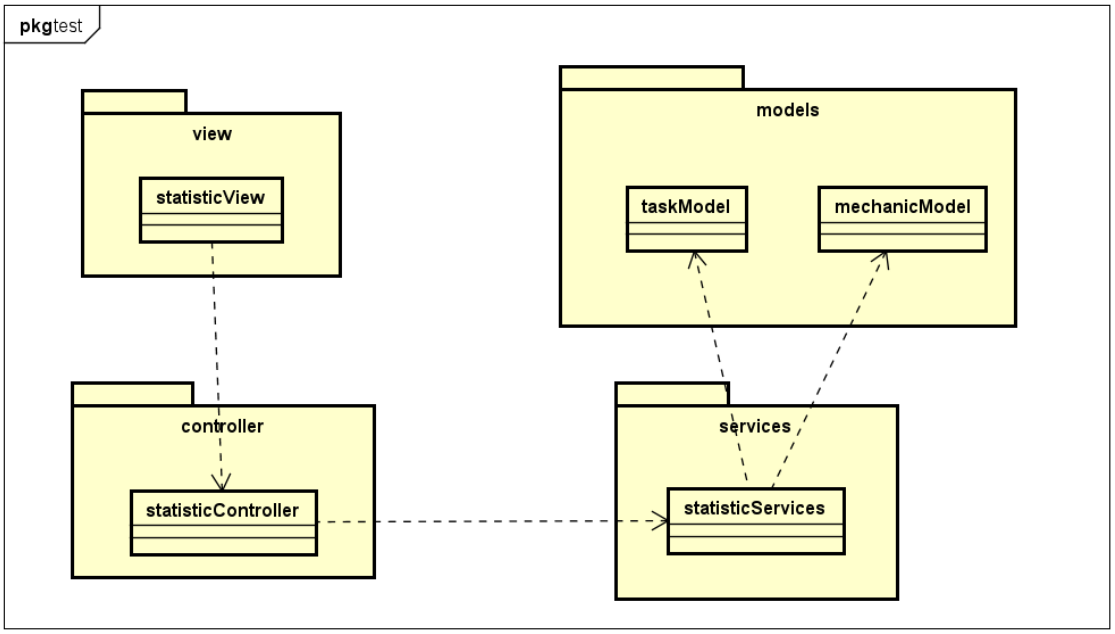
f, Class diagram cho chức năng quản lý xe



Hình 4.9: Class Diagram quản lý xe

Khác hàng sau khi tạo tài khoản sẽ quản lý xe của mình. Class diagram được mô tả ở **hình 4.9**.

g, Class diagram cho chức năng xem thống kê



Hình 4.10: Class Diagram xem thống kê

Hình 4.10 biểu thị class diagram cho chức năng xem thống kê. Garage có thể xem thống kê công việc theo ngày, tháng, năm. Thống kê khối lượng hoàn thành, xếp hạng của thợ sửa chữa trong garage.

4.2 Thiết kế chi tiết

4.2.1 Thiết kế giao diện

Hệ thống hướng tới việc tạo ra giao diện được thiết kế để tương thích tốt với các laptop phổ thông, đảm bảo rằng giao diện hệ thống sẽ hiển thị đẹp và rõ ràng trên hầu hết các thiết bị mà người dùng sử dụng.

Để đạt được sự thống nhất và chuẩn hóa trong thiết kế giao diện, các nguyên tắc đã được áp dụng:

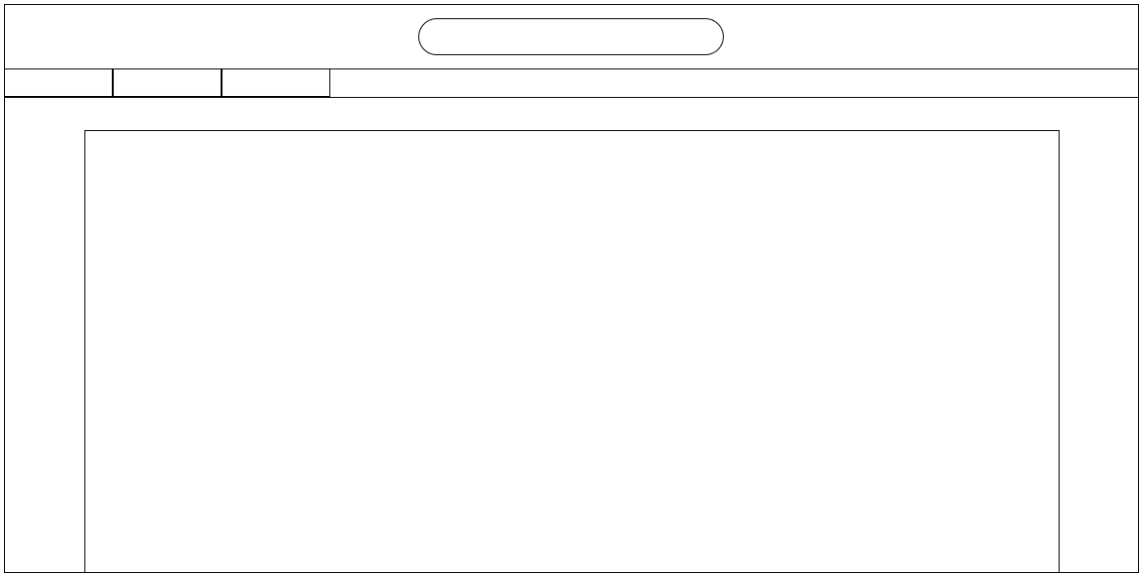
- Các nút và điều khiển được thiết kế với kích thước phù hợp để dễ dàng nhấn và tương tác trên bàn phím và touchpad của laptop. Màu sắc và biên viền được sử dụng để làm nổi bật các nút chức năng và các điều khiển, các vùng thông tin quan trọng được làm sáng màu giúp người dùng nhận diện dễ dàng.
- Thông điệp phản hồi được hiển thị ở góc màn hình với kích thước đủ lớn để người dùng có thể nhận thấy và đọc thông tin trả về từ hệ thống một cách dễ dàng.
- Sử dụng màu sắc hài hòa và tương phản tốt giữa nền và văn bản để đảm bảo độ đọc rõ ràng và thuận tiện.

a, Giao diện chung

Hình 4.11: Thiết kế giao diện chung

Hình 4.11 mô tả giao diện chung cho các vai trò với phần trên cùng chưa logo, thanh tìm kiếm, avatar. Cột điều hướng ở phía bên trái hiển thị các tùy chọn quản lý lịch hẹn, cập nhật thông tin cá nhân và các chức năng khác, trong khi đó cột ở bên phải chứa các thông tin chi tiết, nội dung chính của hệ thống.

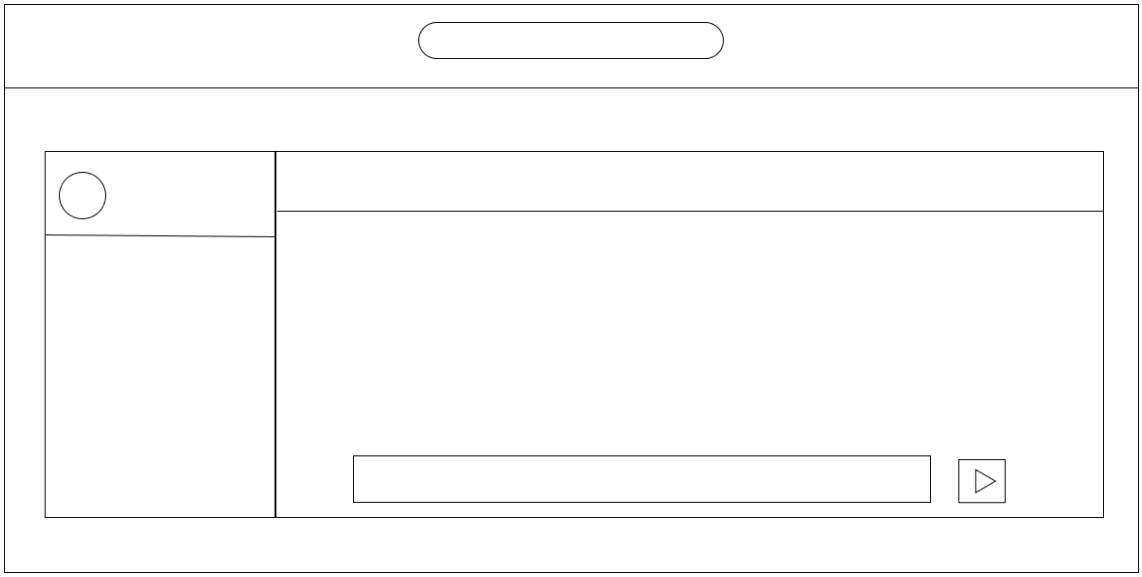
b, Giao diện phía khách hàng



Hình 4.12: Thiết kế giao diện phía khách hàng

Thiết kế giao diện phía khách hàng ở **hình 4.12** được mô tả như sau: Phần khách hàng sẽ có thanh điều hướng đến các dịch vụ của hệ thống, ở giữa là nội dung của dịch vụ như dịch vụ sủa chữa, bảo dưỡng, form điền thông tin.

c, Giao diện nhắn tin

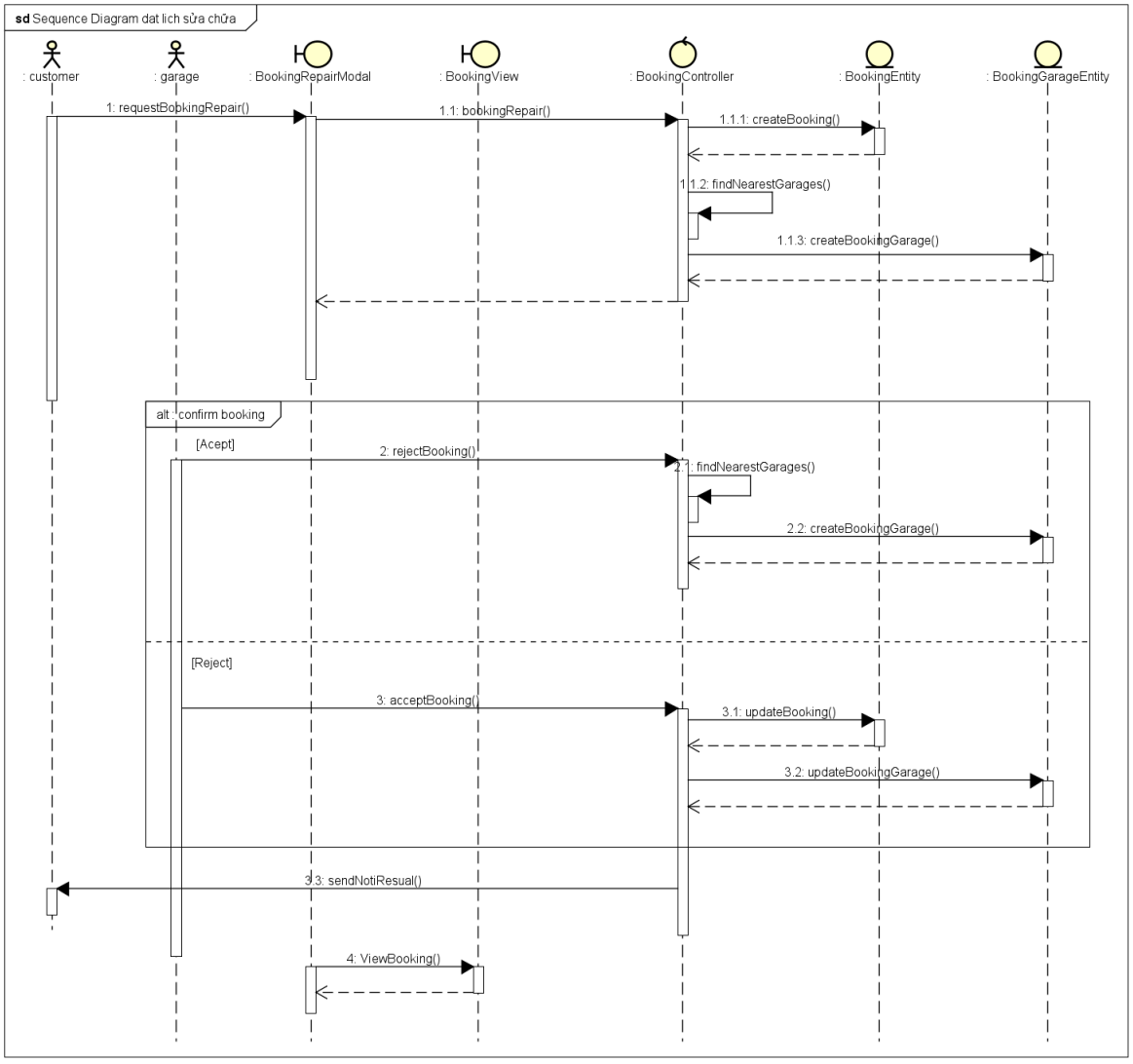


Hình 4.13: Thiết kế giao diện nhắn tin

Hình 4.12 minh họa giao diện tin nhắn với bố cục như sau: Phần nhắn tin được chia làm 2 phần, bên trái là danh sách chat, bên phải là nội dung cuộc trò chuyện.

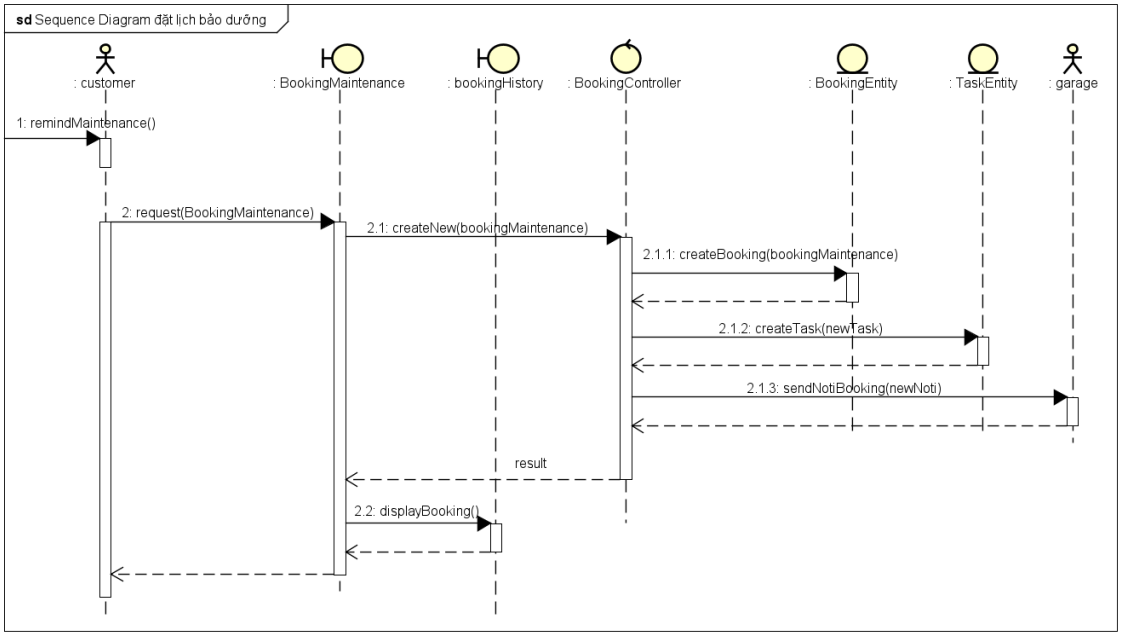
4.2.2 Thiết kế lớp

a, Thiết kế chi tiết các lớp cho các usecase đặt lịch



Hình 4.14: Biểu đồ trình tự đặt lịch sửa chữa

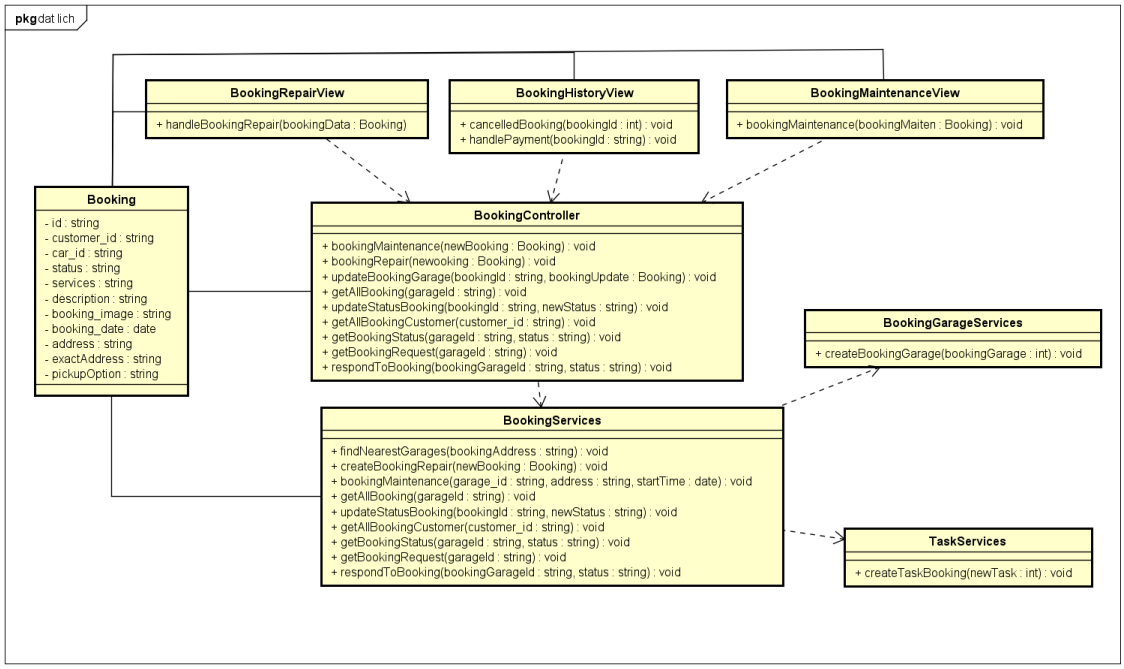
Hình 4.14 trình bày biểu đồ trình tự đặt lịch sửa chữa của khách hàng. Sau khi gửi yêu cầu sửa chữa, hệ thống tiến hành kiểm tra dữ liệu sau đó chọn ra 2 garage để gửi yêu cầu. Nếu cả 2 garage đều từ chối thì gửi tiếp tới 2 garage khác và sẽ dừng nếu không tìm thấy garage nào khác trong phạm vi.



Hình 4.15: Biểu đồ trình tự đặt lịch bảo dưỡng

Hình 4.15 trình bày biểu đồ trình tự đặt lịch bảo dưỡng. Sau khi người dùng sửa chữa, hệ thống lên lịch bảo dưỡng và thông báo cho người dùng khi đến lịch hẹn. Người dùng tiến hành đặt lịch với garage. Đồng thời công việc được thêm vào danh sách công việc của garage. Sau đó sẽ thông báo tới người dùng.

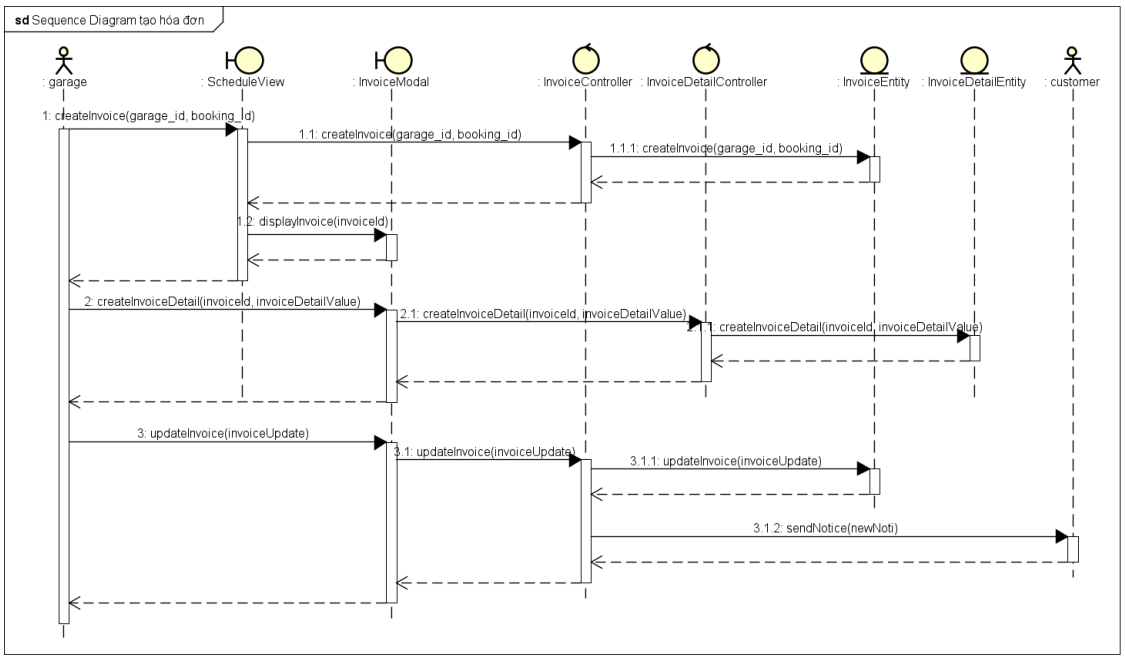
Hình 4.16 thể hiện sơ đồ lớp thiết kế cho chức năng đặt lịch.



Hình 4.16: Lớp thiết kế chức năng đặt lịch

b, Thiết kế chi tiết cho lớp usecase quản lý hóa đơn

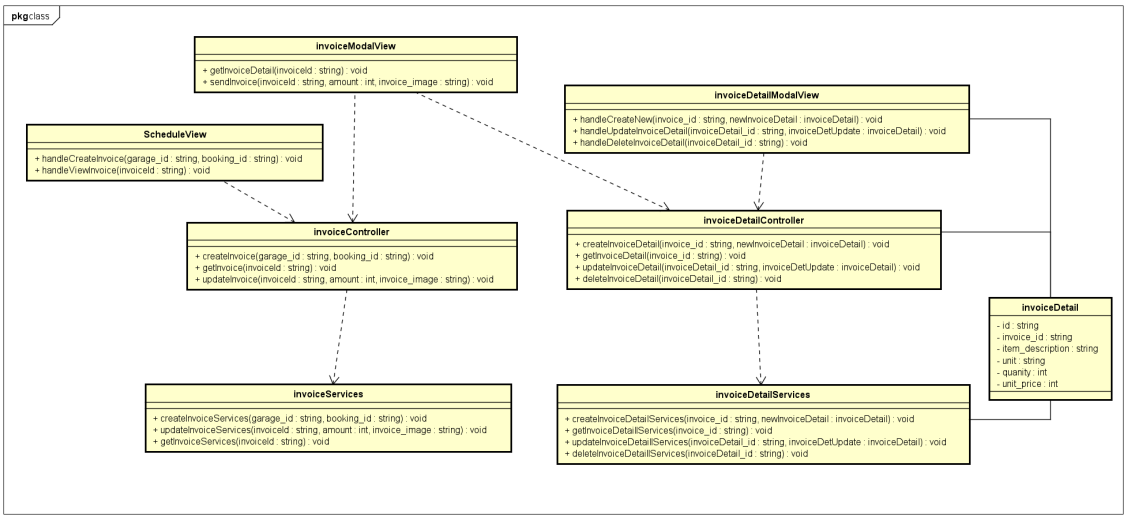
Hình 4.17 trình bày biểu đồ trình tự việc tạo hóa đơn của garage.



Hình 4.17: Biểu đồ trình tự tạo hóa đơn

Các bước tạo hóa đơn được mô tả theo các bước như sau: (i) Garage tạo hóa đơn, (ii) Thêm các sản phẩm, dịch vụ vào hóa đơn, (iii) Gửi hóa đơn đến khách hàng.

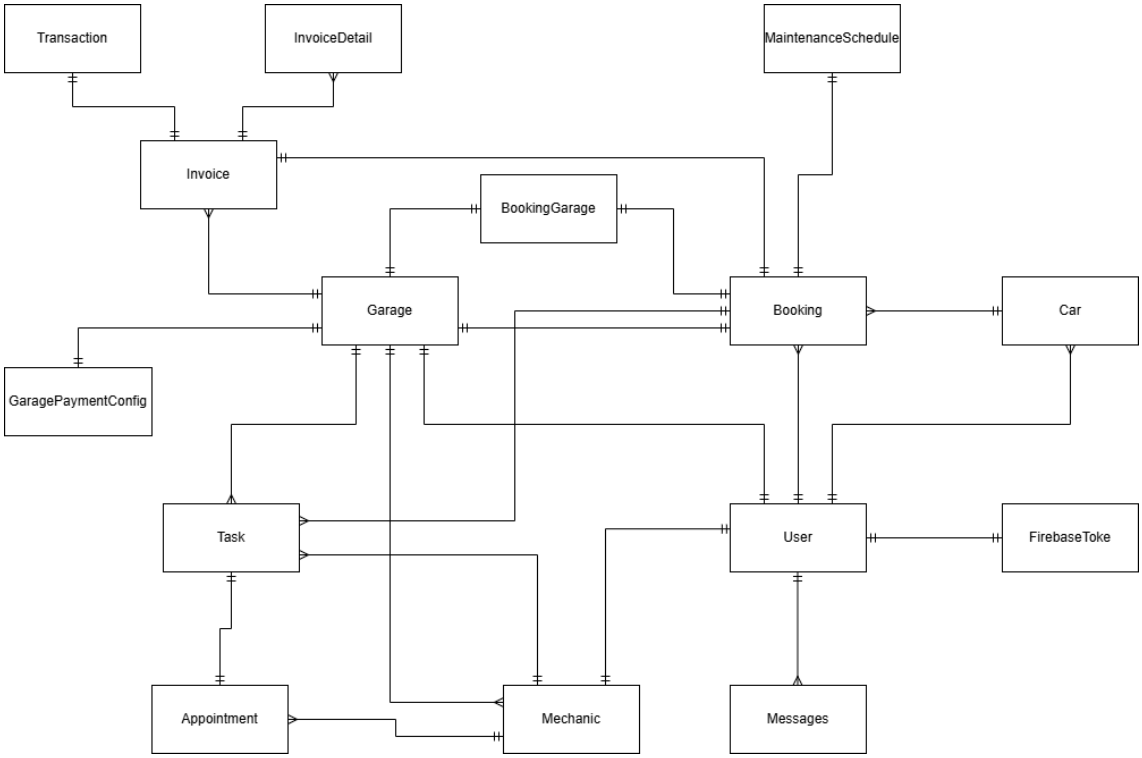
Hình 4.18 thể hiện sơ đồ lớp thiết kế cho lớp chức năng quản lý hóa đơn.



Hình 4.18: Lớp thiết kế quản lý hóa đơn

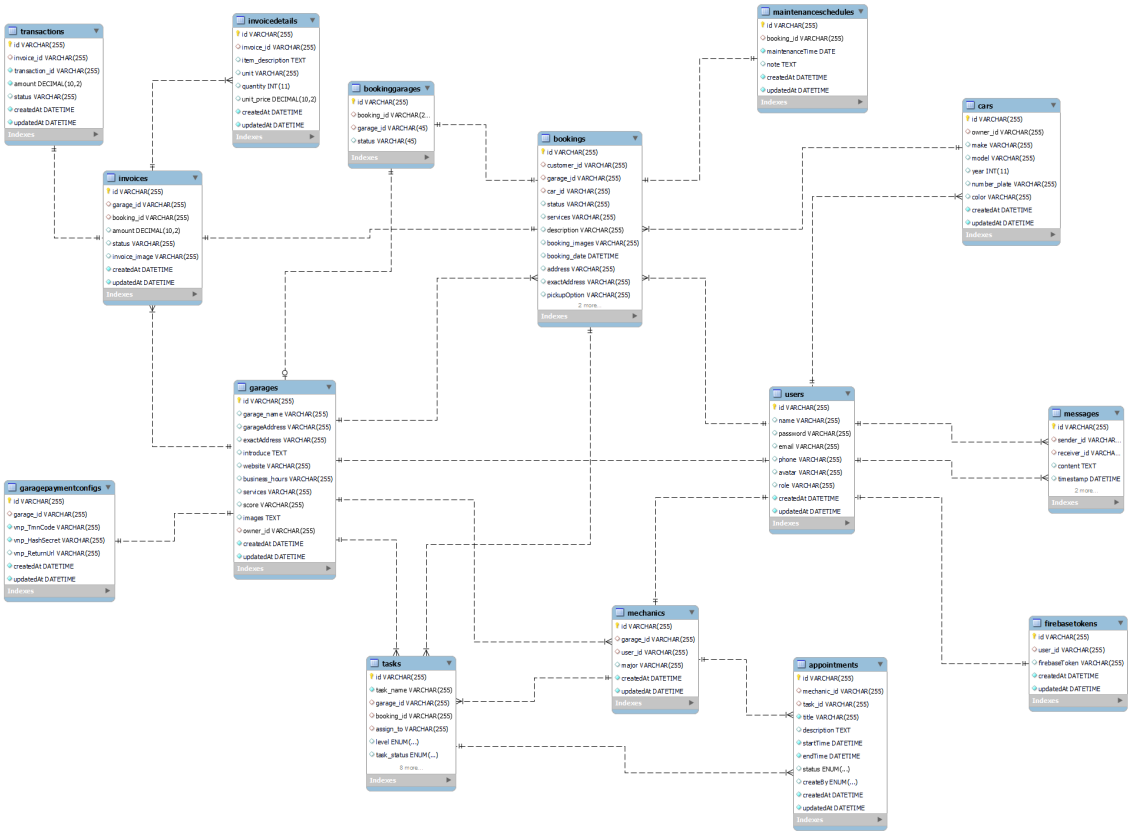
4.2.3 Thiết kế cơ sở dữ liệu

Cơ sở dữ liệu của hệ thống đặt lịch sửa chữa ô tô được thiết kế cho 3 đối tượng là khách hàng, garage và thợ sửa chữa. Sơ đồ thực thể liên kết minh họa ở **hình 4.19**:



Hình 4.19: Sơ đồ thực thể liên kết

Từ sơ đồ thực thể liên kết, em thiết kế cơ sở dữ liệu với hệ quản trị cơ sở dữ liệu mySql. Chi tiết mô tả ở **hình 4.20** :



Hình 4.20: Thiết kế cơ sở dữ liệu

Ý nghĩa của các bảng:

- **Transactions**: Lưu thông tin giao dịch thanh toán.
- **InvoiceDetails:** Thông tin đơn giá của các mặt hàng, dịch vụ trong hóa đơn.
- **Invoice:** Hóa đơn
- **BookingGarages:** Liên kết giữa garage và bookings để ghi nhận quá trình xác nhận lịch đặt.
- **Garages:** Thông tin chi tiết về garage.
- **GaragePaymentConfigs:** Thông tin cấu hình thanh toán của garage.
- **Tasks:** Nhiệm vụ của garage.
- **Mechanics:** Thợ sửa chữa.
- **Appointments:** Lịch làm việc của thợ.
- **Bookings:** Thông tin đặt lịch.
- **MaintenanceSchedule:** Lịch bảo dưỡng.
- **Cars:** Ô tô của khách hàng.
- **Users:** Thông tin cơ bản của người dùng.
- **FirebaseTokens:** Lưu token firebase.
- **Messages:** Tin nhắn.

Mô tả chi tiết:

Bảng 4.1 liệt kê chi tiết các thuộc tính của bảng Bookings.

#	Thuộc tính	Kiểu dữ liệu	Ý nghĩa
1	id	varchar(255)	id
2	customer_id	varchar(255)	customer id
3	garage_id	varchar(255)	garage id
4	car_id	varchar(255)	car id
5	status	enum("request","schedule","reject", "in-progress","complete", "cancelled")	trạng thái lịch đặt
6	services	varchar(255)	dịch vụ đặt lịch
7	description	varchar(255)	mô tả trạng thái xe
8	booking_images	varchar(255)	hình ảnh
9	booking_date	DateTime	ngày đặt lịch
10	address	varchar(255)	địa chỉ
11	exactAddress	varchar(255)	tọa độ khách hàng
12	pickupOption	varchar(255)	lựa chọn kiểu giao xe

Bảng 4.1: Thuộc tính bảng bookings

Bảng 4.2 liệt kê chi tiết các thuộc tính của bảng Tasks.

#	Thuộc tính	Kiểu dữ liệu	Ý nghĩa
1	id	varchar(255)	id
2	task_name	varchar(255)	nội dung nhiệm vụ
3	garage_id	varchar(255)	garage id
4	booking_id	varchar(255)	booking id
5	assign_to	varchar(255)	mechanic id
6	level	enum("easy","medium","hard")	độ khó của nhiệm vụ
7	task_status	enum("pending","assigned", "in_progress","completed")	trạng thái nhiệm vụ
8	allocation_date	date	ngày giao nhiệm vụ
9	estimated_time	int(11)	thời gian dự kiến hoàn thành
10	start_date	date	ngày bắt đầu thực hiện
11	start_time	time	giờ bắt đầu thực hiện
12	end_date	date	ngày hoàn thành
13	end_time	time	giờ hoàn thành

Bảng 4.2: Thuộc tính bảng tasks

Bảng 4.3 liệt kê chi tiết các thuộc tính của bảng Invoices.

#	Thuộc tính	Kiểu dữ liệu	Ý nghĩa
1	id	varchar(255)	id
2	garage_id	varchar(255)	garage id
3	booking_id	varchar(255)	booking id
4	amount	decima	số tiền
5	status	varchar(255)	trạng thái thanh toán
6	invoice_image	varchar(255)	hình ảnh hóa đơn

Bảng 4.3: Thuộc tính bảng invoices

4.3 Xây dựng ứng dụng

4.3.1 Thư viện và công cụ sử dụng

. **Bảng 4.4** liệt kê danh sách thư viện và công cụ sử dụng.

Mục đích	Công cụ	Địa chỉ URL
IDE lập trình	Visual Studio Code 64 bit	htts://code.visualstudio.com
Kiểm thử	Postman	https://www.postman.com
Quản trị cơ sở dữ liệu	MySql Workbench	https://www.mysql.com
Ứng dụng vẽ sơ đồ	Astah UML	https://astah.net
Thư viện UI	Ant Design 5.16.2	https://design
Thư viện CSS	TaiwindCSS	https://tailwindcss.com
Framework backend	NodeJS	htts://nodejs.org
Framework frontend	ReactJS	https://reactjs.org
Gửi thông báo đẩy	Firebase Cloud Messaging	https://firebase.google.com

Bảng 4.4: Danh sách thư viện và công cụ sử dụng

4.3.2 Kết quả đạt được

Hệ thống đặt lịch sửa chữa, bảo dưỡng đã đáp ứng được yêu cầu cơ bản của người dùng và việc quản lý ở phía garage.

Người dùng có thể: (i) đặt lịch hẹn, (ii) theo dõi quá trình, (iii) thanh toán hóa đơn. Garage có các chức năng: (i) quản lý lịch đặt, (ii) quản lý công việc, (iii) tạo hóa đơn, (iv)lên lịch bảo dưỡng, (v) xem thống kê kết quả làm việc của thợ sửa chữa. Thợ sửa chữa có thể: (i) quản lý công việc được giao, (ii) quản lý thời gian làm việc của mình. Hệ thống còn tích hợp nhắn tin giúp người sử dụng giao tiếp thuận tiện hơn.

Một số thông số kỹ thuật về dự án được cung cấp ở **bảng 4.5**:

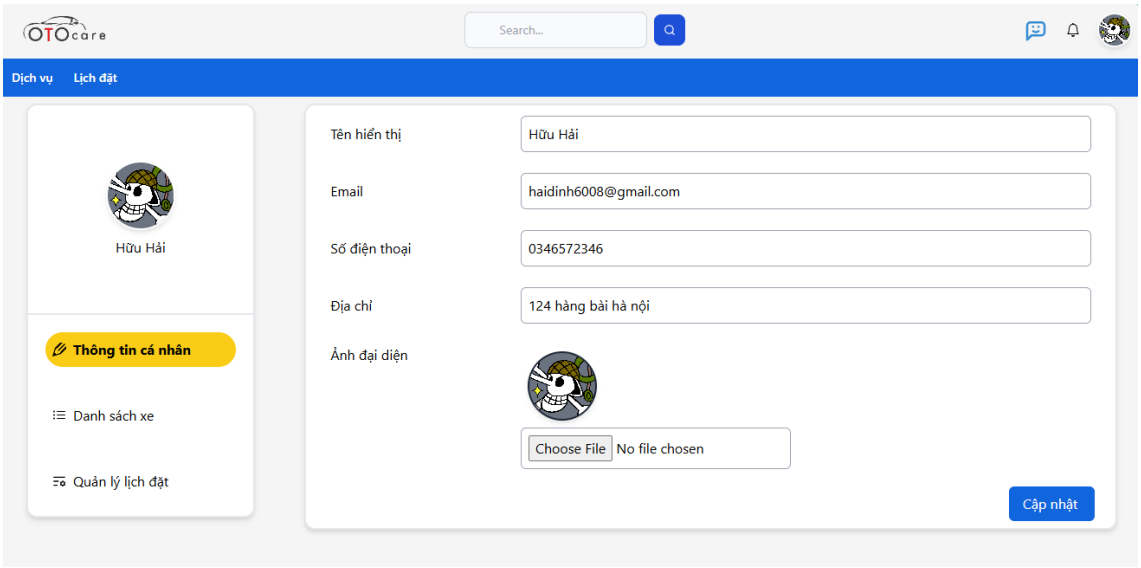
ST	Thông tin	Thống kê
1	Số file mã nguồn (file js)	224
2	Số tệp	32
3	Số bảng CSDL	15

Bảng 4.5: Một số thông số kỹ thuật về dự án

4.3.3 Minh họa các chức năng chính

Dưới đây là giao diện các màn hình chức năng của hệ thống:

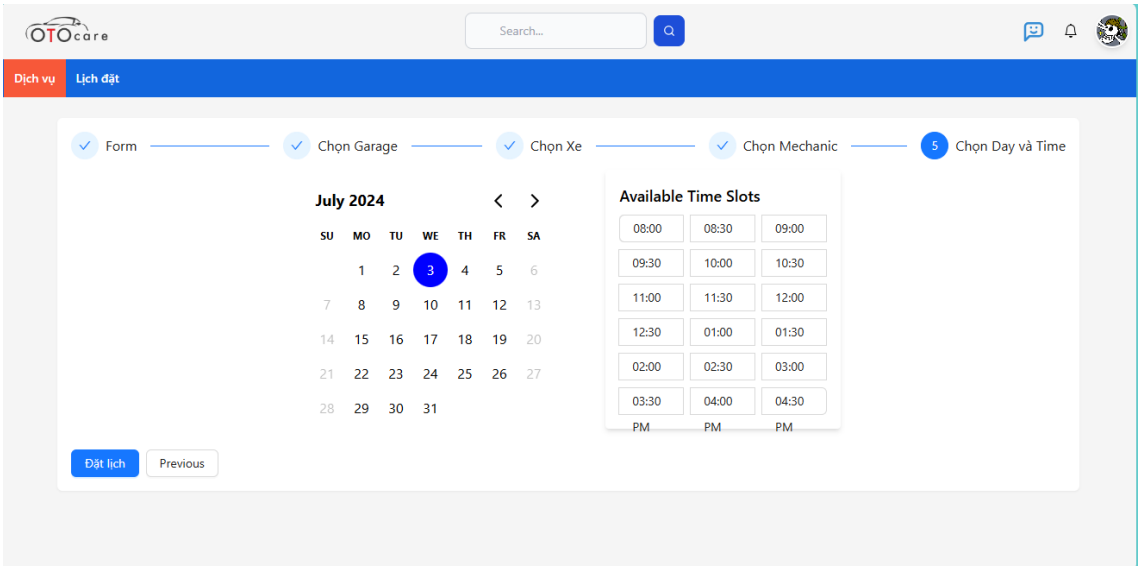
a, Chức năng cập nhật thông tin cá nhân



Hình 4.21: Giao diện thông tin cá nhân

Trang cá nhân gồm các thông tin liên quan đến tên, số điện thoại, địa chỉ được minh họa ở **hình 4.21**.

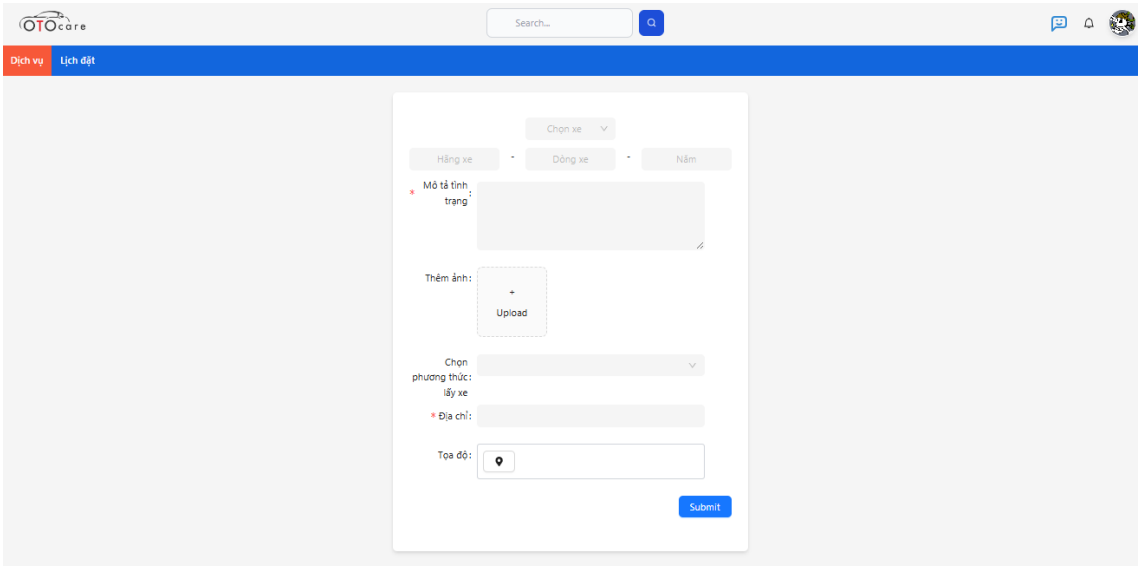
b, Chức năng đặt lịch bảo trì



Hình 4.22: Giao diện đặt lịch bảo trì

Khách hàng hoàn thành theo các bước và nhấn đặt lịch. Minh họa **hình 4.22**.

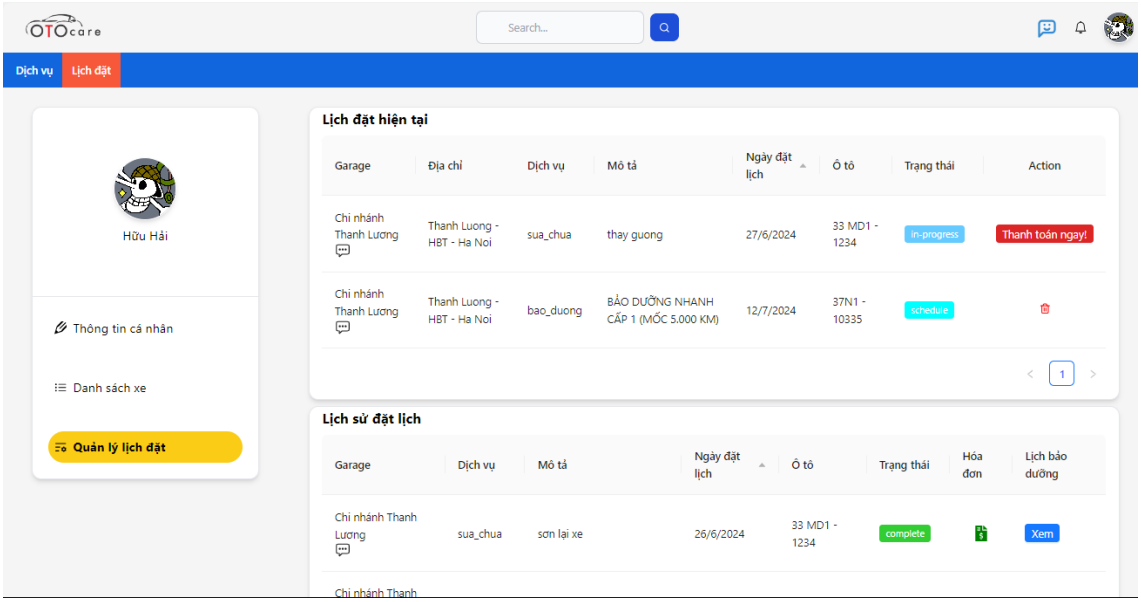
c, Chức năng đặt lịch sửa chữa



Hình 4.23: Giao diện đặt lịch sửa chữa

Hình 4.23 hiển thị form cho người dùng điền các thông tin cần thiết để đặt lịch sửa chữa.

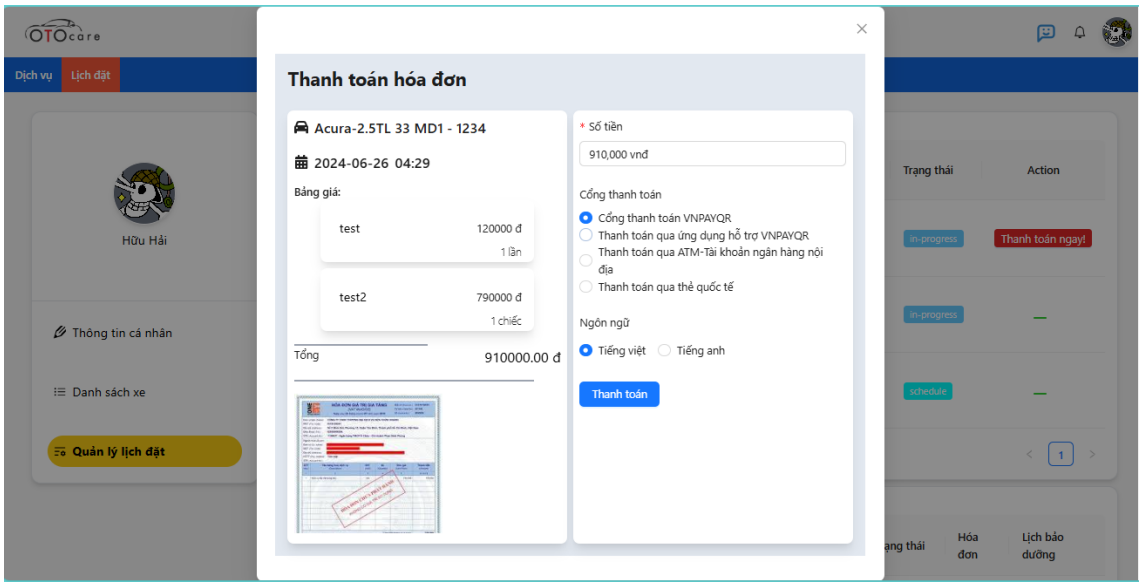
d, Chức năng quản lý lịch đặt



Hình 4.24: Giao diện quản lý lịch đặt

Hình 4.24 minh họa giao diện cho chức năng quản lý lịch đặt. Khách hàng có thể hủy sau khi đặt lịch và thanh toán sau khi garage gửi hóa đơn bằng cách nhấn vào biểu tượng ở cột action. Khi nhấn vào địa chỉ thì màn hình chuyển sang google map tại địa chỉ của garage. Đồng thời có thể chat với garage khi nhấn vào biểu tượng một bên tên gagrage.

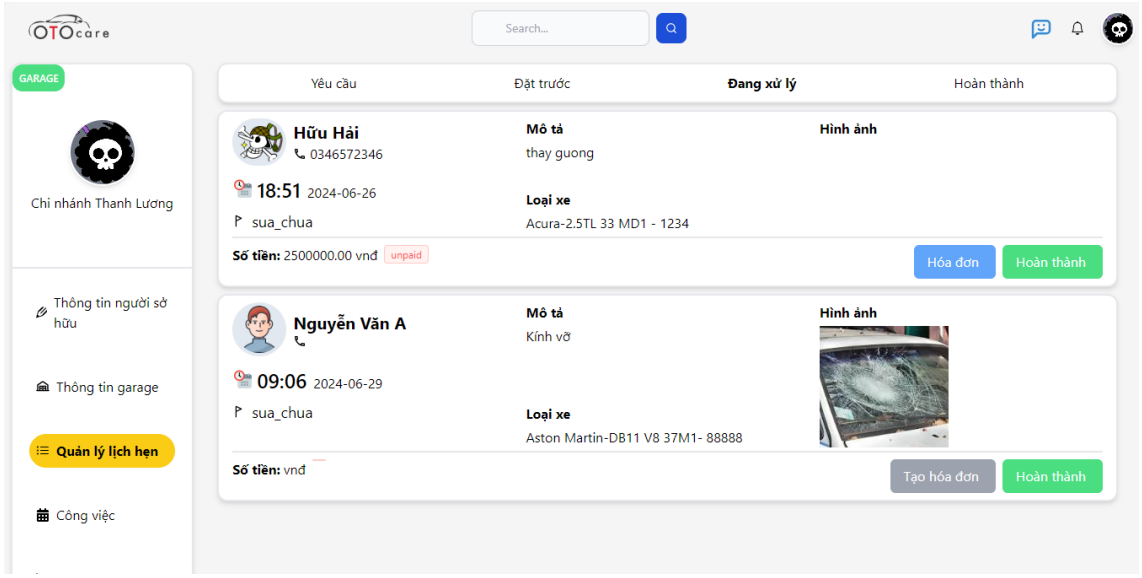
e, Chức năng thanh toán hóa đơn



Hình 4.25: Giao diện nhấn thanh toán hóa đơn

Khách hàng có thể điều chỉnh số tiền thanh toán và chọn phương thức thanh toán qua vnpay. Minh họa ở **hình 4.25**.

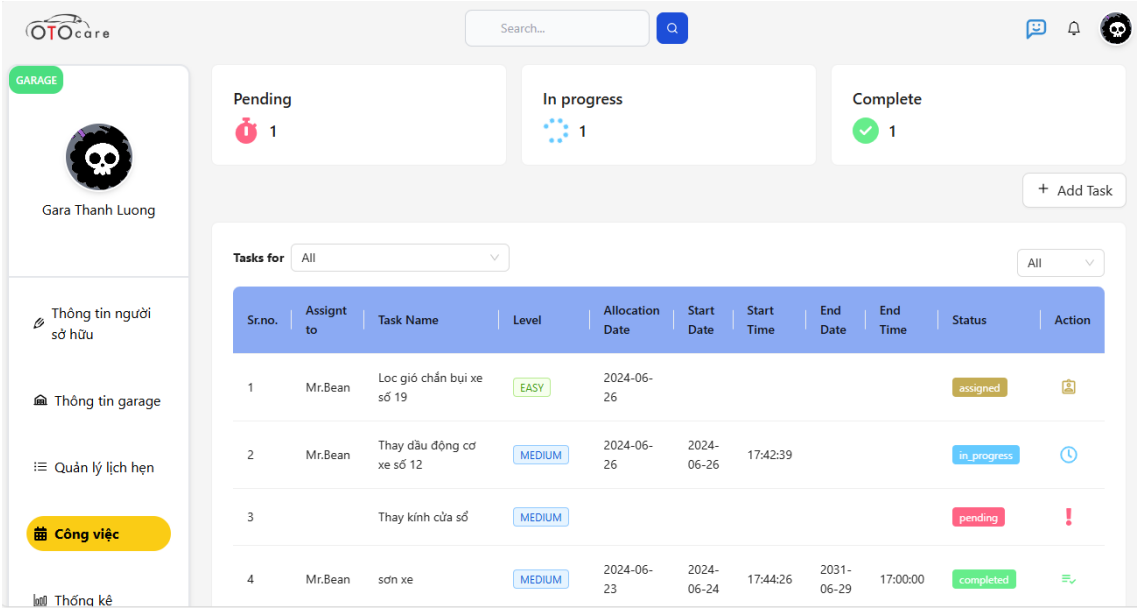
f, Chức năng quản lý lịch hẹn của garage



Hình 4.26: Giao diện garage quản lý lịch hẹn ở trạng thái đang xử lý

Hình 4.26 thể hiện giao diện garage quản lý lịch hẹn ở trạng thái đang xử lý. Garage xác nhận hoàn thành sửa chữa bằng cách nhấn vào button "Hoàn thành". Đồng thời tạo hóa đơn khi nhấn vào button "Tạo hóa đơn", và sau khi tạo thì nhấn vào button "Hóa đơn" để xem chi tiết. Có thể nhắn tin với khách hàng khi nhấn vào tên khách hàng hoặc icon "chat" ở phần header.

g, Chức năng quản lý công việc

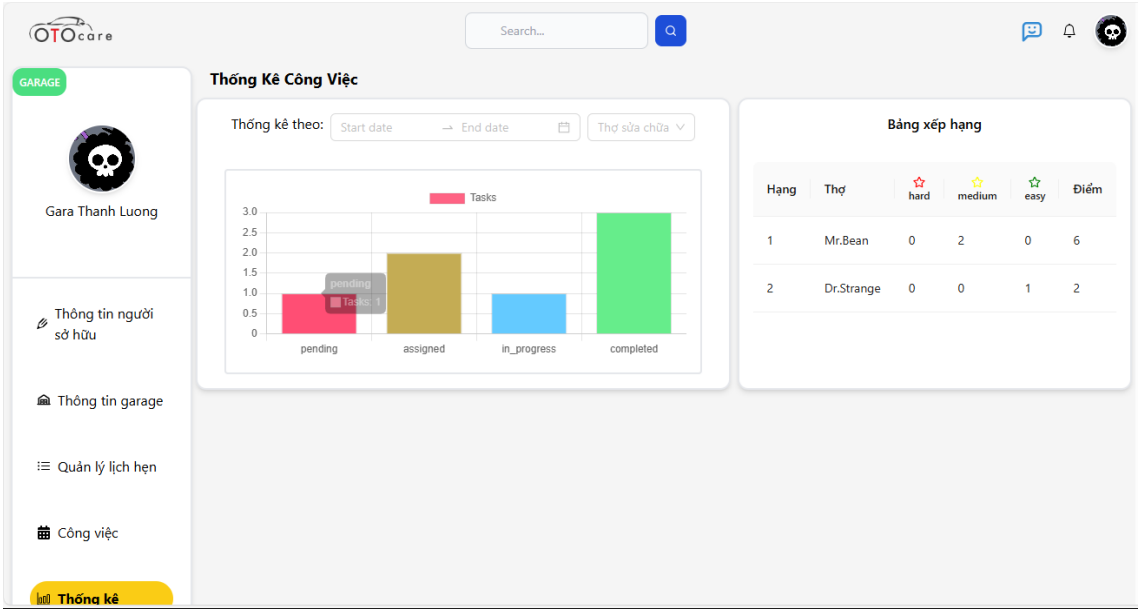


Hình 4.27: Giao diện garage quản lý công việc

Hình 4.27 thể hiện giao diện quả lý công việc phía garage được mô tả như sau: Khi click vào button "Add task" thì form popup hiện lên. Công việc được cập nhật bằng cách click vào các biểu tượng ở cột Action của bảng. Garage có thể lọc task theo xe (bên trái bảng) và theo trạng thái (bên phải bảng).

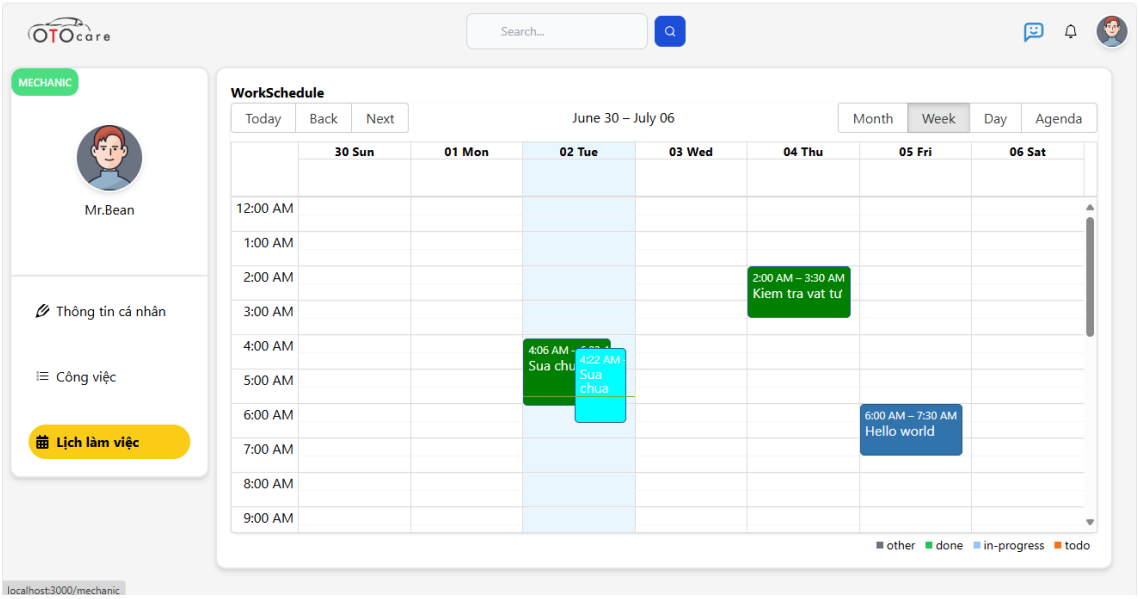
h, Chức năng xem thống kê

Hình 4.28 mô tả giao diện thống kê công việc của garage và thợ sửa chữa.



Hình 4.28: Giao diện thống kê công việc

i, Chức năng quản lý lịch làm việc của thợ

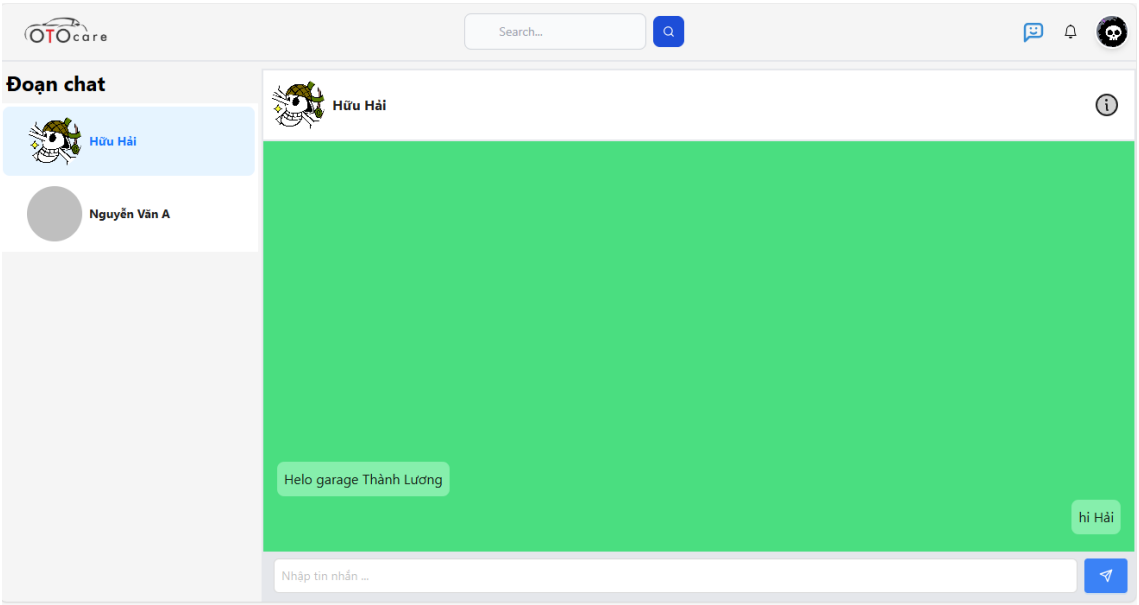


Hình 4.29: Giao diện quản lý lịch làm việc

Thợ sửa chữa có thể thay đổi lịch làm việc bằng cách chọn vùng thời gian sau đó popup điền thông tin hiện lên. Giao diện quản lý lịch làm việc được minh họa ở **hình 4.29**.

j, Chức năng quản nhắn tin

Hình 4.30 minh họa giao diện nhắn tin.



Hình 4.30: Giao diện nhắn tin

4.4 Kiểm thử

4.4.1 Kiểm thử cho chức năng quản lý công việc

Bảng 4.6 mô tả quy trình và kết quả kiểm thử cho chức năng quản lý công việc.

Bảng 4.6: Bảng kiểm thử chức năng quản lý công việc

ID	Trường hợp kiểm thử	Quy trình kiểm thử	Kết quả mong muốn	Kết quả thực hiện	Kết luận
1	Garage tạo công việc	1. Vào màn hình quản lý công việc 2. Nhấn vào nút "Add Task"	Hiển thị form nhập thông tin công việc	Hiển thị modal thành công	PASS
2	Garage giao công việc cho thợ sửa chữa	1. Garage nhấn vào icon ở cột Action 2. Chọn trạng thái Assigned và chọn thợ 3. Nhấn nút cập nhật	Hiển thị modal công việc khi garage nhấn vào icon, thông báo cập nhật thành công	Hiển thị modal thành công, thông báo cập nhật thành công	PASS
3	Garage và thợ cùng cập nhật 1 công việc	1. Garage cập nhật trạng thái complete cho 1 công việc nào đó 2. Thợ sửa chữa cập nhật trạng thái in-progress	Thông báo cập nhật thất bại	Thông báo cập nhật thất bại	PASS
4	Lọc công việc theo trạng thái	1. Garage vào màn hình quản lý công việc 2. Garage chọn trạng thái assigned	Hiển thị các công việc có trạng thái assigned	Hiển thị các công việc có trạng thái assigned thành công	PASS

4.4.2 Kiểm thử cho chức năng thanh toán

Bảng 4.7 mô tả quy trình và kết quả kiểm thử cho chức năng thanh toán.

Bảng 4.7: Bảng kiểm thử chức năng thanh toán

ID	Trường hợp kiểm thử	Quy trình kiểm thử	Kết quả mong muốn	Kết quả thực hiện	Kết luận
1	Xem hóa đơn trước khi thanh toán	1. Vào màn hình quản lý lịch đặt 2. Nhấn vào nút "Thanh toán ngay"	Hiển thị modal hóa đơn và các lựa chọn phương thức thanh toán	Hiển thị modal thành công	PASS
2	Chuyển hướng sang màn hình thanh toán	1. Vào xem hóa đơn trước khi thanh toán 2. Chọn phương thức thanh toán 3. Nhấn nút "thanh toán"	Chuyển hướng sang màn hình thanh toán	Chuyển hướng thành công	PASS
3	Thông báo khi thanh toán thành công	1. Từ màn hình thanh toán của vnpay 2. Nhấn nút thanh toán	Chuyển hướng người dùng về lại ứng dụng, Gửi thông báo kết quả tới garage	Chuyển hướng thành công, Gửi thông báo tới garage thành công	PASS

4.5 Kiểm thử chức năng đặt lịch

Bảng 4.8 mô tả quy trình và kết quả kiểm thử cho chức năng đặt lịch.

Bảng 4.8: Bảng kiểm thử chức năng đặt lịch

ID	Trường hợp kiểm thử	Quy trình kiểm thử	Kết quả mong muốn	Kết quả thực hiện	Kết luận
1	Đặt lịch với thông tin bị thiếu	1. Vào màn hình đặt lịch sửa chữa 2. Điền thiếu thông tin 3. Nhấn gửi	Thông báo thiếu trường cần nhập	Thông báo thiếu trường cần nhập	PASS

Tiếp tục ở trang sau

Bảng 4.8 – tiếp theo từ trang trước

ID	Trường hợp kiểm thử	Quy trình kiểm thử	Kết quả mong muốn	Kết quả thực hiện	Kết luận
2	Đặt lịch với đầy đủ thông tin	1. Vào màn hình đặt lịch sửa chữa 2. Điền đầy đủ thông tin 3. Nhấn gửi	Tạo lịch đặt thành công và thông báo tới garage	Tạo lịch đặt thành công và thông báo thành công	PASS
3	Garage nhận lịch hẹn	1. Vào màn hình quản lý lịch đặt 2. Vào mục yêu cầu 3. Nhấn nút xác nhận	Thông báo xác nhận lịch hẹn tới khách hàng	Thông báo xác nhận lịch hẹn thành công	PASS
4	Garage từ chối lịch hẹn	1. Vào màn hình quản lý lịch đặt 2. Vào mục yêu cầu 3. Nhấn nút từ chối	Gửi yêu cầu tới 2 garage khác khi cả 2 garage đều từ chối	Gửi yêu cầu tới 2 garage khác khi cả 2 garage đều từ chối	PASS
5	Đặt lịch bảo dưỡng thành công với garage	1. Vào màn hình đặt lịch bảo dưỡng 2. Nhập thông tin bảo dưỡng 3. Nhấn nút gửi	Đặt lịch thành công, gửi thông báo tới garage, gửi mail tới khách hàng khi đến lịch	Đặt lịch thành công, gửi thông báo tới garage thành công, gửi mail tới khách hàng thành công	PASS

Tiếp tục ở trang sau

Bảng 4.8 – tiếp theo từ trang trước

ID	Trường hợp kiểm thử	Quy trình kiểm thử	Kết quả mong muốn	Kết quả thực hiện	Kết luận
6	Garage thứ hai chấp nhận lịch sửa chữa sau khi garage thứ nhất đã chấp nhận	1. Garage thứ nhất vào quản lý đặt lịch mục yêu cầu 2. Nhấn nút xác nhận 3. Garage thứ hai vào quản lý đặt lịch mục yêu cầu 4. Nhấn nút xác nhận	Thông báo xác nhận không thành công và xóa lịch đặt	Thông báo xác nhận không thành công và xóa lịch đặt	PASS

4.6 Triển khai

Ứng dụng đang được thực thi cục bộ trên môi trường localhost, với các tham số máy chủ thử nghiệm được cho ở bảng 4.9.

Thông số kỹ thuật	Chi tiết
RAM	16Gb
Hệ điều hành	Window 11
Dung lượng phần cứng	SSD 476Gb
CPU	Intel Core i5-1035G1

Bảng 4.9: Bảng thông số server

CHƯƠNG 5. CÁC GIẢI PHÁP VÀ ĐÓNG GÓP NỔI BẬT

5.1 Quản lý đồng bộ dữ liệu

5.1.1 Đặt vấn đề

Vấn đề: Xung đột dữ liệu khi cả hai garage xử lý lịch đặt cùng lúc.

Trong hệ thống đặt lịch, bảng Booking sẽ lưu các yêu cầu đặt chỗ từ người dùng, bao gồm các trường như customerId và garageId. Khi một người dùng tạo yêu cầu đặt chỗ, garageId chưa được cập nhật ngay lập tức. Thay vào đó, hệ thống sẽ tạo ra hai bản ghi trong bảng BookingGarage, chứa bookingId, garageId. Khi một trong hai garage chấp nhận yêu cầu, garageId trong bảng Booking sẽ được cập nhật.

Vấn đề nảy sinh khi hai garage cùng xử lý một yêu cầu đặt chỗ tại cùng một thời điểm. Điều này có thể dẫn đến việc cập nhật trạng thái của lịch đặt không nhất quán.

Một hệ thống không có cơ chế kiểm soát xung đột hiệu quả có thể gặp phải các vấn đề sau:

1. Cập nhật trạng thái không nhất quán: Khi hai garage xử lý yêu cầu đặt chỗ cùng lúc, có khả năng cả hai cùng chấp nhận hoặc từ chối yêu cầu, dẫn đến trạng thái không nhất quán trong hệ thống.
2. Xung đột dữ liệu: Sự thiếu đồng bộ trong việc xử lý các yêu cầu đặt chỗ có thể gây ra xung đột dữ liệu, làm giảm tính toàn vẹn và tin cậy của hệ thống.
3. Thiếu nhất quán trong cập nhật trạng thái: Khi các sự kiện được xử lý đồng thời mà không có cơ chế kiểm soát xung đột, trạng thái của lịch đặt có thể không phản ánh đúng thực tế, gây nhầm lẫn cho cả người dùng và garage.

5.1.2 Giải pháp và kết quả đạt được

Để giải quyết vấn đề xung đột dữ liệu có nhiều kỹ thuật khác nhau tiêu biểu là:

1. Sử dụng kỹ thuật Locking trên cơ sở dữ liệu: đảm bảo rằng một tài nguyên chỉ có thể được sửa đổi bởi một quá trình tại một thời điểm. Có hai loại khóa phổ biến:
    - Khóa đọc (Read Lock): Cho phép nhiều người đọc nhưng không cho phép viết.
    - Khóa ghi (Write Lock): Chỉ cho phép một quá trình ghi và ngăn chặn tất cả các quá trình đọc và ghi khác cho đến khi quá trình ghi hoàn tất.
2. Sử dụng kỹ thuật Versioning trên cơ sở dữ liệu:

Versioning giúp theo dõi các thay đổi của dữ liệu và giải quyết các xung đột bằng cách duy trì các phiên bản khác nhau của tài liệu hoặc dữ liệu. Mỗi lần một bản ghi được cập nhật, một phiên bản mới của bản ghi đó được tạo ra. Khi một garage chấp nhận yêu cầu đặt chỗ, nó sẽ cập nhật bản ghi với một phiên bản mới. Nếu một garage khác cố gắng cập nhật cùng bản ghi, hệ thống sẽ kiểm tra phiên bản và ngăn chặn cập nhật nếu phiên bản đã thay đổi, yêu cầu garage đó lấy lại dữ liệu mới nhất trước khi thử cập nhật lại.

3. Giao dịch (Transactions) và đảm bảo tính nguyên tử (Atomicity): Sử dụng transactions để đảm bảo rằng tất cả các thao tác trên cơ sở dữ liệu đều nguyên tử, tức là hoặc tất cả các thao tác đều thành công hoặc tất cả đều thất bại. Điều này ngăn chặn trạng thái không nhất quán nếu có lỗi xảy ra trong quá trình cập nhật.

4. Sử dụng cờ trạng thái (Status Flags): Thiết lập các cờ trạng thái trong bảng BookingGarage để chỉ rõ trạng thái xử lý của từng yêu cầu đặt chỗ. Ví dụ, cờ processing có thể được sử dụng để đánh dấu rằng một yêu cầu đang được xử lý bởi một garage, ngăn chặn garage khác xử lý cùng yêu cầu đó cho đến khi trạng thái được cập nhật.

Bằng cách áp dụng các giải pháp này, hệ thống của bạn sẽ giảm thiểu được các xung đột và duy trì tính toàn vẹn của dữ liệu, đảm bảo trạng thái của các yêu cầu đặt chỗ luôn nhất quán và chính xác.

Trong hệ thống của mình em đã áp dụng tạo cờ trạng thái cho bảng Booking-Garage và transaction để đảm bảo tính nhất quán về dữ liệu.

5.2 Tối ưu hóa truy vấn khoảng cách

5.2.1 Đặt vấn đề

Việc tính toán khoảng cách bằng công thức Haversine (**hình 5.1**) được sử dụng để xác định các garage gần nhất với khách hàng.

$$d = 2r \arcsin\left(\sqrt{\frac{1 - \cos(\varphi_2 - \varphi_1) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot (1 - \cos(\lambda_2 - \lambda_1))}{2}}\right)$$

Hình 5.1: Công thức Haversine

với r: bán kính trái đất

φ_1, φ_2: Đây là vĩ độ của hai điểm (latitude) được tính bằng độ.

λ_1, λ_2: Đây là kinh độ của hai điểm (longitude) được tính bằng độ.

$\Delta\varphi$: Độ chênh lệch vĩ độ giữa hai điểm, có thể tính bằng $\varphi_2 - \varphi_1$.
$\Delta\lambda$: Độ chênh lệch kinh độ giữa hai điểm, có thể tính bằng $\lambda_2 - \lambda_1$.

Để minh họa, truy vấn tìm garage gần nhất sử dụng công thức Haversine có thể được thực hiện như sau (**hình 5.2**) :

```sql
SELECT
    id, garage_name, latitude, longitude,
    6371 * acos(
        cos(radians(21.0308)) * cos(radians(latitude)) *
        cos(radians(longitude) - radians(105.8014)) +
        sin(radians(21.0308)) * sin(radians(latitude))
    ) AS distance
FROM
    Garages
HAVING
    distance <= 10
ORDER BY
    distance
LIMIT 2;
```

Hình 5.2: Câu lệnh truy vấn cơ bản

Tính toán khoảng cách sử dụng công thức Haversine trên một tập dữ liệu lớn có thể tốn kém về tính toán, đặc biệt nếu truy vấn được thực hiện thường xuyên. Vì vậy cần phải tối ưu truy vấn để tránh các vấn đề về hiệu suất.

5.2.2 Giải pháp và kết quả đạt được

Giải pháp tối ưu truy vấn khi sử dụng công thức Haversine là tạo một bounding box (là vùng không gian bao quanh điểm cần tính) để giảm số lượng hàng cần kiểm tra trước khi tính khoảng cách chính xác.

Hình 5.3 trình bày hàm boudingbox được viết bằng ngôn ngữ javascript.

```javascript
const boundingBox = (latitude, longitude, distance) => {
    const earthRadius = 6371;

    // Convert từ độ sang radian
    const radLat = (latitude * Math.PI) / 180;
    const radLon = (longitude * Math.PI) / 180;
    const radDist = distance / earthRadius;

    // Tính toán vĩ độ tối thiểu và tối đa
    const minLat = radLat - radDist;
    const maxLat = radLat + radDist;

    // Tính toán kinh độ tối thiểu và tối đa
    const deltaLon = Math.asin(Math.sin(radDist) / Math.cos(radLat));
    const minLon = radLon - deltaLon;
    const maxLon = radLon + deltaLon;

    // Convert từ radian sang độ
    const minLatDeg = (minLat * 180) / Math.PI;
    const maxLatDeg = (maxLat * 180) / Math.PI;
    const minLonDeg = (minLon * 180) / Math.PI;
    const maxLonDeg = (maxLon * 180) / Math.PI;

    return {
        minLat: minLatDeg,
        maxLat: maxLatDeg,
        minLon: minLonDeg,
        maxLon: maxLonDeg,
    };
};
```

Hình 5.3: Boudingbox

Kết quả câu truy vấn sau khi áp dụng bounding box (**hình 5.4**):

```sql
SELECT
    id, garage_name,
    6371 * acos(
        cos(radians(21.0308)) * cos(radians(latitude)) *
        cos(radians(longitude) - radians(105.8014)) +
        sin(radians(21.0308)) * sin(radians(latitude))
    ) AS distance
FROM
    datn2.garages
WHERE
    latitude BETWEEN minLat AND maxLat
AND
    longitude BETWEEN minLon AND maxLon
HAVING
    distance <= 10
ORDER BY
    distance
LIMIT 2;
```

Hình 5.4: Câu lệnh truy vấn với boudingbox

Ngoài ra, để đảm bảo hiệu suất của truy vấn, cần đánh index cho các cột longitude và latitude. Điều này giúp cơ sở dữ liệu tối ưu hóa cách lưu trữ và truy xuất dữ liệu địa lý.

Việc sử dụng công thức Haversine để tính toán khoảng cách giữa khách hàng và garage là một giải pháp hiệu quả, tuy nhiên cần phải tối ưu truy vấn để đảm bảo hiệu suất. Áp dụng bounding box là một trong những cách để cải thiện truy vấn, giảm thiểu số lượng hàng cần phải kiểm tra, và tăng cường hiệu suất của hệ thống.

CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1 Kết luận

Trong suốt quá trình thực hiện ĐATN, em đã phát triển một hệ thống đặt lịch sửa chữa và bảo dưỡng ô tô tích hợp, mang lại sự tiện lợi cho cả khách hàng và quản lý garage. Hệ thống này cho phép khách hàng đặt lịch trực tuyến, quản lý thông tin cá nhân và theo dõi lịch sử bảo dưỡng xe, thanh toán trực tuyến qua hệ thống của vnpay. Đối với garage, hệ thống cung cấp các công cụ để quản lý lịch hẹn, phân công công việc, theo dõi tiến độ, xem thống kê khối lượng công việc. Một cơ chế thông báo tự động qua FCM cũng được tích hợp để nhắc nhở khách hàng về các cuộc hẹn và bảo trì định kỳ. Ngoài ra còn mở rộng thêm tính năng nhắn tin giúp việc trao đổi giữa khách hàng và garage thuận tiện hơn.

Trong suốt quá trình thực hiện ĐATN, mặc dù đã có những cố gắng nhất có thể nhưng kinh nghiệm thiếu sót và việc hoàn thiện một hệ thống hoàn chỉnh, toàn diện đáp ứng nhiều yêu cầu của cả phía khách hàng và phía garage cần bỏ ra nhiều công sức hơn. Hệ thống còn một số hạn chế như chưa phát triển các chức năng liên quan đến quản trị hệ thống, chưa có thống kê doanh thu của garage,...

Từ quá trình thực hiện ĐATN, em đã rút ra được những bài học quý báu bao gồm việc lập kế hoạch chi tiết, quản lý thời gian hiệu quả và sự cần thiết của việc kiểm tra và đánh giá hệ thống một cách liên tục.

6.2 Hướng phát triển

Để triển khai sản phẩm trên thực tế và nâng cao tính ứng dụng, hệ thống cần được cải thiện và bổ sung các chức năng mới để đáp ứng các nhu cầu và mở rộng phạm vi dịch vụ. Đầu tiên, phải phát triển giao diện quản lý cho quản trị viên để họ có thể dễ dàng quản lý thông tin garage, người dùng, và các hoạt động khác trong hệ thống. Chức năng này bao gồm quản lý tài khoản, các dịch vụ và các tính năng báo cáo và thống kê.

Ngoài ra, hệ thống cần cải tiến tính năng phản hồi và đánh giá từ người dùng để thu thập ý kiến và cải thiện chất lượng dịch vụ của garage. Điều này giúp tăng cường mối quan hệ với khách hàng và nâng cao trải nghiệm người dùng.

Cuối cùng, để mở rộng phạm vi dịch vụ, cần cung cấp các tính năng hỗ trợ như tư vấn khách hàng và quản lý kho phụ tùng. Đây là những yếu tố quan trọng giúp nâng cao sự hài lòng của khách hàng và đáp ứng đầy đủ nhu cầu trong lĩnh vực quản lý và vận hành garage.

TÀI LIỆU THAM KHẢO

.

[1] *Learn Reactjs*. [Online]. Available: https://react.dev/learn.

[2] *Setup Firebase Admin*. [Online]. Available: https://firebase.google.com/docs/admin/setup?hl=vi.

[2] *Setup Firebase For React*. [Online]. Available: https://firebase.google.com/docs/web/setup?hl=vi.

[3] *React Image Upload To Cloudinary*. [Online]. Available: https://cloudinary.com/documentation/react_image_and_video_upload.

[4] *NodeJs Tutorial*. [Online]. Available: https://www.w3schools.com/nodejs/.