**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

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



**PHẠM DƯƠNG HÒA**

**THIẾT KẾ XÂY DỰNG WEBSITE**

**QUẢN LÝ BÁN VÉ XE KHÁCH**

**Mã số sinh viên: 1851050051**

**ĐỒ ÁN NGÀNH**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

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

**TP. HỒ CHÍ MINH, 2021**

**LỜI CÁM ƠN**

Trong quá trình học tập tại trường ĐH. Mở TP.Hồ Chí Minh, em đã vẫn luôn nhận được sự quan tâm giúp đỡ của các thầy, các cô trong trường cùng với đó sự động viên hỗ trợ của bạn bè, đàn anh, đàn chị.

Đầu tiên em xin gửi lời cám ơn đến nhà trường vì đã tổ chức, hỗ trợ cho em từ những ngày đầu tiên tiến vào môi trường đại học đến tận bây giờ. Trường đã quan tâm, lo lắng cho em và các bạn sinh viên khác. Đặc biệt vào mùa dịch trường cũng đã tích cực hỗ trợ tạo điều kiện cho bọn em được tiếp tục chương tình học.

Tiếp theo em xin gửi lời cám ơn đến các Khoa Công Nghệ Thông tin cũng như các thầy, các cô của Khoa đã hỗ trợ em trên con đường học vấn, định hướng nghề nghiệp. Nhờ các thầy, cô mà em đã trang bị được lượng lớn kiến thức chuyên ngành.

Em cũng tỏ lòng biết ơn sâu sắc đến gia đình, người thân, những người đã nuôi em khôn lớn. Cũng như những người bạn đã hỗ trợ động viên em hoàn thiện đồ án này.

Mặc dù vẫn còn nhiều yếu em, nhưng em xin hứa vẫn sẽ tiếp tục học tập và cải thiện khả năng của bản thân để không phụ sự quan tâm của mọi người với chính mình.

***Em xin trân trọng cám ơn!***

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

**MỤC LỤC**

[**MỤC LỤC HÌNH ẢNH** 6](#_Toc87309641)

[**MỤC LỤC BẢNG** 7](#_Toc87309642)

[**Chương 1.** **TỔNG QUAN ĐỀ TÀI** 8](#_Toc87309643)

[**1.1.** **Giới thiệu đề tài và lý do chọn đề tài** 8](#_Toc87309644)

[**1.2.** **Các chức năng chính** 8](#_Toc87309645)

[**1.2.1.** **Chức năng đặt vé.** 8](#_Toc87309646)

[**1.2.2.** **Đánh giá chuyến xe.** 9](#_Toc87309647)

[**1.2.3.** **Chức năng quản lý chuyến xe.** 9](#_Toc87309648)

[**1.2.4.** **Chức năng thống kê, quản lý.** 9](#_Toc87309649)

[**Chương 2.** **CƠ SỞ LÝ THUYẾT** 10](#_Toc87309650)

[**2.1.** **Django - Backend** 10](#_Toc87309651)

[**2.1.1.** **Python** 10](#_Toc87309652)

[**2.1.2.** **Django** 10](#_Toc87309653)

[**2.1.3.** **Django REST framework** 11](#_Toc87309654)

[**2.1.4.** **Django OAuth2-provider** 11](#_Toc87309655)

[**2.2.** **Reactjs – Frontend** 13](#_Toc87309656)

[**2.2.1.** **Giới thiệu ReactJS** 13](#_Toc87309657)

[**2.2.2.** **Cây DOM ảo (Virtual DOM)** 13](#_Toc87309658)

[**2.2.3.** **Component** 14](#_Toc87309659)

[**2.2.4.** **React Router** 16](#_Toc87309660)

[**2.2.5.** **React hook** 16](#_Toc87309661)

[**2.3.** **MySQL – Database** 17](#_Toc87309662)

[**Chương 3.** **PHÂN TÍCH THIẾT KẾ HỆ THỐNG** 19](#_Toc87309663)

[**3.1.** **Nghiệp vụ hệ thống** 19](#_Toc87309664)

[**3.2.** **Đặc tả use case** 20](#_Toc87309665)

[**3.4.** **Thiết kế cơ sở dữ liệu** 38](#_Toc87309666)

[**3.4.1.** **Phân tích** 38](#_Toc87309667)

[**3.4.2.** **Bản thiết kế** 42](#_Toc87309668)

[**3.6.** **Thiết kế API backend** 43](#_Toc87309669)

[**3.6.1.** **API xác thực oauth2-provider** 43](#_Toc87309670)

[**3.6.2.** **API mặc định** 43](#_Toc87309671)

[**3.6.3.** **API tùy chỉnh** 44](#_Toc87309672)

[**Chương 4.** **KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN** 52](#_Toc87309673)

[**4.1.** **Vấn đề còn tồn tại** 52](#_Toc87309674)

[**4.2.** **Hướng phát triển** 52](#_Toc87309675)

[**4.2.1.** **Tích hợp định vị GPS** 52](#_Toc87309676)

[**4.2.2.** **Thêm chức năng tạo vé điện tử bằng mã vạch QR** 53](#_Toc87309677)

[**TÀI LIỆU THAM KHẢO** 54](#_Toc87309678)

# MỤC LỤC HÌNH ẢNH

[Hình 1. Mô hình Model-View-Template. 10](file:///C:\Users\ACER\Desktop\baocaodofan.docx#_Toc87309026)

[Hình 2. Luồng hoạt động lấy token Netflix - Google 12](file:///C:\Users\ACER\Desktop\baocaodofan.docx#_Toc87309027)

[Hình 3. Mô hình cây DOM 13](file:///C:\Users\ACER\Desktop\baocaodofan.docx#_Toc87309028)

[Hình 4. So sánh sự khác nhau giữa DOM và Virtual DOM 14](file:///C:\Users\ACER\Desktop\baocaodofan.docx#_Toc87309029)

[Hình 5. Vòng đời component 16](file:///C:\Users\ACER\Desktop\baocaodofan.docx#_Toc87309030)

[Hình 6. Lược đô usecase hệ thống bán vé xe khách 20](file:///C:\Users\ACER\Desktop\baocaodofan.docx#_Toc87309031)

# MỤC LỤC BẢNG

[Bảng 1. Danh sách các actor 20](#_Toc87309610)

[Bảng 2. Đặc tả usecase đăng nhập 21](#_Toc87309611)

[Bảng 3. Đặc tả usecase Đăng ký 22](#_Toc87309612)

[Bảng 4. Đặc tả usecase quản lý tài khoản 23](#_Toc87309613)

[Bảng 5. Đặc tả usecase lịch sử đặt vé 24](#_Toc87309614)

[Bảng 6. Đặc tả usecase tra cứu thông tin chuyến xe 26](#_Toc87309615)

[Bảng 7. Đặc tả usecase đặt vé 28](#_Toc87309616)

[Bảng 8. Đặc tả usecase kết thúc chuyến xe 29](#_Toc87309617)

[Bảng 9. Đặc tả usecase điểm danh 30](#_Toc87309618)

[Bảng 10. Đặc tả usecase quản lý nhân viên 32](#_Toc87309619)

[Bảng 11. Đặc tả usecase cấp tài khoản 33](#_Toc87309620)

[Bảng 12. Đặc tả usecase quản lý xe, model xe, bến, trạm 35](#_Toc87309621)

[Bảng 13. Đặc tả usecase lập lịch chạy 36](#_Toc87309622)

[Bảng 14. Đặc tả usecase thống kê 37](#_Toc87309623)

[Bảng 15. Bảng thiết kế API mặc định 44](#_Toc87309624)

1. **TỔNG QUAN ĐỀ TÀI**
   1. **Giới thiệu đề tài và lý do chọn đề tài**

Ngày nay, xã hội đã có sự phát triển vượt bậc so với các thập niên trước, đời sống của từng cá nhân đã được nâng cao, và việc mỗi cá nhân sở hữu một chiếc điện thoại smartphone là vô cũng dễ dàng. Nước ta đứng thứ 10 trên thế giới về số lượng người sử dụng smartphone năm 2020, với tỉ lệ người sử dụng smartphone là 63.1%, đây là một tỉ lệ lớn trên thị trường [1]. Nhờ có những chiếc điện thoại này, và sự bùng nổ phát triển mạnh mẽ của mạng internet, mà mỗi cá nhân đều có thể tiếp cận được với mạng lưới thông tin khổng lồ. Việc trao đổi thư từ, tin nhắn cũng trở nên nhanh hơn trước.

Vào cuối năm 2019, trên thế giới đã xuất hiện một loại virus mới là Covid-19 [2], virus này có khả năng lây nhiễm nhanh, thời gian ủ bệnh lâu và có thể gây nguy hiểm đến tính mạng của người bệnh. Đất nước đầu tiên bùng dịch là Trung Quốc, và rất nhanh lan rộng ra nhiều nước khác khiến cho các nước này phải phong tỏa, hạn chế sự di chuyển của người dân nhằm giảm mức độ lây lan của dịch bệnh, khiến nhiều người mất đi việc làm và nền kinh tế bị thụt lùi. May thay nhờ sự phát triển của mạng Internet mà một số ngành vẫn có thể duy trì tiếp tục bằng cách mở các dịch vụ online. Trao đổi công việc qua các buổi họp online. Điều này đã giảm được một phần lớn tổn thất mà đại dịch có thể đem lại cho xã hội.

Ngành vận tải hành khách bị cấm lưu thông trong giai đoạn này nhưng lại có vai trò quan trọng trong việc truy vết các ca bệnh đã di chuyển bằng dịch vụ của họ. Tại đây có thể thấy nhiều nhà xe bị thụt lùi, gặp khó khăn trong quá trình truy vết do không họ vẫn sử dụng các mô hình kinh doanh cũ, lưu lại thông tin khách hàng trên giấy tờ. Sự chậm trễ trong việc truy vết đã khiến dịch bệnh bùng phát nặng nề hơn.

Vì vậy tôi quyết định xây dựng một hệ thống giúp các nhà xe có thể quản lý được thông tin các khách hàng và các chuyến xe. Hệ thống sẽ được xây dựng trên nền tảng internet, nên cũng góp phần mở rộng phạm vi kinh doanh của các nhà xe, việc các khách hàng tiếp cận đặt vé cũng trở nên dễ dàng hơn.

* 1. **Các chức năng chính**
     1. **Chức năng đặt vé.**

Đối tượng sử dụng: Khách hàng, nhân viên bán vé.

Nghiệp vụ:

* Tra cứu thông tin của các chuyến xe, giờ chạy.
* Hiển thị sơ đồ xe và các vị trí ghế còn trống.
* Đặt vé xe.
* Đối với đối tượng khách hàng có thể xem lịch sử đặt vé của bản thân đối tượng.
  + 1. **Đánh giá chuyến xe.**

Đối tượng sử dụng: Khách hàng.

Nghiệp vụ: ghi nhận đánh giá của khách hàn.

* + 1. **Chức năng quản lý chuyến xe.**

Đối tượng sử dụng: Tài xế.

Nghiệp vụ:

* Hiển thị sơ đồ xe.
* Điểm danh các vị trí ghế đã có người lên xe.
* Báo cáo kết thúc hành trình khi xe đến đích.
  + 1. **Chức năng thống kê, quản lý.**

Đối tượng sử dụng: Quản trị viên.

Nghiệp vụ:

* Thêm sửa xóa thông tin của các chiếc xe, bến xe, trạm dừng chân.
* Lập lịch chạy các chuyến xe:
  + Chọn xe.
  + Ghi giá vé.
  + Chọn tuyến đường.
  + Chọn danh sách các trạm dừng.
  + Chọn tài xế chạy xe.
* Thống kê báo cáo.
* Quản lý nhân viên:
  + Cấp tài khoản cho nhân viên sử dụng hệ thống.
  + Xem lịch sử công việc của nhân viên.

1. **CƠ SỞ LÝ THUYẾT**
   1. **Django - Backend**
      1. **Python**

Python là ngôn ngữ lập trình mã nguồn mở thế hệ thứ tư (4GL) được thiết kế với 3 tiêu chí là dễ học, dễ đọc và dễ nhớ.

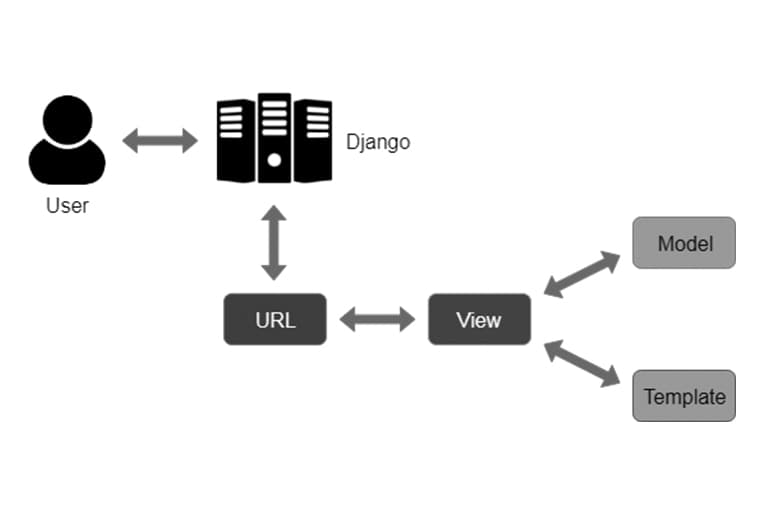
Python hỗ trợ nhiều mẫu hình lập trình khác nhau như: lập trình hướng đối tượng, lập trình câu lệnh, lập trình hàm, thủ tục.

Cứ 100 lập trình viên thì có hơn 48 người là biết ngôn ngữ python, đứng thứ 3 trên bảng xếp hạng các ngôn ngữ lập trình phổ biến [3]. Kéo theo đó là việc python sở hữu nhiều bộ thư viện lớn được tạo ra bởi cộng đồng phục vụ cho nhiều mục đích lập trình khác nhau như: Web Dev, Data Analysis, Machine Learning, Computer Vision, IoT,…

* + 1. **Django**

Django là một framework mạnh mẽ của Python hỗ trợ xây dựng, phát triển phần mềm dễ dàng, nhanh chóng. Django được xây dựng bởi nhiều lập trình viên kinh nghiệm, phục vụ chủ yếu cho mục đích lập trình web.

Django hoạt động theo mô hình MVT (Model View Template).



Hình 1. Mô hình Model-View-Template.

Cơ chế hoạt động:

* Người dùng gửi request về server.
* Server mapping URL request đến View tương ứng.
* View thực hiện chức năng như một controller, xử lý yêu cầu của người dùng, tương tác với model lấy và cập nhật dữ liệu trong cơ sở dữ liệu. Sau khi xử lý xong sẽ trả về response với template tương ứng cho người dùng.

Django hỗ trợ việc xây dựng các lớp model theo mô hình ORM, các lớp này sẽ được ánh xạ xuống bộ cơ sở dữ liệu và xây dựng các bảng tương ứng. bản thân Django cũng cung cấp một bộ API để thực hiện việc truy vấn tương tác cơ sở dữ liệu.

* + 1. **Django REST framework**

Django REST framework (DRF) là một công cụ mạnh mẽ hỗ trợ xây dựng API linh hoạt, nhanh chóng.

Một số điểm mạnh của DRF có thể kể đến là:

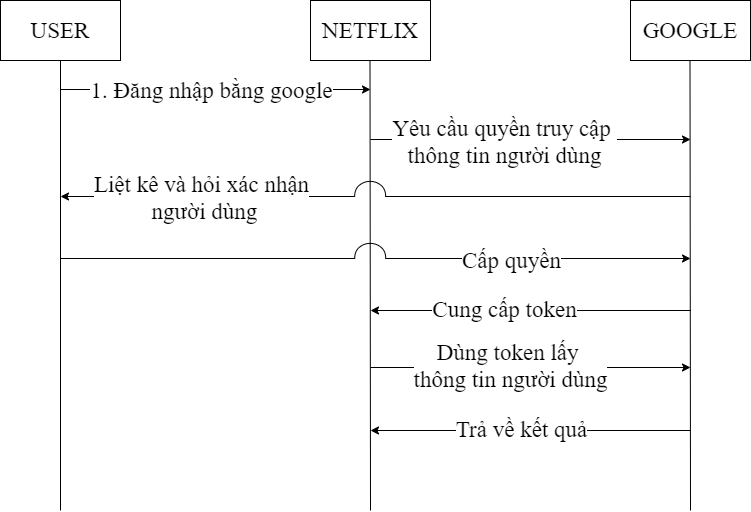
* Hỗ trợ bộ quản lý, kiểm duyệt API.
* Hỗ trợ nhiều chính sách xác thực bao gồm Oauth1a và OAuth2.
* Bộ Serialization hỗ trợ dữ liệu theo ORM và non-ORM
* Nhiều tài liệu và có một cộng đồng lập trình viên lớn.

DRF hỗ trợ sẵn bộ các API cơ bản để tương tác với từng model được khai báo chỉ bằng vài dòng code, lập trình viên cũng có thể ghi đè lên các API này hoặc tự tạo ra các API của riêng họ.

Thông qua Serialization, lập trình viên có thể định nghĩa được định dạng của dữ liệu, chuyển các dữ liệu dạng phức tạp thành các kiểu dữ liệu phổ biến như JSON, XML hay một content type khác. Serialization cũng hỗ trợ phương thức giúp kiểm tra kiểu dữ liệu có chính xác với trường tương ứng của nó hay không. Điều này giúp rất nhiều trong việc kiểm soát lỗi dữ liệu.

* + 1. **Django OAuth2-provider**

**Oauth2** là viết tắt của OPEN, Authentication, Authorization, là một phương thức xác thực người dùng thông qua một token key thay vì dùng username và password như cách thức truyền thống. Token này có thể được sử dụng bởi các bên thứ 3 để sử dụng, truy cập tài nguyên trên hệ thống, mà không làm lộ thông tin tài khoản người dùng.

Ví dụ: Khi đăng nhập một dịch vụ xem phim trực tuyến của Netflix bằng cách dùng tài khoản Facebook, trang web sẽ dẫn bạn tới một trang của facebook và liệt kê các quyền mà trang Netflix muốn sử dụng thông qua tài khoản Facebook. Thông thường thì các quyền như xem địa chỉ email, họ tên, ngày sinh của người dùng sẽ được yêu cầu. Nếu người dùng chấp nhận cho Netflix quyền này, Facebook sẽ sinh ra một token cho Netflix, Netflix sẽ lấy thông tin cần thiết (thực hiện quyền) từ google thông qua token được cấp.

Hình 2. Luồng hoạt động lấy token Netflix - Google

Một số định nghĩa trong oauth2:

* Access token: mã khóa được cung cấp sau khi đăng nhập để truy cập tài nguyên trên hệ thống.
* Refresh token: mã khóa được cung cấp sau khi đăng nhập có thể dùng để yêu cầu cấp một access token mới từ server.
* Expires time: Thời gian mã khóa có hiệu lực, khi hết hiệu lực thì mã sẽ trở nên vô dụng.
* Authentication: Xác thực người dùng.
* Authorization: Ủy quyền các tài nguyên người dùng được truy cập

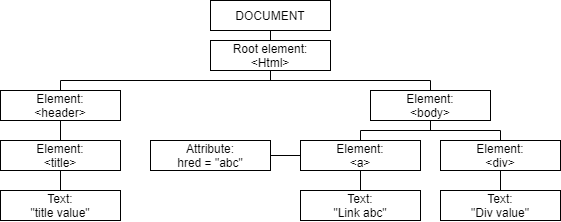
**Django OAuth2-provider** là một bộ công cụ mạnh, được xây dựng để hỗ trợ việc xác thực và ủy quyền người dùng. Nó cung cấp sẵn một bộ API và phương thức xác thực người dùng.

* 1. **Reactjs – Frontend**
     1. **Giới thiệu ReactJS**

ReactJS là một thư viện javascript mã nguồn mở được ra đời và phát triển bởi Facebook, dùng để xây dựng giao diện web.

ReactJS chia một trang web thành nhiều thành phần nhỏ được gọi là component, các component này vừa có khả năng đồng bộ hóa với nhau, vừa có khả năng hoạt động độc lập. Một thành phần có thể được sử dụng nhiều lần tại nhiều trang khác nhau. Nhờ khả năng này mà các thành viên trong cộng đồng sử dụng reactjs có thể chia sẻ cho nhau những component của từng cá nhân họ xây dựng.

Hiện nay số lượng component được cộng đồng reactjs chia sẽ là rất lớn, có thể dễ dàng tìm được những component có các chức năng được tùy chỉnh theo nhu cầu lập trình viên.

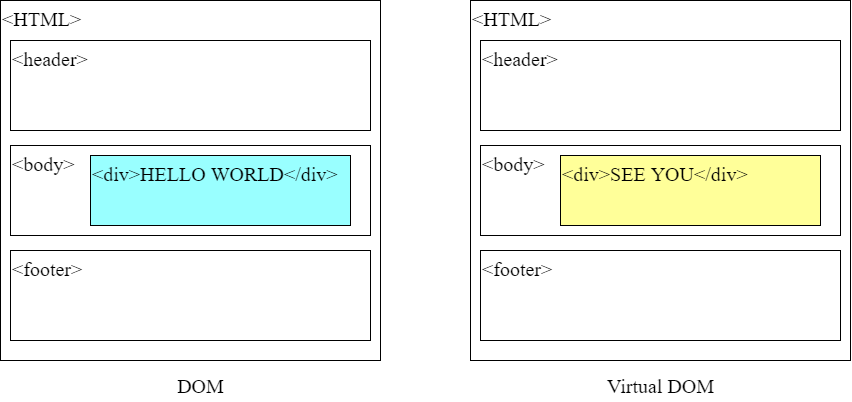
* + 1. **Cây DOM ảo (Virtual DOM)**

Hình 3. Mô hình cây DOM

DOM (**D**ocument **O**bject **M**odel) là một cấu trúc dạng cây được sử dụng trong HTML có các tính chất sau:

* Mỗi node là một thẻ HTML (Element).
* Mỗi node có duy nhất một node cha.
* Một node cha có thể có nhiều node con.
* Mỗi node sẽ có các thuộc tính, giá trị của riêng từng node.

Khi cần render lại một trang web mới, nếu dùng cây DOM truyền thống thì sẽ cần phải render lại tất cả thành phần có trong trang mới đó, kể cả các thành phần giống với trang trước khi render lại, ví dụ như thanh menu, header, footer.

Để cải thiện hiệu năng bằng cách giảm số lượng thành phần cần render, ReactJS sử dụng thêm cây DOM ảo thay vì chỉ dùng một DOM thật. Về cơ bản cây DOM ảo có thể xem là một bản sao trừu tượng kế thừa đầy đủ các tính chất, đặc điểm của cây DOM thật. Nhưng cây DOM ảo chỉ là một phần nền xử lý ẩn bên trong trang web, không được render trực tiếp lên trang web mà phải thông qua cây DOM thật.

Hình 4. So sánh sự khác nhau giữa DOM và Virtual DOM

Khi trang web cần cập nhật một thành phần, cây DOM ảo sẽ render lại thành phần đó và sau đó ReactJS sẽ thực hiện một phép so sánh giữa 2 cây DOM để tìm ra các điểm khác nhau. Sau khi so sánh xong nếu có điểm khác nhau, cây DOM thật sẽ được cập nhật trực tiếp phần khác nhau đó mà không cần render lại cả trang.

* + 1. **Component**

Để cấu thành một trang web cần xây dựng rất nhiều thành phần khác nhau cho nó, trong đó có một số thành phần sẽ được sử dụng lặp đi lặp lại ở nhiều trang như menu, header, footer. Để tránh việc phải code một thành phần nhiều lần ở các trang khác nhau, ReactJS sẽ gom nhóm các thẻ HTML lại và tạo thành một component riêng cho từng phần.

Các thành phần của một component:

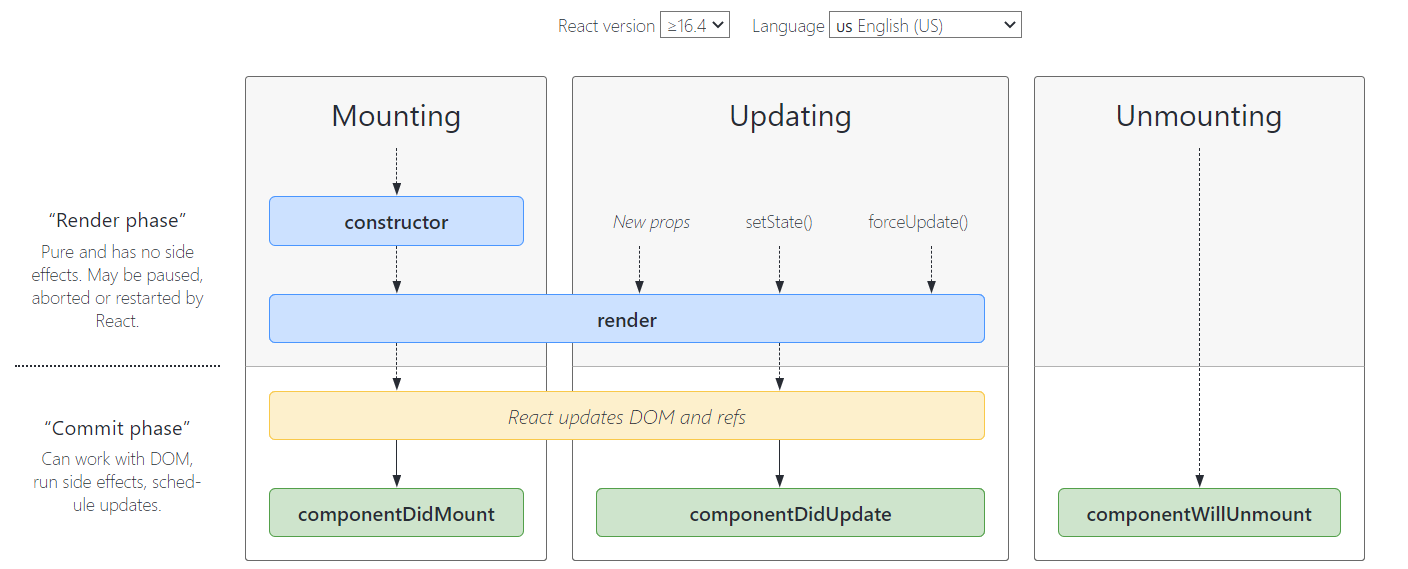
* Props: Tham số truyền vào từ component cha (input).
* State: Tham số được khai báo bên trong component và chỉ sử dụng trong phạm vi component.
* function: Các hàm xử lý vận hành bên trong một.
* Render: mã HTML cuối cùng được trả về khi component được nạp (output).

Các tính chất cơ bản của component:

* Trong một component có thể chèn thêm một component khác.
* Một component có thể được tái sử dụng nhiều lần.
* Các component giao tiếp với nhau thông qua các tham số truyền vào.
* Khi tham số props hoặc state bị thay đổi, component sẽ được render lại.

Mỗi component của ReactJS có 3 giai đoạn thể hiện vòng đời của chính nỏ trong quá trình thực thi mã nguồn:

* Mounting: Khi component vừa được đưa vào DOM, tại bước này component sẽ thực hiện render lần đầu tiên.
* Updating: Phần hoạt động chính trong quá trình component ở trong DOM, Mỗi khi các tham số props hoặc state thay đổi, component sẽ được render lại. Lập trình viên cũng có thể bắt buộc component thực hiện bước này.
* Unmounting: Tháo rời component ra khỏi cây DOM.



Hình 5. Vòng đời component

* + 1. **React Router**

React router là một thư viện hỗ trợ chuyển đổi giữa các component dựa trên URL trang web mà không cần nạp lại cả trang.

Đề tài sử dụng 4 thẻ chính của react router:

* BrowserRouter: Dùng để bọc khu vực xử lý các URL động.
* Route: Chỉ định các cặp component-URL tương ứng với nhau.
* Switch: Dùng để chứa các Route vào đúng vị chí của Switch. Component khi được nạp vào sẽ nằm ở vị trí của Switch trên cây DOM.
* Link: Tạo liên kết chuyển URL nhưng không nạp lại trang.
  + 1. **React hook**

Hook là một tính năng mới của react được hỗ trợ từ phiên bản 16.8, cho phép sử dụng state và các đặc trưng của React trong function.

Có 2 loại hook chính là hook state và hook effect

* + - 1. **Hook State**

Hook state là một cơ chế dùng để cải thiện việc sử dụng state trong một component. Hook state được khai báo theo dạng 1 object – 1 function

import { useState } from “react”

const [value, setValue] = useState(0)

Lấy ví dụ, đoạn code trên có thể hiểu là đang khai báo một biến value giá trị mặc định là 0 và một hàm setValue(x) sẽ được dùng để gán giá trị của x cho value.

Tương tự như state, khi hook state thay đổi, component sẽ được render lại với giá trị state mới.

* + - 1. **Hook Effect**

Hook effect là cơ chế thực hiện các hành động trên function component tương tự như các phương thức trong vòng đời của component mà không sử dụng các phương thức trong đó.

Hook effect có thể sử dụng thông qua việc khai báo useEffect()

import { useEffect } from “react”

useEffect(

effect,

dependency

)

Trong đó

* Effect là một function.
* Dependency là điều kiện để effect được chạy, điều kiện có thể là một hook state, hoặc một dãy các hook state. Nếu không có điều kiện thì effect sẽ được chạy liên tục, còn muốn thực hiện effect duy nhất mọt lần thì điều kiện là 1 mảng rỗng
  1. **MySQL – Database**

MySQL là một hệ thống quản trị cơ sở dữ liệu mã nguồn mở phổ biến trên thế giới, được nhiều công ty và lập trình viên sử dụng.

MySQL hoạt động theo mô hình Client – Server

MySQL lưu trữ dữ liệu theo cấu trúc dạng bảng và các quan hệ

1. **PHÂN TÍCH THIẾT KẾ HỆ THỐNG**
   1. **Nghiệp vụ hệ thống**

Hệ thống đảm nhận một số chức năng sau:

* Tra cứu thông tin các chuyến xe
* Đặt vé
* Quản lý chuyến xe.
* Quản lý tài khoản nhân viên, khách
* Thống kê, báo cáo.

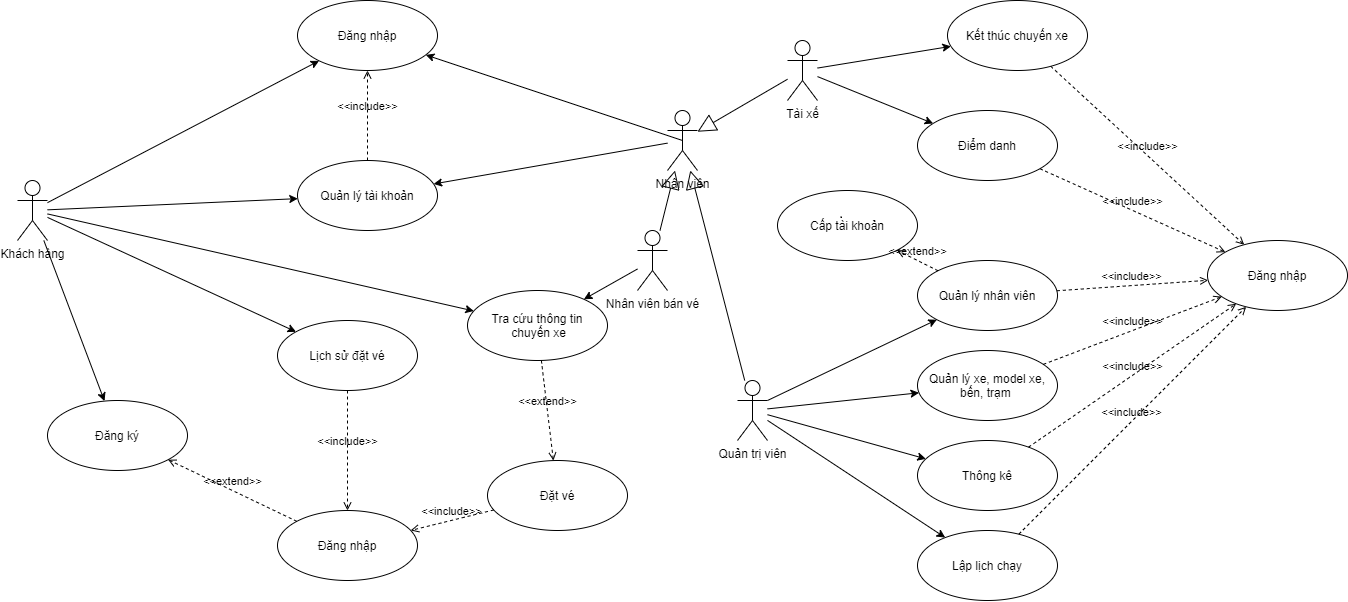
Hệ thống sẽ hoạt động theo dạng website theo dạng client-server. Trong đó client sẽ là trang web mà người dùng có thể truy cập vào và thông qua đó sẽ gửi các request đến server lấy ra các thông tin cần thiết. Mỗi người sử dụng hệ thống có thể sử dụng thiết bị truy cập mạng cá nhân để vào địa chỉ website và tiến hành các thao tác trên đó.

Các khách hàng, những người không có tài khoản trên hệ thống có thể tra cứu lịch trình, thông tin chuyến xe trên hệ thống. Để đặt vé họ cần đăng ký một tài khoản cá nhân hoặc liên hệ nhân viên bán vé để họ thao tác đặt vé hộ. Mỗi chuyến xe sẽ ghi nhận lại các đánh giá, bình luận của khách hàng.

Tài xế lái xe sẽ có thể sử dụng hệ thống để kiểm kê số lượng hành khách lên xe, xuống xe.

Hệ thống cũng có thêm chức năng quản lý thông tin nhân viên, quản trị viên hệ thống có thể cấp và phân quyền cho nhân viên tài khoản để truy cập hệ thống. Quản trị viên cũng có thể xem thống kê, báo cáo doanh thu, số lượng chuyến xe, số vé bán ra,… hằng tháng.

* 1. **Đặc tả use case**



Hình 6. Lược đô usecase hệ thống bán vé xe khách

* + 1. **Danh sách actor**

|  |  |  |
| --- | --- | --- |
| STT | Tên actor | Đối tượng |
| 1 | Khách hàng | Mọi người truy cập vào trang web đều được xem là đối tượng khách hàng. Khách hàng có thể tra cứu thông tin các chuyến xe. Và nếu khách hàng có tài khoản thì có thể đặt vé xe. |
| 2 | Nhân viên | Nhân viên của nhà xe được quản trị viên cấp tài khoản cho sữ dụng. |
| 3 | Nhân viên bán vé | Các nhân viên phụ trách công việc bán vé của nhà xe. |
| 4 | Tài xế | Các nhân viên là tài xế của nhà xe. |
| 5 | Quản trị viên | Người quản lý cấp cao nhất hệ thống. |

Bảng 1. Danh sách các actor

* + 1. **Đặc tả usecase**

|  |  |
| --- | --- |
| Tên | Đăng nhập |
| ID | 1.1 |
| Mô tả | Xác thực vai trò người dùng khi sử dụng hệ thống. |
| Actor | Tất cả actor |
| Priority | Phải có |
| Trigger | * Người dùng chủ động đăng nhập. * Người dùng truy cập chức năng cần xác thực. |
| Pre-condition | * Người dùng sở hữu tài khoản. * Tài khoản vẫn đang hoạt động bình thường. |
| Post-condition | * Người dùng đăng nhập thất bại hoặc thành công |
| Luồng cơ bản | 1. Người dùng được chuyển hướng sang chức năng đăng nhập.  2. Người dùng nhập tài khoản và chọn đăng nhập.  3. Hệ thống xác thực tài khoản hợp lệ và cho phép người dùng quyền truy cập. |
| Luồng thay thế |  |
| Luồng ngoại lệ | 3a. Hệ thống xác thực tài khoản không hợp lệ.  3a1. Hệ thống thông báo cho người dùng.  Usecase thực hiện từ bước 1. |

Bảng 2. Đặc tả usecase đăng nhập

|  |  |
| --- | --- |
| Tên | Đăng xuất |
| ID | 1.2 |
| Mô tả | Hủy xác thực vai trò người dùng. |
| Actor | Tất cả actor. |
| Priority | Phải có |
| Trigger | * Người dùng chủ động đăng xuất. * Người dùng ngừng hoạt động quá 30 phút. |
| Pre-condition | * Người dùng đã được xác thực |
| Post-condition | * Hủy xác thực người dùng. |
| Luồng cơ bản | 1. Người dùng chọn đăng xuất.  2. Hệ thống hủy xác thực người dùng, vô hiệu hóa token.  3. Trình duyệt xóa thông tin người dùng trong cookies |
| Luồng thay thế |  |
| Luồng ngoại lệ |  |

Bảng 3. Đặc tả usecase Đăng ký

|  |  |
| --- | --- |
| Tên | Quản lý tài khoản |
| ID | 1.3 |
| Mô tả | Cho phép người sử dụng xem, chỉnh sửa thông tin tài khoản. |
| Actor | Tất cả actor. |
| Priority | Phải có |
| Trigger | * Người dùng muốn chỉnh sửa thông tin tài khoản. |
| Pre-condition | * Người dùng đã được xác thực. |
| Post-condition | * Thông tin người dùng được cập nhật mới. |
| Luồng cơ bản | 1. Người dùng chọn chỉnh sửa thông tin tài khoản.  2. Người dùng chỉnh sửa các thông tin trong tài khoản.  3. Người dùng nhập mật khẩu xác thực.  4. Hệ thống thực thi cập nhật thông tin người dùng. |
| Luồng thay thế |  |
| Luồng ngoại lệ | 4a, Mật khẩu xác thực người dùng không chính xác  4a1, Hệ thống thông báo người dùng.  Use case thực hiện lại bước 3 |

Bảng 4. Đặc tả usecase quản lý tài khoản

|  |  |
| --- | --- |
| Tên | Lịch sử đặt vé |
| ID | 2.1 |
| Mô tả | Khách hàng xem lại lịch sử đặt vé xe của chính họ. |
| Actor | Khách hàng |
| Priority | Ưu tiên trung bình. |
| Trigger | * Người dùng chủ động xem lịch sử đặt vé. |
| Pre-condition | * Người dùng đã được xác thực thuộc đối tượng actor khách hàng |
| Post-condition | * Người dùng xem được danh sách thông tin của các vé đã đặt. |
| Luồng cơ bản | 1. Người dùng chọn xem lịch sử đặt vé.  2. Hệ thống xác thực vai trò người dùng thành công.  3. Hệ thống trả về và hiện thị ra lịch sử đặt vé. |
| Luồng thay thế |  |
| Luồng ngoại lệ | 2a. Hệ thống xác thực người dùng thất bại.  2a1. Hệ thống thông báo người dùng.  Usecase tiếp tục Usecase 1.1  2b. Hệ thống xác thực người dùng không thuộc actor khách hàng.  2b1. Hệ thống chuyển hướng người dùng. |

Bảng 5. Đặc tả usecase lịch sử đặt vé

|  |  |
| --- | --- |
| Tên | Tra cứu thông tin chuyến xe |
| ID | 3.1 |
| Mô tả | Hỗ trợ người dùng tra cứu thông tin các chuyến xe, lịch chạy, vị trí ghế còn trống. |
| Actor | Tất cả actor |
| Priority | Ưu tiên cao. |
| Trigger | * Người dùng muốn tra cứu thông tin chuyến xe. |
| Pre-condition |  |
| Post-condition | * Người dùng xem được thông tin của các chuyến xe. |
| Luồng cơ bản | 1. Người dùng chọn chức năng tra cứu chuyến xe.  2. Người dùng chọn điểm bắt đầu đi.  3. Hệ thống cập nhật danh sách đích đến tương ứng.  4. Người dùng chọn đích đến.  5. Người dùng chọn ngày xe chạy.  6. Hệ thống cập nhật danh sách giờ chạy tương ứng.  7. Người dùng chọn giờ chạy.  8. Hệ thống hiện thị lên thông tin xe, tuyến đường, sơ đồ xe cùng các trạng thái của từng ghế. |
| Luồng thay thế |  |
| Luồng ngoại lệ | 4a, 5a, 7a. Người dùng chọn điểm bắt đầu đi khác.  Use case tiếp tục từ bước 3.  5b, 7b. Người dùng chọn điểm kết thúc khác.  Use case tiếp tục từ bước 5.  7c. Người dùng chọn ngày xe chạy khác.  Use case tiếp tục từ bước 6.  8d. Hệ thống không tìm được chuyến xe phù hợp, hiển thị rỗng. |

Bảng 6. Đặc tả usecase tra cứu thông tin chuyến xe

|  |  |
| --- | --- |
| Tên | Đặt vé |
| ID | 3.2 |
| Mô tả | Hỗ trợ người dùng chọn vị trí trên ghế xe, và đặt vé |
| Actor | Khách hàng, Nhân viên bán vé. |
| Priority | Ưu tiên cao. |
| Trigger | * Người dùng muốn đặt vé. |
| Pre-condition | * Thực hiện xong use case 3.1. * Người dùng đã được xác thực là khách hàng hoặc nhân viên bán vé. |
| Post-condition | * Đặt vé thành công hoặc thất bại. |
| Luồng cơ bản | 1. Người dùng tra cứu chuyến xe.  2. Người dùng chọn vị trí ghế trên xe.  3. Người dùng xác nhận đặt vé.  4. Hệ thống kiểm tra xác thực người dùng và tiến hành ghi nhận đặt vé. |
| Luồng thay thế | \*Đối với actor Nhân viên bán vé.  3a. Nhân viên nhập thông tin số điện thoại người dùng.  3a1. Hệ thống kiểm tra số điện thoại đã có trong kho dữ liệu chưa  Nếu có thực hiện bước 3a2.  Nếu không thực hiện bước 3b.  3a2. Hệ thống cập nhật hiển thị thông tin khách hàng.  3b. Hệ thống thông báo chưa có thông tin.  3b1. Nhân viên nhập thông tin khách hàng.  4a. Hệ thống thêm thông tin khách hàng  4a1. Hệ thống ghi nhận đặt vé thành công |
| Luồng ngoại lệ | 4c. Hệ thống kiểm tra danh sách vé đặt có những vị trí đã có người đặt trước.  4c1. Hệ thống ghi nhận đặt các vé thành công và gửi thông báo danh sách vé thành công, danh sách vé thất bại.  \*Nếu trong quá trình người dùng thực hiện đổi thông tin trong use case 3.1, thực hiện lại từ bước 1. |

Bảng 7. Đặc tả usecase đặt vé

|  |  |
| --- | --- |
| Tên | Kết thúc chuyến xe |
| ID | 4.1 |
| Mô tả | Xác nhận xe đã tới bến đích |
| Actor | Tài xế |
| Priority | Ưu tiên thấp |
| Trigger | * Tài xế thực hiện kết thúc chuyến xe. |
| Pre-condition | * Người dùng đã được xác thực thuộc actor tài xế |
| Post-condition | * Cập nhật giờ xe đến bến. |
| Luồng cơ bản | 1. Người dùng chọn chuyến xe đang thuộc phận sự quản lý.  2. Người dùng chọn kết thúc chuyến xe.  3. Hệ thống xác thực người dùng thành công.  4. Hệ thống ghi nhận, cập nhật giờ xe đến đích |
| Luồng thay thế |  |
| Luồng ngoại lệ | 1a. Người dùng chọn vào chuyến xe ngoài phận sự.  1a1. Hệ thống báo lỗi.  3b. Hệ thống xác thực người dùng thất bại.  3b1. Hệ thống báo lỗi. |

Bảng 8. Đặc tả usecase kết thúc chuyến xe

|  |  |
| --- | --- |
| Tên | Điểm danh |
| ID | 4.2 |
| Mô tả | Soát vé, kiểm tra những hành hành, vị trí ghế nào đã lên xe. |
| Actor | Tài xế |
| Priority | Ưu tiên thấp |
| Trigger | * Tài xế điểm danh khách lên xe. |
| Pre-condition | * Người dùng đã được xác thực thuộc actor tài xế |
| Post-condition | * Cập nhật tính trạng ghế khách hàng đã lên xe. |
| Luồng cơ bản | 1. Tài xế chọn chuyến xe đang thuộc phận sự quản lý.  2. Tài xế chọn vị trí ghế khách đã lên.  3. Tài xế xác nhận khách lên xe  4. Hệ thống xác thực người dùng thành công.  5. Hệ thống ghi nhận, cập nhật tình trạng ghế. |
| Luồng thay thế |  |
| Luồng ngoại lệ | 1a. Người dùng chọn vào chuyến xe ngoài phận sự.  1a1. Hệ thống báo lỗi.  4b, Hệ thống xác thực người dùng thất bại.  4b1. Hệ thống báo lỗi. |

Bảng 9. Đặc tả usecase điểm danh

|  |  |
| --- | --- |
| Tên | Quản lý nhân viên |
| ID | 5.1 |
| Mô tả | Xem, chỉnh sửa, xóa thông tin các nhân viên của nhà xe. |
| Actor | Quản trị viên |
| Priority | Ưu tiên trung bình |
| Trigger | * Quản trị viên truy cập chức năng quản lý nhân viên. |
| Pre-condition | * Người dùng đã được xác thực thuộc actor quản trị viên. |
| Post-condition | * Cập nhật thông tin mới của nhân viên. |
| Luồng cơ bản | 1. Quản trị viên chọn quản lý nhân viên.  2. Hệ thống xác thực người dùng thành công, hiển thị danh sách nhân viên.  3. Quản trị viên chọn nhân viên cần chỉnh sửa thông tin  4. Quản trị viên chỉnh sửa thông tin.  5. Quản trị viên xác nhận chỉnh sửa.  6. Hệ thống xác thực người dùng thành công.  7. Hệ thống ghi nhận chỉnh sửa thành công, thực hiện lại từ bước 2.  8. Quản trị viên thoát chức năng. |
| Luồng thay thế | 5a. Quản trị viên hủy chỉnh sửa.  Usecase tiếp tục từ bước 2. |
| Luồng ngoại lệ | 2a, Hệ thống xác thực thất bại.  Usecase chuyển sang usecase 1.1  7a. Hệ thống ghi nhận thông tin bị trùng lập.  7a1. Hệ thống thông báo người dùng.  Usecase tiếp tục từ bước 3. |

Bảng 10. Đặc tả usecase quản lý nhân viên

|  |  |
| --- | --- |
| Tên | Cấp tài khoản |
| ID | 5.2 |
| Mô tả | Cấp tài khoản cho nhân viên truy cập vào hệ thống. |
| Actor | Quản trị viên |
| Priority | Ưu tiên cao |
| Trigger | * Quản trị viên truy cập chức năng quản lý nhân viên. Và chọn cấp tài khoản nhân viên. |
| Pre-condition | * Người dùng đã được xác thực thuộc actor quản trị viên. |
| Post-condition | * Tạo mới một tài khoản cho nhân viên tương ứng. |
| Luồng cơ bản | 1. Quản trị viên chọn quản lý nhân viên.  2. Hệ thống xác thực người dùng thành công, hiển thị danh sách nhân viên.  3. Quản trị viên chọn cấp tài khoản cho một nhân viên.  4. Hệ thống tạo mới tài khoản cho nhân viên.  5. Hệ thống thông báo thông tin tài khoản. |
| Luồng thay thế |  |
| Luồng ngoại lệ | 2a. Hệ thống xác thực thất bại.  Usecase chuyển sang usecase 1.1 |

Bảng 11. Đặc tả usecase cấp tài khoản

|  |  |
| --- | --- |
| Tên | Quản lý xe, model xe, bến, trạm |
| ID | 5.3 |
| Mô tả | Quản lý, xem, chỉnh sửa và xóa thông tin mẫu xe, xe, bến xe, trạm dừng chân. |
| Actor | Quản trị viên |
| Priority | Ưu tiên thấp |
| Trigger | * Quản trị viên truy cập chức năng quản lý xe, model xe/bến/trạm |
| Pre-condition | * Người dùng đã được xác thực thuộc actor quản trị viên. |
| Post-condition | * Thông tin được cập nhật, chỉnh sửa, thêm mới. |
| Luồng cơ bản | 1. Quản trị viên chọn chức năng quản lý xe.  2. Hệ thống xác thực thành công.  3. Hệ thống hiển thị danh sách tương ứng.  4. Quản trị viên chọn thêm mới.  5. Quản trị viên nhập các thông tin thêm mới.  6. Quản trị viên xác nhận thêm mới.  7. Hệ thống xác thực người dùng thành công và nạp dữ liệu mới.  8. Hệ thống thông báo thành công. |
| Luồng thay thế | 1a. Quản trị viên chọn chức năng quản lý model xe.  1b. Quản trị viên chọn chức năng quản lý bến.  1c. Quản trị viên chọn chức năng quản lý trạm.  4a. Quản trị viên chọn chỉnh sửa thông tin.  4a1. Quản trị viên nhập thông tin chỉnh sửa.  4a2. Quản trị viên xác nhận cập nhật.  4a3. Hệ thống xác thực người dùng thành công và cập nhật dữ liệu.  Usecase tiếp tục từ bước 8. |
| Luồng ngoại lệ | 2a. Hệ thống xác thực thất bại.  Usecase chuyển sang usecase 1.1  7a. Hệ thống xác thực thất bại.  Hệ thống báo lỗi. |

Bảng 12. Đặc tả usecase quản lý xe, model xe, bến, trạm

|  |  |
| --- | --- |
| Tên | Lập lịch chạy |
| ID | 5.4 |
| Mô tả | Tạo mới một chuyến xe. |
| Actor | Quản trị viên |
| Priority | Ưu tiên thấp |
| Trigger | * Quản trị viên truy cập chức năng lập lịch chạy |
| Pre-condition | * Người dùng đã được xác thực thuộc actor quản trị viên. |
| Post-condition | * Tạo mới một chuyến xe. |
| Luồng cơ bản | 1. Quản trị viên chọn chức năng lập lịch.  2. Hệ thống xác thực người dùng thành công.  3. Quản trị viên nhập các thông tin chuyến xe.  4. Quản trị viên xác nhận thêm mới.  5. Hệ thống xác thực người dùng thành công.  6. Hệ thống thêm mới lịch chạy |
| Luồng thay thế | 4a. Quản trị viên hủy thêm mới.  Usecase trở về bước 1. |
| Luồng ngoại lệ | 2b. Hệ thống xác thực thất bại.  Usecase chuyển sang usecase 1.1  7b. Hệ thống xác thực thất bại.  Hệ thống báo lỗi. |

Bảng 13. Đặc tả usecase lập lịch chạy

|  |  |
| --- | --- |
| Tên | Thống kê |
| ID | 5.5 |
| Mô tả | Thống kê số lượng doanh thu, số vé bán ra,… |
| Actor | Quản trị viên |
| Priority | Ưu tiên thấp |
| Trigger | * Quản trị viên truy cập chức năng thống kê |
| Pre-condition | * Người dùng đã được xác thực thuộc actor quản trị viên. |
| Post-condition | * Hiển thị kết quả thống kê. |
| Luồng cơ bản | 1. Quản trị viên chọn chức năng thống kê.  2. Hệ thống xác thực người dùng thành công.  3. Quản trị viên chọn loại thống kê.  4. Quản trị viên chọn thời gian thống kê.  5. Hệ thống xác thực người dùng thành công.  6. Hệ thống hiển thị kết quả thống kê. |
| Luồng thay thế |  |
| Luồng ngoại lệ | 2a. Hệ thống xác thực thất bại.  Usecase chuyển sang usecase 1.1  7a. Hệ thống xác thực thất bại.  Hệ thống báo lỗi. |

Bảng 14. Đặc tả usecase thống kê

* 1. **Thiết kế cơ sở dữ liệu**
     1. **Phân tích**
        1. Bảng user

Nội dung: Lưu thông tin dùng để xác thực người dùng

Thông tin lưu trữ: Tên tài khoản, mật khẩu, vai trò của tài khoản.

Quan hệ: không có

* + - 1. Bảng nhanvien

Nội dung: Lưu trữ thông tin của nhân viên làm việc thuộc nhà xe.

Thông tin lưu trữ: Thông tin cá nhân, ngày vào làm, chức vụ, của nhân viên.

Quan hệ:

* Quan hệ 1-1 bảng user: 1 nhân viên có thể có 1 tài khoản sử dụng hệ thống hoặc không.
  + - 1. Bảng nhanvienbanve

Nội dung: Lưu thông tin mà chỉ nhân viên bán vé có.

Thông tin lưu trữ: Văn phòng làm việc bán vé của nhân viên.

Quan hệ:

* Quan hệ 1-1 tới bảng nhanvien.
  + - 1. Bảng taixe

Nội dung: Lưu thông tin mà chỉ tài xế có.

Thông tin lưu trữ: Giấy phép lái xe.

Quan hệ:

* Quan hệ 1-1 tới bảng nhanvien.
  + - 1. Bảng khachhang

Nội dung: Lưu thông tin khách hàng đã sữ dụng qua dịch vụ hệ thống.

Thông tin lưu trữ: Thông tin cá nhân khách hàng.

Quan hệ:

* Quan hệ 1-1 tới bảng user, một khách hàng chỉ có duy nhất một tài khoản.
  + - 1. Bảng benxe

Nội dung: Lưu thông tin các bến xe.

Thông tin lưu trữ: Thông tin của bến xe.

Quan hệ: không có

* + - 1. Bảng tramdung

Nội dung: Lưu thông tin các trạm dừng chân.

Thông tin lưu trữ: thông tin của trạm dừng.

Quan hệ:

* Quan hệ n-n với bảng tuyenduong, tạo bảng mới tramdung\_tuyenduong.
  + - 1. Bảng tuyenduong

Nội dung: Lưu thông tin các tuyến đường mà nhà xe có chạy.

Thông tin lưu trữ: Bến xe đầu, bến xe cuối, thời gian chạy, quãng đường.

Quan hệ:

* Quan hệ n-n với bảng benxe (bến bắt đầu), tuyến đường có thể có nhiều điểm xuất phát. Mỗi điểm xuất phát có thể thuộc nhiều tuyến đường.
* Quan hệ n-n với bảng benxe (bến kết thúc), tuyến đường có thể có nhiều điểm kết thúc. Mỗi điểm kết thúc có thể thuộc nhiều tuyến đường.
* Quan hệ n-n với bảng tramdung, tạo bảng mới tramdung\_tuyenduong.
  + - 1. Bảng modelsxe

Nội dung: Lưu thông tin mẫu mã xe, sơ đồ xe.

Thông tin lưu trữ: thông tin mẫu mã , số ghế phục vụ, sơ đồ xe.

Quan hệ: không có.

* + - 1. Bảng xe

Nội dung: Thông tin chi tiết của một chiếc xe.

Thông tin lưu trữ: mẫu mã của xe, biển số xe, ghi chú xe.

Quan hệ: không có.

* Quan hệ 1-n với bảng modelsxe, một xe chỉ được liệt vào duy nhất một mẫu xe, một mẫu xe thì có nhiều xe khác nhau.
  + - 1. Bảng giave

Nội dung: Lưu trữ thông tin của giá vé dựa trên lộ trình và loại xe đưa đón.

Thông tin lưu trữ: Tuyến đường, xe thực hiện chuyến đi, giá vé.

Quan hệ:

* Quan hệ 1-n với bảng tuyenduong, mỗi tuyến đường có thể có nhiều giá vé khác nhau tùy theo loại xe thực hiện chuyến đi
* Quan hệ 1-n với bảng xe, mỗi chiếc xe đưa đón có thể có nhiều giá vé khác nhau tùy theo tuyến đường chạy.
  + - 1. Bảng chuyenxe

Nội dung: Lưu trữ thông tin của một chuyến xe.

Thông tin lưu trữ: Tuyến đường, xe thực thi chuyến đi, tài xế lái xe, giá vé, ngày giờ chạy

Quan hệ:

* Quan hệ 1-n với bảng tuyenduong, 1 chuyến xe chỉ thực hiện một tuyến đường nhưng 1 tuyến đường có thể do nhiều xe thực hiện
* Quan hệ 1- n với bảng taixe, mỗi chuyến xe chỉ có duy nhất 1 tài xế lái xe, nhưng 1 tài xế có thể lái cho nhiều chuyến khác nhau.
* Quan hệ 1-n với bảng xe, 1 chuyến xe chỉ được thực hiện bởi duy nhất 1 chiếc xe nhưng 1 chiếc xe có thể chạy nhiều chuyến khác nhau.
  + - 1. Bảng vexe

Nội dung: Lưu thông tin vé xe.

Thông tin lưu trữ: Người sở hữu vé, tình trạng vé, vị trí ghế, chuyến xe, người tạo vé.

Quan hệ:

* Quan hệ 1-n với bảng khách hàng, 1 khách hàng có thể có nhiều vé nhưng 1 vé chỉ thuộc sở hữu của 1 khách hàng.
* Quan hệ 1-n với bảng user, 1 vé xe chỉ được tạo bởi 1 user, nhưng 1 user có thể tạo nhiều vé xe
* Quan hệ 1-n với bảng chuyến xe, 1 vé xe chỉ thuộc duy nhất một chuyến xe, nhưng 1 chuyến xe có nhiều vé cho nhiều vị trí ghế khác nhau.
  + - 1. Bảng binhluan

Nội dung: Lưu bình luận của khách hàng về dịch vụ.

Thông tin lưu trữ: nội dung bình luận, chuyến xe nhận bình luận, người bình luận.

Quan hệ:

* Quan hệ 1-n với bảng khachhang, 1 khách hàng có thể tạo nhiều bình luận, nhưng 1 bình luận chỉ của duy nhất 1 khách hàng.
* Quan hệ 1-n với bảng chuyenxe, 1 bình luận chỉ áp dụng cho 1 chuyến xe, nhưng 1 chuyến xe có thể có nhiều lời bình luận khác nhau.
  + - 1. Bảng rating

Nội dung: Điểm số khách hàng đánh giá cho dịch vụ.

Thông tin lưu trữ: Điểm đánh giá, người đánh giá, chuyến xe nhận đánh giá.

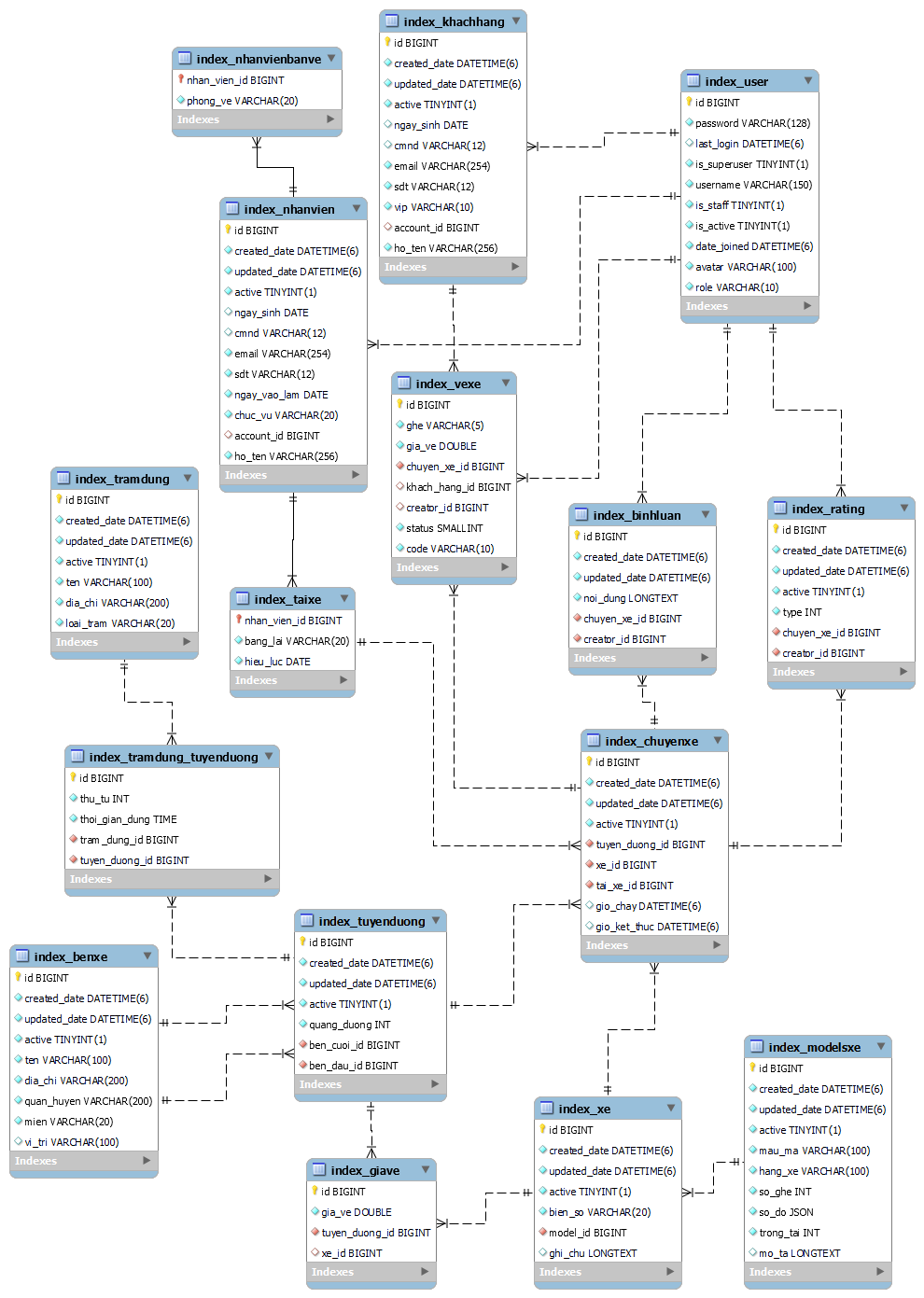
Quan hệ:

* Quan hệ 1-n với bảng khachhang, 1 khách hàng có thể tạo nhiều bình luận, nhưng 1 bình luận chỉ của duy nhất 1 khách hàng.
* Quan hệ 1-n với bảng chuyenxe, 1 bình luận chỉ áp dụng cho 1 chuyến xe, nhưng 1 chuyến xe có thể có nhiều lời bình luận khác nhau.
  + - 1. Bảng tramdung\_tuyenduong

Nội dung: Lưu thông tin các trạm dừng chân mà trên 1 tuyến đường sẽ ghé lại.

Thông tin Lưu trữ: Trạm dừng, thứ tự dừng, tuyến đường, thời gian dừng

Ghi chú: Bảng được tạo ra do quan hệ n-n của 2 bảng tuyenduong và tramdung, một tuyến đường có thể dừng chân tại nhiều trạm khác nhau, và mỗi trạm vẫn có thể là trạm dừng của nhiều tuyến khác nhau.

* + 1. **Bản thiết kế**
  1. **Thiết kế API backend**
     1. **API xác thực oauth2-provider**

Oauth2-provider hỗ trợ người dùng sẵn một số API cung cấp xác thực, và hệ thống sẽ sử dụng các API sau để xác thực tài khoản người dùng, cấp token

* URL: /o/token/
* Methods: POST
* Body data (đăng nhập):

{

“grant\_type”: “password”,

“client\_id”: string,

“client\_secret”: string,

“username”: string,

“password”: string

}

* Body data (Lấy mới token):

{

“grant\_type”: “refresh\_token”,

“client\_id”: string,

“client\_secret”: string,

“refresh\_token”: string

}

* Response data:

{

“expires\_in”: int,

“scope”: string,

“token\_type”: string,

“access\_token”: string,

“refresh\_token”: string

}

* + 1. **API mặc định**

Hệ thống được xây dựng theo hướng code-first tức sẽ dùng code xây dựng lên các model và dùng các model này để tạo lên các bảng, các quan hệ tương ứng trong cơ sở dữ liệu. Vậy nên hệ thống sẽ xây dựng các API xoay quanh các model này. Và mỗi model sẽ có 6 phương thức API mặc định:

|  |  |  |  |
| --- | --- | --- | --- |
| URL | Method | Respone status | Chú thích |
| /{model}/ | GET | 200 | Lấy danh sách tất cả các dòng trong cơ sở dữ liệu |
| /{model}/{id}/ | GET | 200 | Lấy chi tiết một dòng trong sở dữ liệu theo khóa chính |
| /{model}/ | POST | 201 | Thêm mới một dòng vào cơ sở dữ liệu |
| /{model}/{id}/ | PATH | 200 | Cập nhật một vài thuộc tính của một dòng trong cơ sỡ dữ liệu theo khóa chính. |
| /{model}/{id}/ | PUT | 200 | Cập nhật tất cả thuộc tính của một dòng trong cơ sỡ dữ liệu theo khóa chính.  \**Không bao gồm khóa chính* |
| /{model}/{id}/ | DELETE | 204 | Xóa một dòng trong cơ sở dữ liệu |

Bảng 15. Bảng thiết kế API mặc định

* + 1. **API tùy chỉnh**

Các API mà chưa được framework hỗ trợ xây dựng, được tạo mới thủ công hoặc ghi đè lên đường API mặc định.

API đăng ký khách hàng.

* URL: /users/register/
* Methods: POST
* Body data:

{

“username”: string,

“password”: string,

“avatar”: file,

“ho\_ten”: string,

“email”: string,

“sdt”: string

}

* Response status: 201
* Response data:

{

“id”: int,

“username”: string,

“password”: string,

“avatar”: file,

“ho\_ten”: string,

“email”: string,

“sdt”: string,

“created\_date”

}

API lấy thông tin của tài khoản hiện hành.

* URL: /user/current\_user/
* Method: GET
* Request headers:

{

“Authentication”: “bearer <access\_token>”

}

* Response status: 200
* Response data:

{

“id”: int,

“username”: string,

“date\_joined”: string,

“avatar”: string,

“role”: string

}

API lấy thông tin khách hàng.

* URL: /khachhang/cur\_costumer/
* Method: GET
* Request headers:

{

“Authentication”: “bearer <access\_token>”

}

* Response status: 200
* Response data:

{

“user: {

“id”: int,

“username”: string,

“avatar”: string,

“role”: string

},

“khachhang”: {

“id”: int,

“ho\_ten”: string

“ngay\_sinh”: string,

“cmnd”: string,

“sdt”: string,

“email”: string,

“vip”: string,

“account\_id”: int

}

}

API tra cứu bến xe trong tuyến đường khi biết bến đích.

* URL:/tuyenduong/bendau/?bc=?
* Method: GET
* Params: bc là id bến xe đích
* Response status: 200
* Response data:

[

{

“id”: int,

“vi\_tri”: string,

“ten”: string

}

,…]

API tra cứu bến xe trong tuyến đường khi biết bến xuất phát.

* URL:/tuyenduong/bencuoi/?bd=?
* Method: GET
* Params: bc là id bến xe xuất phát
* Response status: 200
* Response data:

[

{

“id”: int,

“vi\_tri”: string,

“ten”: string

}

,…]

API lấy giờ chạy của các chuyến xe theo tuyến đường.

* URL: /tuyenxe/giochay/?diemdi=?&diemden=?&ngay=?
* Method: GET
* Params:
  + diemdi: id bến xuất phát
  + diemden: id bến đích
  + ngay: ngày xe chạy
* Response status: 200
* Response data:

[

{

“id”: int,

“gio\_chay”: string

}

,…]

API lấy danh sách ghế đã đặt của một chuyến xe.

* URL: /chuyenxe/{id}/vexe/
* Method: GET
* Params: id là id của chuyến xe.
* Response status: 200
* Response data:

[

{

“ghe”: string

}

,…]

API đặt vé.

* URL: /chuyenxe/{id}/vexe/
* Method: POST
* Body data:

{

“idkh”: int,

“name”: string,

“sdt”: string,

“ghe”: string

}

\*name và sdt chỉ có khi người tạo vé là nhân viên bán vé

* Request headers:

{

“Authentication”: “bearer <access\_token>”

}

* Response status: 201
* Response data:

{

“id”: int,

“ghe”: string,

“creator”: int,

“khach\_hang”: int,

“chuyen\_xe”: int,

“gia\_ve”: double

}

API lấy thông tin của nhân viên bán vé.

* URL: /nhanvienbanve/info/
* Method: GET
* Request headers:

{

“Authentication”: “bearer <access\_token>”

}

* Response status: 200
* Response data:

{

“nhan\_vien”: {

“id”: int,

“account”: {

“id”: int,

“username”: string,

“avatar”: string,

“date\_joined”: string,

“role”: string

},

“active”: boolean,

“ho\_ten”: string,

“ngay\_sinh”: string,

“cmnd”: string,

“email”: string,

“sdt”: string,

“ngay\_vao\_lam”: string,

“chuc\_vu”: string

},

“phong\_ve”: string,

}

API lấy thông tin của tài xế.

* URL: /taixe/info/
* Method: GET
* Request headers:

{

“Authentication”: “bearer <access\_token>”

}

* Response status: 200
* Response data:

{

"nhan\_vien": {

"id": int,

"account": {

"id": int,

"username": string,

"avatar": string,

"date\_joined": string,

"role": string

},

"active": boolean,

"ho\_ten": string,

"ngay\_sinh": string,

"cmnd": string,

"email": string,

"sdt": string,

"ngay\_vao\_lam": string,

"chuc\_vu": string

},

"phong\_ve": string,

“bang\_lai”: string,

“hieu\_luc”: string,

}

API cập nhật thông tin của nhân viên.

* URL: /nhanvien/{id}/
* Method: PATCH
* Request headers:

{

“Authentication”: “bearer <access\_token>”

}

* Body data:

{

“email”: string,

“sdt”: string,

“ngaysinh”: string,

“oldpassword”: string,

“password”: string,

}

* Response status: 200
* Response data:

{

"nhan\_vien": {

"id": int,

"account": {

"id": int,

"username": string,

"avatar": string,

"date\_joined": string,

"role": string

},

"active": boolean,

"ho\_ten": string,

"ngay\_sinh": string,

"cmnd": string,

"email": string,

"sdt": string,

"ngay\_vao\_lam": string,

"chuc\_vu": string

}

}

1. **KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**
   1. **Vấn đề còn tồn tại**

Chức năng đặt vé chưa hoạt động thực sự ổn định. Trong trường hợp có 2 người dùng cùng lúc truy cập vào cùng một chuyến xe, thì lúc này sơ đồ, vị trí ghế của hai bên sẽ là như nhau, nếu 2 người dùng chọn cùng 1 vị trí ghế thì sẽ xảy ra mâu thuẫn 1 người sẽ đặt được và 1 người sẽ thất bại. Có thể giải quyết bằng cách đồng bộ hóa thời gian thực giữa client và server.

* 1. **Hướng phát triển**
     1. **Tích hợp định vị GPS**

Hiện nay, xe máy cộng nghệ đang ngày càng phát triển vì nó tiện lợi cho mọi người. Hình thức của nó khá đơn giản, vị trí của các tài xế đã được hệ thống nhận biết bằng định vị GPS, Khi khách cần gọi xe, hệ thống sẽ tìm ra tài xế đang hoạt động gần khách nhất và thông báo cho tài xế có một chuyến đi để thực hiện. Khách hàng cũng có thể theo dõi vị trí của tài xế trên bản đồ.

Nếu đem phương pháp trên vào trong việc vận hành bán vé của nhà xe thì như thế nào?

Một nhà xe sẽ chạy rất nhiều tuyến đường khác nhau, mỗi tuyến đường có thể sẽ đi quãng đường rất dài, đi qua nhiều vùng miền, thành phố. Trên đoạn đường này nếu xe vẫn chưa còn những ghế trống thì vẫn có thể đón và thả khách giữa đường. Hiện nay cũng có nhiều nhà xe đã áp dụng việc đưa đón giữa đường nhưng vẫn có vài điểm yếu:

* Khách hàng đặt vé phải thông qua liên hệ tổng đài.
* Tài xế không biết rõ được vị trí của khách hàng để đón.
* Khách hàng không thể chủ động biết được vị trí xe, từ đó mà chuẩn bị trước để đón xe.

Để giải quyết 3 điểm yếu trên, ta có thể tích hợp GPS vào trong hệ thống bán vé này. Cụ thể là:

* Cập nhật thời gian thực vị trí của các chuyến xe trên bản đồ.
* Khách hàng có thể chọn các chuyến xe mà phù hợp với nhu cầu bản thân và đặt vé.
* Tài xế trên xe sẽ nhận được ngay lập tức thông tin của người đặt vé, và tọa độ của người đó trên bản đồ.
* Khách hàng có thể theo dõi vị trí xe trên bản đồ và canh thời gian xe tới.
  + 1. **Thêm chức năng tạo vé điện tử bằng mã vạch QR**

Đa phần các nhà xe hiện tại vẫn sử dụng vé giấy là chính. Kể cả họ có đặt vé online thì vẫn sẽ phải ra điểm phát vé xác nhận thông tin và nhận vé. Vé được dùng khi họ lên xe, ra vào bến xe. Khi kết thúc chuyến đi tấm vé trở nên vô giá trị. Việc dùng vé gây ra một số bất lợi như sau:

* Tốn tài nguyên để sản xuất giấy làm vé.
* Khách hàng dễ đánh mất vé.

Ngày nay đa phần ai cũng sở hữu điện thoại thông minh, ta có thể lợi dụng điều đó, biến những tấm vé truyền thống thành vé điện tử thông qua hình thức mã QR. Khách hàng có thể lưu hình ảnh vé và dùng nó để cho bảo vệ nhà xe, tài xế scan để nhận biết vé.

Một số lợi ích khi dùng vé QR:

* Tiết kiệm tài nguyên môi trường.
* Tránh tình trạng mất vé.
* Hỗ trợ tài xế từ soát vé thủ công sang soát vé điện tử.
* Tiện lợi cho khách hàng, đỡ tốn thời gian nhận vé.

*Lưu ý: Vé xe truyền thống vẫn được sử dụng để phục vụ một số hành khách không sở hữu thiết bị di động thông minh.*

# TÀI LIỆU THAM KHẢO

1. Newzoo, “Top countries by smartphone users,” 2021. [Trực tuyến]. Địa chỉ: https://newzoo.com/insights/rankings/top-countries-by-smartphone-penetration-and-users/ [Truy cập 31/10/2021].
2. WHO, “Coronavirus disease (COVID-19),” 13/05/2021. [Trực tuyến]. Địa chỉ: https://www.who.int/news-room/q-a-detail/coronavirus-disease-covid-19 [Truy cập 31/10/2021].
3. Statista Research Department, “Most used programming languages among developers worldwide, as of 20,” 2021. [Trực tuyến]. Địa chỉ: https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/ [Truy cập 31/10/2021].

Github

1. Sglavoie, “django-rest-framework,” 11/2021. [Trực tuyến]. Địa chỉ: https://github.com/encode/django-rest-framework/tree/master [Truy cập 01/11/2021]
2. Elas7, “react”, 11/2021. [Trực tuyến]. Địa chỉ: https://github.com/facebook/react/ [Truy cập 01/11/2021]

Giáo trình:

1. Dương Hữu Thành, “Giới thiệu django,” Trường ĐH Mở TP.Hồ Chí Minh, 2021
2. Dương Hữu Thành, “Django REST API,” Trường ĐH Mở TP.Hồ Chí Minh, 2021
3. Dương Hữu Thành, “ReactJS,” Trường ĐH Mở TP.Hồ Chí Minh, 2021
4. Hồ Quang Khải, “Phân tích thiết kế hệ thống,” Trường ĐH Mở TP. Hồ Chí Minh, 2019