**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**



**DỰ ÁN LẬP TRÌNH ỨNG DỤNG**

**TÊN ĐỀ TÀI**

**XÂY DỰNG WEBSITE BÁN VÉ MÁY BAY**

Người hướng dẫn**: Ths. Đỗ Thị Tuyết Hoa**

Sinh viên thực hiện**:**

**Nguyễn Văn Lợi LỚP: 23T\_Nhat1 NHÓM: 2**

**Nguyễn Thanh Hậu LỚP: 23T\_Nhat1 NHÓM: 2**

**Trần Minh Nhật LỚP: 23T\_Nhat1 NHÓM: 2**

**Đà Nẵng, 06/2025**

MỤC LỤC

[MỤC LỤC 1](#_efaz5gsmzvc9)

[DANH MỤC HÌNH VẼ 3](#_ajsw17cqmxb3)

[LỜI NÓI ĐẦU 5](#_vj5tcpv5jg3q)

[1.TỔNG QUAN ĐỀ TÀI 7](#_o3imghkmc7k)

[2.CƠ SỞ LÝ THUYẾT 8](#_43y0zh1ntaa8)

[2.1.Ý tưởng 8](#_crk8ya88n5tl)

[2.2.Thuật toán 10](#_1eb267dkirux)

[2.2.1. Thuật toán tìm kiếm chuyến bay(FlightController.searchFlights) 10](#_p1kgvkcvgbwf)

[2.2.2.Thuật toán kiểm tra số ghế trống(FlightService.getAvailableSlots) 12](#_azp7w24igcz4)

[2.2.3. Thuật toán đặt vé (TicketService.addTickets) 14](#_fabjpvgkiw13)

[2.2.4. Thuật toán xử lý đăng nhập (AuthController.login) 18](#_pvo74lobvq1c)

[2.3.Công cụ sử dụng 21](#_bnv3x5l0i7bt)

[3.PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 22](#_b4znz424ecm3)

[3.1. Phát Biểu Yêu Cầu 22](#_8o5p1b40m9j9)

[3.1.1. Đầu Vào (Input) của Hệ Thống 22](#_f69208kw0dm1)

[3.1.2. Đầu Ra (Output) của Hệ Thống 25](#_nwekyjb4galm)

[3.2. Các Biểu Đồ Thiết Kế 29](#_2mu5gb5vr17b)

[3.2.1. Biểu đồ Use Case (Use Case Diagram) 29](#_hc1vj27z3mwv)

[3.2.2. Biểu đồ Lớp (Class Diagram) 31](#_t28wwbzb9f23)

[3.2.3. Biểu đồ Trình Tự (Sequence Diagram) 33](#_yc9ythtsbjof)

[3.2.4. Biểu đồ Hoạt Động (Activity Diagram) 34](#_nxeuldxczsy1)

[3.3.Biểu đồ cơ sở dữ liệu quan hệ hoặc cơ sở dữ liệu phẳng 35](#_y57zfkndij8c)

[3.4.Kiến trúc hệ thống 39](#_nhindnbt8pc4)

[3.4.1. Database Layer (Tầng cơ sở dữ liệu) 39](#_75rdk9ocwg1j)

[3.4.2. Backend Layer (Tầng xử lý phía Server) 41](#_bzkdxt5cxxv)

[3.4.3. Frontend Layer (Tầng giao diện người dùng) 42](#_sl8kfu8ynerj)

[3.4.4. Communication Layer (Tầng giao tiếp) 43](#_fuwno0an4bz3)

[4.CHƯƠNG TRÌNH VÀ KẾT QUẢ 44](#_by21y28efh8)

[4.1.Giao diện chính 44](#_acunuq3qjzej)

[4.1.1.Đăng nhập 44](#_2g6jijmqstz)

[4.1.2.Admin 44](#_drvfpfaxh31p)

[4.2.Kết quả thực thi 48](#_e2sic0v2b3q7)

[4.2.1.User 48](#_wslxl5j2fvvo)

[4.2.2.ADMIN 50](#_we4krxv7b4an)

[4.3.Nhận xét đánh giá 58](#_ya3t152isay1)

[5.KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 59](#_axb0t9nizl2w)

[5.1. Kết luận 59](#_4l7sqvumm8h6)

[5.2. Hướng phát triển 60](#_e759mztj5qbi)

[TÀI LIỆU THAM KHẢO 62](#_dqphp96gob8s)

[PHỤ LỤC 63](#_1a74lgk29d76)

DANH MỤC HÌNH VẼ

Hình 1:Sơ đồ kiến trúc hệ thống (client-server) 7

Hình 2.Bảng logo các công nghệ (HTML5, CSS3, JavaScript, Java, Spring Boot, v.v.). 22

Hình 3: Sơ đồ chức năng hệ thống 31

Hình 4.Database. 38

Hình 5.Đăng Nhập 44

Hình 6.Giao diện Admin và Máy bay 44

Hình 7.Giao diện Chuyến Bay 45

Hình 8.Giao diện người dùng 45

Hình 9.Giao diện đã booking 46

Hình 10.Giao diện tài khoảng-role 46

Hình 11.Giao diện tài khoản 47

Hình 12.Giao diện user 48

Hình 13.Tìm vé 49

Hình 14.Chọn vé 49

Hình 15.Nhập thông tin hành khách đi 50

Hình 16.Click thanh toán 50

Hình 17.Thêm Máy Bay 51

Hình 18.Chỉnh Sửa 51

Hình 19.Thêm chuyến bay 52

Hình 20.Xem sơ đồ ghế 52

Hình 21.Xem chuyến bay 53

Hình 22.Chỉnh sửa chuyến bay 53

Hình 23.Thêm người dùng 54

Hình 24.Chỉnh sửa 54

Hình 25.Chi tiết vé booked 55

Hình 26.Chỉnh sửa vai trò 55

Hình 27.Thêm vai trò 56

Hình 28.Thêm tài khoản 56

Hình 29.Tài khoản sẽ tồn tại ở danh mục tài khoản người dùng 57

Hình 30.Chỉnh sửa tài khoản 57

# LỜI NÓI ĐẦU

Trong thời đại cách mạng công nghiệp lần thứ tư – hay còn gọi là **kỷ nguyên công nghệ 4.0**, thế giới đang chứng kiến sự thay đổi mạnh mẽ và toàn diện trong mọi lĩnh vực của đời sống xã hội. Những tiến bộ vượt bậc trong các lĩnh vực như trí tuệ nhân tạo (AI), học máy (Machine Learning), dữ liệu lớn (Big Data), Internet vạn vật (IoT), công nghệ di động và điện toán đám mây (Cloud Computing) đã làm thay đổi căn bản cách con người làm việc, học tập, giao tiếp và tiếp cận thông tin. Công nghệ thông tin không chỉ đóng vai trò là công cụ hỗ trợ mà còn là **trung tâm cốt lõi trong chiến lược phát triển** của mọi tổ chức, doanh nghiệp và quốc gia.

Việc **ứng dụng công nghệ thông tin vào đời sống** đã và đang góp phần nâng cao hiệu suất làm việc, giảm thiểu rủi ro, tăng cường khả năng quản lý, đồng thời tiết kiệm thời gian và chi phí cho cả cá nhân và tổ chức. Sự phát triển không ngừng của các nền tảng phần mềm đã giúp tự động hóa hàng loạt quy trình truyền thống vốn phụ thuộc vào sức người, từ đó tạo ra một xã hội thông minh, linh hoạt và kết nối.

Một trong những minh chứng rõ ràng nhất cho sự tác động mạnh mẽ của công nghệ thông tin là **sự thay đổi trong hành vi tiêu dùng và tương tác của người dùng với các dịch vụ trực tuyến**. Với sự phổ biến của mạng Internet và thiết bị di động thông minh, người dùng ngày càng có xu hướng thực hiện mọi giao dịch, tra cứu và hoạt động hàng ngày thông qua các nền tảng trực tuyến – từ mua sắm, học tập, làm việc cho đến đi lại và giải trí.

Đặc biệt, trong lĩnh vực **vận tải hàng không**, nhu cầu **đặt vé máy bay trực tuyến** đang ngày càng tăng mạnh. Người tiêu dùng hiện đại không còn muốn phải trực tiếp đến các đại lý hoặc sân bay để mua vé như trước đây, mà mong muốn có thể tự mình tìm kiếm, so sánh giá vé, chọn chuyến bay phù hợp và thực hiện thanh toán chỉ bằng vài thao tác đơn giản trên điện thoại hoặc máy tính. Việc đặt vé online không chỉ giúp tiết kiệm thời gian, công sức mà còn tạo ra sự **chủ động, tiện lợi và linh hoạt** trong việc lên kế hoạch di chuyển.

Xuất phát từ thực tế đó, nhóm chúng em đã lựa chọn và triển khai đề tài: **“Xây dựng website bán vé máy bay trực tuyến”** với mục tiêu thiết kế và phát triển một hệ thống phần mềm có thể hỗ trợ người dùng trong việc tra cứu và đặt vé một cách **nhanh chóng, tiện lợi và chính xác**. Website được xây dựng nhằm cung cấp một nền tảng thân thiện với người dùng, tích hợp đầy đủ các tính năng cần thiết như tìm kiếm chuyến bay, xem chi tiết thông tin hành trình, đặt vé, thanh toán, quản lý thông tin cá nhân, cũng như hỗ trợ chức năng dành riêng cho người quản trị như quản lý người dùng, phân quyền, quản lý chuyến bay, máy bay, địa điểm, và các tùy chọn về hành lý, ghế ngồi…

Ngoài ra, hệ thống cũng được chú trọng đến khía cạnh bảo mật thông tin và trải nghiệm người dùng (UX/UI), nhằm đảm bảo rằng cả người dùng cuối và quản trị viên đều có thể thao tác một cách dễ dàng, an toàn và hiệu quả.

Thông qua đề tài này, nhóm chúng em mong muốn không chỉ vận dụng những kiến thức đã học về lập trình, cơ sở dữ liệu, phân tích hệ thống, thiết kế giao diện, bảo mật và phát triển ứng dụng web, mà còn rèn luyện kỹ năng làm việc nhóm, quản lý thời gian, phân chia công việc và giải quyết vấn đề trong quá trình thực hiện một dự án thực tế. Đây là cơ hội quý báu để chúng em tiếp cận và trải nghiệm quy trình phát triển phần mềm theo hướng chuyên nghiệp, từ giai đoạn khảo sát, phân tích yêu cầu, thiết kế, lập trình đến kiểm thử và triển khai sản phẩm.

Chúng em hy vọng rằng hệ thống website đặt vé máy bay trực tuyến mà nhóm đã xây dựng không chỉ mang lại giá trị thực tiễn, phục vụ tốt nhu cầu của người dùng, mà còn có thể trở thành nền tảng phát triển mở rộng trong tương lai, hướng tới tích hợp thêm các tính năng thông minh như đề xuất chuyến bay tối ưu, tích hợp bản đồ sân bay, đồng bộ hóa với ứng dụng di động, v.v.

Cuối cùng, nhóm chúng em xin gửi lời cảm ơn chân thành đến quý thầy cô, đặc biệt là giảng viên hướng dẫn, đã tận tình chỉ bảo, định hướng và hỗ trợ trong suốt quá trình thực hiện đề tài. Đồng thời, chúng em cũng mong muốn nhận được những ý kiến đóng góp quý báu từ thầy cô, bạn bè và người dùng để tiếp tục cải thiện và hoàn thiện sản phẩm.

## 1.TỔNG QUAN ĐỀ TÀI

• Mục tiêu dự án

• Mục tiêu của dự án là tạo ra một nền tảng trực tuyến cho phép người dùng dễ dàng đặt vé máy bay, đồng thời giúp các doanh nghiệp quản lý hoạt động bán vé một cách hiệu quả.

• Cấu trúc dự án

• Dự án được tổ chức thành hai phần chính:

* **Frontend**: Chịu trách nhiệm về giao diện người dùng, được xây dựng bằng các công nghệ như HTML, CSS và JavaScript.
* **Backend**: Xử lý logic nghiệp vụ và quản lý dữ liệu, sử dụng ngôn ngữ Java.

• Công nghệ sử dụng

* **Ngôn ngữ lập trình**: Java , JavaScript, HTML ,CSS .
* **Cơ sở dữ liệu**: Chưa được chỉ định rõ trong kho lưu trữ, nhưng có thể sử dụng các hệ quản trị cơ sở dữ liệu phổ biến như MySQL hoặc SQLserver.

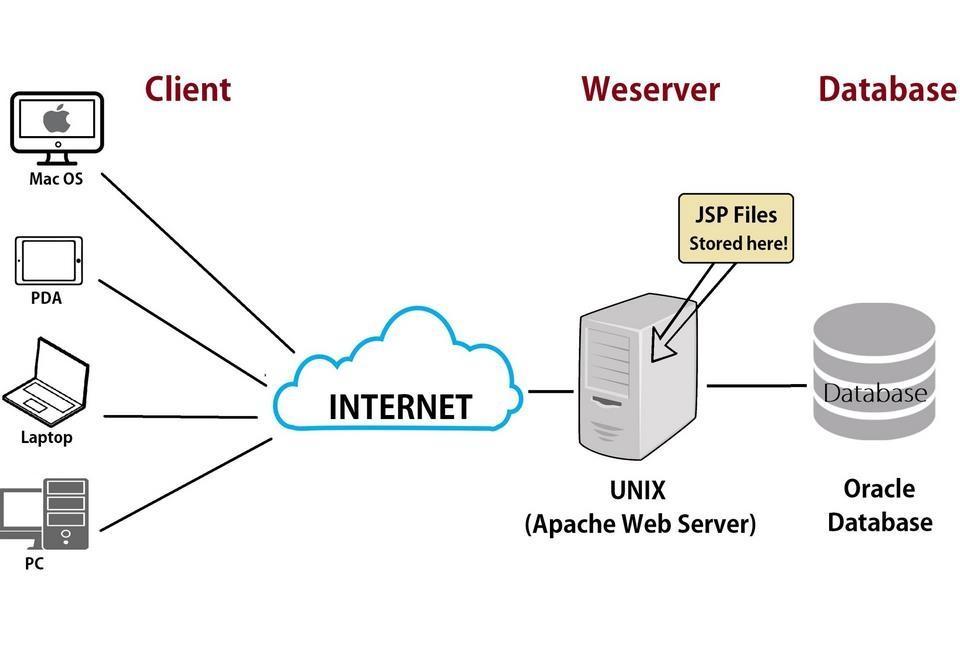
• Chức năng chính

• Dựa trên cấu trúc và mục tiêu của dự án, các chức năng chính có thể bao gồm:

* **Đặt vé máy bay trực tuyến**: Cho phép người dùng tìm kiếm và đặt vé máy bay một cách dễ dàng…
* **Quản lý thông tin chuyến bay**: Cập nhật và quản lý thông tin về các chuyến bay…
* **Quản lý người dùng**: Đăng ký, đăng nhập và quản lý thông tin cá nhân của người dùng…

• Kết luận

• Dự án PBL3\_Web nhằm mục đích cung cấp một giải pháp trực tuyến cho việc đặt vé máy bay, giúp người dùng tiết kiệm thời gian và nâng cao trải nghiệm mua vé. Đồng thời, hệ thống cũng hỗ trợ các doanh nghiệp trong việc quản lý và tối ưu hóa hoạt động bán vé.



**Hình 1:Sơ đồ kiến trúc hệ thống (client-server).**

## 2.CƠ SỞ LÝ THUYẾT

### 2.1.Ý tưởng

Trong những năm gần đây, nhu cầu di chuyển bằng đường hàng không của người dân ngày càng tăng cao, không chỉ phục vụ cho mục đích công tác, học tập mà còn để du lịch, thăm thân hoặc khám phá thế giới. Song song với đó là sự phát triển mạnh mẽ của công nghệ thông tin và Internet, đặc biệt là xu hướng tiêu dùng số – nơi mà người dùng ưu tiên thực hiện mọi giao dịch một cách **nhanh chóng, tiện lợi và linh hoạt** thông qua các nền tảng trực tuyến. Chính vì thế, mô hình **đặt vé máy bay online** ngày càng trở nên phổ biến và được ưa chuộng.

Tuy nhiên, trên thực tế vẫn còn tồn tại nhiều **hạn chế trong các hệ thống đặt vé hiện tại**. Một số website có **giao diện phức tạp, khó sử dụng**, **thiếu tính năng so sánh hoặc lọc thông minh**, **quá trình thanh toán rườm rà**, hoặc chưa tối ưu cho thiết bị di động. Ngoài ra, phía nhà quản trị – gồm các đại lý hoặc nhân viên hãng hàng không – cũng gặp khó khăn trong việc **quản lý chuyến bay, tình trạng chỗ ngồi, hành lý và lịch sử đặt vé** nếu hệ thống không hỗ trợ đầy đủ.

Xuất phát từ thực tế đó, nhóm chúng em quyết định lựa chọn và triển khai đề tài **“Xây dựng website bán vé máy bay trực tuyến”** với mục tiêu phát triển một nền tảng **đa chức năng, thân thiện với người dùng**, giúp nâng cao hiệu quả trong quá trình tìm kiếm, đặt vé và quản lý thông tin chuyến bay.

Cụ thể, hệ thống được xây dựng với các **mục tiêu chính** như sau:

* **Đối với người dùng:**
  + Cho phép tra cứu chuyến bay theo tiêu chí như điểm đi, điểm đến, ngày khởi hành, hãng bay, loại vé.
  + Hỗ trợ lọc kết quả theo mức giá, thời gian bay, hạng vé,...
  + Hiển thị thông tin chi tiết về chuyến bay: giờ khởi hành, thời lượng, số ghế trống, hành lý kèm theo,...
  + Thực hiện thao tác đặt vé đơn giản, rõ ràng và an toàn.
  + Hỗ trợ chọn ghế ngồi mong muốn nếu còn chỗ.
  + Gửi thông báo xác nhận qua email và lưu trữ lịch sử đặt vé cá nhân.
  + Giao diện responsive, tối ưu hiển thị trên điện thoại, máy tính bảng và máy tính.
* **Đối với quản trị viên (admin/đại lý):**
  + Quản lý danh sách các chuyến bay, điểm đi – điểm đến, thời gian và loại máy bay.
  + Quản lý thông tin vé, hành lý, ghế ngồi, bảng giá.
  + Xử lý đơn đặt vé, trạng thái vé (đã thanh toán, đang chờ, bị hủy,...).
  + Thống kê doanh thu, số lượng vé bán ra, lịch sử hoạt động.
  + Quản lý người dùng, phân quyền truy cập, đảm bảo bảo mật hệ thống.

Ngoài ra, một điểm quan trọng của hệ thống là **cơ chế phân quyền rõ ràng** giữa người dùng thông thường và người quản trị, nhằm đảm bảo rằng mỗi nhóm người dùng chỉ thực hiện được các thao tác phù hợp với quyền hạn được cấp. Điều này giúp tăng cường tính bảo mật, ổn định và chuyên nghiệp cho hệ thống.

Về mặt kỹ thuật, nhóm lựa chọn phát triển hệ thống theo kiến trúc client-server. Phía frontend sử dụng các công nghệ như **HTML5, CSS3,** để tạo giao diện hiện đại, dễ sử dụng. Phía backend sử dụng **ngôn ngữ Java kết hợp Spring Boot**, đảm nhận xử lý nghiệp vụ và kết nối cơ sở dữ liệu **SQLSEVER**. Dữ liệu về người dùng, chuyến bay, vé, hành lý,... được lưu trữ trong một **cơ sở dữ liệu quan hệ** với mô hình thiết kế tối ưu, tránh dư thừa và đảm bảo tính toàn vẹn dữ liệu.

Ý tưởng của hệ thống không chỉ hướng đến việc **tạo ra một sản phẩm có giá trị thực tiễn**, dễ ứng dụng trong đời sống mà còn là cơ hội để nhóm sinh viên rèn luyện kỹ năng chuyên môn về lập trình, thiết kế giao diện, xử lý dữ liệu, làm việc nhóm và quản lý dự án phần mềm theo hướng thực chiến.

### 2.2.Thuật toán

• Trong quá trình xây dựng hệ thống website bán vé máy bay trực tuyến, nhóm đã xác định một số chức năng cốt lõi cần thuật toán xử lý hiệu quả để đảm bảo tính chính xác, tốc độ phản hồi và trải nghiệm người dùng. Dưới đây là mô tả các thuật toán chính được áp dụng trong hệ thống:

## 2.2.1. Thuật toán tìm kiếm chuyến bay(FlightController.searchFlights)

• Mô tả: Phương thức searchFlights trong FlightController cho phép tìm kiếm các chuyến bay dựa trên điểm khởi hành (fromLocationId), điểm đến (toLocationId), và ngày khởi hành (departureDate).

Mã nguồn liên quan:

@GetMapping("/find\_flight")

public List<FlightDTO> searchFlights(

@RequestParam int fromLocationId,

@RequestParam int toLocationId,

@RequestParam @DateTimeFormat(iso = DateTimeFormat.ISO.DATE) LocalDate departureDate) {

Location from = locationRepo.findById(fromLocationId).orElseThrow();

Location to = locationRepo.findById(toLocationId).orElseThrow();

List<Flight> flights = flightRepo.findByFromLocationAndToLocationAndDepartureDate(from, to, java.sql.Date.valueOf(departureDate));

List<FlightDTO> flightDTOs = new ArrayList<>();

for(Flight flight : flights){

FlightDTO flightDTO = new FlightDTO(flight);

flightDTO.setAvailableEconomyTicket(flightService.getAvailableSlots(flight.getIdFlight(), TicketType.ECONOMY));

flightDTO.setAvailableBusinessTicket(flightService.getAvailableSlots(flight.getIdFlight(), TicketType.BUSINESS));

flightDTOs.add(flightDTO);

}

return flightDTOs;

}

Thuật toán:

1. Nhập liệu:
   * Nhận fromLocationId, toLocationId (ID của điểm khởi hành và điểm đến), và departureDate (ngày khởi hành).
   * Các tham số được truyền qua query parameters trong request HTTP.
2. Kiểm tra tính hợp lệ:
   * Sử dụng locationRepo.findById để tìm kiếm thông tin của điểm khởi hành và điểm đến trong cơ sở dữ liệu.
   * Nếu một trong hai địa điểm không tồn tại, ném ra ngoại lệ (sử dụng orElseThrow).
3. Truy vấn cơ sở dữ liệu:
   * Gọi flightRepo.findByFromLocationAndToLocationAndDepartureDate để lấy danh sách các chuyến bay thỏa mãn điều kiện: điểm khởi hành, điểm đến, và ngày khởi hành.
   * Phương thức này sử dụng Spring Data JPA để thực hiện truy vấn SQL trên bảng Flight.
4. Xử lý kết quả:
   * Duyệt qua danh sách các chuyến bay (flights) và chuyển đổi mỗi chuyến bay thành đối tượng FlightDTO.
   * Tính toán số vé còn lại cho hạng phổ thông (ECONOMY) và hạng thương gia (BUSINESS) bằng cách gọi flightService.getAvailableSlots.
   * Thêm các FlightDTO vào danh sách kết quả.
5. Trả về:
   * Trả về danh sách FlightDTO dưới dạng JSON thông qua ResponseEntity.

Đánh giá thuật toán:

* Độ phức tạp:
  + Truy vấn cơ sở dữ liệu: O(1) cho tìm kiếm theo ID (findById) và O(n) cho truy vấn các chuyến bay (n là số lượng chuyến bay trong cơ sở dữ liệu).
  + Vòng lặp xử lý danh sách chuyến bay: O(m), với m là số lượng chuyến bay trả về.
  + Tổng độ phức tạp: Phụ thuộc vào cơ sở dữ liệu và số lượng chuyến bay, thường là O(n + m).
* Ưu điểm:
  + Sử dụng Spring Data JPA giúp đơn giản hóa việc truy vấn cơ sở dữ liệu.
  + Kiểm tra tính hợp lệ của địa điểm trước khi truy vấn, giảm nguy cơ lỗi.
* Nhược điểm:
  + Nếu số lượng chuyến bay trong cơ sở dữ liệu lớn, truy vấn có thể chậm nếu không có index phù hợp trên các cột fromLocation, toLocation, và departureDate.
  + Không có bộ nhớ đệm (cache) để lưu kết quả truy vấn thường xuyên, có thể ảnh hưởng đến hiệu suất nếu API được gọi nhiều lần.

Cải tiến tiềm năng:

* Thêm index trên các cột liên quan trong cơ sở dữ liệu để tăng tốc độ truy vấn.
* Sử dụng caching (ví dụ: Spring Cache hoặc Redis) để lưu kết quả tìm kiếm phổ biến.
* Thêm phân trang (pagination) để giới hạn số lượng chuyến bay trả về, tránh tải quá nhiều dữ liệu.

## 2.2.2.Thuật toán kiểm tra số ghế trống(FlightService.getAvailableSlots)

• Mô tả: Phương thức getAvailableSlots trong FlightService tính toán số lượng vé còn lại (hạng phổ thông hoặc thương gia) cho một chuyến bay cụ thể.

• Mã nguồn liên quan:

public int getAvailableSlots(int flightId, TicketType ticketType) {

SeatType seatType = SeatType.ECONOMY;

if (ticketType == TicketType.BUSINESS) {

seatType = SeatType.BUSINESS;

}

List<Flights\_Seat> allFSOfSeatType = getFlightSeats(flightId, seatType, null);

List<Ticket> bookedTickets = getBookedTickets(flightId, ticketType);

*// Trả về số lượng ghế trống*

return allFSOfSeatType.size() - bookedTickets.size();

}

Thuật toán:

1. Xác định loại ghế:
   * Nếu ticketType là BUSINESS, đặt seatType thành BUSINESS; nếu không, đặt thành ECONOMY.
2. Lấy danh sách ghế:
   * Gọi getFlightSeats(flightId, seatType, null) để lấy tất cả các ghế (Flights\_Seat) của chuyến bay theo loại ghế (seatType).
3. Lấy danh sách vé đã đặt:
   * Gọi getBookedTickets(flightId, ticketType) để lấy danh sách vé đã đặt cho chuyến bay và loại vé tương ứng.
4. Tính toán số ghế trống:
   * Trả về số ghế trống bằng cách lấy số ghế có sẵn (allFSOfSeatType.size()) trừ đi số vé đã đặt (bookedTickets.size()).

Đánh giá thuật toán:

* Độ phức tạp:
  + getFlightSeats: O(n), với n là số lượng ghế trong chuyến bay.
  + getBookedTickets: O(m), với m là số lượng vé đã đặt.
  + Tổng độ phức tạp: O(n + m).
* Ưu điểm:
  + Đơn giản, dễ hiểu, và trực tiếp tính toán số ghế trống.
  + Tận dụng các truy vấn từ cơ sở dữ liệu thông qua Spring Data JPA.
* Nhược điểm:
  + Hiệu suất có thể bị ảnh hưởng nếu chuyến bay có nhiều ghế hoặc vé đã đặt, đặc biệt khi không có index trên bảng Flights\_Seat và Ticket.
  + Không kiểm tra trạng thái ghế (SeatStatus) trong hàm này, mà dựa vào hàm getFlightSeats, có thể gây nhầm lẫn nếu trạng thái ghế không được cập nhật đúng.

Cải tiến tiềm năng:

* Thêm index trên các cột liên quan trong bảng Flights\_Seat và Ticket để tăng tốc độ truy vấn.
* Caching kết quả số ghế trống để giảm truy vấn cơ sở dữ liệu.
* Kiểm tra rõ ràng hơn về trạng thái ghế (SeatStatus.NOT\_BOOKED) trong hàm này để đảm bảo tính chính xác.

#### 2.2.3. Thuật toán đặt vé (TicketService.addTickets)

• Mô tả: Phương thức addTickets trong TicketService xử lý việc đặt vé cho một chuyến bay, bao gồm kiểm tra ghế trống, cập nhật trạng thái ghế, và lưu vé vào cơ sở dữ liệu.

• Mã nguồn liên quan:

@Transactional

List<Ticket> addTickets(BookingRequest bookingRequest, Booking booking) {

*// bước kiểm tra*

try {

checkCustomerLimit(bookingRequest);

} catch (RuntimeException e) {

throw new RuntimeException(e.getMessage());

}

*// bước thực hiện*

List<TicketRequest> requestTickets = bookingRequest.getTickets();

Integer departureFlightId = bookingRequest.getDepartureFlightId();

Integer returnFlightId = bookingRequest.getReturnFlightId();

TicketType departureTicketType = bookingRequest.getDepartureTicketType();

TicketType returnTicketType = bookingRequest.getReturnTicketType();

List<Ticket> tickets = new ArrayList<>();

Flight departureFlight = flightRepo.findByIdFlight(departureFlightId);

Flight returnFlight = null;

SeatType seatType = departureTicketType == TicketType.ECONOMY ? SeatType.ECONOMY : SeatType.BUSINESS;

Map<String, Flights\_Seat> departureSeats = flightService

.getFlightSeats(departureFlightId, seatType, SeatStatus.NOT\_BOOKED).stream()

.collect(Collectors.toMap(seat -> seat.getSeat().getSeatNumber(), seat -> seat));

Map<String, Flights\_Seat> returnSeats = null;

if (returnFlightId != null) {

returnFlight = flightRepo.findByIdFlight(returnFlightId);

returnSeats = flightService

.getFlightSeats(returnFlightId, seatType, SeatStatus.NOT\_BOOKED).stream()

.collect(Collectors.toMap(seat -> seat.getSeat().getSeatNumber(), seat -> seat));

}

Flights\_Seat departureSeat = null;

ReturnTicket returnTicket = null;

for (TicketRequest ticketRequest : requestTickets) {

departureSeat = null;

returnTicket = null;

if (ticketRequest.getSeatId() != null) {

departureSeat = departureSeats.get(ticketRequest.getSeatId());

if (departureSeat == null) {

throw new RuntimeException("Seat not available");

}

departureSeat.setSeatStatus(SeatStatus.BOOKED);

}

if (ticketRequest.getReturnTicket() != null) {

Flights\_Seat returnSeat = null;

if (ticketRequest.getReturnTicket().getSeatId() != null) {

returnSeat = returnSeats.get(ticketRequest.getReturnTicket().getSeatId());

if (returnSeat == null) {

throw new RuntimeException("Seat not available");

}

returnSeat.setSeatStatus(SeatStatus.BOOKED);

}

returnTicket = new ReturnTicket(null, returnFlight, returnTicketType, returnSeat);

}

ticketRequest.getUser().setUserType(UserType.CUSTOMER);

User user = userService.createUpdateUser(ticketRequest.getUser());

Ticket ticket = new Ticket(user, departureFlight, departureTicketType, departureSeat, returnTicket);

if (returnTicket != null) {

returnTicket.setTicket(ticket);

}

ticket.setBooking(booking);

ticket.setPrice(departureFlight.getPrice(departureTicketType));

ticketRepo.save(ticket);

tickets.add(ticket);

}

return tickets;

}

Thuật toán:

1. Kiểm tra giới hạn ghế:
   * Gọi checkCustomerLimit để đảm bảo số lượng vé yêu cầu không vượt quá số ghế trống.
   * Nếu vượt quá, ném ra ngoại lệ RuntimeException.
2. Chuẩn bị dữ liệu:
   * Lấy danh sách các yêu cầu vé (requestTickets), ID chuyến bay đi (departureFlightId), ID chuyến bay về (nếu có, returnFlightId), và loại vé (departureTicketType, returnTicketType).
   * Xác định loại ghế (seatType) dựa trên departureTicketType (ECONOMY hoặc BUSINESS).
   * Lấy danh sách ghế chưa đặt (SeatStatus.NOT\_BOOKED) cho chuyến bay đi và chuyến bay về (nếu có) bằng flightService.getFlightSeats.
3. Xử lý từng yêu cầu vé:
   * Duyệt qua danh sách requestTickets.
   * Nếu vé yêu cầu một ghế cụ thể (seatId), kiểm tra xem ghế có sẵn trong departureSeats. Nếu không, ném ngoại lệ.
   * Cập nhật trạng thái ghế thành BOOKED.
   * Nếu có vé khứ hồi (returnTicket), thực hiện tương tự cho ghế của chuyến bay về.
   * Tạo hoặc cập nhật thông tin người dùng (User) và đặt loại người dùng là CUSTOMER.
   * Tạo đối tượng Ticket, liên kết với booking, departureFlight, departureSeat, và returnTicket (nếu có).
   * Lưu vé vào cơ sở dữ liệu bằng ticketRepo.save.
4. Trả về:
   * Trả về danh sách các vé đã được tạo.

Đánh giá thuật toán:

* Độ phức tạp:
  + Kiểm tra giới hạn ghế (checkCustomerLimit): O(n + m), với n là số ghế, m là số vé đã đặt.
  + Lấy danh sách ghế và vé: O(n + m).
  + Vòng lặp xử lý yêu cầu vé: O(k), với k là số lượng vé yêu cầu.
  + Tổng độ phức tạp: O(n + m + k).
* Ưu điểm:
  + Sử dụng @Transactional để đảm bảo tính toàn vẹn dữ liệu (hoặc tất cả các vé được lưu, hoặc không lưu gì nếu có lỗi).
  + Kiểm tra kỹ lưỡng trạng thái ghế và số lượng vé trống trước khi đặt.
* Nhược điểm:
  + Nếu số lượng ghế hoặc vé lớn, việc tải toàn bộ danh sách ghế và vé có thể gây chậm.
  + Không có cơ chế khóa (locking) rõ ràng để xử lý đồng thời (concurrency), có thể dẫn đến lỗi nếu nhiều người dùng đặt vé cùng lúc.
  + Không kiểm tra trạng thái chuyến bay (ví dụ: chuyến bay đã bị hủy hoặc đã khởi hành).

Cải tiến tiềm năng:

* Thêm cơ chế khóa lạc quan (optimistic locking) hoặc khóa bi quan (pessimistic locking) để xử lý đồng thời.
* Tối ưu hóa truy vấn cơ sở dữ liệu bằng cách chỉ lấy các ghế cần thiết thay vì toàn bộ danh sách.
* Thêm kiểm tra trạng thái chuyến bay trước khi đặt vé.

#### 2.2.4. Thuật toán xử lý đăng nhập (AuthController.login)

• Mô tả: Phương thức login trong AuthController xử lý việc đăng nhập người dùng, tạo JWT token, và lưu token vào cookie.

• Mã nguồn liên quan:

@PostMapping("/login")

public ResponseEntity<?> login(@RequestBody Account account,

HttpServletResponse response) {

String jwtToken = null;

try {

jwtToken = jwtService.generateToken(account.getUsername(), account.getPassword());

} catch (LoginFailedException e) {

return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body(e.getMessage());

}

*// Tạo cookie*

ResponseCookie cookie = ResponseCookie.from("token", jwtToken)

.httpOnly(true)

.secure(false)

.sameSite("Lax")

.path("/")

.maxAge(Duration.ofHours(1))

.build();

*// Đưa cookie vào response*

response.addHeader(HttpHeaders.SET\_COOKIE, cookie.toString());

*// Trả response (tùy bạn)*

return ResponseEntity.ok(jwtService.getPermissions(jwtToken));

}

• Thuật toán:

1. Nhập liệu:
   * Nhận username và password từ request body thông qua đối tượng Account.
2. Tạo JWT token:
   * Gọi jwtService.generateToken để xác thực thông tin đăng nhập và tạo JWT token.
   * Nếu xác thực thất bại, ném LoginFailedException và trả về lỗi 401 (Unauthorized).
3. Tạo cookie:
   * Tạo một ResponseCookie chứa JWT token với các thuộc tính:
     + httpOnly: true (ngăn chặn truy cập cookie qua JavaScript).
     + secure: false (cho phép sử dụng trên HTTP, không chỉ HTTPS).
     + sameSite: "Lax" (bảo vệ chống CSRF).
     + path: "/" (có thể truy cập trên toàn bộ ứng dụng).
     + maxAge: 1 giờ.
   * Thêm cookie vào header của response.
4. Trả về:
   * Trả về danh sách quyền (permissions) liên quan đến token thông qua jwtService.getPermissions.

Đánh giá thuật toán:

* Độ phức tạp: O(1) cho việc tạo token và cookie, phụ thuộc vào triển khai của JwtService.
* Ưu điểm:
  + Sử dụng JWT để xác thực, an toàn và phổ biến.
  + Cookie được cấu hình với các thuộc tính bảo mật như httpOnly và sameSite.
* Nhược điểm:
  + Thiếu cơ chế làm mới token (refresh token) để duy trì phiên đăng nhập lâu dài.
  + Thiết lập secure: false có thể gây rủi ro bảo mật nếu ứng dụng chạy trên HTTPS.
  + Không kiểm tra trạng thái tài khoản (ví dụ: tài khoản bị khóa).

Cải tiến tiềm năng:

* Thêm cơ chế refresh token để kéo dài thời gian đăng nhập.
* Đặt secure: true khi triển khai trên HTTPS.
* Thêm kiểm tra trạng thái tài khoản trước khi tạo token.

### 2.3.Công cụ sử dụng

• Hệ thống được xây dựng dựa trên kiến trúc Client-Server, chia thành hai phần chính: Frontend (giao diện người dùng) và Backend (xử lý nghiệp vụ và dữ liệu). Dưới đây là các công nghệ được sử dụng trong từng phần:

• Frontend (Giao diện người dùng)

* **HTML5**: Ngôn ngữ đánh dấu siêu văn bản, tạo cấu trúc cho các trang web.
* **CSS3**: Ngôn ngữ tạo kiểu cho các phần tử HTML, giúp thiết kế giao diện đẹp và thân thiện với người dùng.
* **JavaScript**: Ngôn ngữ lập trình phía client, tạo các hiệu ứng động và xử lý sự kiện trên trình duyệt.
* **Bootstrap**: Framework CSS giúp thiết kế giao diện responsive, tương thích với nhiều loại thiết bị.

• Backend (Xử lý nghiệp vụ và dữ liệu)

* **Java**: Ngôn ngữ lập trình hướng đối tượng, được sử dụng để xây dựng các chức năng xử lý nghiệp vụ của hệ thống.
* **Spring Boot**: Framework giúp phát triển ứng dụng Java nhanh chóng, cung cấp các tính năng như Dependency Injection, RESTful API, và tích hợp dễ dàng với các công nghệ khác.
* **SQLSEVER**: Hệ quản trị cơ sở dữ liệu quan hệ, lưu trữ thông tin về người dùng, chuyến bay, vé máy bay, v.v.
* **Hibernate**: Framework ORM (Object-Relational Mapping) giúp ánh xạ các đối tượng Java với các bảng trong cơ sở dữ liệu, giảm thiểu việc viết các câu lệnh SQL thủ công.
* **Maven**: Công cụ quản lý dự án và phụ thuộc, giúp tự động hóa quá trình build và quản lý thư viện.

• Công cụ và môi trường phát triển

* **IntelliJ IDEA**: Môi trường phát triển tích hợp (IDE) hỗ trợ lập trình Java, cung cấp nhiều tính năng hữu ích như tự động hoàn thành mã, kiểm tra lỗi, và tích hợp với các công cụ khác.
* **Git**: Hệ thống quản lý phiên bản phân tán, giúp theo dõi lịch sử thay đổi của mã nguồn và hỗ trợ làm việc nhóm hiệu quả.
* **GitHub**: Nền tảng lưu trữ mã nguồn trực tuyến, hỗ trợ quản lý dự án và cộng tác giữa các thành viên trong nhóm.
* **Postman**: Công cụ kiểm thử API, giúp gửi các yêu cầu HTTP đến server và kiểm tra phản hồi, hỗ trợ quá trình phát triển và debug.

A person sitting at a computer

Description automatically generated

**Hình 2.Bảng logo các công nghệ (HTML5, CSS3, JavaScript, Java, Spring Boot, v.v.).**

## 3.PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

### 3.1. Phát Biểu Yêu Cầu

#### 3.1.1. Đầu Vào (Input) của Hệ Thống

• Hệ thống "Flight App" nhận các đầu vào từ người dùng thông qua các API RESTful và giao diện người dùng (frontend). Các đầu vào chính bao gồm:

• Thông tin xác thực người dùng (Authentication):

• Đăng nhập (Login):

• Tên người dùng (username) và mật khẩu (password) được gửi qua endpoint /login (POST) để xác thực người dùng và tạo JWT token.

• Ví dụ: Account object chứa {username, password}.

• Đăng xuất (Logout):

• Không yêu cầu thông tin cụ thể, chỉ cần gửi yêu cầu tới endpoint /logoutApp (POST) để xóa cookie chứa JWT token.

• Quản lý tài khoản (Account Management):

• Tìm kiếm tài khoản:

• Email (username) và tùy chọn roleId qua endpoint /account/find-account-by-email (GET).

• Lấy thông tin tài khoản theo ID:

• ID tài khoản (id) qua endpoint /account/{id} (GET).

• Tạo tài khoản mới:

• Đối tượng AccountDTO chứa thông tin như username, password, role, v.v. qua endpoint /account (POST).

• Cập nhật tài khoản:

• ID tài khoản (id) và đối tượng Account chứa thông tin cập nhật qua endpoint /account/{id} (PUT).

• Cập nhật hồ sơ người dùng:

• ID tài khoản (id) và đối tượng User chứa thông tin hồ sơ qua endpoint /account/{id}/change-profile (PUT).

• Xóa tài khoản:

• ID tài khoản (id) qua endpoint /account/{id} (DELETE).

• Quản lý chuyến bay (Flight Management):

• Tạo chuyến bay mới:

• Đối tượng Flight chứa thông tin như điểm khởi hành, điểm đến, ngày giờ khởi hành, v.v. qua endpoint /flight (POST).

• Cập nhật chuyến bay:

• ID chuyến bay (idFlight) và đối tượng Flight chứa thông tin cập nhật qua endpoint /flight/{id} (PUT).

• Xóa chuyến bay:

• ID chuyến bay (id) qua endpoint /flight/{id} (DELETE).

• Tìm kiếm chuyến bay:

• ID điểm khởi hành (fromLocationId), ID điểm đến (toLocationId), và ngày khởi hành (departureDate) qua endpoint /flight/find\_flight (GET).

• Lấy thông tin chuyến bay:

• ID chuyến bay (idFlight) qua endpoint /flight/{idFlight} hoặc /flight/{idFlight}/information (GET).

• Lấy ghế trên chuyến bay:

• ID chuyến bay (flightId) qua endpoint /flight/flight\_seats (GET).

• Quản lý đặt vé (Booking Management):

• Tạo đặt vé:

• Đối tượng BookingRequest chứa thông tin về chuyến bay, khách hàng, và vé qua endpoint /bookings (POST).

• Lấy thông tin đặt vé:

• ID đặt vé (id) qua endpoint /bookings/{id} (GET).

• Xóa đặt vé:

• ID đặt vé (id) qua endpoint /bookings/{id} (DELETE).

• Tìm đặt vé theo thông tin khách hàng:

• Email (email), số điện thoại (phone), hoặc số thẻ (cardNumber) qua endpoint /bookings/findByCustomerInformation (GET).

• Tìm đặt vé theo ID khách hàng:

• ID khách hàng (customerId) qua endpoint /bookings/findByCustomerId/{customerId} (GET).

• Tìm đặt vé theo khoảng thời gian:

• Ngày bắt đầu (fromDate) và ngày kết thúc (toDate) qua endpoint /bookings/fromDate/{fromDate}/toDate/{toDate} (GET).

• Lấy danh sách vé của một đặt vé:

• ID đặt vé (id) qua endpoint /bookings/{id}/tickets (GET).

• Quản lý máy bay (Plane Management):

• Tạo máy bay mới:

• Đối tượng Plane chứa thông tin như tên máy bay, số ghế, trạng thái qua endpoint /plane (POST).

• Cập nhật máy bay:

• Đối tượng Plane chứa thông tin cập nhật qua endpoint /plane (PUT).

• Xóa máy bay:

• ID máy bay (id) qua endpoint /plane/{id} (DELETE).

• Cập nhật trạng thái máy bay:

• ID máy bay (id) và trạng thái (status) qua endpoint /plane/{id}/status (PATCH).

• Lấy thông tin máy bay:

• ID máy bay (id) qua endpoint /plane/{id} (GET).

• Lấy tất cả máy bay:

• Không cần tham số, qua endpoint /plane/all\_plane (GET).

• Quản lý địa điểm (Location Management):

• Lấy tất cả địa điểm:

• Không cần tham số, qua endpoint /location/all\_location (GET).

• Quản lý vai trò (Role Management):

• Lấy tất cả vai trò:

• Không cần tham số, qua endpoint /role (GET).

• Lấy vai trò theo ID:

• ID vai trò (id) qua endpoint /role/{id} (GET).

• Tạo vai trò mới:

• Đối tượng Role qua endpoint /role (POST).

• Cập nhật vai trò:

• ID vai trò (id) và đối tượng Role qua endpoint /role/{id} (PUT).

• Xóa vai trò:

• ID vai trò (id) qua endpoint /role/{id} (DELETE).

• Giao diện người dùng (Frontend):

• Tìm kiếm chuyến bay:

• Người dùng nhập thông tin điểm đi, điểm đến, và ngày khởi hành trên giao diện timChuyenBay.html để gửi yêu cầu đến backend.

#### 3.1.2. Đầu Ra (Output) của Hệ Thống

• Hệ thống "Flight App" trả về các đầu ra thông qua các phản hồi HTTP từ API hoặc hiển thị trên giao diện người dùng. Các đầu ra chính bao gồm:

• Xác thực người dùng (Authentication):

• Đăng nhập (Login):

• Phản hồi HTTP 200 với JWT token được lưu trong cookie (token) và danh sách quyền (permissions) của người dùng.

• Nếu thất bại: Phản hồi HTTP 401 với thông báo lỗi (ví dụ: thông báo từ LoginFailedException).

• Đăng xuất (Logout):

• Phản hồi HTTP 200 với thông báo "Đăng xuất thành công" và xóa cookie chứa JWT token.

• Quản lý tài khoản (Account Management):

• Tìm kiếm tài khoản:

• Phản hồi HTTP 200 với danh sách AccountDTO nếu tìm thấy tài khoản, hoặc HTTP 404 với thông báo "No accounts found" nếu không tìm thấy.

• Lấy thông tin tài khoản theo ID:

• Phản hồi HTTP 200 với đối tượng Account nếu tìm thấy, hoặc Optional.empty() nếu không tìm thấy.

• Tạo tài khoản mới:

• Phản hồi HTTP 200 với đối tượng Account được tạo, hoặc HTTP 400 với thông báo lỗi và mã lỗi (AccountException hoặc UserException).

• Cập nhật tài khoản:

• Phản hồi HTTP 200 với đối tượng Account đã cập nhật, hoặc HTTP 400 với thông báo lỗi và mã lỗi (AccountException).

• Cập nhật hồ sơ người dùng:

• Phản hồi HTTP 200 với thông tin người dùng đã cập nhật, hoặc HTTP 404 với thông báo "Account not found".

• Xóa tài khoản:

• Phản hồi HTTP 200 với thông báo "Deleted successfully" nếu xóa thành công, hoặc HTTP 404 với thông báo "Account not found".

• Quản lý chuyến bay (Flight Management):

• Tạo hoặc cập nhật chuyến bay:

• Phản hồi HTTP 200 (không có nội dung cụ thể, chỉ xác nhận hành động thành công).

• Xóa chuyến bay:

• Phản hồi HTTP 200 (không có nội dung cụ thể, chỉ xác nhận hành động thành công).

• Lấy thông tin chuyến bay:

• Phản hồi HTTP 200 với đối tượng Flight (cho endpoint /flight/{idFlight}) hoặc một Map chứa thông tin chi tiết (FlightDTO, danh sách vé đã đặt, số ghế economy/business) (cho endpoint /flight/{idFlight}/information).

• Nếu không tìm thấy: Lỗi NotFoundException với HTTP 404.

• Tìm kiếm chuyến bay:

• Phản hồi HTTP 200 với danh sách FlightDTO, mỗi đối tượng bao gồm thông tin chuyến bay và số vé còn lại (economy/business).

• Lấy tất cả chuyến bay:

• Phản hồi HTTP 200 với danh sách tất cả Flight.

• Lấy ghế trên chuyến bay:

• Phản hồi HTTP 200 với danh sách FlightSeatDTO chứa thông tin ghế của chuyến bay.

• Quản lý đặt vé (Booking Management):

• Tạo đặt vé:

• Phản hồi HTTP 200 với đối tượng Booking được tạo.

• Lấy thông tin đặt vé:

• Phản hồi HTTP 200 với đối tượng Booking, hoặc ném RuntimeException nếu không tìm thấy.

• Xóa đặt vé:

• Phản hồi HTTP 200 (không có nội dung cụ thể, chỉ xác nhận hành động thành công).

• Tìm đặt vé theo ID khách hàng hoặc thông tin khách hàng:

• Phản hồi HTTP 200 với danh sách BookingDTO, hoặc ném RuntimeException nếu không tìm thấy khách hàng.

• Tìm đặt vé theo khoảng thời gian:

• Phản hồi HTTP 200 với danh sách BookingDTO.

• Lấy danh sách vé của một đặt vé:

• Phản hồi HTTP 200 với danh sách TicketDTO.

• Quản lý máy bay (Plane Management):

• Lấy tất cả máy bay:

• Phản hồi HTTP 200 với danh sách Plane.

• Lấy máy bay theo ID:

• Phản hồi HTTP 200 với đối tượng Plane.

• Tạo, cập nhật, hoặc xóa máy bay:

• Phản hồi HTTP 200 (không có nội dung cụ thể, chỉ xác nhận hành động thành công).

• Cập nhật trạng thái máy bay:

• Phản hồi HTTP 200 (không có nội dung cụ thể, chỉ xác nhận hành động thành công).

• Quản lý địa điểm (Location Management):

• Lấy tất cả địa điểm:

• Phản hồi HTTP 200 với danh sách Location.

• Quản lý vai trò (Role Management):

• Lấy tất cả vai trò:

• Phản hồi HTTP 200 với danh sách Role.

• Lấy vai trò theo ID:

• Phản hồi HTTP 200 với đối tượng Role hoặc Optional.empty() nếu không tìm thấy.

• Tạo hoặc cập nhật vai trò:

• Phản hồi HTTP 200 với đối tượng Role được tạo hoặc cập nhật.

• Xóa vai trò:

• Phản hồi HTTP 200 với thông báo "Deleted successfully" nếu xóa thành công, hoặc "Role not found" nếu không tìm thấy.

• Giao diện người dùng (Frontend):

• Hiển thị danh sách chuyến bay, thông tin đặt vé, hoặc thông tin tài khoản trên giao diện timChuyenBay.html dựa trên dữ liệu từ backend.

• Hiển thị thông báo lỗi hoặc thành công tương ứng với phản hồi từ API.

### 3.2. Các Biểu Đồ Thiết Kế

#### 3.2.1. Biểu đồ Use Case (Use Case Diagram)

• Mô tả: Biểu đồ Use Case mô tả các chức năng chính của hệ thống và sự tương tác giữa các tác nhân (Actors) như Người dùng (User) và Quản trị viên (Admin) với hệ thống.

• Các tác nhân:

• Người dùng (User): Khách hàng tìm kiếm và đặt vé máy bay.

• Quản trị viên (Admin): Quản lý tài khoản, chuyến bay, máy bay, vai trò, và địa điểm.

• Các trường hợp sử dụng (Use Cases):

• Người dùng:

• Tìm kiếm chuyến bay (/flight/find\_flight)

• Đặt vé máy bay (/bookings, POST)

• Xem thông tin đặt vé (/bookings/findByCustomerId/{customerId})

• Xem vé theo đặt vé (/bookings/{id}/tickets)

• Tìm vé theo thông tin khách hàng (/bookings/findByCustomerInformation)

• Đăng nhập (/login)

• Đăng xuất (/logoutApp)

• Quản trị viên:

• Quản lý tài khoản (/account): Tạo, cập nhật, xóa, tìm kiếm tài khoản

• Quản lý chuyến bay (/flight): Thêm, cập nhật, xóa, xem chi tiết chuyến bay

• Quản lý máy bay (/plane): Thêm, cập nhật, xóa, cập nhật trạng thái máy bay

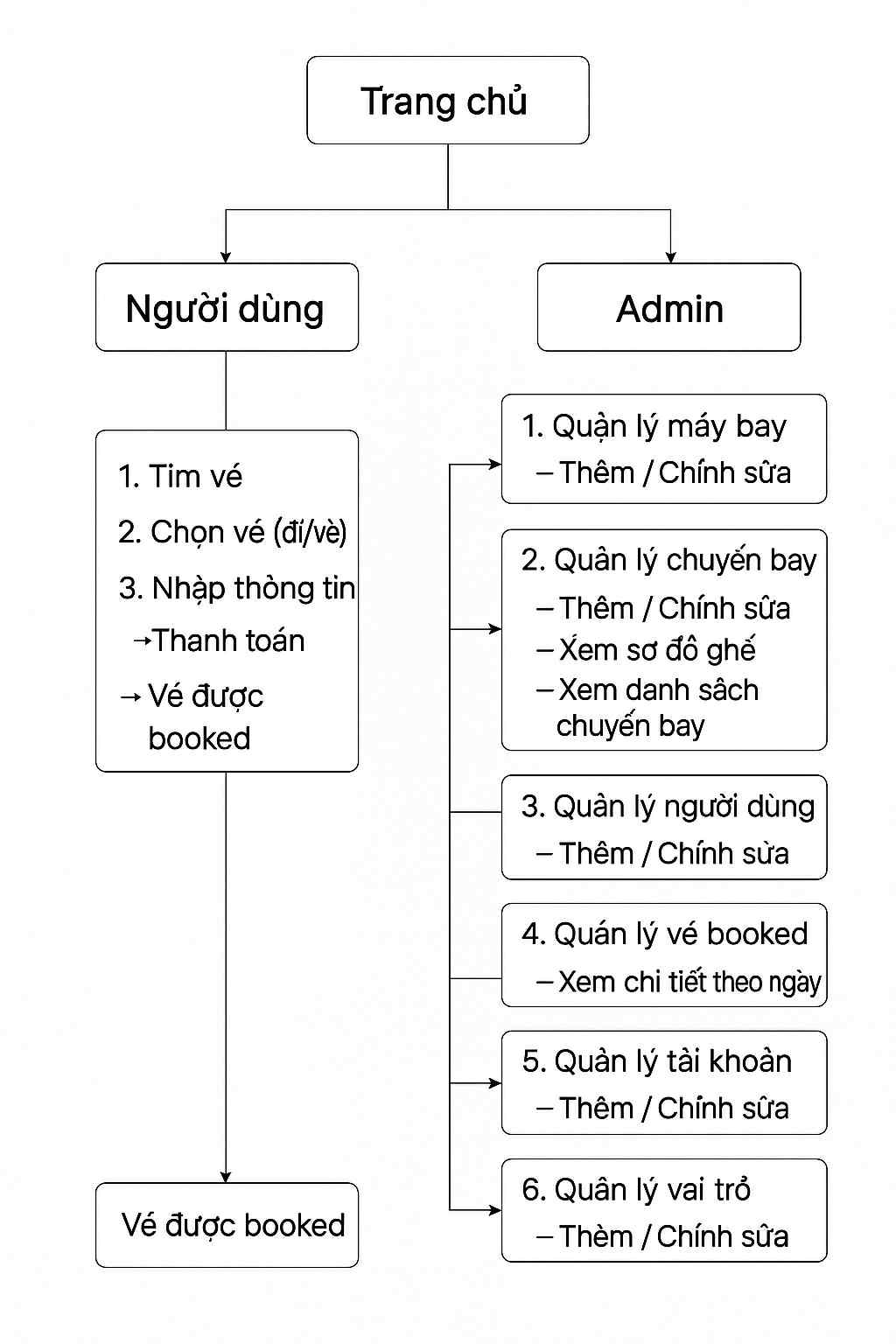
• Quản lý vai trò (/role): Tạo, cập nhật, xóa, xem vai trò

• Quản lý địa điểm (/location): Xem tất cả địa điểm

• Mối quan hệ:

• Include: Đăng nhập thường là điều kiện tiên quyết để đặt vé hoặc quản lý.

• Extend: Tìm kiếm chuyến bay có thể mở rộng để xem chi tiết ghế (/flight/flight\_seats).



**Hình 3: Sơ đồ chức năng hệ thống.**

#### 3.2.2. Biểu đồ Lớp (Class Diagram)

• Mô tả: Hiển thị các lớp chính trong hệ thống, thuộc tính, phương thức và mối quan hệ giữa chúng.

• Các lớp chính (dựa trên mã Java):

\*Account:

• Thuộc tính: id, username, password, roleId (liên kết với Role)

• Phương thức: CRUD qua AccountService (tạo, lấy, cập nhật, xóa)

\*User:

• Thuộc tính: idUser, email, phone, cardNumber

• Phương thức: Tìm theo thông tin khách hàng (UserService.findByCustomerInformation)

\*Flight:

• Thuộc tính: idFlight, fromLocation, toLocation, departureDate

• Phương thức: Thêm, cập nhật, xóa, tìm kiếm (FlightService)

\*Plane:

• Thuộc tính: idPlane, status (Enum Status)

• Phương thức: CRUD, cập nhật trạng thái (PlaneService)

\*Booking:

• Thuộc tính: id, customerId (liên kết với User)

• Phương thức: Tạo, lấy, xóa, tìm theo customerId hoặc ngày (BookingService)

\*Ticket:

• Thuộc tính: id, bookingId, ticketType (Enum TicketType)

• Phương thức: Lấy vé theo bookingId (BookingService.getTicketsByBookingId)

\*Location:

• Thuộc tính: id, name

• Phương thức: Lấy tất cả địa điểm (LocationRepo.findAll)

\*Role:

• Thuộc tính: id, name

• Phương thức: CRUD qua RoleService

\*Flights\_Seat:

• Thuộc tính: flightId, seatType (Enum SeatType)

• Phương thức: Lấy ghế theo chuyến bay (Flights\_SeatService.getFlightSeatsByIdFlight)

\*Mối quan hệ:

• Kết hợp (Association):

• Account 1:n Booking (một tài khoản có thể liên kết với nhiều đặt vé qua User).

• Flight 1:n Ticket (một chuyến bay có nhiều vé).

• Booking 1:n Ticket (một đặt vé có nhiều vé).

• Flight 1:n Flights\_Seat (một chuyến bay có nhiều ghế).

• Liên kết (Dependency):

• Account phụ thuộc vào Role (roleId).

• Kế thừa (Inheritance): Không rõ ràng trong mã, nhưng DTOs (AccountDTO, FlightDTO, etc.) có thể coi là biến thể của các model.

#### 3.2.3. Biểu đồ Trình Tự (Sequence Diagram)

• Mô tả: Mô tả dòng tương tác giữa các đối tượng (controllers, services, repositories) khi thực hiện một chức năng cụ thể.

• Ví dụ 1: Đăng nhập (/login):

• Người dùng gửi POST request đến AuthController với username và password.

• AuthController gọi JwtService.generateToken để tạo JWT.

• JwtService kiểm tra thông tin đăng nhập, nếu thất bại ném LoginFailedException.

• Nếu thành công, tạo cookie chứa token và trả về permissions.

• Trình duyệt nhận cookie và response.

• Ví dụ 2: Đặt vé máy bay (/bookings, POST):

• Người dùng gửi POST request với BookingRequest đến BookingController.

• BookingController gọi BookingService.createBooking.

• BookingService lưu thông tin đặt vé vào cơ sở dữ liệu.

• Trả về đối tượng Booking cho BookingController.

• BookingController trả response thành công cho người dùng.

• Ví dụ 3: Tìm kiếm chuyến bay (/flight/find\_flight):

• Người dùng gửi GET request với fromLocationId, toLocationId, departureDate đến FlightController.

• FlightController gọi LocationRepo.findById để lấy thông tin địa điểm.

• FlightController gọi FlightService.getFlightByFromAndToAndDepartureDate để tìm chuyến bay.

• FlightService trả về danh sách Flight.

• FlightController chuyển đổi sang FlightDTO, tính số vé còn trống (economy, business).

• Trả danh sách FlightDTO cho người dùng.

#### 3.2.4. Biểu đồ Hoạt Động (Activity Diagram)

• Mô tả: Diễn tả luồng hoạt động cho một chức năng cụ thể, bao gồm các bước và quyết định.

• Ví dụ: Đặt vé máy bay

• Bắt đầu: Người dùng truy cập /bookings.

• Tìm chuyến bay: Gửi yêu cầu tìm chuyến bay (/flight/find\_flight) với điểm đi, điểm đến, ngày khởi hành.

• Kiểm tra chuyến bay:

• Nếu không có chuyến bay: Thông báo "Không tìm thấy chuyến bay" → Kết thúc.

• Nếu có chuyến bay: Hiển thị danh sách chuyến bay.

• Chọn chuyến bay: Người dùng chọn chuyến bay và loại vé (economy/business).

• Gửi yêu cầu đặt vé: Gửi POST request với BookingRequest đến /bookings.

• Xác nhận đặt vé:

• BookingService kiểm tra ghế trống và thông tin.

• Nếu hợp lệ: Lưu đặt vé, trả về Booking.

• Nếu không hợp lệ: Trả lỗi.

• Thanh toán: Người dùng cung cấp thông tin thanh toán (cardNumber).

• Xác nhận: Hệ thống lưu booking và gửi xác nhận.

• Kết thúc: Người dùng nhận vé.

### 3.3.Biểu đồ cơ sở dữ liệu quan hệ hoặc cơ sở dữ liệu phẳng

**\* Bảng users**

• Lưu thông tin người dùng.Bao gồm các trường:

• id\_user: mã người dùng (khóa chính).

• full\_name: họ tên người dùng.

• address: địa chỉ.

• phone: số điện thoại.

• email: địa chỉ email.

• card\_number: số CMND/CCCD.

• day\_of\_birth: ngày sinh.

• sex: giới tính.

• user\_type: loại người dùng (ví dụ: khách hàng, nhân viên, quản trị).

**\*Bảng account**

• Quản lý tài khoản đăng nhập của người dùng. Bao gồm:

• id\_account: mã tài khoản (khóa chính).

• username: tên đăng nhập.

• password: mật khẩu.

• created\_at: ngày tạo tài khoản.

• role: vai trò trong hệ thống.

• user\_id: mã người dùng (khóa ngoại liên kết với users).

**\*Bảng role**

• Xác định các vai trò trong hệ thống. Có 2 cột:

• id\_role: mã vai trò (khóa chính).

• role: tên vai trò (ví dụ: admin, customer, staff...).

\***Bảng role\_permissions**

• Quản lý quyền của từng vai trò. Gồm:

• role\_id: khóa ngoại liên kết với role.

• permission: tên quyền (ví dụ: thêm người dùng, xem vé...).

**\*Bảng bookings**

• Lưu thông tin các phiếu đặt vé. Các trường gồm:

• booking\_id: mã đặt vé (khóa chính).

• amount: tổng số tiền.

• payment\_date: ngày thanh toán.

• payment\_method: phương thức thanh toán.

• id\_user: người đặt vé (khóa ngoại liên kết với users).

**\* Bảng ticket**

• Lưu thông tin các vé máy bay. Gồm:

• id\_ticket: mã vé (khóa chính).

• created\_at: ngày tạo vé.

• price: giá vé.

• ticket\_type: loại vé (một chiều, khứ hồi).

• id\_booking: mã đặt vé (khóa ngoại).

• id\_baggage: mã hành lý đi kèm (khóa ngoại).

• flight\_seat\_id: mã ghế ngồi của chuyến bay.

• id\_user: người mua vé (khóa ngoại).

**\*Bảng return\_ticket**

• Lưu thông tin vé khứ hồi. Gồm:

• id\_return\_ticket: mã vé về (khóa chính).

• price: giá vé về.

• ticket\_type: loại vé.

• id\_baggage: hành lý đi kèm.

• flight\_seat\_id: ghế ngồi.

• id\_flight: mã chuyến bay về.

• id\_ticket: mã vé đi (khóa ngoại liên kết với ticket).

**\*Bảng baggage**

• Lưu thông tin hành lý. Gồm:

• id\_baggage: mã hành lý (khóa chính).

• baggage\_type: loại hành lý (xách tay, ký gửi...).

**\*Bảng flights**

• Lưu thông tin các chuyến bay. Các trường:

• id\_flight: mã chuyến bay (khóa chính).

• common\_fare: giá ghế thường.

• vip\_fare: giá ghế VIP.

• created\_at: ngày tạo chuyến bay.

• departure\_date: ngày khởi hành.

• departure\_time: giờ khởi hành.

• duration\_minutes: thời gian bay.

• from, to: địa điểm đi và đến.

• id\_plane: máy bay được sử dụng (khóa ngoại).

**\*Bảng plane**

• Thông tin về máy bay. Gồm:

• id\_plane: mã máy bay (khóa chính).

• name\_plane: tên máy bay.

• flight\_hours: tổng số giờ bay.

**\*Bảng seat**

• Thông tin các ghế ngồi trên máy bay. Gồm:

• id\_seat: mã ghế (khóa chính).

• seat\_number: số ghế.

• seat\_type: loại ghế (VIP, thường...).

**\*Bảng flights\_seat**

• Bảng liên kết giữa chuyến bay và ghế ngồi (nhiều - nhiều). Gồm:

• id\_flight: mã chuyến bay (khóa ngoại).

• id\_seat: mã ghế (khóa ngoại).

**\*Bảng location**

• Lưu tên các địa điểm, dùng để xác định sân bay đi/đến. Gồm:

• id: mã địa điểm (khóa chính).

• name: tên địa điểm (Hà Nội, Đà Nẵng, Tokyo...).

A computer screen shot of a computer

Description automatically generated

**Hình 4.Database**

• Tổng kết các mối quan hệ giữa các bảng

• Một người dùng có thể có nhiều đơn đặt vé và nhiều vé.

• Một người dùng có một tài khoản.

• Mỗi tài khoản gắn với một vai trò (role) và có một danh sách quyền (role\_permissions).

• Một đặt vé (booking) có thể chứa nhiều vé (ticket).

• Mỗi vé có thể có hành lý, ghế ngồi, và liên kết tới một chuyến bay.

• Nếu vé là khứ hồi, thì có thêm thông tin ở bảng return\_ticket.

• Một chuyến bay có thể có nhiều ghế (flights\_seat) và gắn với một máy bay (plane).

• Các sân bay (địa điểm đi/đến) được lưu trong bảng location.

### 3.4.Kiến trúc hệ thống

#### 3.4.1. Database Layer (Tầng cơ sở dữ liệu)

• Công nghệ: Microsoft SQL Server (mssql-jdbc:12.4.2).

• Connection Pooling: HikariCP (HikariCP:5.1.0) quản lý kết nối hiệu quả.

• ORM: Spring Data JPA ánh xạ các thực thể Java (User, Account, Flight, Booking, Ticket, Location, Plane, Role, Flights\_Seat…) với bảng cơ sở dữ liệu.

• users: Lưu trữ thông tin người dùng, bao gồm các trường như: id\_user, fullname, card\_number, date\_of\_birth, address, phone, user\_type.

• account: Đại diện cho tài khoản đăng nhập của người dùng. Có các trường: id\_account, username, password, email, liên kết với bảng users qua user\_id.

• role: Bảng lưu các vai trò như admin, nhân viên, khách hàng,... Mỗi role có id\_role và name.

• role\_permissions: Liên kết các vai trò với các quyền cụ thể (permission). Mỗi dòng chứa role\_id và một permission.

• bookings: Đại diện cho các đơn đặt vé, với thông tin: booking\_id, amount, payment\_method, account\_id.

• ticket: Thông tin về vé đã được đặt, gồm: id\_ticket, created\_at, price, ticket\_type, id\_booking, flight\_id, flight\_seat\_id, id\_user, id\_baggage.

• return\_ticket: Thông tin vé khứ hồi, liên kết đến vé chính qua id\_ticket, chứa các trường: id\_return\_ticket, price, ticket\_type, id\_baggage, flight\_seat\_id, flight\_return\_id.

• baggage: Hành lý đi kèm vé, gồm: id\_baggage, baggage\_type, weight.

• lights: Đại diện cho các chuyến bay. Gồm: id\_flight, departure\_date, departure\_time, duration\_minutes, commercial\_fare, id\_plane, from và to (liên kết đến bảng location).

• plane: Thông tin về máy bay sử dụng trong chuyến bay. Có: id\_plane, plane\_name, flight\_hours.

• seat: Ghế ngồi trên máy bay. Gồm: id\_seat, seat\_number, seat\_type.

• flights\_seat: Liên kết giữa chuyến bay và ghế ngồi. Có: id\_flight, id\_seat.

• location: Danh sách các sân bay (địa điểm đi/đến). Trường chính: id\_location, name.

• Mối quan hệ giữa các bảng:

• Một user có thể có nhiều booking.

• Mỗi account liên kết một user và một role.

• Mỗi role có thể có nhiều quyền qua bảng role\_permissions.

• Mỗi booking có thể chứa nhiều ticket.

• Mỗi ticket có thể có một vé khứ hồi đi kèm (return\_ticket).

• Mỗi flight có nhiều ghế thông qua bảng flights\_seat.

• Mỗi vé có thể có hành lý đi kèm thông qua baggage.

• Mỗi flight liên kết với một máy bay và hai địa điểm (đi và đến).

• Ghế (seat) được chia sẻ giữa các chuyến bay thông qua bảng trung gian flights\_seat.

#### 3.4.2. Backend Layer (Tầng xử lý phía Server)

* **Công nghệ:**
  + Spring Boot 3.4.3 (spring-boot-starter-parent).
  + Spring Web (spring-boot-starter-web) cho REST API.
  + Spring Security (spring-boot-starter-security) và JWT (jjwt:0.11.5) cho xác thực/phân quyền.
  + Spring Data JPA truy vấn cơ sở dữ liệu.
  + Maven quản lý build (pom.xml).
* **Cấu trúc**:
  + **Controllers**: Xử lý yêu cầu HTTP.
    - AuthController: Đăng nhập (/login), đăng xuất (/logoutApp).
    - FlightController: Quản lý chuyến bay (/flight).
    - BookingController: Quản lý đặt vé (/bookings).
    - AccountController, PlaneController, LocationController, RoleController.
  + **Services**: Logic nghiệp vụ (FlightService, BookingService, AccountService, etc.).
  + **Repositories**: Truy vấn dữ liệu (LocationRepo, etc.).
  + **DTOs**: Chuyển đổi dữ liệu (FlightDTO, BookingDTO, etc.).
  + **Enums**: SeatType (ECONOMY, BUSINESS), TicketType, Status.
* **Bảo mật:**
  + JWT token lưu trong cookie (HTTP-only, maxAge 1 giờ).
  + @PreAuthorize kiểm tra quyền (MANAGE\_USER, MANAGE\_PLANE).
  + GlobalExceptionHandler xử lý lỗi (NotFoundException, etc.).
* **API chính:**
  + **Tìm chuyến bay:** GET /flight/find\_flight.
  + **Đặt vé:** POST /bookings.
  + **Quản lý tài khoản:** POST/PUT/DELETE /account/{id}.
  + **Thông tin chuyến bay:** GET /flight/{idFlight}/information.

#### 3.4.3. Frontend Layer (Tầng giao diện người dùng)

• Công nghệ:

• Vite 6.3.4: Công cụ quản lý build, cung cấp khả năng phát triển nhanh và tối ưu hóa quá trình xây dựng ứng dụng.

• Tailwind CSS 4.1.4: Framework CSS để thiết kế giao diện responsive, hiện đại và dễ tùy chỉnh.

• HTML/JavaScript: Sử dụng để xây dựng cấu trúc và logic giao diện người dùng.

• Node.js: Quản lý các phụ thuộc và chạy môi trường phát triển.

• Cấu trúc:

• vite.config.js: Cấu hình entry point cho ứng dụng, chỉ định file user/timChuyenBay.html làm điểm vào chính cho quá trình build.

• Thư mục APIs: Chứa các file JavaScript (account.js, auth.js, booking.js, flight.js, location.js, plane.js, role.js, user.js) xử lý các yêu cầu API tới backend sử dụng Axios và Fetch API.

• Thư mục admin/contents: Chứa các file HTML (accountView.html, booked\_view.html, create\_booking\_view.html, flight.html, plane.html) định nghĩa giao diện quản lý tài khoản, đặt vé, chuyến bay và máy bay.

• Chức năng:

• Tìm chuyến bay: Gửi yêu cầu GET tới endpoint /flight/find\_flight với các tham số như fromLocationId, toLocationId, và departureDate để lấy danh sách chuyến bay phù hợp.

• Đặt vé: Gửi yêu cầu POST tới endpoint /bookings với dữ liệu bookingRequest để tạo booking mới, hỗ trợ chọn ghế và loại vé (phổ thông/thương gia).

• Xem vé: Gửi yêu cầu GET tới endpoint /bookings/{id}/tickets để lấy danh sách vé liên quan đến booking cụ thể.

• Đăng nhập/Đăng xuất:

• Đăng nhập: Gửi yêu cầu POST tới /login với email và password.

• Đăng xuất: Gửi yêu cầu POST tới /logoutApp để kết thúc phiên làm việc.

• Quản lý tài khoản: Hỗ trợ tạo, sửa, xóa tài khoản (POST /account, PUT /account/{id}, DELETE /account/{id}) và tìm kiếm tài khoản theo email/role (GET /account/find-account-by-email).

• Quản lý chuyến bay và máy bay:

• Chuyến bay: Xem, thêm, sửa, xóa (GET /flight/{id}, POST /flight, PUT /flight/{id}, DELETE /flight/{id}) và xem sơ đồ ghế (GET /flight/flight\_seats).

• Máy bay: Xem, thêm, sửa, xóa (GET /plane/{id}, POST /plane, PUT /plane, DELETE /plane/{id}).

• Giao diện:

• Giao diện tìm kiếm chuyến bay (create\_booking\_view.html): Cho phép chọn điểm đi/đến, ngày đi/về, loại vé, số lượng hành khách và tìm chuyến bay.

• Giao diện quản lý đơn đặt vé (booked\_view.html): Hiển thị danh sách booking, chi tiết khách hàng, thanh toán và danh sách vé.

• Giao diện quản lý chuyến bay (flight.html): Hiển thị thông tin chuyến bay, sơ đồ ghế và thống kê vé.

• Giao diện quản lý tài khoản (accountView.html): Cho phép tìm kiếm, thêm, sửa, xóa tài khoản và cập nhật thông tin người dùng.

• Giao diện quản lý máy bay (plane.html): Quản lý danh sách máy bay với thông tin tên, trạng thái, giờ bay và ghế.

#### 3.4.4. Communication Layer (Tầng giao tiếp)

• Giao thức: HTTP/REST API, dữ liệu JSON.

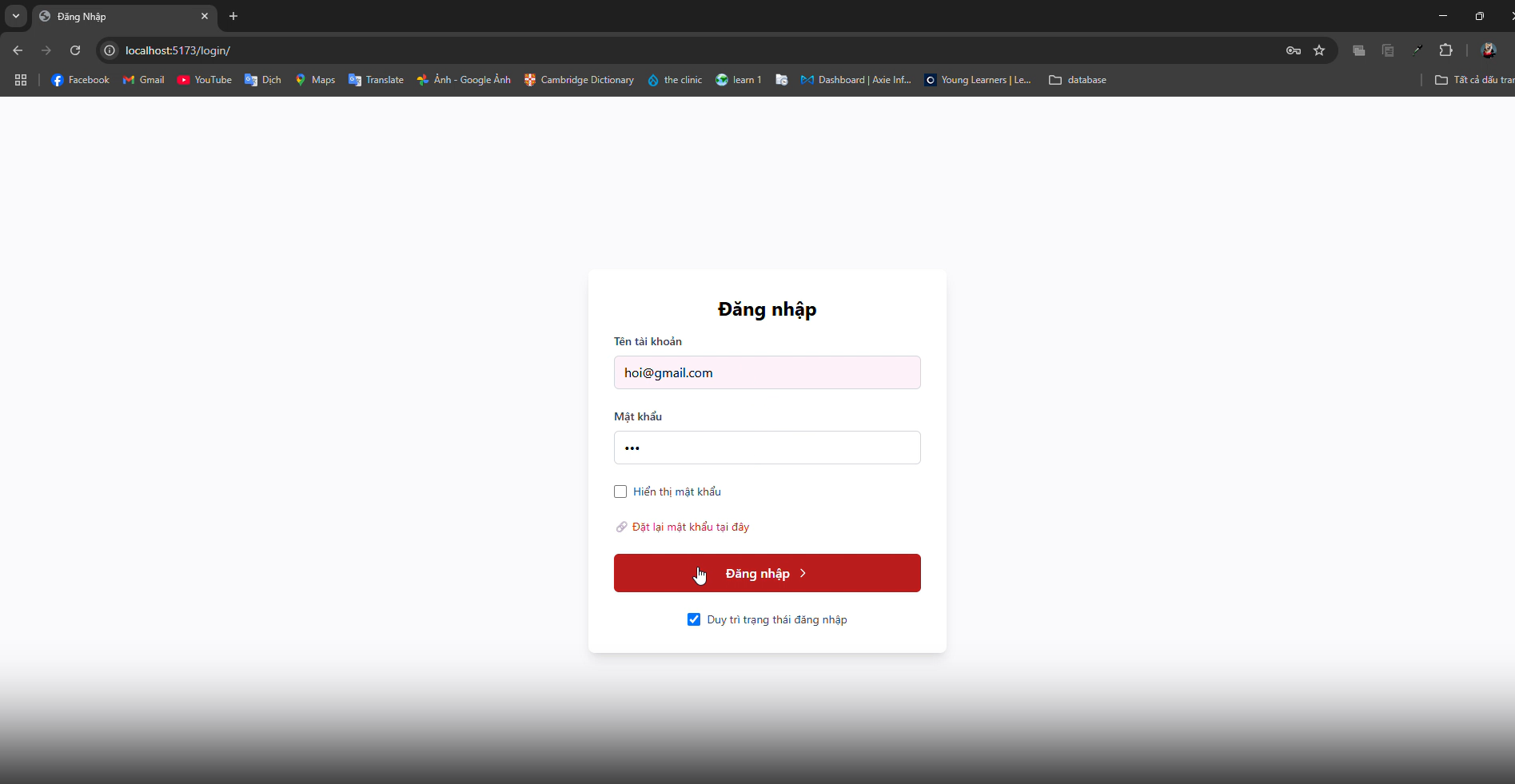
• CORS: Hỗ trợ cross-origin (@CrossOrigin(origins = "\*")).

• Xác thực: JWT token trong cookie hoặc header Authorization.

## 4.CHƯƠNG TRÌNH VÀ KẾT QUẢ

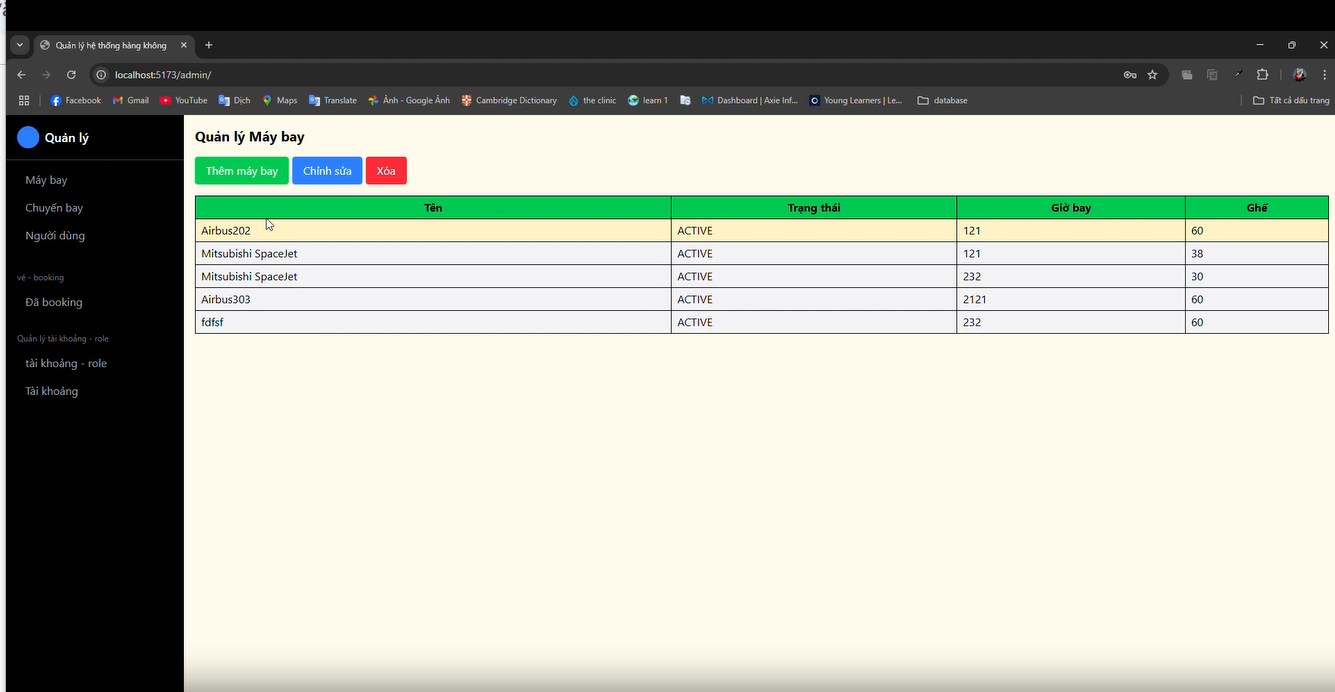
### 4.1.Giao diện chính

#### 4.1.1.Đăng nhập



**Hình 5.Đăng Nhập**

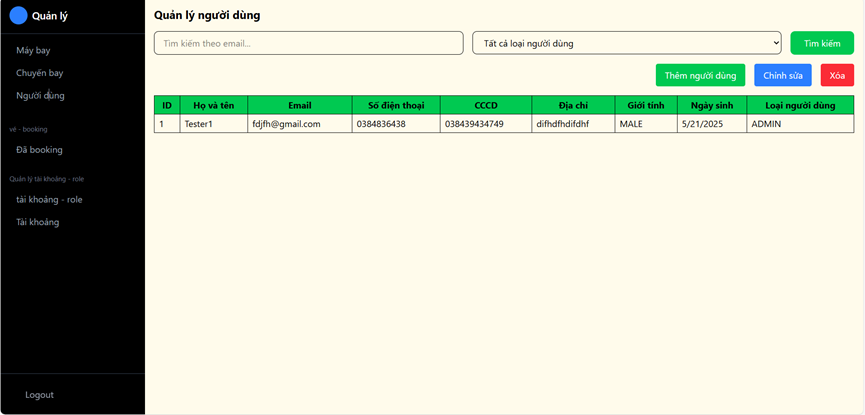
#### 4.1.2.Admin



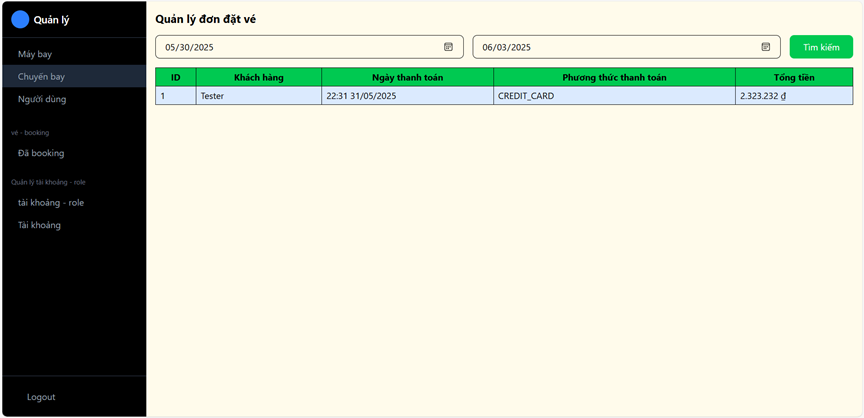
**Hình 6.Giao diện Admin và Máy bay**

****

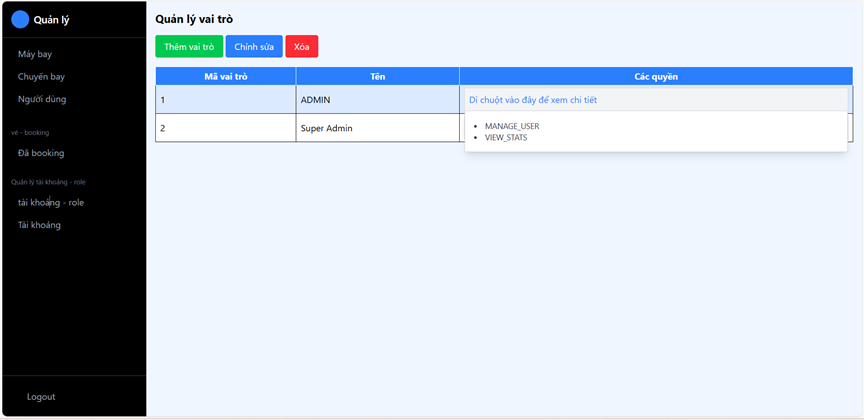
**Hình 7.Giao diện Chuyến Bay**

****

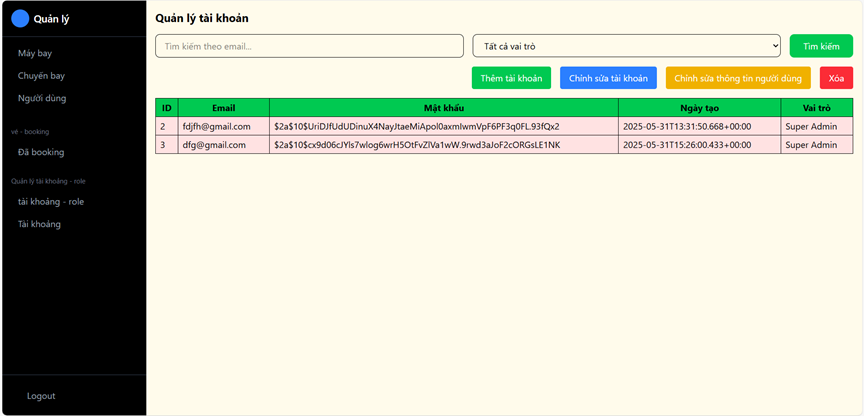
**Hình 8.Giao diện người dùng**

****

**Hình 9.Giao diện đã booking**

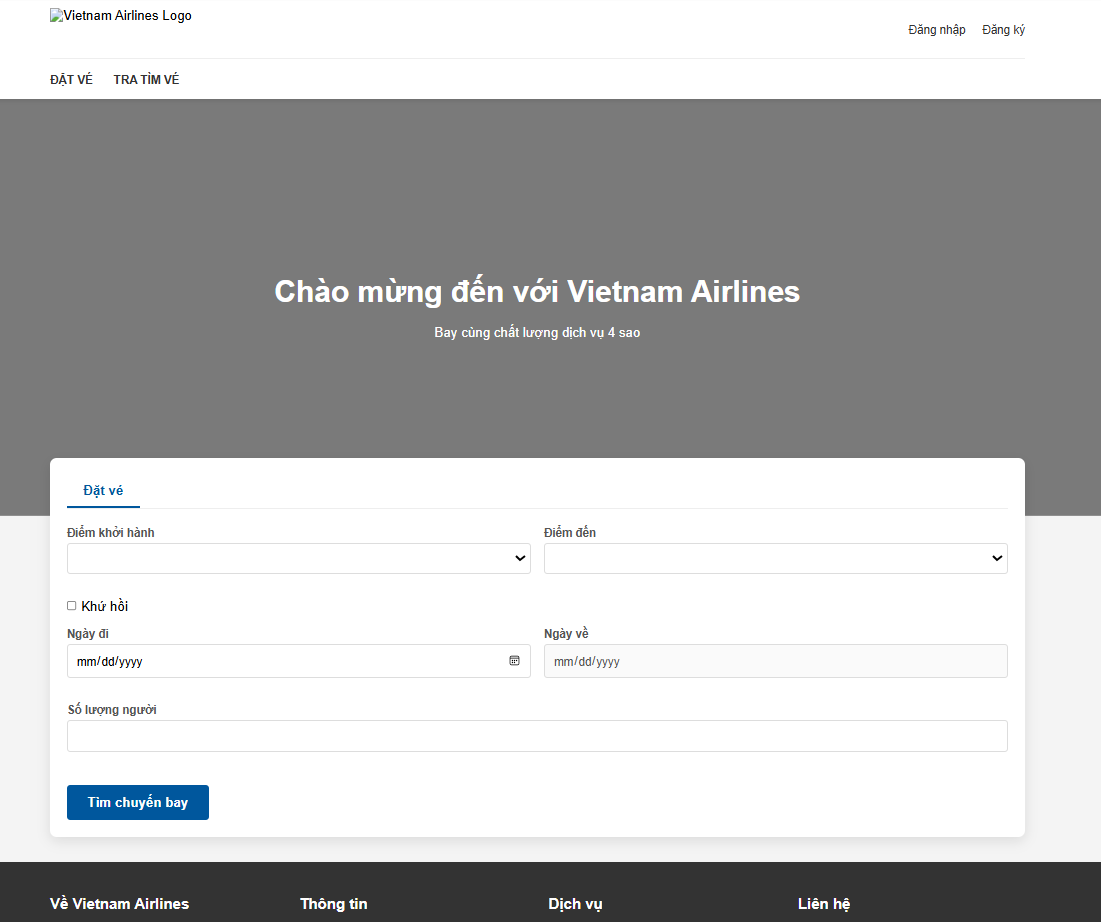
****

**Hình 10.Giao diện tài khoảng-role**

****

**Hình 11.Giao diện tài khoản**

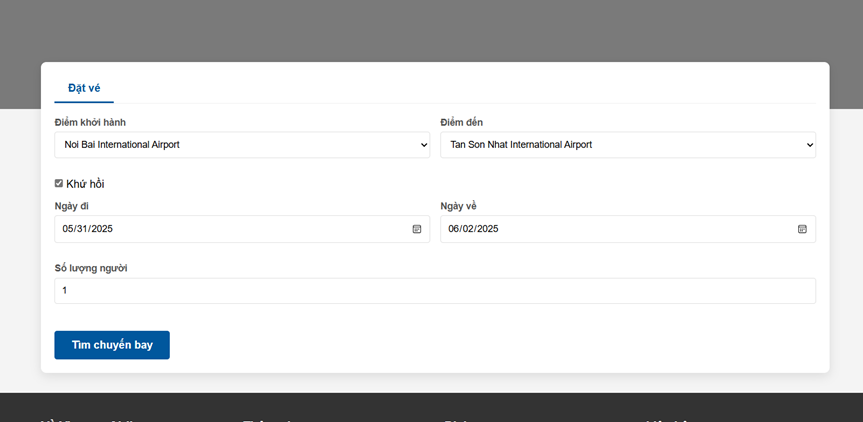
**4.1.3.User**



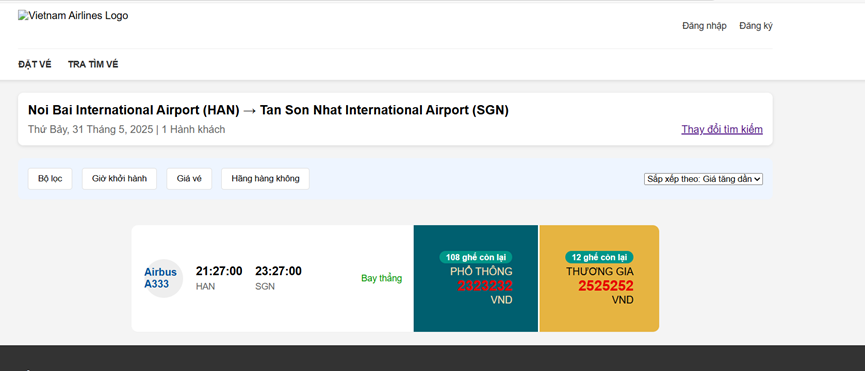
**Hình 12:Giao diện user**

### 4.2.Kết quả thực thi

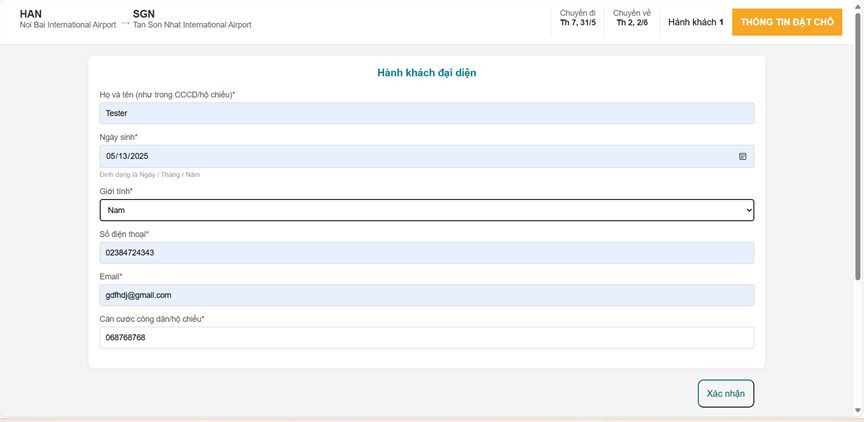
#### 4.2.1.User



**Hình 13.Tìm vé**

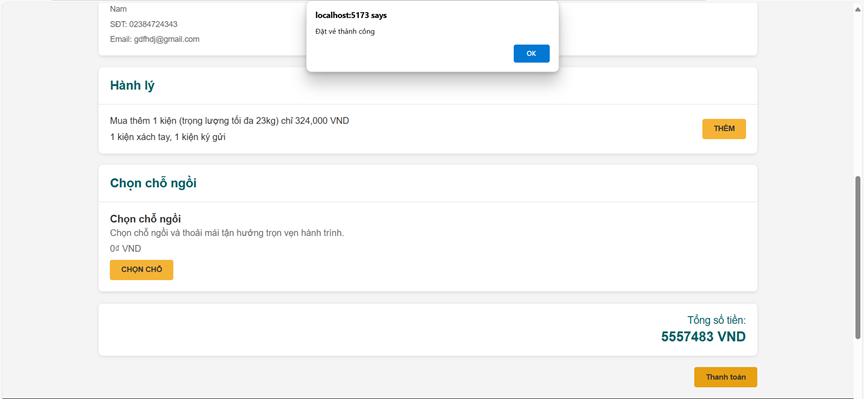
****

**Hình 14.Chọn vé**

****

**Hình 15.Nhập thông tin hành khách đi**

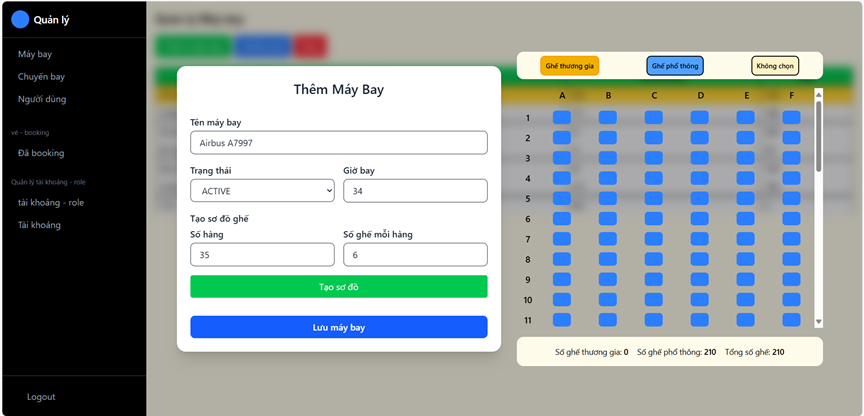
• Click thanh toán

****

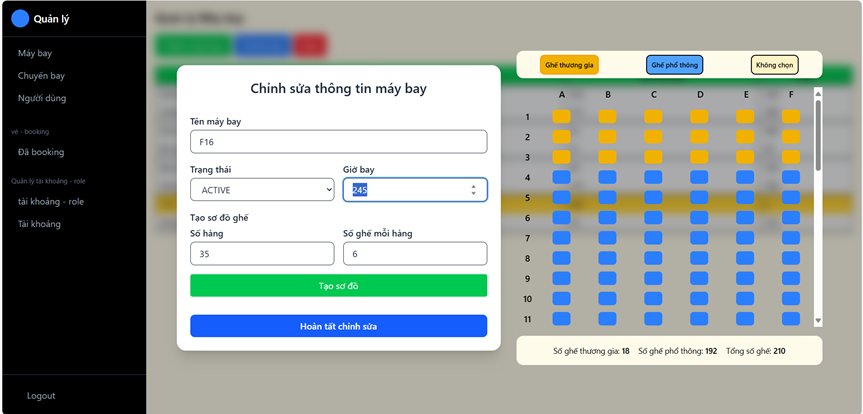
**Hình 16.Click thanh toán**

#### 4.2.2.ADMIN

• Máy Bay

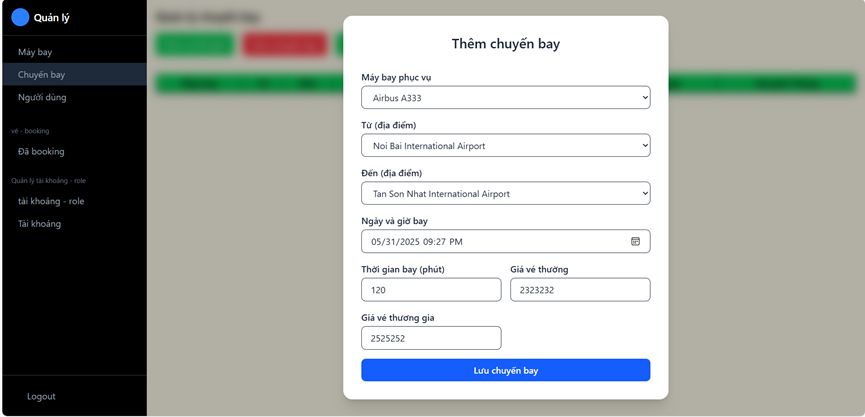
****

**Hình 17.Thêm Máy Bay**

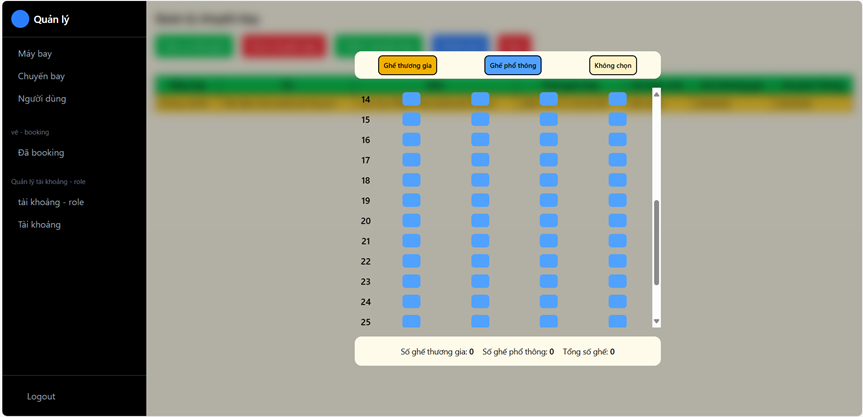
****

**Hình 18.Chỉnh Sửa**

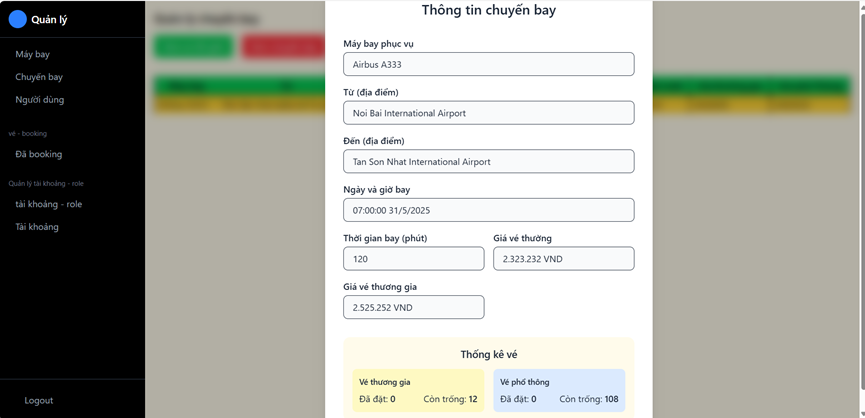
• Chuyến bay

****

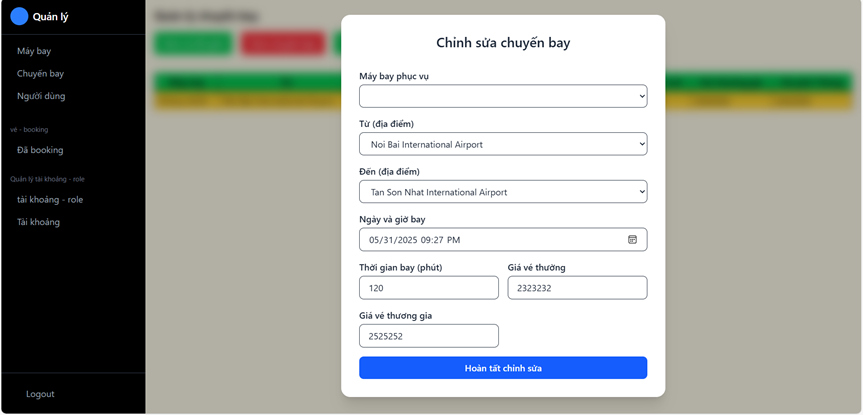
**Hình 19.Thêm chuyến bay**

****

**Hình 20.Xem sơ đồ ghế**

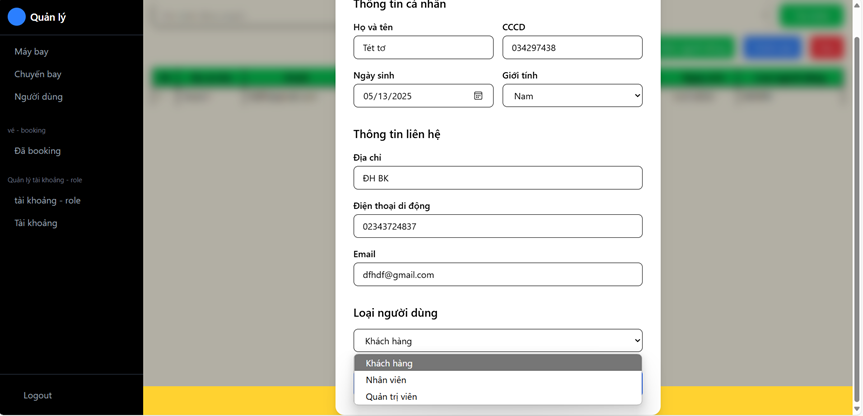
****

**Hình 21.Xem chuyến bay**

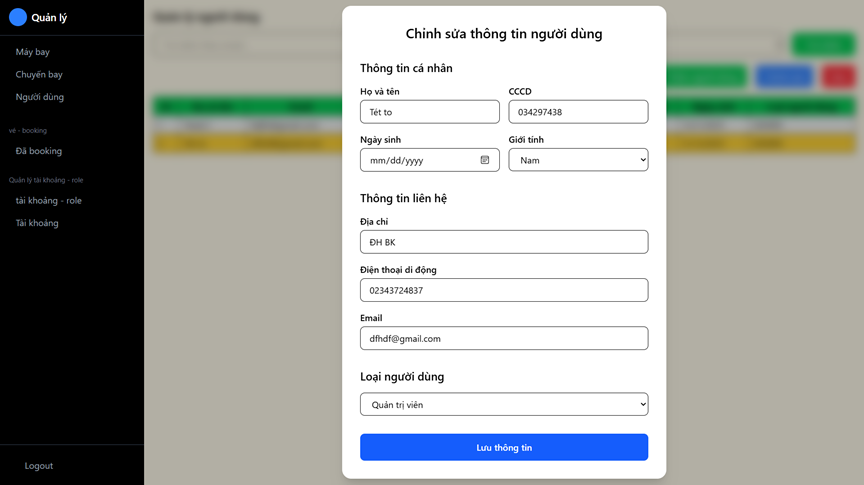
****

**Hình 22.Chỉnh sửa chuyến bay**

• Người dùng

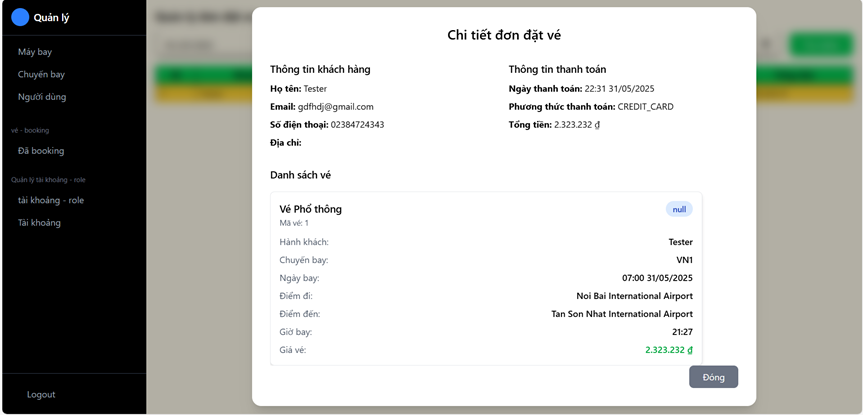
****

**Hình 23.Thêm người dùng**

****

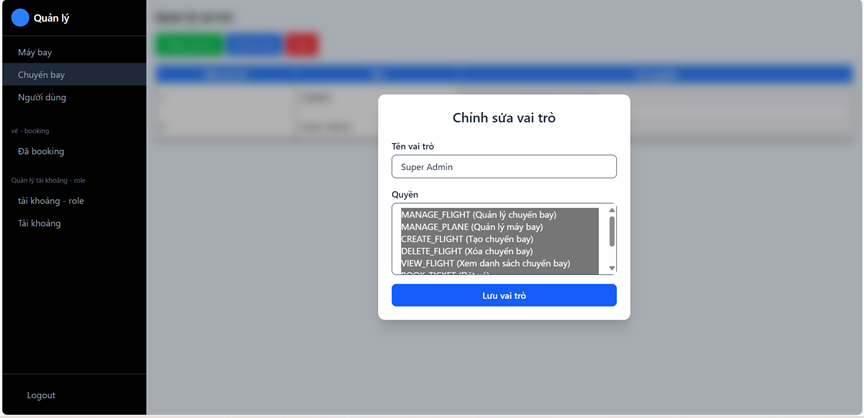
**Hình 24.Chỉnh sửa**

• Vé book

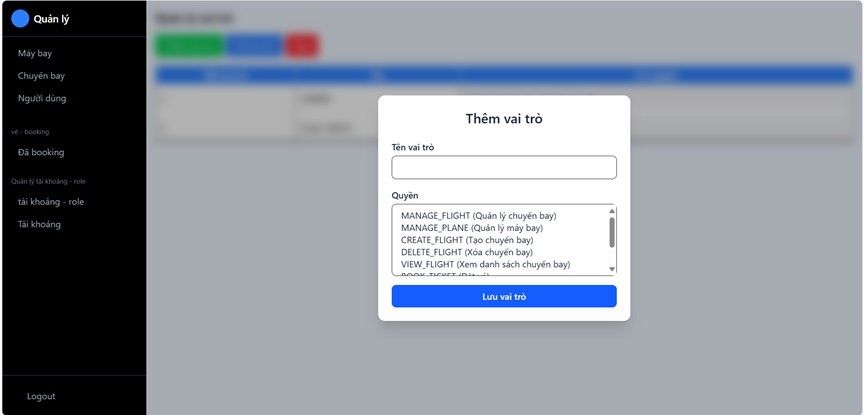
****

**Hình 25.Chi tiết vé booked**

• Vai trò tài khoản

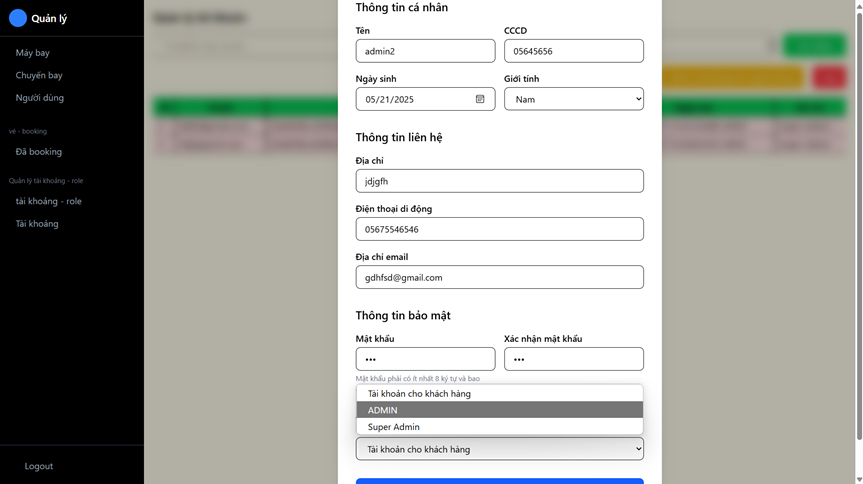
****

**Hình 26.Chỉnh sửa vai trò**

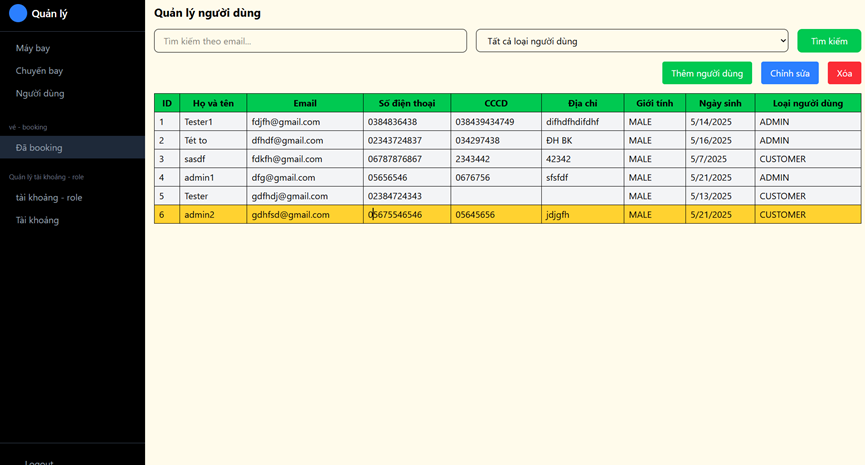
****

**Hình 27.Thêm vai trò**

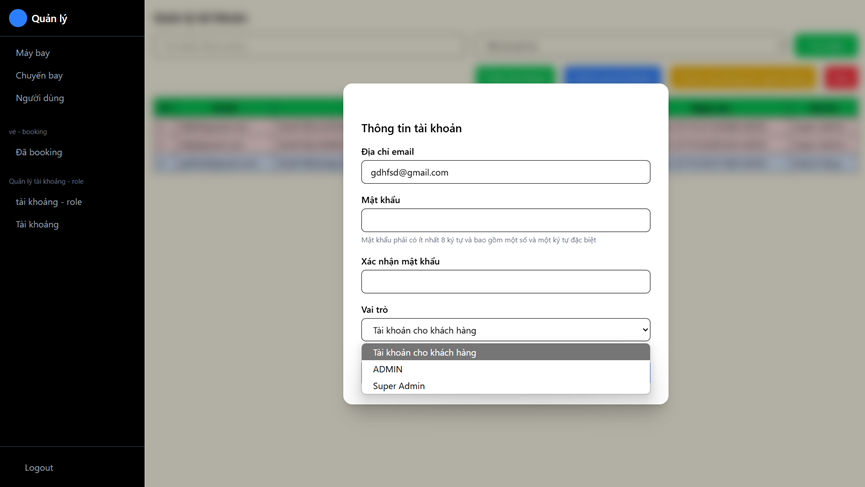
• Tài khoản

****

**Hình 28.Thêm tài khoản**

****

**Hình 29.Tài khoản sẽ tồn tại ở danh mục tài khoản người dùng**

****

**Hình 30.Chỉnh sửa tài khoản**

### 4.3.Nhận xét đánh giá

• Sau quá trình triển khai và thực nghiệm hệ thống, nhóm đã thu được những kết quả khả quan, đồng thời rút ra một số nhận xét, đánh giá tổng quan như sau:

• Ưu điểm

**Hệ thống hoạt động ổn định**, các chức năng như tìm kiếm chuyến bay, đặt vé, quản lý thông tin hoạt động chính xác theo thiết kế.

• Giao diện người dùng trực quan, thân thiện, hỗ trợ tốt trên cả desktop và thiết bị di động.

• Phân quyền rõ ràng giữa người dùng và quản trị viên, đảm bảo tính bảo mật và kiểm soát chức năng trong hệ thống.

• Tốc độ phản hồi nhanh, trải nghiệm người dùng mượt mà nhờ kiến trúc client-server hiệu quả và sử dụng công nghệ hiện đại như Spring Boot, Tailwind, Vite.

• Cấu trúc mã nguồn rõ ràng, dễ mở rộng và bảo trì. Việc phân chia lớp (Controller, Service, Repository, DTO) tuân thủ theo mô hình chuẩn trong phát triển phần mềm.

• Hạn chế

• Chưa tích hợp thanh toán trực tuyến, người dùng mới chỉ đặt vé chứ chưa thực hiện thanh toán qua cổng thanh toán thực tế.

• Thiếu kiểm soát đồng thời (concurrency) khi nhiều người dùng đặt vé cùng lúc, có thể dẫn đến trùng ghế nếu hệ thống không khóa đúng lúc.

• Chưa có tính năng tự động đề xuất chuyến bay, lọc chuyến tối ưu hay tích hợp bản đồ sân bay.

• Bảo mật mức cao (HTTPS, refresh token) chưa được triển khai đầy đủ trong phiên bản hiện tại.

## 5.KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

### 5.1. Kết luận

• Dự án “Xây dựng website bán vé máy bay trực tuyến” đã được nhóm thực hiện với mục tiêu cung cấp một nền tảng tiện lợi, thân thiện với người dùng, hỗ trợ người dùng trong việc tìm kiếm, đặt vé máy bay, và quản lý thông tin liên quan một cách hiệu quả. Qua quá trình phát triển, nhóm đã đạt được những kết quả đáng ghi nhận như sau:

* **Hoàn thành các chức năng cốt lõi**: Hệ thống đã triển khai đầy đủ các chức năng chính như tìm kiếm chuyến bay, đặt vé trực tuyến, quản lý thông tin chuyến bay, vé, hành lý, ghế ngồi, và phân quyền người dùng. Các tính năng này đáp ứng tốt nhu cầu của cả người dùng cuối (khách hàng) và quản trị viên (admin/đại lý).
* **Ứng dụng công nghệ hiện đại**: Sử dụng kiến trúc Client-Server với frontend được xây dựng bằng HTML5, CSS3, JavaScript, và Bootstrap, cùng backend dựa trên Java, Spring Boot, và SQL Server, đảm bảo hệ thống hoạt động ổn định, bảo mật, và có khả năng mở rộng.
* **Tối ưu trải nghiệm người dùng**: Giao diện responsive, dễ sử dụng trên nhiều thiết bị (máy tính, điện thoại, máy tính bảng) và tích hợp các tính năng như thông báo xác nhận qua email, quản lý lịch sử đặt vé, và cơ chế bảo mật với JWT token.
* **Rèn luyện kỹ năng chuyên môn**: Nhóm đã áp dụng thành công kiến thức về lập trình, cơ sở dữ liệu, thiết kế hệ thống, và quản lý dự án phần mềm. Đồng thời, quá trình làm việc nhóm đã giúp cải thiện kỹ năng giao tiếp, phân chia công việc, và giải quyết vấn đề thực tế.
* **Đáp ứng yêu cầu thực tiễn**: Hệ thống mang lại giá trị thực tế bằng cách đơn giản hóa quy trình đặt vé máy bay, tiết kiệm thời gian và công sức cho người dùng, đồng thời hỗ trợ quản trị viên quản lý dữ liệu hiệu quả.

• Tuy nhiên, dự án vẫn còn một số hạn chế, như hiệu suất truy vấn cơ sở dữ liệu có thể bị ảnh hưởng khi khối lượng dữ liệu lớn, thiếu cơ chế xử lý đồng thời (concurrency control) chặt chẽ, và một số tính năng nâng cao chưa được tích hợp (ví dụ: đề xuất chuyến bay thông minh hoặc tích hợp bản đồ sân bay).

### 5.2. Hướng phát triển

• Để nâng cao chất lượng và khả năng ứng dụng thực tiễn của hệ thống, nhóm đề xuất các hướng phát triển sau trong tương lai:

• Tích hợp trí tuệ nhân tạo và học máy:

* + Phát triển các thuật toán đề xuất chuyến bay tối ưu dựa trên lịch sử tìm kiếm, sở thích của người dùng, hoặc các yếu tố như giá vé, thời gian bay, và số lần quá cảnh.
  + Sử dụng AI để dự đoán nhu cầu đặt vé hoặc tối ưu hóa giá vé dựa trên dữ liệu thị trường.

• Tối ưu hóa hiệu suất hệ thống:

* + Thêm index cho các cột thường xuyên được truy vấn trong cơ sở dữ liệu (như fromLocation, toLocation, departureDate) để tăng tốc độ tìm kiếm.
  + Triển khai cơ chế caching (ví dụ: sử dụng Redis) để lưu trữ kết quả tìm kiếm phổ biến, giảm tải cho cơ sở dữ liệu.
  + Áp dụng cơ chế khóa lạc quan hoặc bi quan để xử lý các yêu cầu đặt vé đồng thời, tránh xung đột dữ liệu.

• Mở rộng tính năng cho người dùng:

* + Tích hợp bản đồ sân bay để hỗ trợ người dùng định vị cổng ra máy bay, khu vực nhận hành lý, hoặc các tiện ích tại sân bay.
  + Phát triển ứng dụng di động đồng bộ với website để tăng khả năng tiếp cận và tiện lợi cho người dùng.
  + Thêm tính năng so sánh giá vé giữa các hãng hàng không hoặc gợi ý các gói ưu đãi, khuyến mãi.

• Tăng cường bảo mật:

* + Chuyển sang sử dụng HTTPS và đặt thuộc tính secure: true cho cookie chứa JWT token để tăng cường bảo mật.
  + Triển khai cơ chế refresh token để duy trì phiên đăng nhập lâu dài mà không yêu cầu người dùng đăng nhập lại thường xuyên.
  + Thêm kiểm tra trạng thái tài khoản (ví dụ: tài khoản bị khóa) trước khi cấp token hoặc xử lý yêu cầu.

• Cải tiến giao diện và trải nghiệm người dùng:

* + Tích hợp các hiệu ứng động mượt mà hơn bằng cách sử dụng các thư viện JavaScript hiện đại (như React hoặc Vue.js) thay vì chỉ dùng HTML/CSS thuần.
  + Cung cấp tùy chọn đa ngôn ngữ (ví dụ: tiếng Anh, tiếng Việt) để phục vụ người dùng quốc tế.
  + Tối ưu hóa giao diện cho các thiết bị có màn hình nhỏ hoặc cấu hình thấp.

• Tích hợp thanh toán trực tuyến:

* + Kết nối với các cổng thanh toán phổ biến (như VNPay, Momo, hoặc PayPal) để hỗ trợ thanh toán trực tuyến nhanh chóng và an toàn.
  + Thêm tính năng hoàn tiền hoặc hủy vé trực tuyến với quy trình rõ ràng và minh bạch.

• Mở rộng chức năng quản trị:

* + Phát triển bảng điều khiển (dashboard) cho quản trị viên với các biểu đồ thống kê trực quan về doanh thu, số lượng vé bán ra, hoặc xu hướng đặt vé.
  + Thêm tính năng quản lý khuyến mãi, giảm giá, hoặc chương trình khách hàng thân thiết.

• Triển khai và bảo trì:

* + Triển khai hệ thống trên nền tảng đám mây (như AWS, Azure) để đảm bảo khả năng mở rộng và tính sẵn sàng cao.
  + Thiết lập quy trình giám sát và bảo trì định kỳ để phát hiện và khắc phục lỗi nhanh chóng.

• Nhóm hy vọng rằng những cải tiến và hướng phát triển này sẽ giúp hệ thống trở thành một giải pháp toàn diện, đáp ứng tốt hơn nhu cầu của người dùng và doanh nghiệp trong lĩnh vực đặt vé máy bay trực tuyến. Đồng thời, đây cũng là cơ hội để nhóm tiếp tục học hỏi và hoàn thiện kỹ năng trong việc phát triển các ứng dụng thực tế.

TÀI LIỆU THAM KHẢO

* Spring Documentation. (2025). *Spring Boot Reference Documentation*. Truy cập tại: <https://docs.spring.io/spring-boot/docs/current/reference/html/>  
  *(Hướng dẫn chính thức về Spring Boot, bao gồm cách xây dựng RESTful API, tích hợp Hibernate, và quản lý cơ sở dữ liệu.)*
* Oracle. (2025). *Java SE Documentation*. Truy cập tại: <https://docs.oracle.com/en/java/javase/17/>  
  *(Tài liệu tham khảo về Java SE, hỗ trợ trong việc lập trình backend và xử lý logic nghiệp vụ.)*
* Coursera. *Web Development with HTML, CSS, JavaScript*. University of Michigan.  
  *(Khóa học cung cấp kiến thức cơ bản về phát triển web frontend, hỗ trợ nhóm trong việc xây dựng giao diện.)*
* Udemy. *Spring Boot and Hibernate for Beginners*. Instructor: Chad Darby.  
  *(Khóa học hướng dẫn cách sử dụng Spring Boot và Hibernate để phát triển backend và quản lý cơ sở dữ liệu.)*
* YouTube. *REST API with Spring Boot*. Kênh Amigoscode.  
  *(Video hướng dẫn về cách xây dựng RESTful API, áp dụng trong việc thiết kế các endpoint của hệ thống.)*
* Maven. (2025). *Apache Maven Documentation*. Truy cập tại: <https://maven.apache.org/guides/>  
  *(Tài liệu về Maven, được sử dụng để quản lý phụ thuộc và build dự án.)*
* Postman. (2025). *Postman API Documentation*. Truy cập tại: <https://learning.postman.com/docs/>  
  Các tài liệu nội bộ của Trường Đại học Bách Khoa, Khoa Công nghệ Thông tin, bao gồm giáo trình môn Lập trình ứng dụng, Cơ sở dữ liệu, và Phân tích thiết kế hệ thống.
* Các bài giảng và tài liệu hướng dẫn từ TS. Đỗ Thị Tuyết Hoa, người hướng dẫn dự án.
* Các diễn đàn công nghệ như Stack Overflow (<https://stackoverflow.com/>) và GitHub Issues để tham khảo cách giải quyết các vấn đề kỹ thuật trong quá trình lập trình.

PHỤ LỤC