**Trường Đại Học Xây Dựng Hà Nội**

**Khoa Công nghệ Thông tin**

**--o0o--**

**Icon

Description automatically generated**

**Báo cáo đồ án môn học**

**LẬP TRÌNH ỨNG DỤNG MẠNG**

***Tên đề tài: Website tạo cassette cá nhân***

**Lớp: 66MHT1**

**Nhóm thực hiện: Nhóm 6**

**Các thành viên:**

**1) Vũ Hoàng Giang - 0188166**

**2) Nguyễn Xuân Sang - 0202766**

**Giáo viên hướng dẫn : Thầy Lê Đức Quang**

**Hà Nội, 30/5/2024**

Mục lục

[A. PHÂN TÍCH THIẾT KẾ HỆ THỐNG 4](#_Toc167966594)

[I Đặt vấn đề - Xác định yêu cầu khách hàng 4](#_Toc167966595)

[1. Đặt vấn đề 4](#_Toc167966596)

[1.1 Hệ thống dự định làm 4](#_Toc167966597)

[1.2 Tóm tắt dự án: 4](#_Toc167966598)

[2. Xác định yêu cầu khách hàng 6](#_Toc167966599)

[2.1 Sơ đồ usecase 6](#_Toc167966600)

[2.2 Phân tích yêu cầu 7](#_Toc167966601)

[2.3 Biểu đồ sequence 26](#_Toc167966602)

[3. Thiết kế cơ sở dữ liệu 28](#_Toc167966603)

[4. Thiết kế chi tiết 32](#_Toc167966604)

[4.1 Danh sách chức năng 32](#_Toc167966605)

[4.2 Thiết kế chi tiết 33](#_Toc167966606)

[I Hướng dẫn sử dụng phần user 63](#_Toc167966607)

[1. Đăng nhập 63](#_Toc167966608)

[2. Đăng kí 64](#_Toc167966609)

[3. Đăng xuất 65](#_Toc167966610)

[4. Chỉnh sửa thông tin cá nhân 66](#_Toc167966611)

[5. Tạo cassette 67](#_Toc167966612)

[6. Thêm giỏ hàng 69](#_Toc167966613)

[7. Xem giỏ hàng 69](#_Toc167966614)

[8. Tạo đơn hàng 70](#_Toc167966615)

[III Hướng dẫn sử dụng phần admin 70](#_Toc167966616)

[1. Đăng nhập 70](#_Toc167966617)

[2. Đăng xuất 70](#_Toc167966618)

[3. Quản lý mẫu băng 71](#_Toc167966619)

[3.1 Thêm mới 71](#_Toc167966620)

[3.2 Cập nhập 72](#_Toc167966621)

[3.3 Xóa 72](#_Toc167966622)

[4. Quản lý đơn hàng 73](#_Toc167966623)

[4.1 Xem đơn hàng 73](#_Toc167966624)

[4.2 Xóa đơn hàng 74](#_Toc167966625)

[5. Báo cáo thống kê 74](#_Toc167966626)

[6. Quản lý user 74](#_Toc167966627)

[6.1 Block user 74](#_Toc167966628)

1. PHÂN TÍCH THIẾT KẾ HỆ THỐNG

# I Đặt vấn đề - Xác định yêu cầu khách hàng

## 1. Đặt vấn đề

### 1.1 Hệ thống dự định làm

Sau khi nghiên cứu về thị trường tại Việt Nam, em thấy ngành băng đĩa đang rất phát triển và đáng được lưu tâm vì sự phát triển của các nghệ sĩ trẻ, nghệ sĩ underground, .... Hiện nay, sự cá nhân hóa đang ngày càng được xem trọng, ai cũng muốn có một “chất” riêng trong các món đồ mình sử dụng. Hoặc là nhu cầu tạo băng đĩa những nhóm nhạc mới chưa có kinh phí lớn để sản xuất đại trà. Từ đó, nhóm em đã thực hiện phân tích và thiết kế 1 web tạo băng cassette cá nhân gồm ngoại hình và mixtape do khách hàng quyết định.

Ngôn ngữ lập trình: Java Spring Boot MVC, Angular

Hệ quản trị CSDL: MySQL

### 1.2 Tóm tắt dự án:

a, Mô tả sơ lược:

· Khi khách truy cập trang web, khách có thể chọn đăng kí, đăng nhập hoặc chế độ khách.

o Khách có thể đăng kí tài khoản (Tài khoản; Mật Khẩu; Tên; SĐT; Email, địa chỉ)

o Ở chế độ khách, khách có thể xem và thêm vào giỏ hàng

o Khi mua hàng, khách phải đăng nhập mới được phép thanh toán khi đã có đăng kí tài khoản

· Khách có thể lựa chọn chức năng tạo mixtape cá nhân:

o Khách được chọn ngoại hình của Cassette (Được thêm vào do nhà sản xuất (quản trị))

o Khách được chọn mixtape thu vào Cassette

§ Mixtape là playlist do nhà sản xuất có sẵn.

§ Mixtape là âm thanh ngoài được thu âm và gửi link cho nhà sản xuất (theo quy định của nhà sản xuất)

· Khi khách chọn được sản phầm ưng ý, khách có thể thêm sản phầm vào giỏ hàng

· Khi khách ấn thanh toán, khách cần chuyển khoản toàn bộ số tiền đến 1 stk nhất định, bên admin sẽ check và xác nhận thanh toán, sẽ gửi thông báo về tài khoản người dùng.

· Admin khách sạn có tài khoản riêng để quản lí băng đĩa, quản lí đơn đặt hàng, xem báo cáo thống kê

b. Nhu cầu quản lí

· Quản lý băng đĩa

· Ghi nhận đặt hàng

· Ghi nhận thanh toán

· Báo cáo thống kê

c. Đối tượng

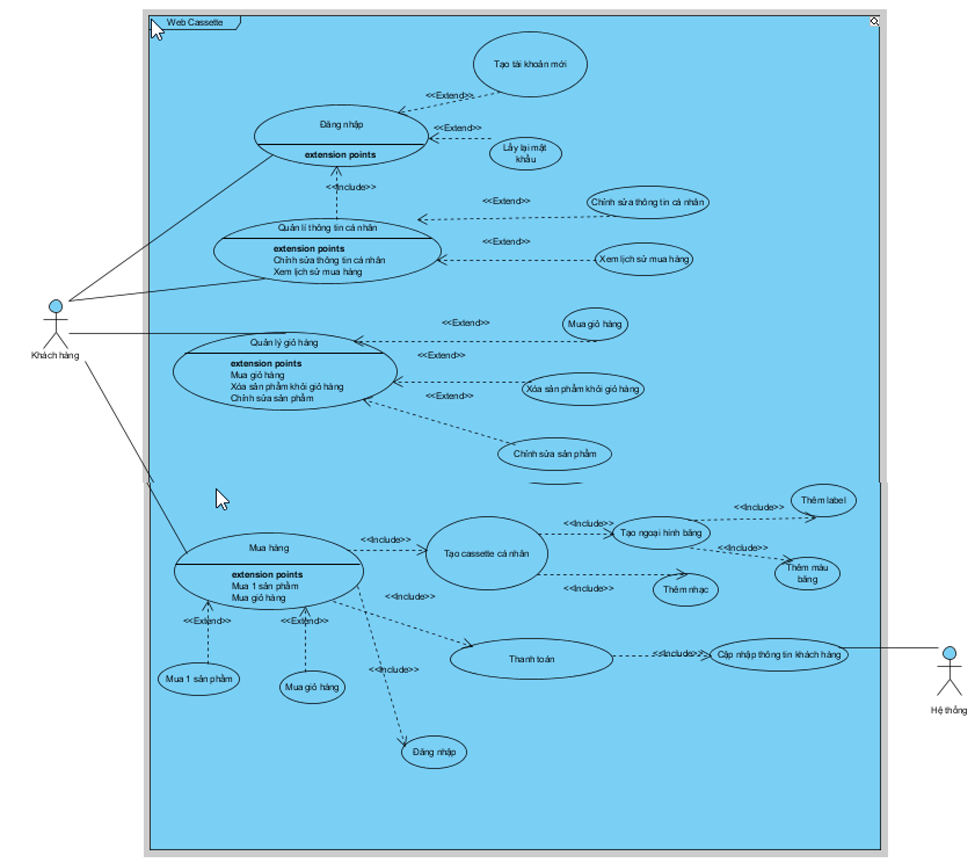
· Admin: Quản lí băng đĩa, quản lí đơn đặt hàng, Báo cáo thống kê

· Khách hàng: Đăng kí, đăng nhập, Tạo mixtape (ngoại hình, âm thanh), Thanh toán, Giỏ hàng

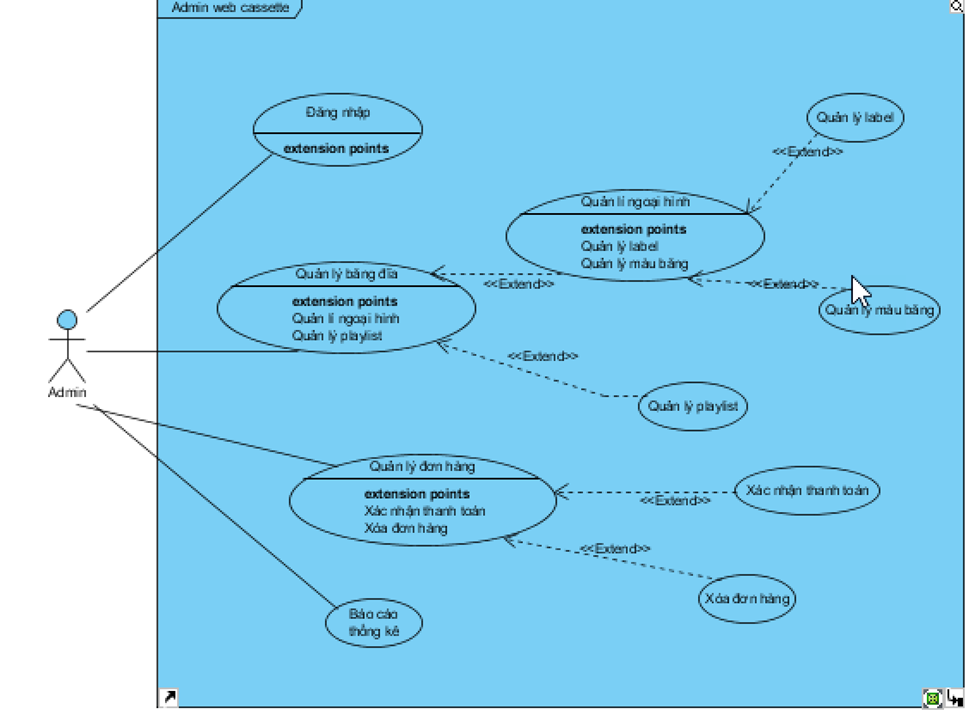
## 2. Xác định yêu cầu khách hàng

### 2.1 Sơ đồ usecase

#### 2.1.1 Website



#### 2.1.2 Admin



### 2.2 Phân tích yêu cầu

#### 2.2.1 Khách hàng

##### 2.2.1.1 Chức năng đăng nhập

|  |  |
| --- | --- |
| **Tên chức năng** | Đăng nhập |
| **Mô tả** | Chức năng cho phép khách hàng đăng nhập vào tài khoản đã có của mình |
| **Tác nhân** | Khách hàng |
| **Điều kiện trước** | Khách hàng nhấn tùy chọn đăng nhập ở trang chủ |
| **Điều kiện sau** | Thành công : Quay trở lại trang chủ  Thất bại : Thông báo thất bại, hiển thị tùy chọn cho khách hàng |
| **Ngoại lệ** | N/A |
| **Yêu cầu đặc biệt** | N/A |

- Mô tả dòng sự kiện chính

|  |  |
| --- | --- |
| **Hành động của tác nhân** | **Phản ứng hệ thống** |
| 1. Nhấn tùy chọn đăng nhập website 2. Nhập thông tin đăng nhập |  |
|  | 3.Hệ thống xác nhận, chuyển về giao diện đăng nhập thất bại hoặc thành công |

##### 2.2.1.2 Tạo tài khoản mới

|  |  |
| --- | --- |
| **Tên chức năng** | Tạo tài khoản |
| **Mô tả** | Chức năng cho phép khách hàng tạo 1 tài khoản mới |
| **Tác nhân** | Khách hàng |
| **Điều kiện trước** | Khách hàng nhấn tùy chọn đăng kí |
| **Điều kiện sau** | Thành công : Hiển thị giao diện trang chủ  Thất bại : Thông báo thất bại, hiển thị tùy chọn cho khách hàng |
| **Ngoại lệ** | N/A |
| **Yêu cầu đặc biệt** | N/A |

- Mô tả dòng sự kiện chính

|  |  |
| --- | --- |
| **Hành động của tác nhân** | **Phản ứng hệ thống** |
| 1. Nhấn tùy chọn đăng kí tài khoản 2. Nhập các thông tin hệ thống yêu cầu |  |
|  | 3.Hệ thống cập nhật dữ liệu, hiển thị giao diện trang chủ |

##### 2.2.1.3 Xem lịch sử mua hàng

|  |  |
| --- | --- |
| **Tên chức năng** | Lịch sử mua hàng |
| **Mô tả** | Chức năng cho phép khách hàng tra cứu lịch sử các giao dịch |
| **Tác nhân** | Khách hàng |
| **Điều kiện trước** | Khách hàng đã đăng nhập vào hệ thống, nhấn tùy chọn tra cứu lịch sử mua hang |
| **Điều kiện sau** | Hiển thị lịch sử giao dịch của khách hàng |
| **Ngoại lệ** | N/A |
| **Yêu cầu đặc biệt** | N/A |

- Mô tả dòng sự kiện chính

|  |  |
| --- | --- |
| **Hành động của tác nhân** | **Phản ứng hệ thống** |
| 1. Khách hàng nhấn tra cứu lịch sử giao dịch |  |
|  | 1. Hệ thống ghi nhận mã khách hàng đưa ra yêu cầu 2. Hệ thống hiển thị lịch sử giao dịch theo mã khách hàng |

##### 2.2.1.4 Giỏ hàng

|  |  |
| --- | --- |
| **Tên chức năng** | Giỏ Hàng |
| **Mô tả** | Chức năng cho phép khách hàng xem giỏ hàng của mình và thanh toán cho giỏ hàng |
| **Tác nhân** | Khách hàng |
| **Điều kiện trước** | Khách hàng đã đăng nhập vào hệ thống, nhấn tùy chọn giỏ hàng |
| **Điều kiện sau** | Khách hàng có thể xem thông tin giỏ hàng của mình |
| **Ngoại lệ** | N/A |
| **Yêu cầu đặc biệt** | N/A |

- Mô tả dòng sự kiện chính

|  |  |
| --- | --- |
| **Hành động của tác nhân** | **Phản ứng hệ thống** |
| 1. Khách hàng nhấn giỏ hàng |  |
|  | 1. Hệ thống ghi nhận mã khách hàng đưa ra yêu cầu 2. Hệ thống hiển thị giỏ hàng |

##### 2.2.1.5 Xóa sản phẩm khỏi giỏ hàng

|  |  |
| --- | --- |
| **Tên chức năng** | Xóa sản phẩm |
| **Mô tả** | Chức năng cho phép khách hàng xóa những sản phẩm không mong muốn trong giỏ hàng |
| **Tác nhân** | Khách hàng |
| **Điều kiện trước** | Khách hàng đã đăng nhập vào hệ thống, nhấn tùy chọn giỏ hàng |
| **Điều kiện sau** | Khách hàng xem được giỏ hàng sau khi xóa |
| **Ngoại lệ** | N/A |
| **Yêu cầu đặc biệt** | N/A |

- Mô tả dòng sự kiện chính

|  |  |
| --- | --- |
| **Hành động của tác nhân** | **Phản ứng hệ thống** |
| 1. Khách hàng nhấn xóa sản phẩm ở danh mục giỏ hàng |  |
|  | 1. Hệ thống ghi nhận mã khách hàng đưa ra yêu cầu 2. Hệ thống thực hiện xóa sản phẩm và trả lại giao diện giỏ hàng mới |

##### 2.2.1.6 Tạo cassette cá nhân

|  |  |
| --- | --- |
| **Tên chức năng** | Tạo cassette cá nhân |
| **Mô tả** | Chức năng cho phép khách hàng tạo 1 bản casstte cá nhân cho bản thân |
| **Tác nhân** | Khách hàng |
| **Điều kiện trước** | Khách hàng đã đăng nhập vào hệ thống, chọn tùy chọn tạo cassette cá nhân trong giao diện trang chủ |
| **Điều kiện sau** | Khách hàng tùy chọn tạo chức năng tạo casstte |
| **Ngoại lệ** | N/A |
| **Yêu cầu đặc biệt** | N/A |

- Mô tả dòng sự kiện chính

|  |  |
| --- | --- |
| **Hành động của tác nhân** | **Phản ứng hệ thống** |
| 1. Khách hàng nhấn tạo cassette cá nhân |  |
|  | 1. Hệ thống hiển thị tùy chọn cho khách hàng |

##### 2.2.1.7 Tạo ngoại hình băng

|  |  |
| --- | --- |
| **Tên chức năng** | Tạo ngoại hình băng casette |
| **Mô tả** | Chức năng cho phép khách hàng tạo ngoại hình cho sản phẩm cassette của mình |
| **Tác nhân** | Khách hàng |
| **Điều kiện trước** | Khách hàng đã đăng nhập vào hệ thống, chọn tùy chọn tạo băng cassette cá nhân, chọn tùy chọn tạo ngoại hình |
| **Điều kiện sau** | N/A |
| **Ngoại lệ** | N/A |
| **Yêu cầu đặc biệt** | N/A |

- Mô tả dòng sự kiện chính

|  |  |
| --- | --- |
| **Hành động của tác nhân** | **Phàn ứng hệ thống** |
| 1. Khách hàng nhấn tạo ngoại hình |  |
|  | 1. Hệ thống hiển thị giao diện các ngoại hình cho khách hàng lựa chọn |

##### 2.2.1.8 Chức năng thêm nhạc

|  |  |
| --- | --- |
| **Tên chức năng** | Thêm nhạc |
| **Mô tả** | Chức năng cho phép khách hàng thêm các ca khúc vào băng cassette của mình |
| **Tác nhân** | Khách hàng |
| **Điều kiện trước** | Khách hàng đã đăng nhập vào hệ thống, nhấn tùy chọn tạo băng cassette cá nhân |
| **Điều kiện sau** | khách hàng có thể thêm nhạc vào cassette |
| **Ngoại lệ** | N/A |
| **Yêu cầu đặc biệt** | N/A |

- Mô tả dòng sự kiện chính

|  |  |
| --- | --- |
| **Hành động của tác nhân** | **Phàn ứng hệ thống** |
| 1. Khách hàng nhấn thêm nhạc |  |
|  | 1. Hệ thống hiển thị giao diện cho khách hàng thêm nhạc |

##### 2.2.1.8 Thanh toán

|  |  |
| --- | --- |
| **Tên chức năng** | Thanh toán |
| **Mô tả** | Chức năng cho phép khách hàng thanh toán cho sản phẩm của mình |
| **Tác nhân** | Khách hàng |
| **Điều kiện trước** | Khách hàng đã đăng nhập vào hệ thống, nhấn thanh toán |
| **Điều kiện sau** | Khách hàng có thể xem được giỏ hàng của |
| **Ngoại lệ** | N/A |
| **Yêu cầu đặc biệt** | N/A |

##### 2.2.1.9 Cập nhật sản phẩm

|  |  |
| --- | --- |
| **Tên chức năng** | Cập nhật sản phẩm |
| **Mô tả** | Chức năng cho phép khách hàng chỉnh sửa lại sản phẩm mình đã tạo |
| **Tác nhân** | Khách hàng |
| **Điều kiện trước** | Khách hàng đã đăng nhập vào hệ thống, chọn tùy chọn cập nhật sản phẩm |
| **Điều kiện sau** | Khách hàng có thể tùy chọn chỉnh sửa sản phẩm |
| **Ngoại lệ** | N/A |
| **Yêu cầu đặc biệt** | N/A |

- Mô tả dòng sự kiện chính

|  |  |
| --- | --- |
| **Hành động của tác nhân** | **Phàn ứng hệ thống** |
| 1. Khách hàng nhấn cập nhật sản phẩm |  |
|  | 1. Hệ thống chuyển đến giao diện cập nhật sản phẩm khách hàng yêu cầu |

#### 2.2.2 Admin

##### 2.2.2.1 Đăng nhập

|  |  |
| --- | --- |
| **Tên chức năng** | Đăng nhập |
| **Mô tả** | Chức năng cho phép admin đăng nhập vào hệ thống quản lý |
| **Tác nhân** | admin |
| **Điều kiện trước** | Admin chọn tùy chọn đăng nhập |
| **Điều kiện sau** | Admin xem được giao diện hệ thống quản lý |
| **Ngoại lệ** | N/A |
| **Yêu cầu đặc biệt** | N/A |

- Mô tả dòng sự kiện chính

|  |  |
| --- | --- |
| **Hành động của tác nhân** | **Phàn ứng hệ thống** |
| 1. Admin nhấn tùy chọn đăng nhập 2. Admin nhập thông tin đăng nhập |  |
|  | 1. Hệ thống hiển thị giao diện quản lý |

##### 2.2.2.2 Quản lý băng đĩa

|  |  |
| --- | --- |
| **Tên chức năng** | Quản lý băng đĩa |
| **Mô tả** | Chức năng cho phép admin quản lý băng đĩa cassette có trên hệ thống |
| **Tác nhân** | admin |
| **Điều kiện trước** | Admin đã đăng nhập vào hệ thống |
| **Điều kiện sau** | Admin được tùy chọn quản lý băng đĩa |
| **Ngoại lệ** | N/A |
| **Yêu cầu đặc biệt** | N/A |

- Mô tả dòng sự kiện chính

|  |  |
| --- | --- |
| **Hành động của tác nhân** | **Phàn ứng hệ thống** |
| 1. Khách hàng nhấn thanh toán |  |
|  | 1. Hệ thống ghi nhận mã khách hàng đưa ra yêu cầu 2. Hệ thống hiển thị giỏ hàng, tổng tiền |

##### 2.2.2.3 Quản lý màu băng

|  |  |
| --- | --- |
| **Tên chức năng** | Quản lý màu băng |
| **Mô tả** | Chức năng cho phép admin có thể xem được các màu băng đang có trên hệ thống |
| **Tác nhân** | admin |
| **Điều kiện trước** | Admin đã đăng nhập vào hệ thống, chọn tùy chọn quản lý màu băng |
| **Điều kiện sau** | Admin có thể xem được thông tin các màu băng, có các tùy chọn thêm, xóa màu băng |
| **Ngoại lệ** | N/A |
| **Yêu cầu đặc biệt** | N/A |

- Mô tả dòng sự kiện chính

|  |  |
| --- | --- |
| **Hành động của tác nhân** | **Phàn ứng hệ thống** |
| 1. Admin nhấn quản lý màu băng |  |
|  | 1. Hệ thống hiển thị thông tin màu băng trên hệ thống |

* + - 1. Quản lý playlist

|  |  |
| --- | --- |
| **Tên chức năng** | Quản lý playlist |
| **Mô tả** | Chức năng cho phép admin quản lý danh sách playlist trên hệ thống |
| **Tác nhân** | admin |
| **Điều kiện trước** | Admin đăng nhập vào hệ thống, nhấn tùy chọn quản lý băng đĩa, chọn tùy chọn quản lý playlist |
| **Điều kiện sau** | Admin có thể xem được thông tin, thực hiện các chức năng với playlist |
| **Ngoại lệ** | N/A |
| **Yêu cầu đặc biệt** | N/A |

- Mô tả dòng sự kiện chính

|  |  |
| --- | --- |
| **Hành động của tác nhân** | **Phàn ứng hệ thống** |
| 1. Admin nhấn quản lý playlist |  |
|  | 1. Hệ thống hiển thị danh sách playlist |

##### 2.2.2.4Quản lý đơn hàng

|  |  |
| --- | --- |
| **Tên chức năng** | Quản lý đơn hàng |
| **Mô tả** | Chức năng cho phép admin quản lý chọn các tùy chọn để quản lý các đơn hàng |
| **Tác nhân** | admin |
| **Điều kiện trước** | Admin đăng nhập vào hệ thống, chọn tùy chọn quản lý đơn hàng |
| **Điều kiện sau** | Admin có thể xem được các tùy chọn quản lý đơn hàng |
| **Ngoại lệ** | N/A |
| **Yêu cầu đặc biệt** | N/A |

- Mô tả dòng sự kiện chính

|  |  |
| --- | --- |
| **Hành động của tác nhân** | **Phàn ứng hệ thống** |
| 1. Admin nhấn quản lý đơn hàng |  |
|  | 1. Hệ thống hiển thị danh sách đơn hàng, các tùy chọn xử lý đơn hàng |

#### 

##### 2.2.2.5 Xác nhận thanh toán

|  |  |
| --- | --- |
| **Tên chức năng** | Xác nhận thanh toán |
| **Mô tả** | Chức năng cho phép admin xác nhận các đơn hàng đã được thanh toán |
| **Tác nhân** | admin |
| **Điều kiện trước** | Admin đăng nhập vào hệ thống, chọn tùy chọn quản lý đơn hàng |
| **Điều kiện sau** | Admin xem được giao diện các đơn hàng đã được xác nhận |
| **Ngoại lệ** | N/A |
| **Yêu cầu đặc biệt** | N/A |

- Mô tả dòng sự kiện chính

|  |  |
| --- | --- |
| **Hành động của tác nhân** | **Phàn ứng hệ thống** |
| 1. Admin nhấn xác nhận thanh toán |  |
|  | 1. Hệ thống hiển thị danh sách đơn hàng được chấp nhận |

2.2.2.6 Xóa đơn hàng

|  |  |
| --- | --- |
| **Tên chức năng** | Xóa đơn hàng |
| **Mô tả** | Chức năng cho phép admin xóa các đơn hàng |
| **Tác nhân** | admin |
| **Điều kiện trước** | Admin đăng nhập vào hệ thống, chọn tùy chọn quản lý đơn hàng |
| **Điều kiện sau** | Admin xem được danh sách các đơn hàng sau khi xóa |
| **Ngoại lệ** | N/A |
| **Yêu cầu đặc biệt** | N/A |

- Mô tả dòng sự kiện chính

|  |  |
| --- | --- |
| **Hành động của tác nhân** | **Phàn ứng hệ thống** |
| 1. Admin nhấn xóa đơn hàng |  |
|  | 1. Hệ thống hiển thị danh sách đơn hàng mới sau khi xóa |

##### 2.2.2.6 Báo cáo thống kê

|  |  |
| --- | --- |
| **Tên chức năng** | Báo cáo thống kê |
| **Mô tả** | Chức năng cho phép admin xem được thống kê trong các tháng |
| **Tác nhân** | admin |
| **Điều kiện trước** | Admin đăng nhập vào hệ thống, chọn tùy chọn báo cáo thống kê |
| **Điều kiện sau** | Admin xem được giao diện báo cáo |
| **Ngoại lệ** | N/A |
| **Yêu cầu đặc biệt** | N/A |

- Mô tả dòng sự kiện chính

|  |  |
| --- | --- |
| **Hành động của tác nhân** | **Phàn ứng hệ thống** |
| 1. Admin nhấn báo cáo thống kê |  |
|  | 1. Hệ thống hiển thị bảng báo cáo thống kê |

## 

### 2.3 Biểu đồ sequence

#### 2.3.1 Tạo đơn hàng

A screenshot of a computer

Description automatically generated

#### 2.3.2 Tạo giỏ hàng

A diagram of a diagram

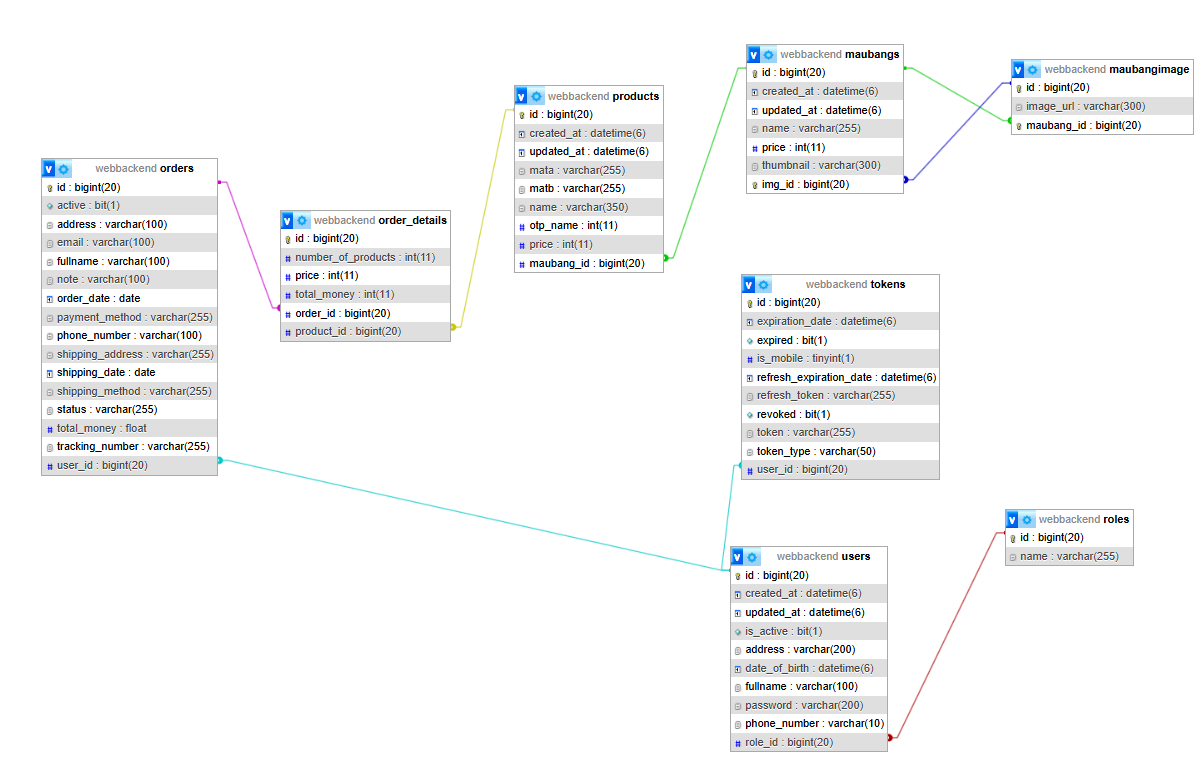
Description automatically generated

#### 2.3.3 Tạo, cập nhập mẫu băng

A diagram of a company

Description automatically generated

## 3. Thiết kế cơ sở dữ liệu



3.1 maubangimage

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Column** | **Type** | **Attributes** | **Null** | **Default** | **Extra** | **Links to** | **Comments** | **MIME** |
| id | bigint(20) |  | No |  | auto\_inc rement |  |  |  |
| image\_url | varchar(300  ) |  | Yes | NULL |  |  |  |  |
| maubang\_id | bigint(20) |  | Yes | NULL |  | -> maubangs.id ON UPDATE RESTRICT  ON DELETE RESTRICT |  |  |

3.2 maubangs

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Column** | **Type** | **Attributes** | **Null** | **Default** | **Extra** | **Links to** | **Comments** | **MIME** |
| id | bigint(20) |  | No |  | auto\_inc rement |  |  |  |
| created\_at | datetime(6) |  | Yes | NULL |  |  |  |  |
| updated\_at | datetime(6) |  | Yes | NULL |  |  |  |  |
| name | varchar(255  ) |  | Yes | NULL |  |  |  |  |
| price | int(11) |  | No |  |  |  |  |  |
| thumbnail | varchar(300  ) |  | Yes | NULL |  |  |  |  |
| img\_id | bigint(20) |  | Yes | NULL |  | ->  maubangimage.id ON UPDATE RESTRICT  ON DELETE RESTRICT |  |  |

3.3 orders

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Column** | **Type** | **Attributes** | **Null** | **Default** | **Extra** | **Links to** | **Comments** | **MIME** |
| id | bigint(20) |  | No |  | auto\_inc rement |  |  |  |
| active | bit(1) |  | Yes | NULL |  |  |  |  |
| address | varchar(100  ) |  | Yes | NULL |  |  |  |  |
| email | varchar(100  ) |  | Yes | NULL |  |  |  |  |
| fullname | varchar(100  ) |  | Yes | NULL |  |  |  |  |
| note | varchar(100  ) |  | Yes | NULL |  |  |  |  |
| order\_date | date |  | Yes | NULL |  |  |  |  |
| payment\_m ethod | varchar(255  ) |  | Yes | NULL |  |  |  |  |
| phone\_num ber | varchar(100  ) |  | No |  |  |  |  |  |
| shipping\_ad dress | varchar(255  ) |  | Yes | NULL |  |  |  |  |
| shipping\_da te | date |  | Yes | NULL |  |  |  |  |
| shipping\_me thod | varchar(255  ) |  | Yes | NULL |  |  |  |  |
| status | varchar(255  ) |  | Yes | NULL |  |  |  |  |
| total\_money | float |  | Yes | NULL |  |  |  |  |
| tracking\_nu mber | varchar(255  ) |  | Yes | NULL |  |  |  |  |
| user\_id | bigint(20) |  | Yes | NULL |  | -> users.id ON UPDATE RESTRICT ON DELETE RESTRICT |  |  |

3.4 order\_details

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Column** | **Type** | **Attributes** | **Null** | **Default** | **Extra** | **Links to** | **Comments** | **MIME** |
| id | bigint(20) |  | No |  | auto\_inc rement |  |  |  |
| number\_of\_ products | int(11) |  | No |  |  |  |  |  |
| price | int(11) |  | No |  |  |  |  |  |
| total\_money | int(11) |  | No |  |  |  |  |  |
| order\_id | bigint(20) |  | Yes | NULL |  | -> orders.id ON UPDATE RESTRICT ON DELETE RESTRICT |  |  |
| product\_id | bigint(20) |  | Yes | NULL |  | -> products.id ON UPDATE RESTRICT  ON DELETE RESTRICT |  |  |

3.5 products

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Column** | **Type** | **Attributes** | **Null** | **Default** | **Extra** | **Links to** | **Comments** | **MIME** |
| id | bigint(20) |  | No |  | auto\_inc rement |  |  |  |
| created\_at | datetime(6) |  | Yes | NULL |  |  |  |  |
| updated\_at | datetime(6) |  | Yes | NULL |  |  |  |  |
| bia | varchar(255  ) |  | Yes | NULL |  |  |  |  |
| mata | varchar(255  ) |  | Yes | NULL |  |  |  |  |
| matb | varchar(255  ) |  | Yes | NULL |  |  |  |  |
| name | varchar(350  ) |  | No |  |  |  |  |  |
| otp\_name | int(11) |  | No |  |  |  |  |  |
| price | int(11) |  | No |  |  |  |  |  |
| maubang\_id | bigint(20) |  | Yes | NULL |  | -> maubangs.id ON UPDATE RESTRICT  ON DELETE RESTRICT |  |  |

3.6 roles

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Column** | **Type** | **Attributes** | **Null** | **Default** | **Extra** | **Links to** | **Comments** | **MIME** |
| id | bigint(20) |  | No |  | auto\_inc rement |  |  |  |
| name | varchar(255  ) |  | No |  |  |  |  |  |

3.7 tokens

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Column** | **Type** | **Attributes** | **Null** | **Default** | **Extra** | **Links to** | **Comments** | **MIME** |
| id | bigint(20) |  | No |  | auto\_inc rement |  |  |  |
| expiration\_d ate | datetime(6) |  | Yes | NULL |  |  |  |  |
| expired | bit(1) |  | No |  |  |  |  |  |
| is\_mobile | tinyint(1) |  | Yes | NULL |  |  |  |  |
| refresh\_expi ration\_date | datetime(6) |  | Yes | NULL |  |  |  |  |
| refresh\_toke n | varchar(255  ) |  | Yes | NULL |  |  |  |  |
| revoked | bit(1) |  | No |  |  |  |  |  |
| token | varchar(255  ) |  | Yes | NULL |  |  |  |  |
| token\_type | varchar(50) |  | Yes | NULL |  |  |  |  |
| user\_id | bigint(20) |  | Yes | NULL |  | -> users.id ON UPDATE RESTRICT ON DELETE RESTRICT |  |  |

3.8 users

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Column** | **Type** | **Attributes** | **Null** | **Default** | **Extra** | **Links to** | **Comments** | **MIME** |
| id | bigint(20) |  | No |  | auto\_inc rement |  |  |  |
| created\_at | datetime(6) |  | Yes | NULL |  |  |  |  |
| updated\_at | datetime(6) |  | Yes | NULL |  |  |  |  |
| is\_active | bit(1) |  | Yes | NULL |  |  |  |  |
| address | varchar(200  ) |  | Yes | NULL |  |  |  |  |
| date\_of\_birt h | datetime(6) |  | Yes | NULL |  |  |  |  |
| fullname | varchar(100  ) |  | Yes | NULL |  |  |  |  |
| password | varchar(200  ) |  | No |  |  |  |  |  |
| phone\_num ber | varchar(10) |  | No |  |  |  |  |  |
| role\_id | bigint(20) |  | Yes | NULL |  | -> roles.id ON UPDATE RESTRICT ON DELETE RESTRICT |  |  |

## 4. Thiết kế chi tiết

### 4.1 Danh sách chức năng

|  |  |  |
| --- | --- | --- |
| STT | Chức năng | Tác nhân |
| 1 | Đăng nhập | User,admin |
| 2 | Đăng kí | User |
| 3 | Đổi mật khẩu, thông tin | User |
| 3 | Tạo cassette | User |
| 4 | Tạo giỏ hàng | User |
| 5 | Xem giỏ hàng | User |
| 6 | Tạo đơn hàng | User |
| 7 | Quản lý đơn hàng | Admin |
| 8 | Quản lý mẫu băng | Admin |
| 9 | Báo cáo thống kê | Admin |

### 4.2 Thiết kế chi tiết

#### 4.2.1 Đăng nhập

4.2.1.1 Giới thiệu chung

- Chức năng giúp user đăng nhập vào website bằng tài khoản đã có hoặc admin đăng nhập hệ thống quản lý

4.2.1.2 Giao diện hiển thị

A screenshot of a phone

Description automatically generated

4..2.1.3 Mô tả thành phần

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Tên** | **Kiểu dữ liệu** | **Input/output** | **Giá trị khởi tạo** | **Mô tả** |
| 1 | txtemail | Textbox | Input | null | Dữ liệu được nhập từ bàn phím |
| 2 | Textpass | Textbox | input | null | Dữ liệu được nhập từ bàn phím |
| 3 | cbxghinho | checkbox |  | null | Dữ liệu được lưu |
| 4 | btndangnhap | button | enable | null | Button chuyển sang giao diện trang chủ |
| 5 | linkquenpass | link | enable | null | Link chuyển sang giao diện lấy lại mật khẩu |

4.2.1.4 Luồng nghiệp vụ

A diagram of a diagram

Description automatically generated

#### 4.2.2 Đăng kí

4.2.2.1 Giới thiệu chung

- Chức năng giúp khách hàng tạo tài khoản đăng nhập cho trang web

4.2.2.2 Giao diện hiển thị

A screenshot of a computer

Description automatically generated

4.2.2.3 Mô tả thành phần

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Tên** | **Kiểu dữ liệu** | **Input/output** | **Giá trị khởi tạo** | **Mô tả** |
| 1 | Textname | Textbox | Input | null | Dữ liệu được nhập từ bàn phím |
| 2 | Textsđt | Textbox | input | null | Dữ liệu được nhập từ bàn phím |
| 3 | Textdiachi | Textbox | Input | null | Dữ liệu được nhập từ bàn phím |
| 4 | Textemail | Textbox | Input | null | Dữ liệu được nhập từ bàn phím |
| 5 | Textloginname | Textbox | input | null | Dữ liệu được nhập từ bàn phím |
| 6 | Textpass | Textbox | Input | Null | Dữ liệu được nhập từ bàn phím |
| 7 | Textxacnhan | Textbox | Input | Null | Dữ liệu được nhập từ bàn phím |
| 8 | btndangki | button | enable | null | Button chuyển về giao diện trang chủ |

4.2.2.4 Luồng xử lý nghiệp vụ

A yellow rectangular object with black text

Description automatically generated

#### 4.2.3 Đổi thông tin tài khoản

4.2.3.1 Giới thiệu chung

- Chức năng giúp khách hàng có thể thay đổi thông tin cá nhân

4.2.3.2 Giao diện

A screenshot of a computer

Description automatically generated

4.2.3.3 Mô tả thành phần

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Tên** | **Kiểu dữ liệu** | **Input/output** | **Giá trị khởi tạo** | **Mô tả** |
| 1 | Textname | Textbox | Input | null | Dữ liệu được nhập từ bàn phím |
| 2 | Textsđt | Textbox | input | null | Dữ liệu được nhập từ bàn phím |
| 3 | Textdiachi | Textbox | Input | null | Dữ liệu được nhập từ bàn phím |
| 4 | Textemail | Textbox | Input | null | Dữ liệu được nhập từ bàn phím |
| 5 | Textloginname | Textbox | input | null | Dữ liệu được nhập từ bàn phím |
| 6 | Textpass | Textbox | Input | Null | Dữ liệu được nhập từ bàn phím |
| 7 | Textxacnhan | Textbox | Input | Null | Dữ liệu được nhập từ bàn phím |
| 8 | btnluu | button | enable | null | Button lưu dữ liệu |

4.2.3.4 Luồng xử lý nghiệp vụ

A yellow rectangular object with black text

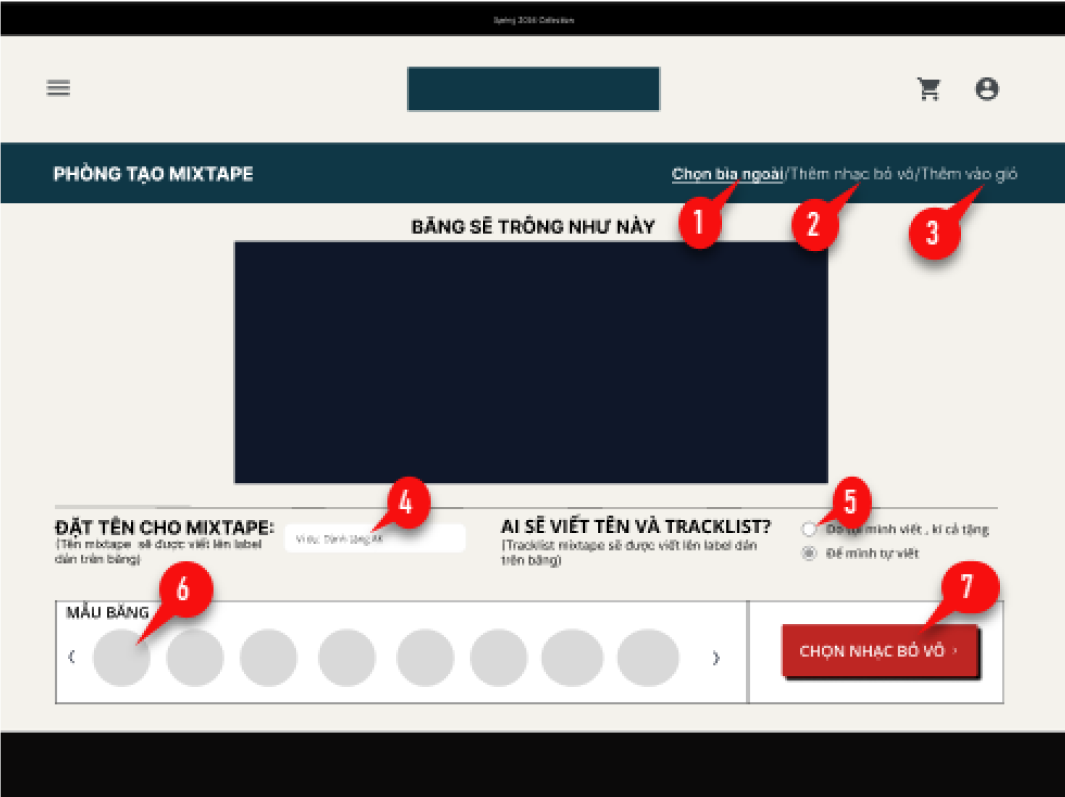
Description automatically generated

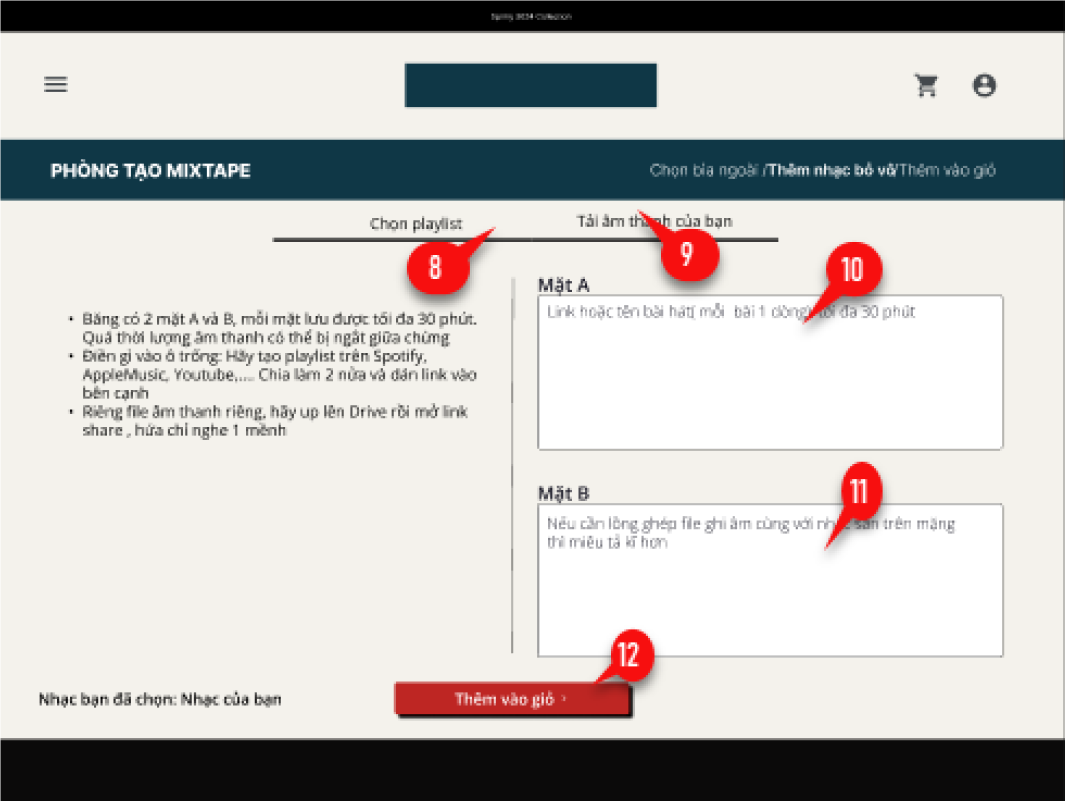
#### 4.2.4 Tạo cassette

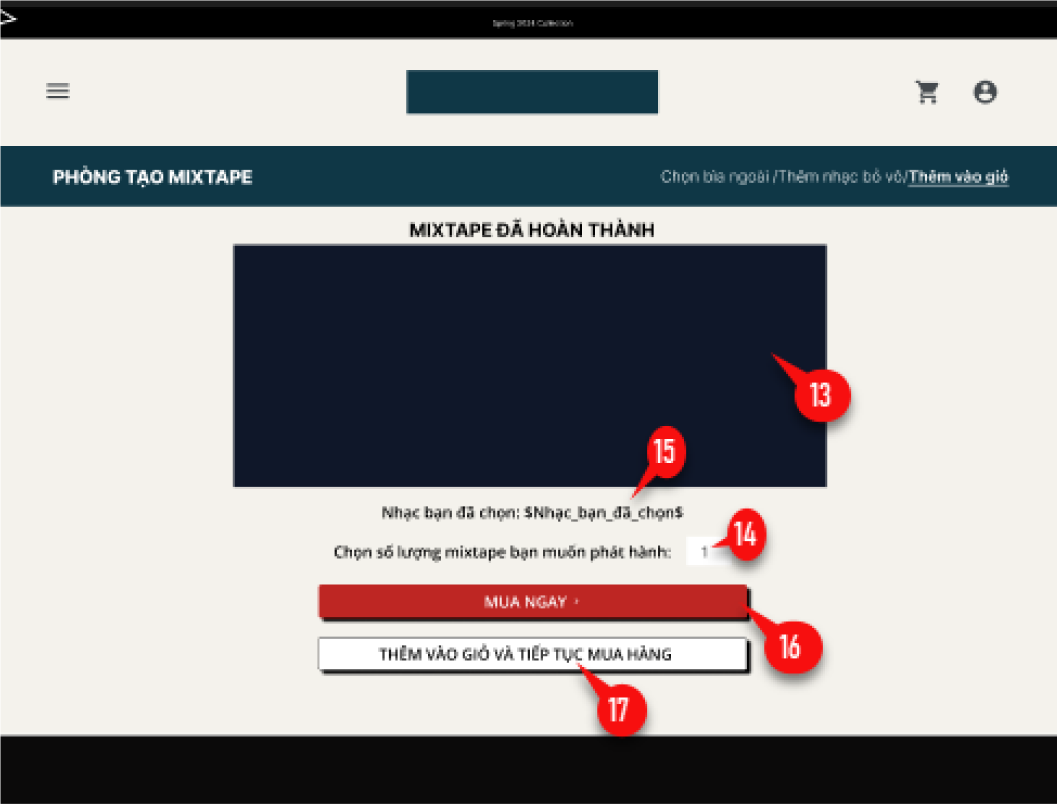
4.2.4.1 Giới thiệu chung

-Chức năng giúp user tạo cassette theo sở thích dựa

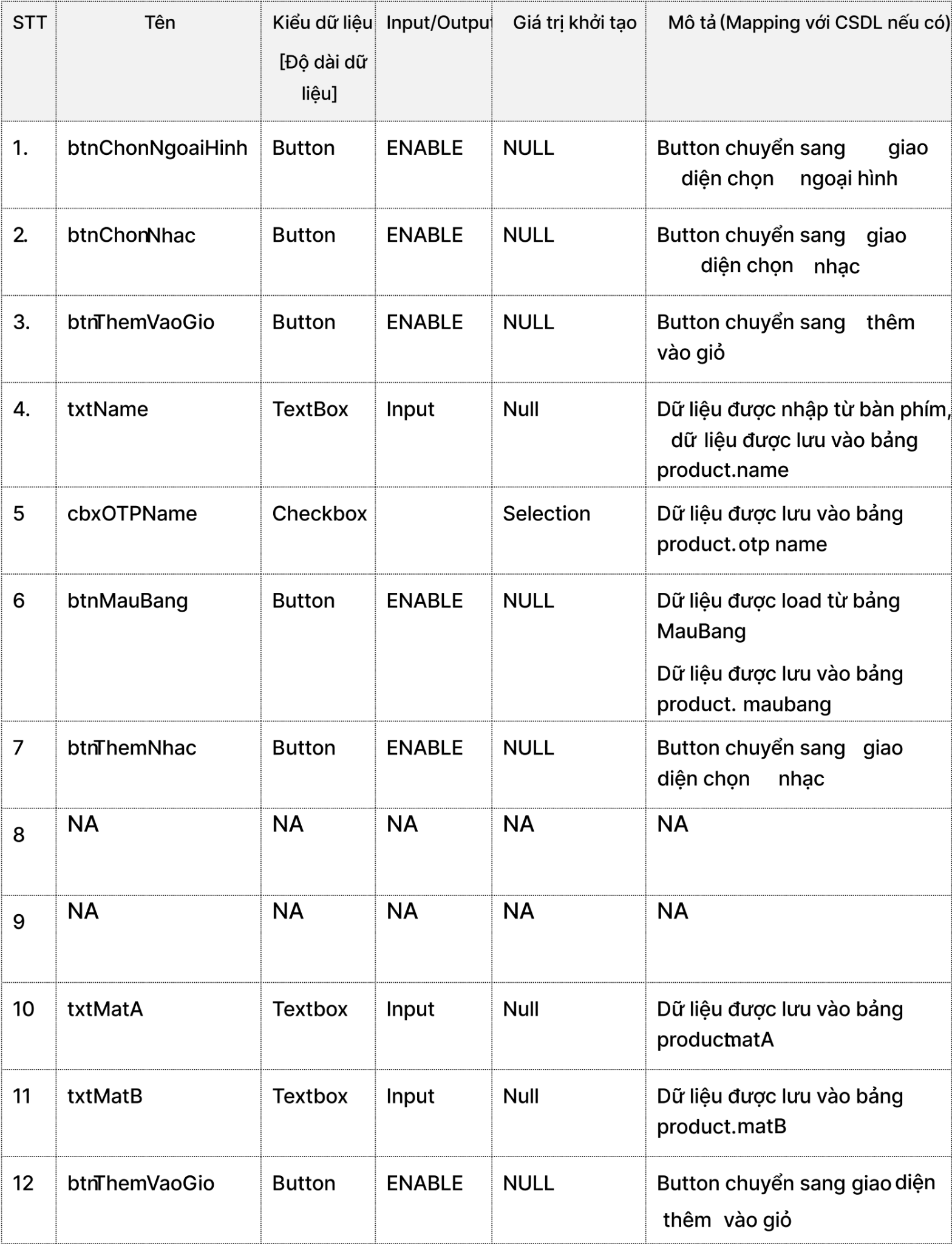
4.2.4.2 Màn hình





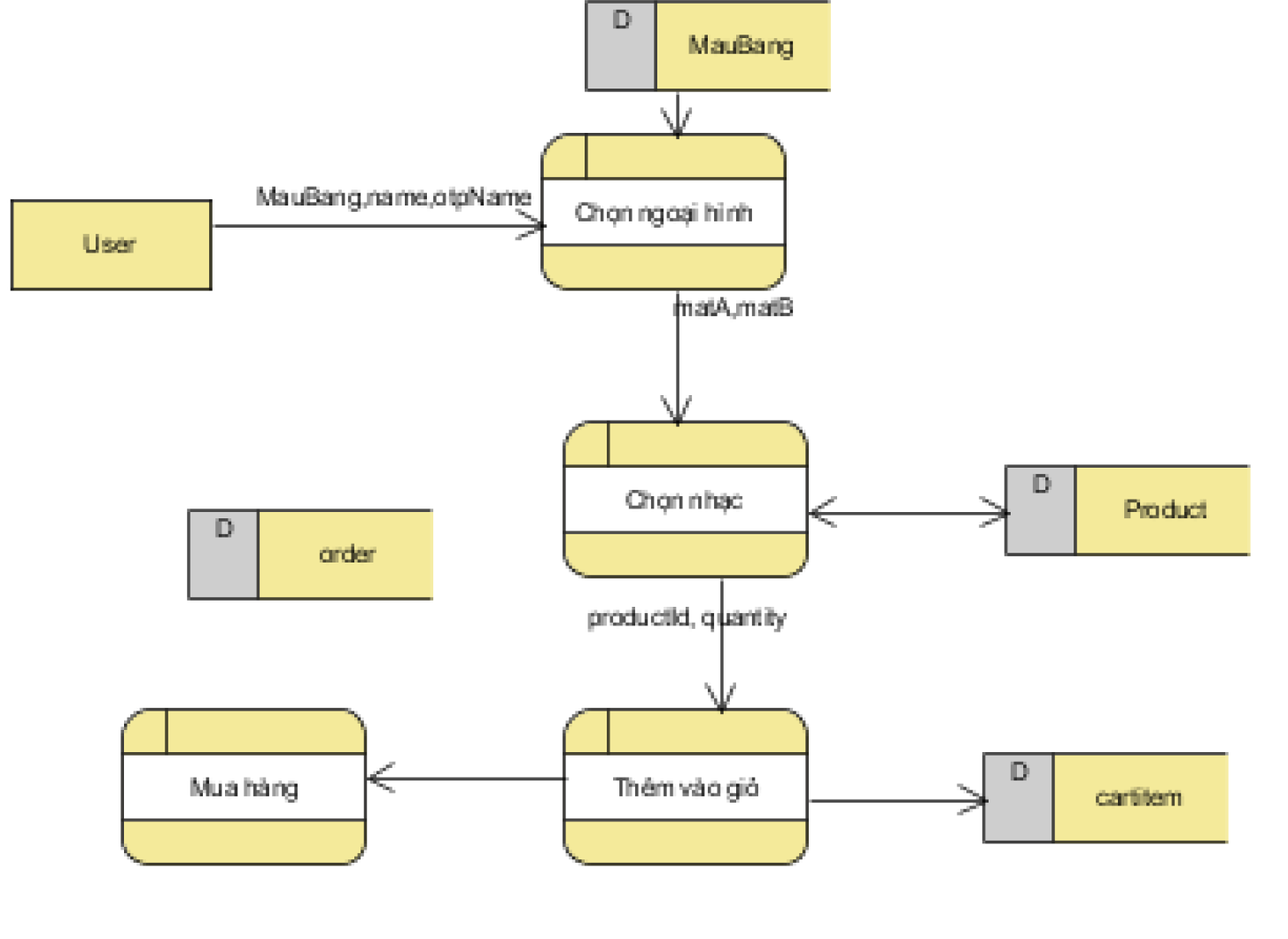


4.2.4.3 Mô tả chi tiết thành phần





4.2.4.4 Luồng nghiệp vụ



1. User chọn ngoại hình cho băng
2. User chọn nhạc cho băng
3. User chọn số lượng phát hành

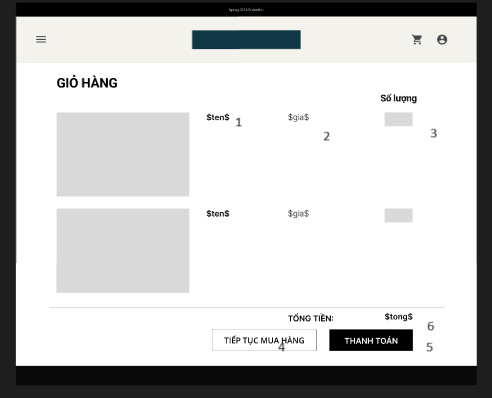
#### 4.2.5 Tạo giỏ hàng

#### 4.1.6 Xem giỏ hàng

4.2.6.1 Giới thiệu chung

* Chức năng giúp user xem giỏ hàng

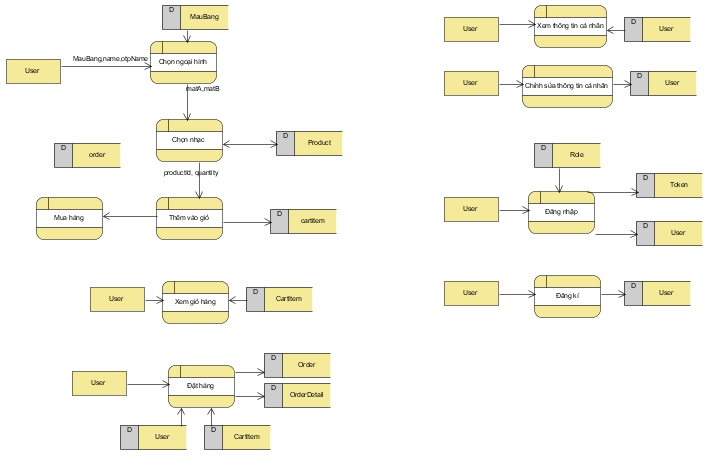
4.2.6.2 Màn hình



4.2.6.3 Mô tả chi tiết thành phần

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Tên | Kiểu dữ liệu | I/O | Giá trị khởi tạo | Mô tả |
| 1 | lbTen | Label | O |  | Dữ liệu được load từ bảng product.name |
| 2 | lbGia | Label | O |  | Dữ liệu được load từ bảng product.name |
| 3 | txtSoLuong | Textbox | I |  | Dữ liệu được lưu vào bảng order-detail.number-of-products |
| 4 | btnTiepTucMuaHang | Button |  |  | Button chuyển về trang chủ |
| 5 | btnThanhToan | Button |  |  | Button chuyển sang giao diện hóa đơn |
| 6 | lbTongTien | Label | O |  | Dữ liệu được load từ bảng order-detail.total-money |

4.2.6.4 Luồng nghiệp vụ



User xem giỏ hàng

4.2.7 Tạo đơn hàng

4.2.7.1 Giới thiệu chung

- Chức năng cho phép khách hàng điền các thông tin cho đơn hàng

4.2.7.2 Giao diện hiển thị

A screenshot of a computer

Description automatically generated

4.2.7.3 Mô tả chung

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Tên** | **Kiểu dữ liệu** | **Input/output** | **Giá trị khởi tạo** | **Mô tả** |
| 1 | Textname | Textbox | Input | null | Dữ liệu được nhập từ bàn phím |
| 2 | Textsđt | Textbox | input | null | Dữ liệu được nhập từ bàn phím |
| 3 | Textemail | Textbox | Input | null | Dữ liệu được nhập từ bàn phím |
| 4 | Textdiachi | Textbox | Input | null | Dữ liệu được nhập từ bàn phím |
| 5 | lbthongtindonhang | label | output |  | Dữ liệu được load từ CSDL |
| 6 | Textghichu | Textbox | Input | Null | Dữ liệu được nhập từ bàn phím |
| 7 | btnxacnhan | button |  |  | Dữ liệu được lưu vào CSDL |
| 8 | lbbillcost | label | Output |  | Dữ liệu được load từ CSDL |

4.2.7.4 Luồng xử lý nghiệp vụ

A diagram of a computer

Description automatically generated

#### 4.2.8 Quản lý đơn hàng

4.2.8.1 Giới thiệu chung

- Chức năng cho phép admin quản lý các đơn hàng trên hệ thống

4.2.8.2 Giao diện hiển thị

A screenshot of a computer

Description automatically generated

4.2.8.3 Mô tả chung

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Tên** | **Kiểu dữ liệu** | **Input/output** | **Giá trị khởi tạo** | **Mô tả** |
| 1 | Textdate | Textbox | Input | null | Dữ liệu được nhập từ bàn phím |
| 2 | Textmadonhang | Textbox | input | null | Dữ liệu được nhập từ bàn phím |
| 3 | tabledonhang | table | output |  | Dữ liệu được load từ CSDL |
| 4 | Linkqlbangdia | link |  |  | Link chuyển sang giao diện quản lý băng đĩa |
| 5 | Linkbaocaothongke | Link | output |  | Link chuyển sang giao diện báo cáo |

#### 4.2.9 Quản lý mẫu băng

4.2.9.1 Giới thiệu chức năng

- Chức năng giúp admin xem thông tin, chỉnh sửa thông tin mẫu băng trên hệ thống

4.2.9.2 Giao diện hiển thị

A screenshot of a computer

Description automatically generated

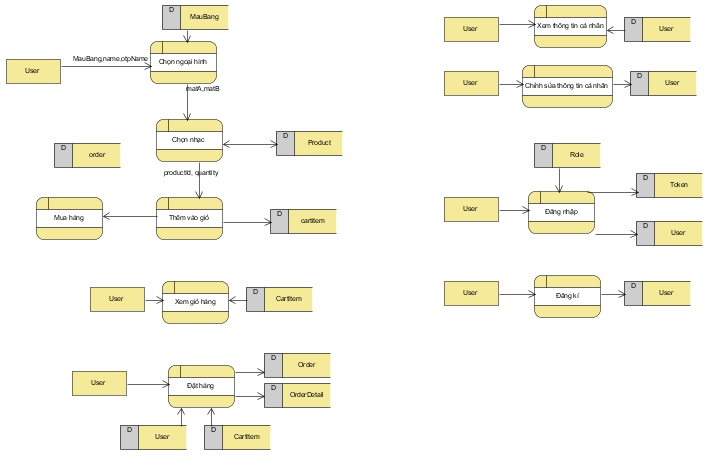
A screenshot of a phone

Description automatically generated

4.2.9.3 Mô tả thành phần

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Tên** | **Kiểu dữ liệu** | **Input/output** | **Giá trị khởi tạo** | **Mô tả** |
| 1 | Table | Table | output |  | Dữ liệu được load từ CSDL |
| 2 | btnthem | button |  |  | Button chuyển sang giao diện thêm băng đĩa |
| 3 | btnsua | button |  |  | Button chuyển sâng giao diện sửa thông tin băng đĩa |
| 4 | btnxoa | button |  |  | Button thực hiện chức năng xóa băng đĩa |
| 5 | Textlabel | Textbox | input | null | Dữ liệu được thêm vào CSDL |
| 6 | lbimg | img | output |  | Hiển thị ảnh admin tải lên |
| 7 | btnupload | button |  |  | Button giúp admin tải lên ảnh mẫu label |
| 8 | btnxacnhan | button |  |  | Button xác nhận tải ảnh lên hệ thống, lưu ảnh vào CSDl |

4.2.10 Báo cáo thống kê

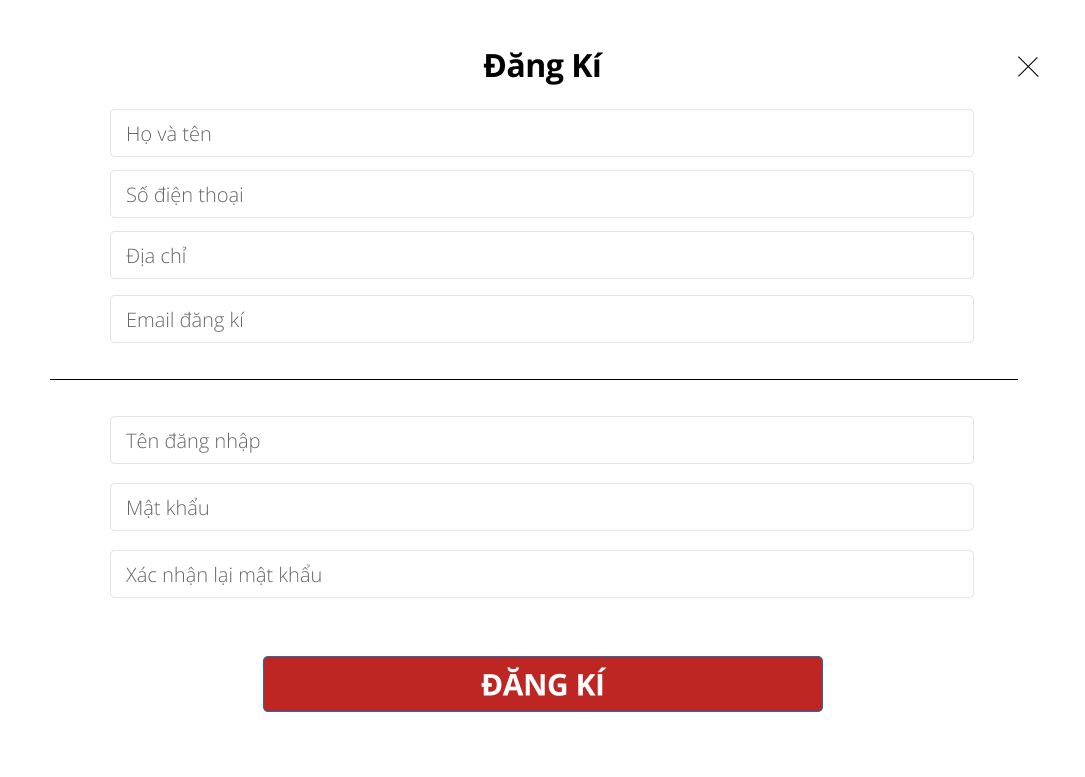
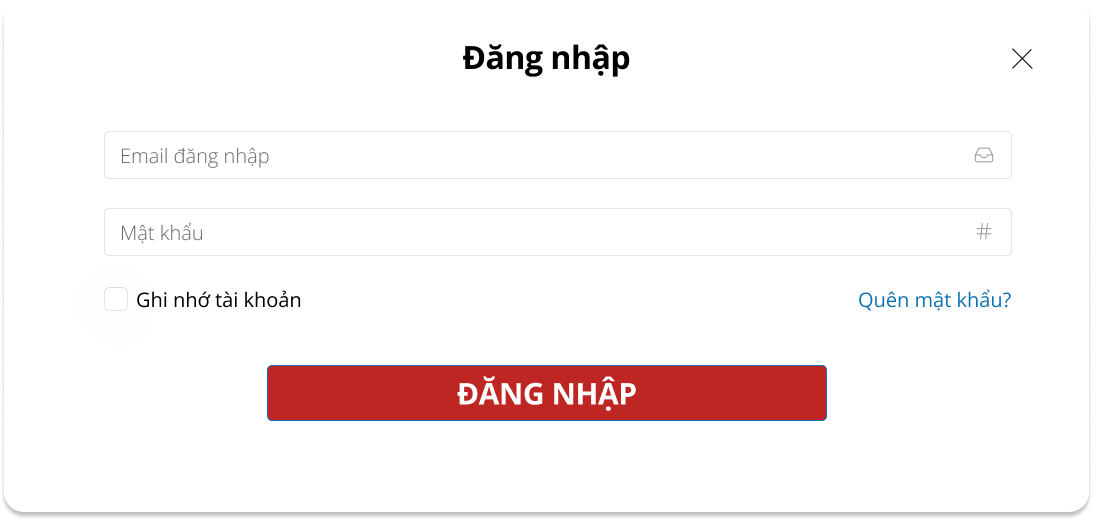


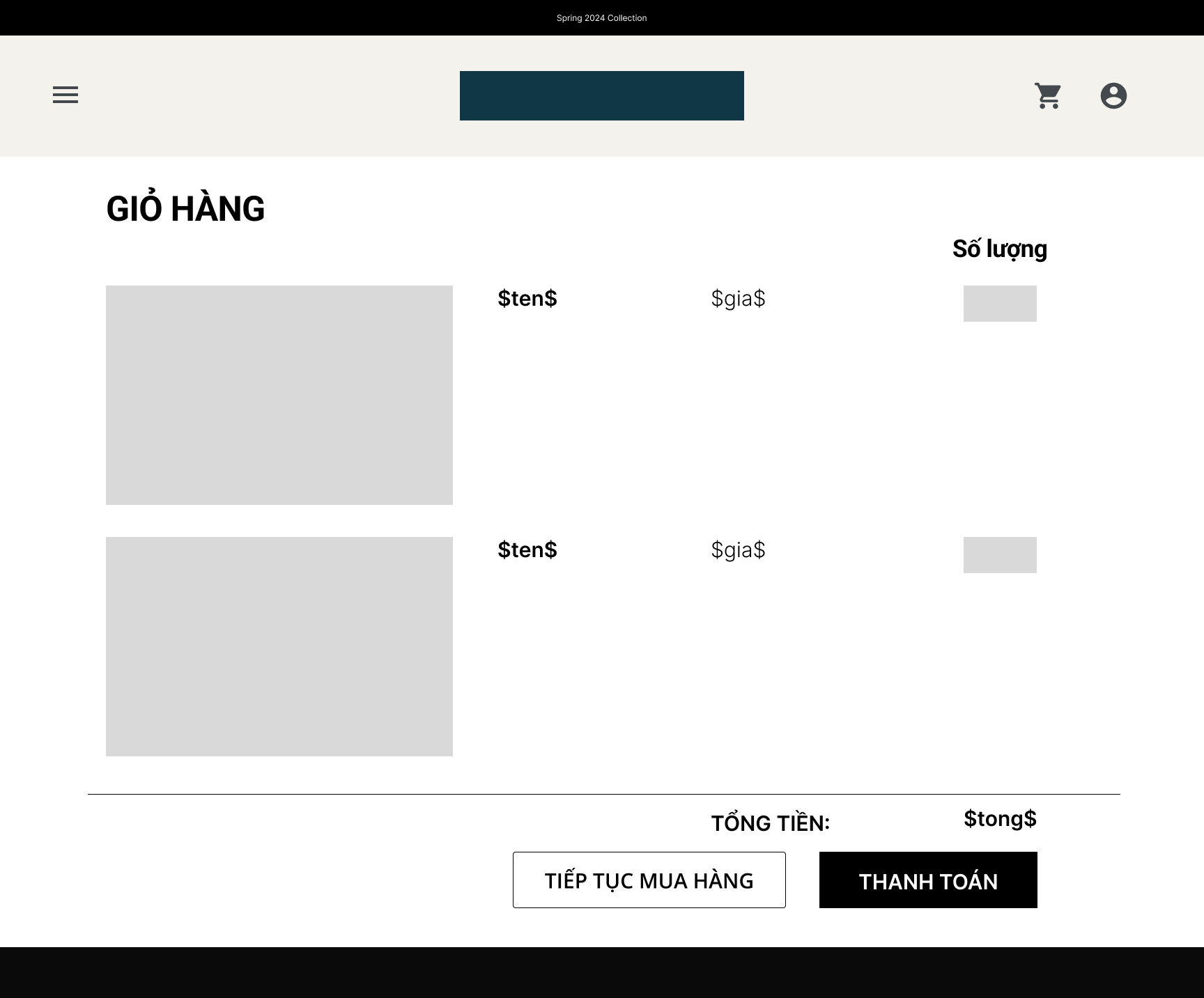
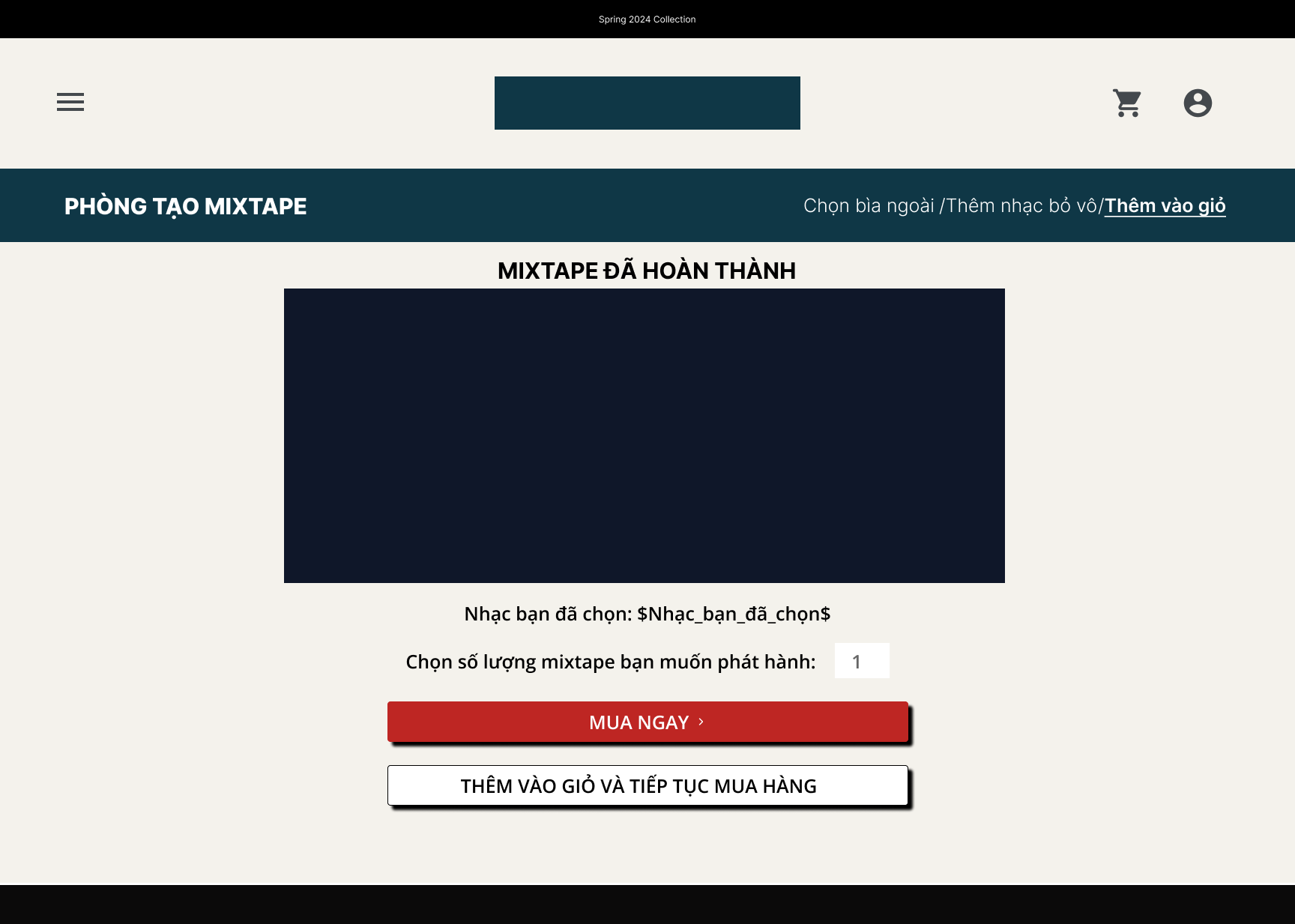
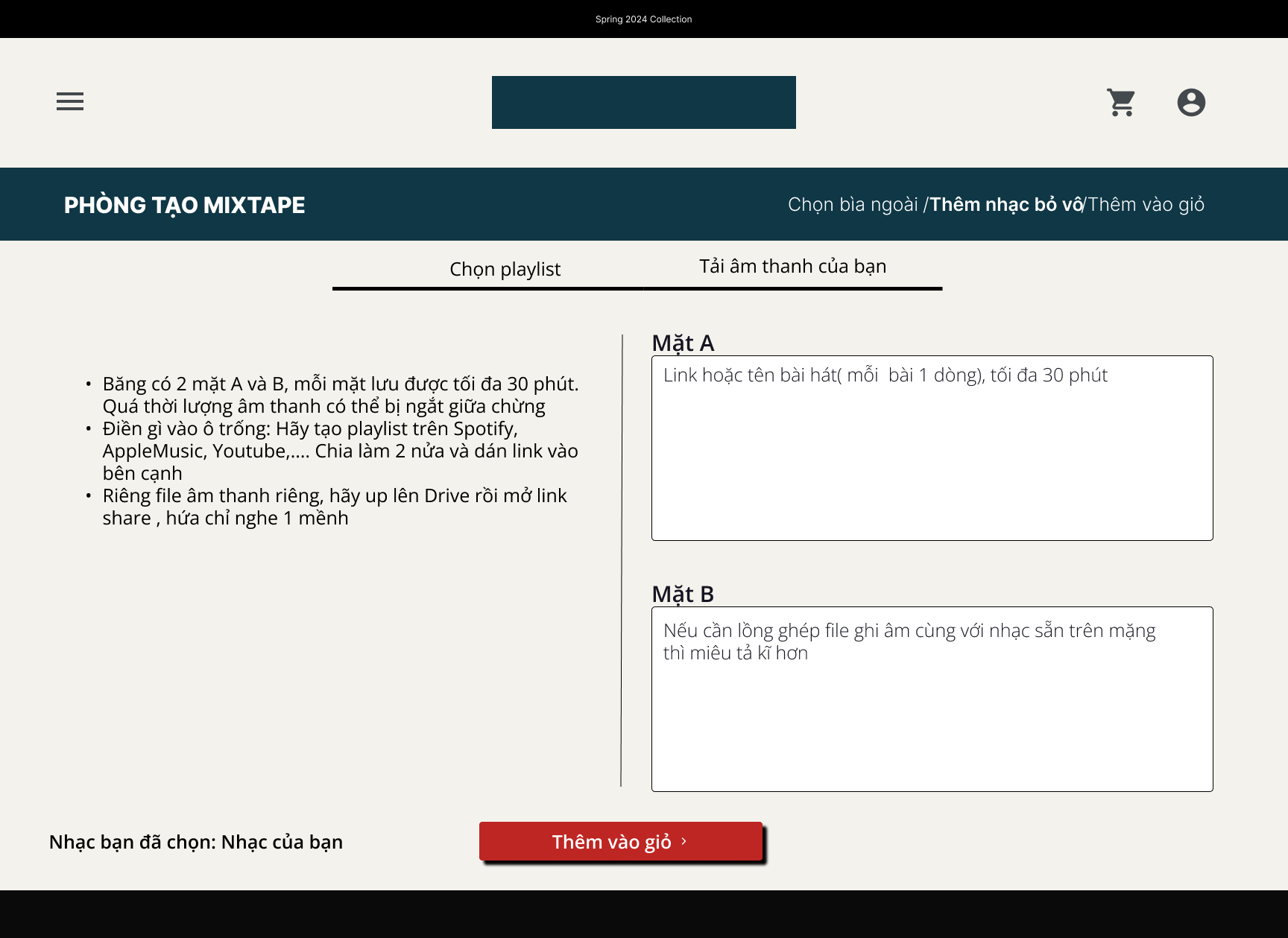
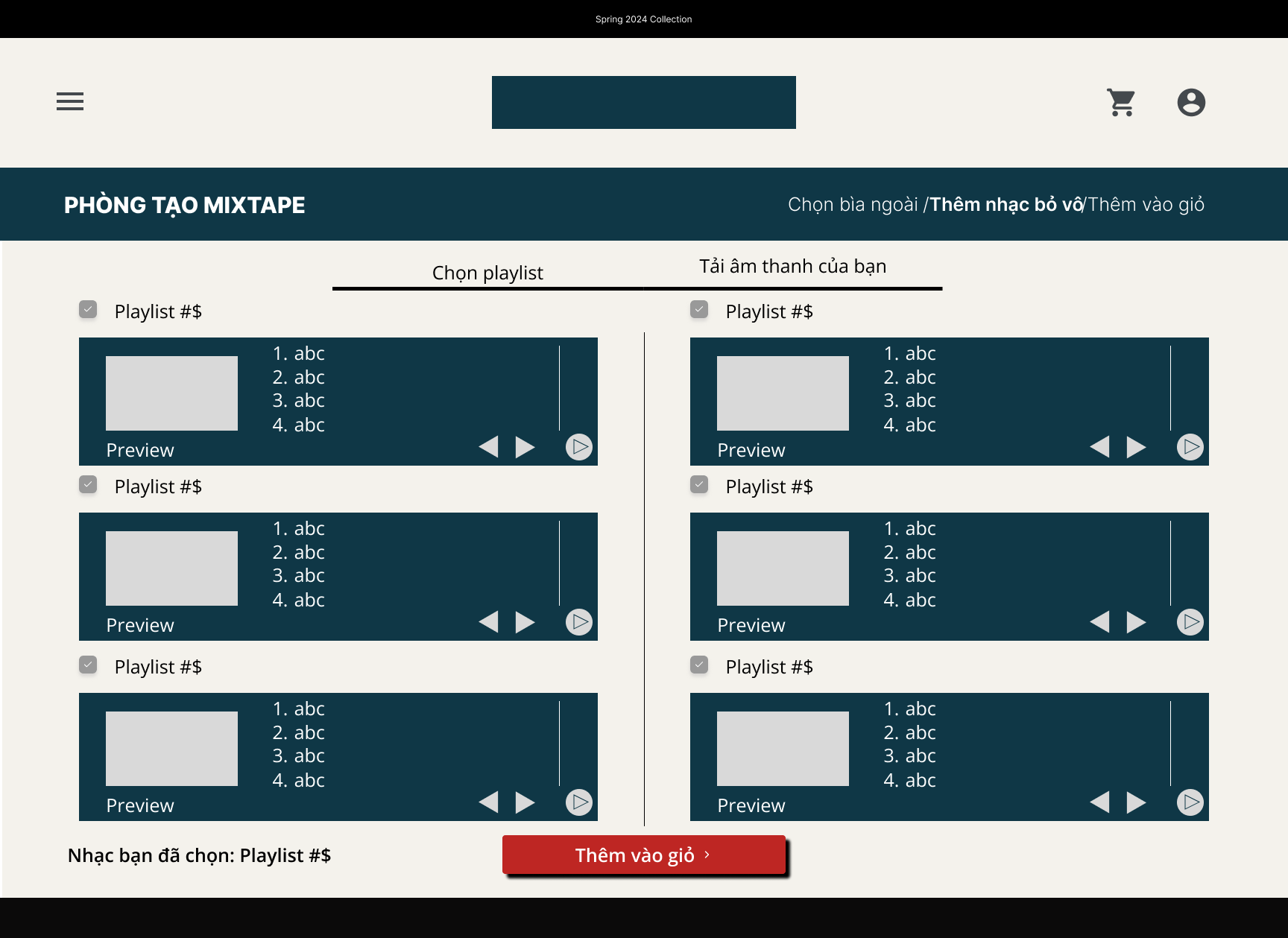
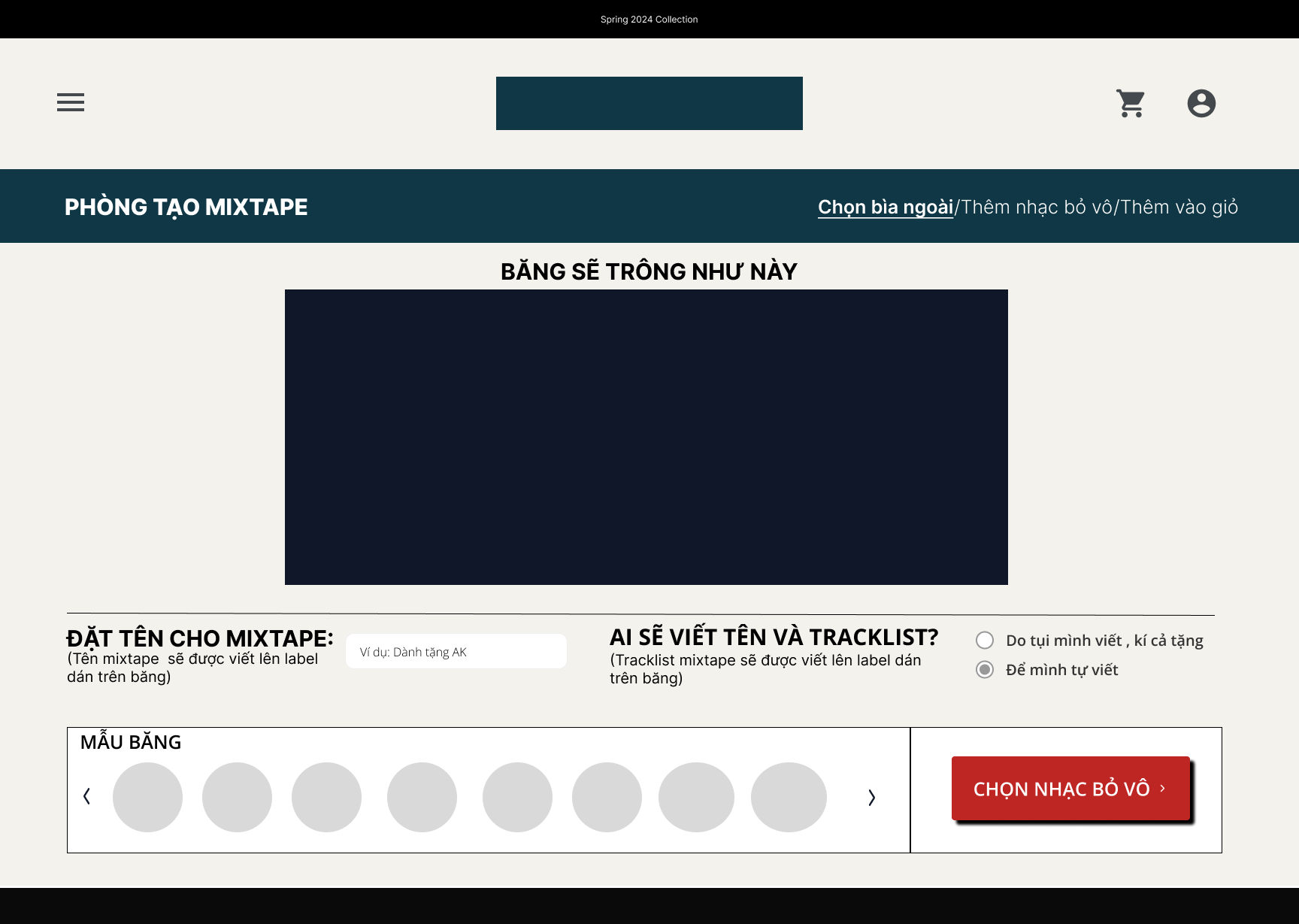
5. Danh sách màn hình

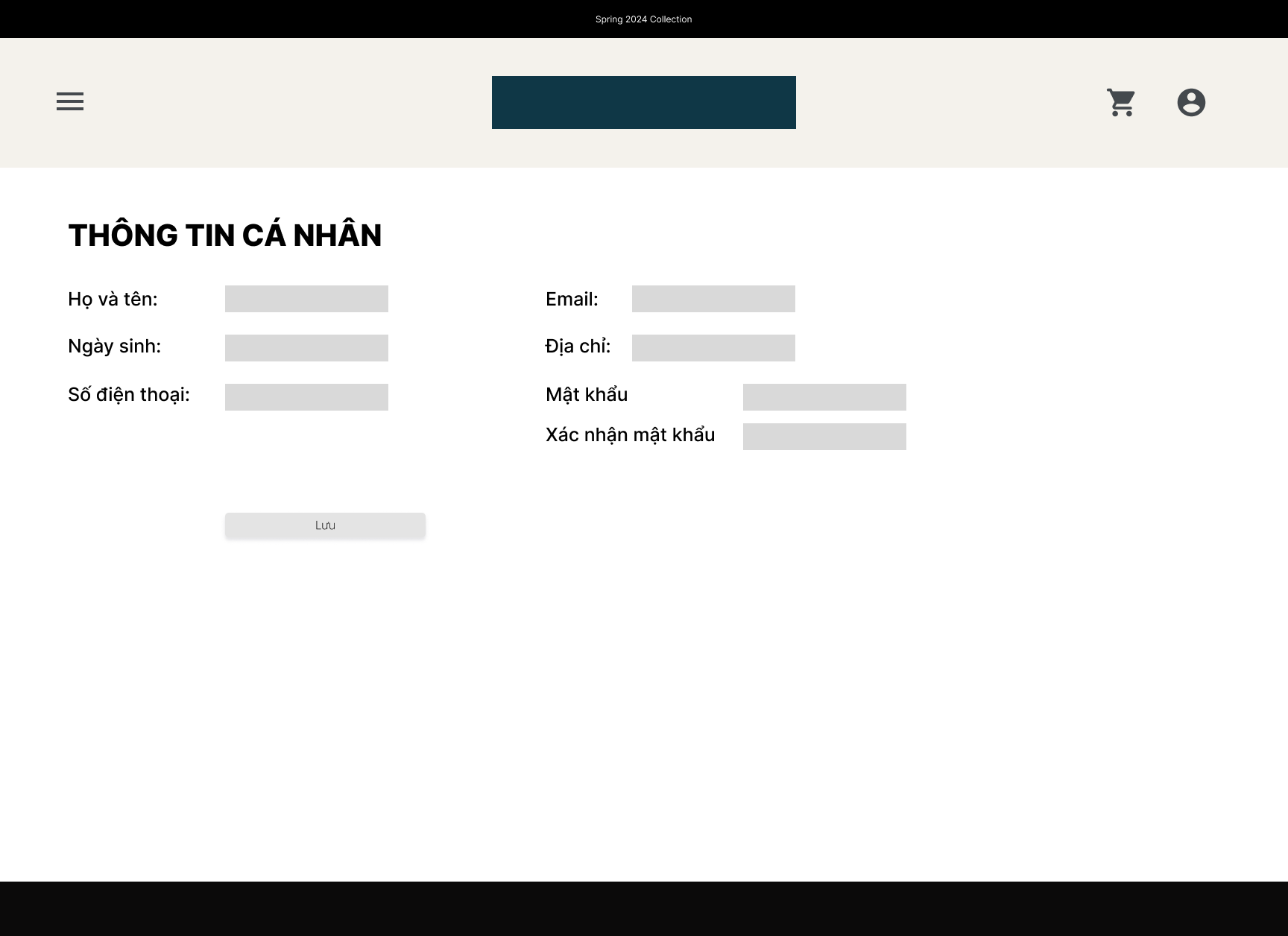
5.1 Website



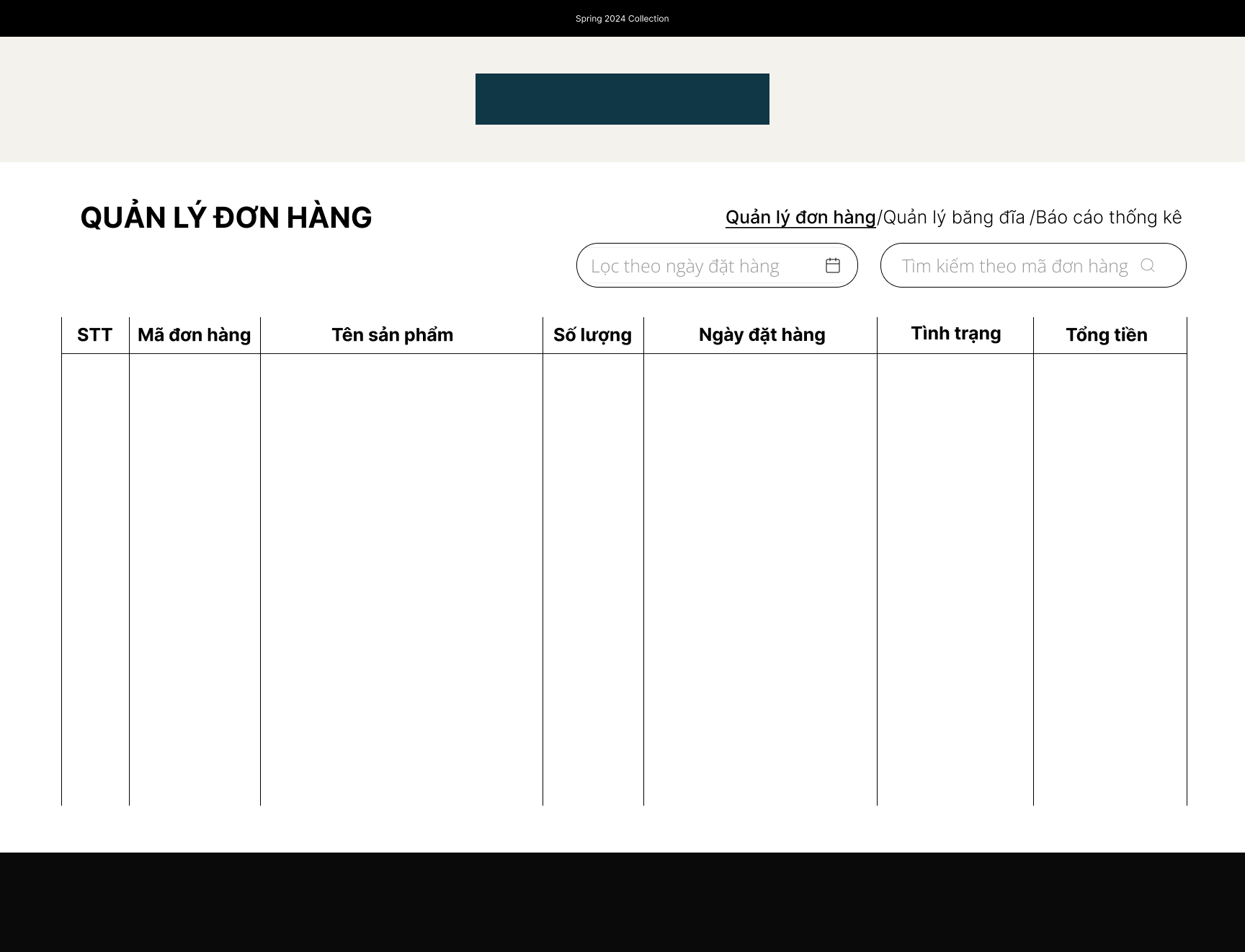


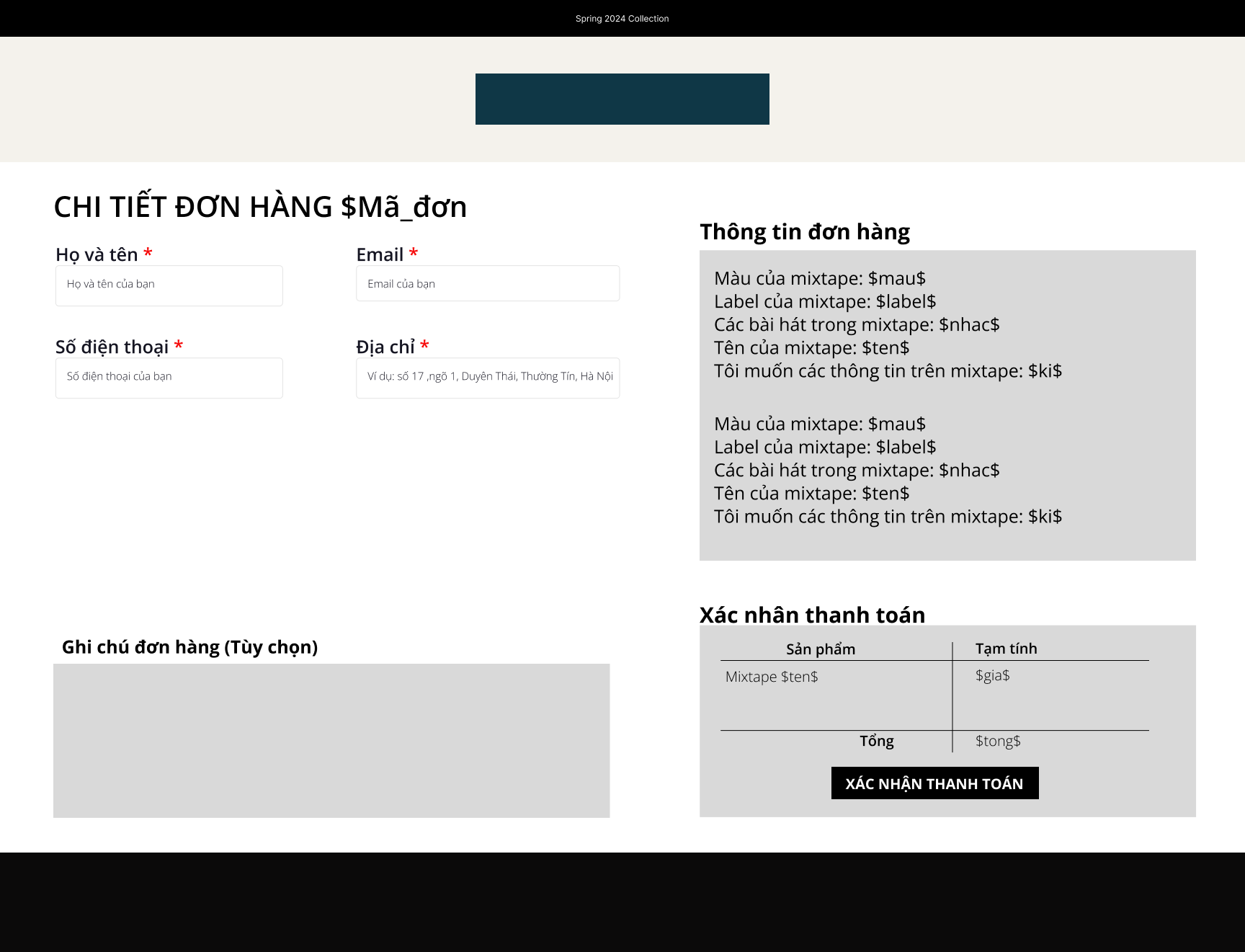


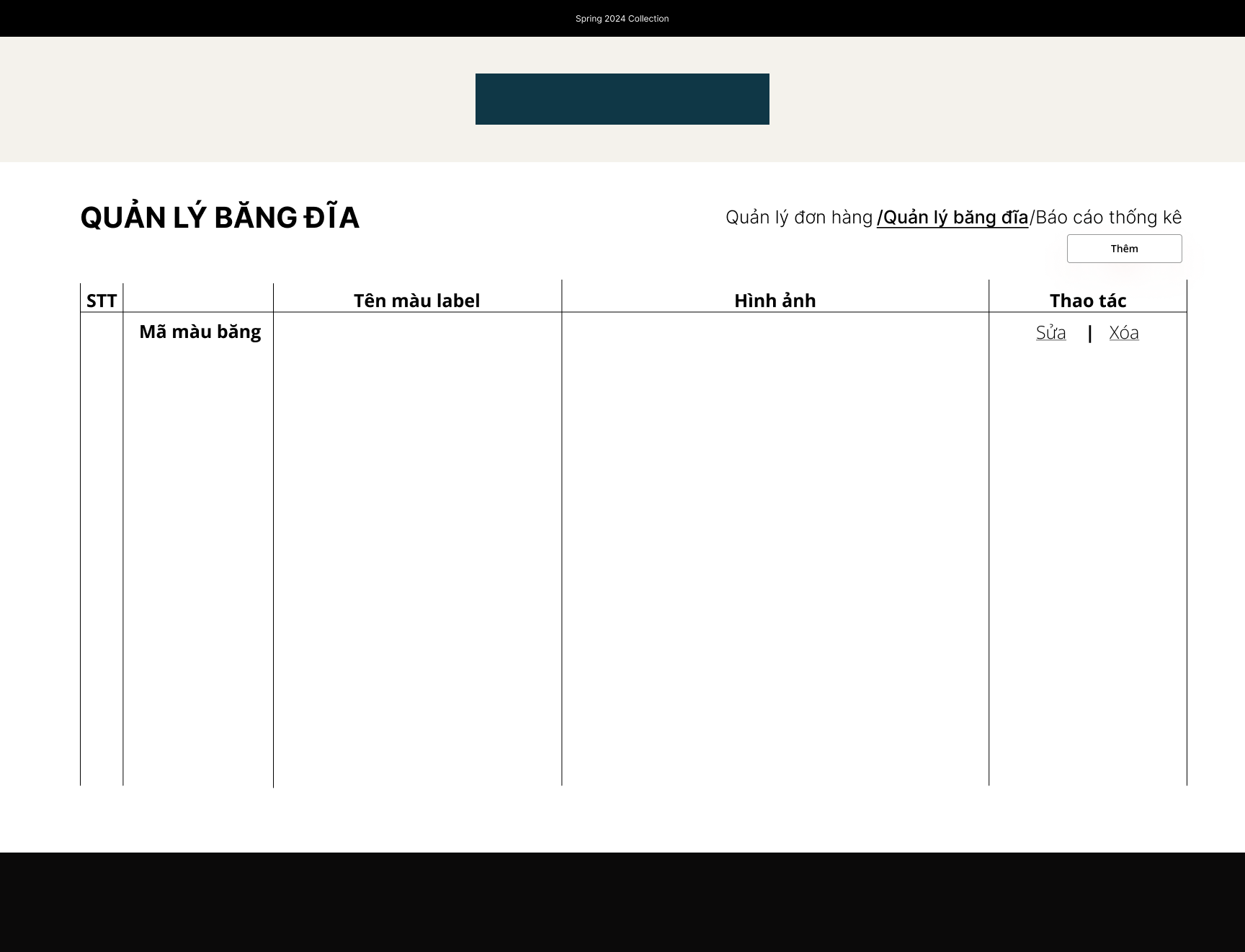


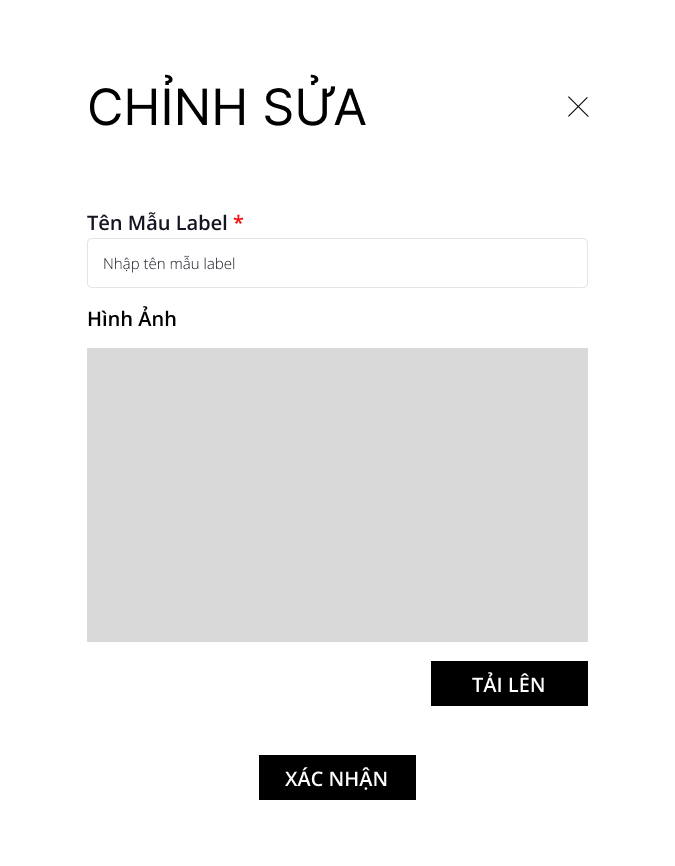


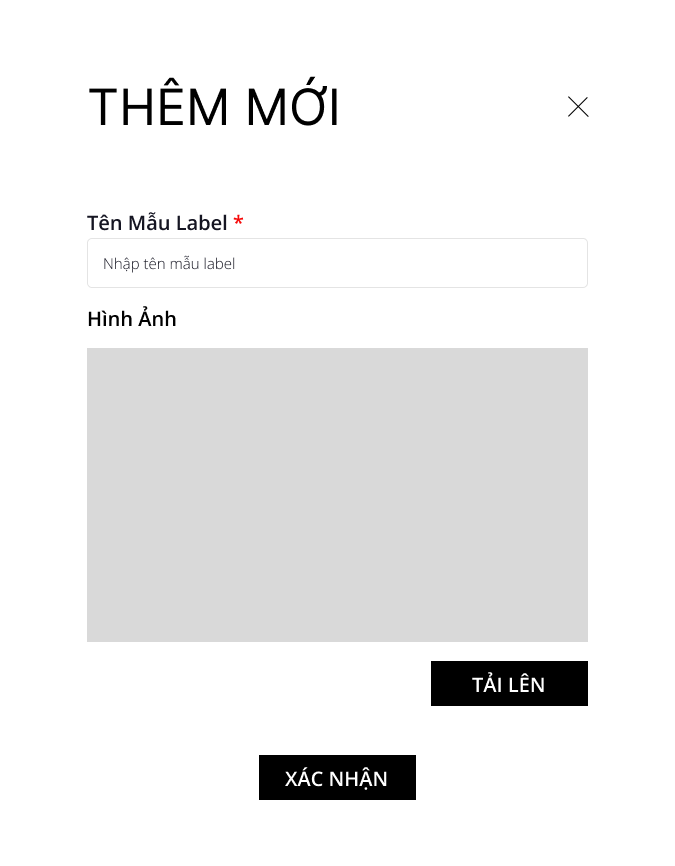
5.2 Admin











 B. TRÌNH BÀY CODE

# **I Code Backend: Ngôn ngữ sử dụng Spring Boots Restful JPA**

## **Config**

### MapperConfiguration

/\*\*

\* Lớp cấu hình cho ModelMapper trong ứng dụng Spring Boot.

\*/

@Configuration

public class MapperConfiguration {

/\*\*

\* Tạo và cấu hình bean ModelMapper.

\* @return một đối tượng ModelMapper mới.

\*/

@Bean

public ModelMapper modelMapper() {

return new ModelMapper();

}

}

### SecurityConfig

/\*\*

\* Lớp cấu hình bảo mật cho ứng dụng Spring Boot.

\*/

@Configuration

@RequiredArgsConstructor

public class SecurityConfig {

private final UserRepository userRepository;

/\*\*

\* Bean UserDetailsService để tải thông tin chi tiết người dùng từ cơ sở dữ liệu.

\* @return UserDetailsService triển khai.

\*/

@Bean

public UserDetailsService userDetailsService() {

return phoneNumber -> userRepository

.findByPhoneNumber(phoneNumber)

.orElseThrow(() ->

new UsernameNotFoundException(

"Không thể tìm thấy người dùng với số điện thoại = " + phoneNumber));

}

/\*\*

\* Bean PasswordEncoder để mã hóa mật khẩu người dùng.

\* @return PasswordEncoder triển khai BCrypt.

\*/

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

/\*\*

\* Bean AuthenticationProvider để cấu hình nhà cung cấp xác thực.

\* @return DaoAuthenticationProvider.

\*/

@Bean

public AuthenticationProvider authenticationProvider() {

DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();

authProvider.setUserDetailsService(userDetailsService());

authProvider.setPasswordEncoder(passwordEncoder());

return authProvider;

}

/\*\*

\* Bean AuthenticationManager để quản lý xác thực.

\* @param config Cấu hình xác thực.

\* @return AuthenticationManager.

\* @throws Exception nếu có lỗi trong quá trình lấy AuthenticationManager.

\*/

@Bean

public AuthenticationManager authenticationManager(

AuthenticationConfiguration config

) throws Exception {

return config.getAuthenticationManager();

}

}

### WebSecurityConfiguration

@Configuration

@EnableWebSecurity(debug = true) // Kích hoạt bảo mật web với chế độ debug bật

@EnableWebMvc // Kích hoạt hỗ trợ MVC

@RequiredArgsConstructor // Tự động tạo constructor cho các biến final

public class WebSecurityConfig {

private final JwtTokenFilter jwtTokenFilter; // Bộ lọc JWT token

private final CustomOAuth2UserService oauth2UserService; // Dịch vụ người dùng OAuth2 tùy chỉnh

@Value("${api.prefix}")

private String apiPrefix; // Tiền tố API được cấu hình trong application.properties

@Bean

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http

.addFilterBefore(jwtTokenFilter, UsernamePasswordAuthenticationFilter.class) // Thêm bộ lọc JWT trước bộ lọc xác thực

.authorizeHttpRequests(requests -> {

requests

.requestMatchers(

String.format("%s/users/register", apiPrefix), // Định nghĩa các endpoint không yêu cầu xác thực

String.format("%s/users/login", apiPrefix),

"/api-docs",

"/api-docs/\*\*",

"/swagger-resources",

"/swagger-resources/\*\*",

"/configuration/ui",

"/configuration/security",

"/swagger-ui/\*\*",

"/swagger-ui.html",

"/webjars/swagger-ui/\*\*",

"/swagger-ui/index.html"

).permitAll()

.anyRequest().permitAll(); // Cho phép tất cả các yêu cầu khác

})

.csrf(AbstractHttpConfigurer::disable); // Tắt bảo vệ CSRF

http.securityMatcher(String.valueOf(EndpointRequest.toAnyEndpoint())); // Áp dụng bảo mật cho tất cả các endpoint

return http.build(); // Xây dựng chuỗi bộ lọc bảo mật

}

}

## component

### JwtTokenUtlis

/\*\*

\* Lớp tiện ích để làm việc với JWT tokens.

\*/

@Component

@RequiredArgsConstructor

public class JwtTokenUtils {

@Value("${jwt.expiration}")

private int expiration; // Thời gian hết hạn của JWT (lưu trong biến môi trường)

@Value("${jwt.expiration-refresh-token}")

private int expirationRefreshToken; // Thời gian hết hạn của refresh token (lưu trong biến môi trường)

@Value("${jwt.secretKey}")

private String secretKey; // Khóa bí mật để ký JWT (lưu trong biến môi trường)

private static final Logger logger = LoggerFactory.getLogger(JwtTokenUtils.class);

private final TokenRepository tokenRepository; // Repository để làm việc với token trong cơ sở dữ liệu

/\*\*

\* Tạo JWT token cho người dùng.

\* @param user Thông tin người dùng

\* @return JWT token

\* @throws Exception nếu có lỗi khi tạo token

\*/

public String generateToken(User user) throws Exception {

Map<String, Object> claims = new HashMap<>(); // Các thông tin bổ sung đưa vào token

claims.put("phoneNumber", user.getPhoneNumber());

claims.put("userId", user.getId());

try {

String token = Jwts.builder()

.setClaims(claims) // Thiết lập các claims

.setSubject(user.getPhoneNumber()) // Thiết lập subject là số điện thoại người dùng

.setExpiration(new Date(System.currentTimeMillis() + expiration \* 1000L)) // Thiết lập thời gian hết hạn

.signWith(getSignInKey(), SignatureAlgorithm.HS256) // Ký token với khóa bí mật

.compact();

return token;

} catch (Exception e) {

throw new InvalidParamException("Cannot create jwt token, error: " + e.getMessage());

}

}

/\*\*

\* Lấy khóa ký từ khóa bí mật.

\* @return Key để ký JWT

\*/

private Key getSignInKey() {

byte[] bytes = Decoders.BASE64.decode(secretKey);

return Keys.hmacShaKeyFor(bytes);

}

/\*\*

\* Trích xuất tất cả các claims từ JWT token.

\* @param token JWT token

\* @return Claims chứa thông tin của token

\*/

private Claims extractAllClaims(String token) {

return Jwts.parserBuilder()

.setSigningKey(getSignInKey())

.build()

.parseClaimsJws(token)

.getBody();

}

/\*\*

\* Trích xuất claim cụ thể từ JWT token.

\* @param token JWT token

\* @param claimsResolver Hàm để trích xuất claim cụ thể

\* @param <T> Loại của claim

\* @return Claim cụ thể

\*/

public <T> T extractClaim(String token, Function<Claims, T> claimsResolver) {

final Claims claims = this.extractAllClaims(token);

return claimsResolver.apply(claims);

}

/\*\*

\* Kiểm tra token đã hết hạn hay chưa.

\* @param token JWT token

\* @return true nếu token đã hết hạn, ngược lại false

\*/

public boolean isTokenExpired(String token) {

Date expirationDate = this.extractClaim(token, Claims::getExpiration);

return expirationDate.before(new Date());

}

/\*\*

\* Trích xuất số điện thoại từ JWT token.

\* @param token JWT token

\* @return Số điện thoại của người dùng

\*/

public String extractPhoneNumber(String token) {

return extractClaim(token, Claims::getSubject);

}

/\*\*

\* Xác thực tính hợp lệ của JWT token.

\* @param token JWT token

\* @param userDetails Thông tin chi tiết của người dùng

\* @return true nếu token hợp lệ, ngược lại false

\*/

public boolean validateToken(String token, User userDetails) {

try {

String phoneNumber = extractPhoneNumber(token);

Token existingToken = tokenRepository.findByToken(token);

if (existingToken == null || existingToken.isRevoked() || !userDetails.isActive()) {

return false;

}

return (phoneNumber.equals(userDetails.getUsername())) && !isTokenExpired(token);

} catch (MalformedJwtException e) {

logger.error("Invalid JWT token: {}", e.getMessage());

} catch (ExpiredJwtException e) {

logger.error("JWT token is expired: {}", e.getMessage());

} catch (UnsupportedJwtException e) {

logger.error("JWT token is unsupported: {}", e.getMessage());

} catch (IllegalArgumentException e) {

logger.error("JWT claims string is empty: {}", e.getMessage());

}

return false;

}

}

### LocalizationUtil

/\*\*

\* Lớp tiện ích để lấy các thông điệp được bản địa hóa (localized messages).

\*/

@RequiredArgsConstructor

@Component

public class LocalizationUtils {

private final MessageSource messageSource; // Đối tượng MessageSource để lấy các thông điệp

private final LocaleResolver localeResolver; // Đối tượng LocaleResolver để xác định Locale của yêu cầu hiện tại

/\*\*

\* Lấy thông điệp được bản địa hóa dựa trên khóa thông điệp và tham số.

\* @param messageKey Khóa của thông điệp cần lấy

\* @param params Tham số để chèn vào thông điệp (nếu có)

\* @return Thông điệp được bản địa hóa

\*/

public String getLocalizedMessage(String messageKey, Object... params) {

HttpServletRequest request = RequestContextUtils.getRequestAttributes().getRequest(); // Lấy yêu cầu hiện tại

Locale locale = localeResolver.resolveLocale(request); // Xác định Locale từ yêu cầu

return messageSource.getMessage(messageKey, params, locale); // Lấy thông điệp được bản địa hóa

}

}

### SecurityUtlis

/\*\*

\* Lớp tiện ích để làm việc với bảo mật, cung cấp các phương thức để lấy thông tin người dùng đã đăng nhập.

\*/

@Component

public class SecurityUtils {

/\*\*

\* Lấy người dùng hiện tại đã đăng nhập.

\* @return Đối tượng User của người dùng đã đăng nhập hoặc null nếu không có người dùng nào đăng nhập

\*/

public User getLoggedInUser() {

Authentication authentication = SecurityContextHolder.getContext().getAuthentication(); // Lấy thông tin xác thực từ SecurityContext

if (authentication != null && authentication.getPrincipal() instanceof User) { // Kiểm tra xem có thông tin xác thực và principal là đối tượng User

return (User) authentication.getPrincipal(); // Trả về đối tượng User

}

return null; // Trả về null nếu không có người dùng nào đăng nhập

}

}

### YourCorsFilter

/\*\*

\* Lớp lọc để xử lý CORS (Cross-Origin Resource Sharing).

\* Lớp này sẽ chạy trước tất cả các bộ lọc khác vì nó được thiết lập với độ ưu tiên cao nhất.

\*/

@Component

@Order(Ordered.HIGHEST\_PRECEDENCE)

public class YourCorsFilter extends OncePerRequestFilter {

/\*\*

\* Phương thức xử lý lọc yêu cầu HTTP và thêm các header cần thiết để xử lý CORS.

\* @param request Yêu cầu HTTP

\* @param response Phản hồi HTTP

\* @param filterChain Chuỗi lọc để chuyển tiếp yêu cầu và phản hồi

\* @throws ServletException Nếu có lỗi xảy ra trong quá trình xử lý lọc

\* @throws IOException Nếu có lỗi I/O xảy ra trong quá trình xử lý lọc

\*/

@Override

protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain) throws ServletException, IOException {

// Thiết lập các header CORS

response.setHeader("Access-Control-Allow-Origin", "\*"); // Cho phép tất cả các nguồn gốc (origin)

response.setHeader("Access-Control-Allow-Methods", "GET, POST, PUT, DELETE, OPTIONS"); // Cho phép các phương thức HTTP

response.setHeader("Access-Control-Max-Age", "3600"); // Thời gian mà trình duyệt có thể cache phản hồi của preflight request (trong giây)

response.setHeader("Access-Control-Allow-Headers", "authorization, content-type, xsrf-token"); // Các header cho phép trong yêu cầu

response.addHeader("Access-Control-Expose-Headers", "xsrf-token"); // Header có thể được client truy cập

// Xử lý yêu cầu với phương thức OPTIONS (preflight request)

if ("OPTIONS".equals(request.getMethod())) {

response.setStatus(HttpServletResponse.SC\_OK); // Trả về trạng thái OK cho yêu cầu OPTIONS

} else {

filterChain.doFilter(request, response); // Chuyển tiếp yêu cầu và phản hồi đến các bộ lọc khác

}

}

}

## filter: JwtTokenFilter

/\*\*

\* Lớp lọc JWT để xử lý xác thực người dùng dựa trên token JWT.

\*/

@Component

@RequiredArgsConstructor

public class JwtTokenFilter extends OncePerRequestFilter {

@Value("${api.prefix}")

private String apiPrefix;

private final UserDetailsService userDetailsService;

private final JwtTokenUtils jwtTokenUtil;

/\*\*

\* Phương thức lọc yêu cầu HTTP và xử lý xác thực người dùng dựa trên token JWT.

\* @param request Yêu cầu HTTP

\* @param response Phản hồi HTTP

\* @param filterChain Chuỗi lọc để chuyển tiếp yêu cầu và phản hồi

\* @throws ServletException Nếu có lỗi xảy ra trong quá trình xử lý lọc

\* @throws IOException Nếu có lỗi I/O xảy ra trong quá trình xử lý lọc

\*/

@Override

protected void doFilterInternal(@NonNull HttpServletRequest request,

@NonNull HttpServletResponse response,

@NonNull FilterChain filterChain)

throws ServletException, IOException {

try {

// Bỏ qua token cho các yêu cầu nhất định

if (isBypassToken(request)) {

filterChain.doFilter(request, response);

return;

}

final String authHeader = request.getHeader("Authorization");

if (authHeader == null || !authHeader.startsWith("Bearer ")) {

response.sendError(

HttpServletResponse.SC\_UNAUTHORIZED,

"authHeader null or not started with Bearer");

return;

}

final String token = authHeader.substring(7);

final String phoneNumber = jwtTokenUtil.extractPhoneNumber(token);

if (phoneNumber != null && SecurityContextHolder.getContext().getAuthentication() == null) {

User userDetails = (User) userDetailsService.loadUserByUsername(phoneNumber);

if (jwtTokenUtil.validateToken(token, userDetails)) {

UsernamePasswordAuthenticationToken authenticationToken =

new UsernamePasswordAuthenticationToken(

userDetails,

null,

userDetails.getAuthorities()

);

authenticationToken.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));

SecurityContextHolder.getContext().setAuthentication(authenticationToken);

}

}

filterChain.doFilter(request, response);

} catch (Exception e) {

response.setStatus(HttpServletResponse.SC\_UNAUTHORIZED);

response.getWriter().write(e.getMessage());

}

}

/\*\*

\* Kiểm tra xem yêu cầu hiện tại có nằm trong danh sách các yêu cầu bỏ qua token hay không.

\* @param request Yêu cầu HTTP

\* @return true nếu yêu cầu cần bỏ qua token, ngược lại false

\*/

private boolean isBypassToken(@NonNull HttpServletRequest request) {

final List<Pair<String, String>> bypassTokens = Arrays.asList(

// Healthcheck request, no JWT token required

Pair.of(String.format("%s/healthcheck/health", apiPrefix), "GET"),

Pair.of(String.format("%s/actuator/\*\*", apiPrefix), "GET"),

Pair.of(String.format("%s/users/register", apiPrefix), "POST"),

Pair.of(String.format("%s/users/login", apiPrefix), "POST"),

Pair.of(String.format("%s/users/refreshToken", apiPrefix), "POST"),

Pair.of(String.format("%s/users", apiPrefix), "GET"),

Pair.of(String.format("%s/users/\*\*", apiPrefix), "PUT"),

// Swagger

Pair.of("/api-docs", "GET"),

Pair.of("/api-docs/\*\*", "GET"),

Pair.of("/swagger-resources", "GET"),

Pair.of("/swagger-resources/\*\*", "GET"),

Pair.of("/configuration/ui", "GET"),

Pair.of("/configuration/security", "GET"),

Pair.of("/swagger-ui/\*\*", "GET"),

Pair.of("/swagger-ui.html", "GET"),

Pair.of("/swagger-ui/index.html", "GET")

);

String requestPath = request.getServletPath();

String requestMethod = request.getMethod();

for (Pair<String, String> token : bypassTokens) {

String path = token.getFirst();

String method = token.getSecond();

// Kiểm tra xem đường dẫn và phương thức yêu cầu có khớp với bất kỳ cặp nào trong danh sách bypassTokens không

if (requestPath.matches(path.replace("\*\*", ".\*"))

&& requestMethod.equalsIgnoreCase(method)) {

return true;

}

}

return false;

}

}

## **Entity**

### BaseEntity

public class BaseEntity {

@Column(name = "created\_at")

private LocalDateTime createdAt;

@Column(name = "updated\_at")

private LocalDateTime updatedAt;

@PrePersist

protected void onCreate() {

createdAt = LocalDateTime.now();

updatedAt = LocalDateTime.now();

}

@PreUpdate

protected void onUpdate() {

updatedAt = LocalDateTime.now();

}

}

### MauBang

public class MauBang extends BaseEntity{

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "name")

private String name;

private int price;

@Column(name = "thumbnail", length = 300)

private String thumbnail;

@OneToOne(fetch = FetchType.LAZY, cascade = CascadeType.ALL)

@JoinColumn(name = "img\_id")

private MauBangImage img;

}

### MauBangImage

public class MauBangImage {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@OneToOne

@JsonIgnore

@JoinColumn(name = "maubang\_id")

private MauBang mauBang;

@Column(name = "image\_url", length = 300)

@JsonProperty("image\_url")

private String imageUrl;

}

### Order

public class Order {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@ManyToOne

@JoinColumn(name = "user\_id")

private User user;

@Column(name = "fullname", length = 100)

private String fullName;

@Column(name = "email", length = 100)

private String email;

@Column(name = "phone\_number",nullable = false, length = 100)

private String phoneNumber;

@Column(name = "address", length = 100)

private String address;

@Column(name = "note", length = 100)

private String note;

@Column(name="order\_date")

private LocalDate orderDate;

@Column(name = "status")

private String status;

@Column(name = "total\_money")

private Float totalMoney;

@Column(name = "shipping\_method")

private String shippingMethod;

@Column(name = "shipping\_address")

private String shippingAddress;

@Column(name = "shipping\_date")

private LocalDate shippingDate;

@Column(name = "tracking\_number")

private String trackingNumber;

@Column(name = "payment\_method")

private String paymentMethod;

@Column(name = "active")

private Boolean active;//thuộc về admin

@OneToMany(mappedBy = "order", cascade = CascadeType.ALL, fetch = FetchType.LAZY)

@JsonManagedReference

private List<OrderDetail> orderDetails;

}

### OrderDetail

public class OrderDetail {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@ManyToOne

@JoinColumn(name = "order\_id")

private Order order;

@ManyToOne

@JoinColumn(name = "product\_id")

private Product product;

@Column(name = "number\_of\_products", nullable = false)

private int numberOfProducts;

@Column(name = "price", nullable = false)

private int price;

@Column(name = "total\_money", nullable = false)

private int totalMoney;

}

### OrderStatus

public class OrderStatus {

public static final String PENDING = "pending";

public static final String PROCESSING = "processing";

public static final String SHIPPED = "shipped";

public static final String DELIVERED = "delivered";

public static final String CANCELLED = "cancelled";

}

### Product

public class Product extends BaseEntity{

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "name", nullable = false, length = 350)

private String name;

private int price;

private int otpName;

@OneToOne

@JoinColumn(name = "maubang\_id")

private MauBang mauBang;

private String matA;

private String matB;

private String bia;

}

### Role

public class Role {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "name", nullable = false)

private String name;

public static String ADMIN = "ADMIN";

public static String USER = "USER";

}

### Token

public class Token {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "token", length = 255)

private String token;

@Column(name = "refresh\_token", length = 255)

private String refreshToken;

@Column(name = "token\_type", length = 50)

private String tokenType;

@Column(name = "expiration\_date")

private LocalDateTime expirationDate;

@Column(name = "refresh\_expiration\_date")

private LocalDateTime refreshExpirationDate;

@Column(name = "is\_mobile", columnDefinition = "TINYINT(1)")

private boolean isMobile;

private boolean revoked;

private boolean expired;

@ManyToOne

@JoinColumn(name = "user\_id")

private User user;

}

### User

public class User extends BaseEntity implements UserDetails, OAuth2User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "fullname", length = 100)

private String fullName;

@Column(name = "phone\_number", length = 10, nullable = false)

private String phoneNumber;

@Column(name = "address", length = 200)

private String address;

@Column(name = "password", length = 200, nullable = false)

private String password;

@Column(name = "is\_active")

private boolean active;

@Column(name = "date\_of\_birth")

private Date dateOfBirth;

@ManyToOne

@JoinColumn(name = "role\_id")

private Role role;

@Override

public Collection<? extends GrantedAuthority> getAuthorities() {

List<SimpleGrantedAuthority> authorityList = new ArrayList<>();

authorityList.add(new SimpleGrantedAuthority( "ROLE\_"

+getRole().getName().toUpperCase()));

//authorityList.add(new SimpleGrantedAuthority("ROLE\_ADMIN"));

return authorityList;

}

@Override

public String getUsername() {

return phoneNumber;

}

@Override

public boolean isAccountNonExpired() {

return true;

}

@Override

public boolean isAccountNonLocked() {

return true;

}

@Override

public boolean isCredentialsNonExpired() {

return true;

}

@Override

public boolean isEnabled() {

return true;

}

//Login facebook

@Override

public Map<String, Object> getAttributes() {

return new HashMap<String, Object>();

}

@Override

public String getName() {

return getAttribute("name");

}

}

## **Repository**

### MauBangImageRepository

@Repository

public interface MauBangImageRepository extends JpaRepository<MauBangImage, Long> {

Optional<MauBangImage> findByMauBangId(Long mauBangId);

}

### MauBangRepository

@Repository

public interface MauBangRepository extends JpaRepository<MauBang, Long> {

boolean existsByName(String name);

Optional<MauBang> findByName(String name);

@Query("SELECT p FROM MauBang p WHERE " +

"(:keyword IS NULL OR :keyword = '' OR p.name LIKE %:keyword%)")

Page<MauBang> searchProducts

(@Param("keyword") String keyword, Pageable pageable);

@Query("SELECT p FROM MauBang p LEFT JOIN FETCH p.img WHERE p.id = :mauBangid")

Optional<MauBang> getImg(@Param("mauBangid") Long mauBangid);

@Query("SELECT p FROM MauBang p WHERE p.id IN :productIds")

List<MauBang> findProductsByIds(@Param("productIds") List<Long> productIds);

}

### OrderDetailRepository

@Repository

public interface OrderDetailRepository extends JpaRepository<OrderDetail, Long> {

List<OrderDetail> findByOrderId(Long orderId);

@Query("SELECT p FROM OrderDetail p WHERE p.id IN :productIds")

List<OrderDetail> findOrderDetailsByIds(@Param("productIds") List<Long> productIds);

}

### OrderRepository

@Repository

public interface OrderRepository extends JpaRepository<Order, Long> {

//Tìm các đơn hàng của 1 user nào đó

List<Order> findByUserId(Long userId);

@Query("SELECT o FROM Order o WHERE "+

"(:keyword IS NULL OR :keyword = '' OR " +

"o.fullName LIKE %:keyword% " +

"OR o.address LIKE %:keyword% " +

"OR o.note LIKE %:keyword% " +

"OR o.email LIKE %:keyword%)")

Page<Order> findByKeyword(@Param("keyword") String keyword, Pageable pageable);

}

### ProductRepository

@Repository

public interface ProductRepository extends JpaRepository<Product, Long> {

boolean existsByName(String name);

@Query("SELECT p FROM Product p WHERE p.id IN :productIds")

List<Product> findProductsByIds(@Param("productIds") List<Long> productIds);

}

### RoleRepository

@Repository

public interface RoleRepository extends JpaRepository<Role, Long> {

}

### TokenRepository

@Repository

public interface TokenRepository extends JpaRepository<Token, Long> {

List<Token> findByUser(User user);

Token findByToken(String token);

Token findByRefreshToken(String token);

}

### UserRepositoy

@Repository

public interface UserRepository extends JpaRepository<User, Long> {

boolean existsByPhoneNumber(String phoneNumber);

Optional<User> findByPhoneNumber(String phoneNumber);

//SELECT \* FROM users WHERE phoneNumber=?

//query command

@Query("SELECT o FROM User o WHERE o.active = true AND (:keyword IS NULL OR :keyword = '' OR " +

"o.fullName LIKE %:keyword% " +

"OR o.address LIKE %:keyword% " +

"OR o.phoneNumber LIKE %:keyword%) " +

"AND LOWER(o.role.name) = 'user'")

Page<User> findAll(@Param("keyword") String keyword, Pageable pageable);

}

## **Service**

### MauBangImageService

/\*\*

\* Lớp dịch vụ để xử lý các thao tác liên quan đến hình ảnh của mẫu bảng.

\*/

@Service

@RequiredArgsConstructor

public class MauBangImageService {

@Autowired

private MauBangImageRepository mauBangImageRepository;

/\*\*

\* Phương thức xóa hình ảnh của mẫu bảng dựa trên ID.

\* @param id ID của hình ảnh mẫu bảng cần xóa

\* @return Mẫu bảng hình ảnh đã bị xóa

\* @throws Exception Nếu có lỗi xảy ra trong quá trình xóa

\*/

@Transactional

public MauBangImage deleteProductImage(Long id) throws Exception {

// Tìm kiếm hình ảnh mẫu bảng theo ID

Optional<MauBangImage> productImage = mauBangImageRepository.findByMauBangId(id);

// Nếu không tìm thấy hình ảnh mẫu bảng, ném ngoại lệ DataNotFoundException

if(productImage.isEmpty()) {

throw new DataNotFoundException(

String.format("Cannot find mau bang image with id: %ld", id)

);

}

// Xóa hình ảnh mẫu bảng và trả về nó

mauBangImageRepository.deleteById(productImage.get().getId());

return productImage.get();

}

}

### MauBangService

@Service

@RequiredArgsConstructor

public class MauBangService {

@Autowired

private MauBangRepository mauBangRepository;

@Autowired

private MauBangImageRepository mauBangImageRepository;

private static String UPLOADS\_FOLDER = "uploads";

@Transactional

public MauBang createMauBang(MauBangDTO MauBangDTO) throws DataAccessException {

// Tạo một mẫu bảng mới từ DTO và lưu vào cơ sở dữ liệu

MauBang newMauBang = MauBang.builder()

.name(MauBangDTO.getName())

.price(MauBangDTO.getPrice())

.build();

return mauBangRepository.save(newMauBang);

}

public MauBang getMauBangById(long MauBangId) throws Exception {

// Lấy mẫu bảng từ cơ sở dữ liệu dựa trên ID

Optional<MauBang> optionalMauBang = mauBangRepository.getImg(MauBangId);

if (optionalMauBang.isPresent()) {

return optionalMauBang.get();

}

throw new DataNotFoundException("Cannot find MauBang with id =" + MauBangId);

}

@Transactional

public MauBang updateProduct(long id, MauBangDTO MauBangDTO) throws Exception {

// Cập nhật thông tin của mẫu bảng

MauBang existingMauBang = getMauBangById(id);

if (existingMauBang != null) {

// Copy các thuộc tính từ DTO -> MauBang

if (MauBangDTO.getName() != null && !MauBangDTO.getName().isEmpty()) {

existingMauBang.setName(MauBangDTO.getName());

}

return mauBangRepository.save(existingMauBang);

}

return null;

}

@Transactional

public void deleteMauBang(long id) {

// Xóa mẫu bảng từ cơ sở dữ liệu dựa trên ID

Optional<MauBang> optionalProduct = mauBangRepository.findById(id);

optionalProduct.ifPresent(mauBangRepository::delete);

}

public boolean existsByName(String name) {

// Kiểm tra xem mẫu bảng có tồn tại trong cơ sở dữ liệu không

return mauBangRepository.existsByName(name);

}

@Transactional

public MauBangImage createProductImage(Long productId, MauBangImageDTO productImageDTO) throws Exception {

// Tạo hình ảnh mới cho mẫu bảng và lưu vào cơ sở dữ liệu

MauBang existingProduct = mauBangRepository.findById(productId)

.orElseThrow(() -> new DataNotFoundException("Cannot find product with id: " + productImageDTO.getMaubangid()));

MauBangImage newProductImage = MauBangImage.builder()

.mauBang(existingProduct)

.imageUrl(productImageDTO.getImageUrl())

.build();

if (existingProduct.getThumbnail() == null) {

existingProduct.setThumbnail(newProductImage.getImageUrl());

}

mauBangRepository.save(existingProduct);

return mauBangImageRepository.save(newProductImage);

}

public List<MauBang> findProductsByIds(List<Long> productIds) {

// Tìm kiếm mẫu bảng dựa trên danh sách các ID

return mauBangRepository.findProductsByIds(productIds);

}

public void deleteFile(String filename) throws IOException {

// Xóa một file từ thư mục uploads

java.nio.file.Path uploadDir = Paths.get(UPLOADS\_FOLDER);

java.nio.file.Path filePath = uploadDir.resolve(filename);

if (Files.exists(filePath)) {

Files.delete(filePath);

} else {

throw new FileNotFoundException("File not found: " + filename);

}

}

private boolean isImageFile(MultipartFile file) {

// Kiểm tra xem file có phải là file hình ảnh không

String contentType = file.getContentType();

return contentType != null && contentType.startsWith("image/");

}

public String storeFile(MultipartFile file) throws IOException {

// Lưu trữ một file vào thư mục uploads

if (!isImageFile(file) || file.getOriginalFilename() == null) {

throw new IOException("Invalid image format");

}

String filename = StringUtils.cleanPath(Objects.requireNonNull(file.getOriginalFilename()));

String uniqueFilename = UUID.randomUUID().toString() + "\_" + System.nanoTime();

java.nio.file.Path uploadDir = Paths.get(UPLOADS\_FOLDER);

if (!Files.exists(uploadDir)) {

Files.createDirectories(uploadDir);

}

java.nio.file.Path destination = Paths.get(uploadDir.toString(), uniqueFilename);

Files.copy(file.getInputStream(), destination, StandardCopyOption.REPLACE\_EXISTING);

return uniqueFilename;

}

public Page<MauBangResponse> getAllProducts(String keyword, PageRequest pageRequest) {

// Lấy danh sách mẫu bảng dựa trên từ khóa và yêu cầu trang

Page<MauBang> productsPage = mauBangRepository.searchProducts(keyword, pageRequest);

return productsPage.map(MauBangResponse::fromProduct);

}

}

### OderDetailService

@RequiredArgsConstructor

@Service

public class OrderDetailService implements IOrderDetailService {

// Inject các repository cần thiết

private final OrderRepository orderRepository;

private final OrderDetailRepository orderDetailRepository;

private final ProductRepository productRepository;

@Override

@Transactional

public OrderDetail createOrderDetail(OrderDetailDTO orderDetailDTO) throws Exception {

// Tìm Order theo ID, nếu không tồn tại sẽ ném ra ngoại lệ

Order order = orderRepository.findById(orderDetailDTO.getOrderId())

.orElseThrow(() -> new DataNotFoundException(

"Cannot find Order with id : " + orderDetailDTO.getOrderId()));

// Tìm Product theo ID, nếu không tồn tại sẽ ném ra ngoại lệ

Product product = productRepository.findById(orderDetailDTO.getProductId())

.orElseThrow(() -> new DataNotFoundException(

"Cannot find product with id: " + orderDetailDTO.getProductId()));

// Tạo OrderDetail từ dữ liệu DTO và lưu vào cơ sở dữ liệu

OrderDetail orderDetail = OrderDetail.builder()

.order(order)

.product(product)

.numberOfProducts(orderDetailDTO.getNumberOfProducts())

.totalMoney(orderDetailDTO.getTotalMoney())

.build();

return orderDetailRepository.save(orderDetail);

}

// Các phương thức khác như update, delete và tìm kiếm OrderDetail...

}

### OrderService

@RequiredArgsConstructor

@Service

public class OrderService implements IOrderService {

// Inject các repository và ModelMapper

private final UserRepository userRepository;

private final OrderRepository orderRepository;

private final ProductRepository productRepository;

private final OrderDetailRepository orderDetailRepository;

private final ModelMapper modelMapper;

@Override

@Transactional

public Order createOrder(OrderDTO orderDTO) throws Exception {

// Tìm User theo ID, nếu không tồn tại sẽ ném ra ngoại lệ

User user = userRepository.findById(orderDTO.getUserId())

.orElseThrow(() -> new DataNotFoundException("Cannot find user with id: " + orderDTO.getUserId()));

// Tạo một đối tượng Order từ dữ liệu DTO

Order order = modelMapper.map(orderDTO, Order.class);

order.setUser(user);

order.setOrderDate(LocalDate.now());

order.setStatus(OrderStatus.PENDING);

// Kiểm tra và thiết lập shipping date

LocalDate shippingDate = Optional.ofNullable(orderDTO.getShippingDate()).orElse(LocalDate.now());

if (shippingDate.isBefore(LocalDate.now())) {

throw new DataNotFoundException("Date must be at least today !");

}

order.setShippingDate(shippingDate);

order.setActive(true);

order.setTotalMoney(orderDTO.getTotalMoney());

// Tạo danh sách các OrderDetail từ CartItemDTO

List<OrderDetail> orderDetails = new ArrayList<>();

for (CartItemDTO cartItemDTO : orderDTO.getCartItems()) {

Long productId = cartItemDTO.getProductId();

int quantity = cartItemDTO.getQuantity();

// Tìm thông tin sản phẩm từ cơ sở dữ liệu

Product product = productRepository.findById(productId)

.orElseThrow(() -> new DataNotFoundException("Product not found with id: " + productId));

// Tạo OrderDetail từ CartItemDTO

OrderDetail orderDetail = new OrderDetail();

orderDetail.setOrder(order);

orderDetail.setProduct(product);

orderDetail.setNumberOfProducts(quantity);

orderDetail.setPrice(product.getPrice());

orderDetails.add(orderDetail);

}

// Lưu danh sách OrderDetail vào cơ sở dữ liệu

orderDetailRepository.saveAll(orderDetails);

// Lưu Order vào cơ sở dữ liệu

return orderRepository.save(order);

}

@Override

@Transactional

public Order updateOrderWithDetails(OrderWithDetailsDTO orderWithDetailsDTO) {

// Mapping từ OrderWithDetailsDTO sang Order

Order order = modelMapper.map(orderWithDetailsDTO, Order.class);

Order savedOrder = orderRepository.save(order);

// Cập nhật các OrderDetail cho Order

for (OrderDetailDTO orderDetailDTO : orderWithDetailsDTO.getOrderDetailDTOS()) {

// Cập nhật order cho mỗi order detail (chưa implement)

}

// Lưu hoặc cập nhật OrderDetail

List<OrderDetail> savedOrderDetails = orderDetailRepository.saveAll(order.getOrderDetails());

savedOrder.setOrderDetails(savedOrderDetails);

return savedOrder;

}

// Các phương thức khác như update, delete, find...

}

### ProductService

package com.project.webbackend.Service.product;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.project.webbackend.Entity.MauBang;

import com.project.webbackend.Entity.Product;

import com.project.webbackend.Repository.MauBangRepository;

import com.project.webbackend.Repository.ProductRepository;

import com.project.webbackend.dtos.ProductDTO;

import com.project.webbackend.exceptions.DataNotFoundException;

import java.util.List;

import java.util.Optional;

@Service

@RequiredArgsConstructor

public class ProductService{

private final ProductRepository productRepository;

private final MauBangRepository mauBangRepository;

// Phương thức tạo mới một sản phẩm

@Transactional

public Product createProduct(ProductDTO productDTO) throws DataNotFoundException {

// Tìm mẫu bảng tồn tại dựa trên ID trong ProductDTO

MauBang existingMauBang = mauBangRepository

.findById(productDTO.getMaubangid())

.orElseThrow(() ->

new DataNotFoundException(

"Cannot find mau bang with id: "+productDTO.getMaubangid()));

// Tạo một sản phẩm mới từ dữ liệu trong ProductDTO và mẫu bảng tìm được

Product newProduct = Product.builder()

.name(productDTO.getName())

.price(productDTO.getPrice())

.otpName(productDTO.getOtpName())

.matA(productDTO.getMatA())

.matB(productDTO.getMatB())

.bia(existingMauBang.getThumbnail())

.mauBang(existingMauBang)

.build();

// Lưu sản phẩm mới vào cơ sở dữ liệu và trả về

return productRepository.save(newProduct);

}

// Phương thức lấy thông tin sản phẩm dựa trên ID

public Product getProductById(long ProductId) throws Exception {

Optional<Product> optional = productRepository.findById(ProductId);

if(optional.isPresent()) {

return optional.get();

}

throw new DataNotFoundException("Cannot find Product with id =" + ProductId);

}

// Phương thức tìm kiếm danh sách sản phẩm dựa trên danh sách ID

public List<Product> findProductsByIds(List<Long> productIds) {

return productRepository.findProductsByIds(productIds);

}

// Phương thức kiểm tra sự tồn tại của sản phẩm dựa trên tên

public boolean existsByName(String name) {

return productRepository.existsByName(name);

}

}

### RoleService

@Service

@RequiredArgsConstructor

public class RoleService implements IRoleService{

private final RoleRepository roleRepository;

@Override

public List<Role> getAllRoles() {

return roleRepository.findAll();

}

}

### TokenService

@Service

@RequiredArgsConstructor

public class TokenService implements ITokenService{

private static final int MAX\_TOKENS = 3;

@Value("${jwt.expiration}")

private int expiration; //save to an environment variable

@Value("${jwt.expiration-refresh-token}")

private int expirationRefreshToken;

private final TokenRepository tokenRepository;

private final JwtTokenUtils jwtTokenUtil;

// Phương thức làm mới token

@Transactional

@Override

public Token refreshToken(String refreshToken, User user) throws Exception{

// Tìm token hiện tại dựa trên refreshToken

Token existingToken = tokenRepository.findByRefreshToken(refreshToken);

if(existingToken == null) {

throw new DataNotFoundException("Refresh token does not exist");

}

// Kiểm tra xem token đã hết hạn chưa

if(existingToken.getRefreshExpirationDate().compareTo(LocalDateTime.now()) < 0){

tokenRepository.delete(existingToken);

throw new ExpiredTokenException("Refresh token is expired");

}

// Tạo mới token và refreshToken

String token = jwtTokenUtil.generateToken(user);

LocalDateTime expirationDateTime = LocalDateTime.now().plusSeconds(expiration);

existingToken.setExpirationDate(expirationDateTime);

existingToken.setToken(token);

existingToken.setRefreshToken(UUID.randomUUID().toString());

existingToken.setRefreshExpirationDate(LocalDateTime.now().plusSeconds(expirationRefreshToken));

return existingToken;

}

// Phương thức thêm mới token

@Transactional

@Override

public Token addToken(User user,String token, boolean isMobileDevice) {

// Lấy danh sách token của người dùng

List<Token> userTokens = tokenRepository.findByUser(user);

int tokenCount = userTokens.size();

// Nếu số lượng token vượt quá MAX\_TOKENS, xóa một token cũ

if (tokenCount >= MAX\_TOKENS) {

// Kiểm tra xem có token nào không phải là thiết bị di động không

boolean hasNonMobileToken = !userTokens.stream().allMatch(Token::isMobile);

Token tokenToDelete;

if (hasNonMobileToken) {

tokenToDelete = userTokens.stream()

.filter(userToken -> !userToken.isMobile())

.findFirst()

.orElse(userTokens.get(0));

} else {

// Nếu tất cả token đều là thiết bị di động, xóa token đầu tiên

tokenToDelete = userTokens.get(0);

}

tokenRepository.delete(tokenToDelete);

}

long expirationInSeconds = expiration;

LocalDateTime expirationDateTime = LocalDateTime.now().plusSeconds(expirationInSeconds);

// Tạo mới một token cho người dùng

Token newToken = Token.builder()

.user(user)

.token(token)

.revoked(false)

.expired(false)

.tokenType("Bearer")

.expirationDate(expirationDateTime)

.isMobile(isMobileDevice)

.build();

newToken.setRefreshToken(UUID.randomUUID().toString());

newToken.setRefreshExpirationDate(LocalDateTime.now().plusSeconds(expirationRefreshToken));

tokenRepository.save(newToken);

return newToken;

}

}

### UserService

@RequiredArgsConstructor

@Service

public class UserService implements IUserService{

private final UserRepository userRepository;

private final RoleRepository roleRepository;

private final TokenRepository tokenRepository;

private final PasswordEncoder passwordEncoder;

private final JwtTokenUtils jwtTokenUtil;

private final AuthenticationManager authenticationManager;

private final LocalizationUtils localizationUtils;

@Override

@Transactional

public User createUser(UserDTO userDTO) throws Exception {

//register user

String phoneNumber = userDTO.getPhoneNumber();

// Kiểm tra xem số điện thoại đã tồn tại hay chưa

if(userRepository.existsByPhoneNumber(phoneNumber)) {

throw new DataIntegrityViolationException("Số điện thoại đã tồn tại");

}

Role role =roleRepository.findById(userDTO.getRoleId())

.orElseThrow(() -> new DataNotFoundException(

localizationUtils.getLocalizedMessage(MessageKeys.ROLE\_DOES\_NOT\_EXISTS)));

if(role.getName().toUpperCase().equals(Role.ADMIN)) {

throw new PermissionDenyException("Không được phép đăng ký tài khoản Admin");

}

//convert from userDTO => user

User newUser = User.builder()

.fullName(userDTO.getFullName())

.phoneNumber(userDTO.getPhoneNumber())

.password(userDTO.getPassword())

.address(userDTO.getAddress())

.dateOfBirth(userDTO.getDateOfBirth())

.active(true)

.build();

String password = userDTO.getPassword();

String encodedPassword = passwordEncoder.encode(password);

newUser.setPassword(encodedPassword);

newUser.setRole(role);

return userRepository.save(newUser);

}

@Override

public String login(

String phoneNumber,

String password,

Long roleId

) throws Exception {

Optional<User> optionalUser = userRepository.findByPhoneNumber(phoneNumber);

if(optionalUser.isEmpty()) {

throw new DataNotFoundException(localizationUtils.getLocalizedMessage(MessageKeys.WRONG\_PHONE\_PASSWORD));

}

//return optionalUser.get();//muốn trả JWT token ?

User existingUser = optionalUser.get();

//check password

/\*

Optional<Role> optionalRole = roleRepository.findById(roleId);

if(optionalRole.isEmpty() || !roleId.equals(existingUser.getRole().getId())) {

throw new DataNotFoundException(localizationUtils.getLocalizedMessage(MessageKeys.ROLE\_DOES\_NOT\_EXISTS));

}

\*/

if(!optionalUser.get().isActive()) {

throw new DataNotFoundException(localizationUtils.getLocalizedMessage(MessageKeys.USER\_IS\_LOCKED));

}

UsernamePasswordAuthenticationToken authenticationToken = new UsernamePasswordAuthenticationToken(

phoneNumber, password,

existingUser.getAuthorities()

);

//authenticate with Java Spring security

authenticationManager.authenticate(authenticationToken);

return jwtTokenUtil.generateToken(existingUser);

}

@Transactional

@Override

public User updateUser(Long userId, UpdateUserDTO updatedUserDTO) throws Exception {

// Find the existing user by userId

User existingUser = userRepository.findById(userId)

.orElseThrow(() -> new DataNotFoundException("User not found"));

// Check if the phone number is being changed and if it already exists for another user

String newPhoneNumber = updatedUserDTO.getPhoneNumber();

if (!existingUser.getPhoneNumber().equals(newPhoneNumber) &&

userRepository.existsByPhoneNumber(newPhoneNumber)) {

throw new DataIntegrityViolationException("Phone number already exists");

}

// Update user information based on the DTO

if (updatedUserDTO.getFullName() != null) {

existingUser.setFullName(updatedUserDTO.getFullName());

}

if (newPhoneNumber != null) {

existingUser.setPhoneNumber(newPhoneNumber);

}

if (updatedUserDTO.getAddress() != null) {

existingUser.setAddress(updatedUserDTO.getAddress());

}

if (updatedUserDTO.getDateOfBirth() != null) {

existingUser.setDateOfBirth(updatedUserDTO.getDateOfBirth());

}

// Update the password if it is provided in the DTO

if (updatedUserDTO.getPassword() != null

&& !updatedUserDTO.getPassword().isEmpty()) {

if(!updatedUserDTO.getPassword().equals(updatedUserDTO.getRetypePassword())) {

throw new DataNotFoundException("Password and retype password not the same");

}

String newPassword = updatedUserDTO.getPassword();

String encodedPassword = passwordEncoder.encode(newPassword);

existingUser.setPassword(encodedPassword);

}

//existingUser.setRole(updatedRole);

// Save the updated user

return userRepository.save(existingUser);

}

@Override

public User getUserDetailsFromToken(String token) throws Exception {

if(jwtTokenUtil.isTokenExpired(token)) {

throw new ExpiredTokenException("Token is expired");

}

String phoneNumber = jwtTokenUtil.extractPhoneNumber(token);

Optional<User> user = userRepository.findByPhoneNumber(phoneNumber);

if (user.isPresent()) {

return user.get();

} else {

throw new Exception("User not found");

}

}

@Override

public User getUserDetailsFromRefreshToken(String refreshToken) throws Exception {

Token existingToken = tokenRepository.findByRefreshToken(refreshToken);

return getUserDetailsFromToken(existingToken.getToken());

}

@Override

public Page<User> findAll(String keyword, Pageable pageable) {

return userRepository.findAll(keyword, pageable);

}

@Override

@Transactional

public void resetPassword(Long userId, String newPassword)

throws InvalidPasswordException, DataNotFoundException {

User existingUser = userRepository.findById(userId)

.orElseThrow(() -> new DataNotFoundException("User not found"));

String encodedPassword = passwordEncoder.encode(newPassword);

existingUser.setPassword(encodedPassword);

userRepository.save(existingUser);

//reset password => clear token

List<Token> tokens = tokenRepository.findByUser(existingUser);

for (Token token : tokens) {

tokenRepository.delete(token);

}

}

@Override

@Transactional

public void blockOrEnable(Long userId, Boolean active) throws DataNotFoundException {

User existingUser = userRepository.findById(userId)

.orElseThrow(() -> new DataNotFoundException("User not found"));

existingUser.setActive(active);

userRepository.save(existingUser);

}

}

## **Controller**

### MauBangController

@RestController

@RequestMapping("${api.prefix}/maubangs")

public class MauBangController {

private static final Logger logger = LoggerFactory.getLogger(MauBangController.class);

@Autowired

private MauBangService mauBangService;

private LocalizationUtils localizationUtils;

private SecurityUtils securityUtils;

@PostMapping("")

@PreAuthorize("hasRole('ROLE\_ADMIN')")

//POST http://localhost:8088/v1/api/products

public ResponseEntity<ResponseObject> createProduct(

@Valid @RequestBody MauBangDTO mauBangDTO,

BindingResult result

) throws Exception {

if(result.hasErrors()) {

List<String> errorMessages = result.getFieldErrors()

.stream()

.map(FieldError::getDefaultMessage)

.toList();

return ResponseEntity.badRequest().body(

ResponseObject.builder()

.message(String.join("; ", errorMessages))

.status(HttpStatus.BAD\_REQUEST)

.build()

);

}

MauBang newProduct = mauBangService.createMauBang(mauBangDTO);

return ResponseEntity.ok(

ResponseObject.builder()

.message("Create new mau bang successfully")

.status(HttpStatus.CREATED)

.data(newProduct)

.build());

}

@PostMapping(value = "uploads/{id}",

consumes = MediaType.MULTIPART\_FORM\_DATA\_VALUE)

@PreAuthorize("hasRole('ROLE\_ADMIN')")

public ResponseEntity<ResponseObject> uploadImages(

@PathVariable("id") Long productId,

@ModelAttribute("files") List<MultipartFile> files

) throws Exception {

MauBang existingProduct = mauBangService.getMauBangById(productId);

files = files == null ? new ArrayList<MultipartFile>() : files;

List<MauBangImage> productImages = new ArrayList<>();

for (MultipartFile file : files) {

if(file.getSize() == 0) {

continue;

}

// Kiểm tra kích thước file và định dạng

if(file.getSize() > 10 \* 1024 \* 1024) { // Kích thước > 10MB

return ResponseEntity.status(HttpStatus.PAYLOAD\_TOO\_LARGE)

.body(ResponseObject.builder()

.message(localizationUtils

.getLocalizedMessage(MessageKeys.UPLOAD\_IMAGES\_FILE\_LARGE))

.status(HttpStatus.PAYLOAD\_TOO\_LARGE)

.build());

}

String contentType = file.getContentType();

if(contentType == null || !contentType.startsWith("image/")) {

return ResponseEntity.status(HttpStatus.UNSUPPORTED\_MEDIA\_TYPE)

.body(ResponseObject.builder()

.message(localizationUtils

.getLocalizedMessage(MessageKeys.UPLOAD\_IMAGES\_FILE\_MUST\_BE\_IMAGE))

.status(HttpStatus.UNSUPPORTED\_MEDIA\_TYPE)

.build());

}

// Lưu file và cập nhật thumbnail trong DTO

String filename = mauBangService.storeFile(file); // Thay thế hàm này với code của bạn để lưu file

//lưu vào đối tượng product trong DB

MauBangImage productImage = mauBangService.createProductImage(

existingProduct.getId(),

MauBangImageDTO.builder()

.imageUrl(filename)

.build()

);

productImages.add(productImage);

}

return ResponseEntity.ok().body(ResponseObject.builder()

.message("Upload image successfully")

.status(HttpStatus.CREATED)

.data(productImages)

.build());

}

@GetMapping("/images/{imageName}")

public ResponseEntity<?> viewImage(@PathVariable String imageName) {

try {

java.nio.file.Path imagePath = Paths.get("uploads/"+imageName);

UrlResource resource = new UrlResource(imagePath.toUri());

if (resource.exists()) {

return ResponseEntity.ok()

.contentType(MediaType.IMAGE\_JPEG)

.body(resource);

} else {

return ResponseEntity.ok()

.contentType(MediaType.IMAGE\_JPEG)

.body(new UrlResource(Paths.get("uploads/notfound.jpeg").toUri()));

//return ResponseEntity.notFound().build();

}

} catch (Exception e) {

return ResponseEntity.notFound().build();

}

}

@GetMapping("")

public ResponseEntity<ResponseObject> getProducts(

@RequestParam(defaultValue = "") String keyword,

@RequestParam(defaultValue = "0") int page,

@RequestParam(defaultValue = "10") int limit

) {

try {

PageRequest pageRequest = PageRequest.of(

page, limit,

Sort.by("id").ascending()

);

logger.info(String.format("keyword = %s, page = %d, limit = %d", keyword, page, limit));

Page<MauBangResponse> productPage = mauBangService.getAllProducts(keyword,pageRequest);

int totalPages = productPage.getTotalPages();

List<MauBangResponse> productResponses = productPage.getContent();

MauBangListResponse productListResponse = MauBangListResponse.builder()

.products(productResponses)

.totalPages(totalPages)

.build();

return ResponseEntity.ok().body(ResponseObject.builder()

.message("Lấy sản phẩm thành công")

.status(HttpStatus.OK)

.data(productListResponse)

.build());

} catch (Exception e) {

return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).body(

ResponseObject.builder()

.message("Đã xảy ra lỗi: " + e.getMessage())

.status(HttpStatus.INTERNAL\_SERVER\_ERROR)

.data(null)

.build()

);

}

}

//http://localhost:8088/api/v1/products/6

@GetMapping("/{id}")

public ResponseEntity<ResponseObject> getProductById(

@PathVariable("id") Long productId

) throws Exception {

MauBang existingProduct = mauBangService.getMauBangById(productId);

return ResponseEntity.ok(ResponseObject.builder()

.data(MauBangResponse.fromProduct(existingProduct))

.message("Get detail product successfully")

.status(HttpStatus.OK)

.build());

}

@GetMapping("/by-ids")

public ResponseEntity<ResponseObject> getProductsByIds(@RequestParam("ids") String ids) {

//eg: 1,3,5,7

// Tách chuỗi ids thành một mảng các số nguyên

List<Long> productIds = Arrays.stream(ids.split(","))

.map(Long::parseLong)

.collect(Collectors.toList());

List<MauBang> products = mauBangService.findProductsByIds(productIds);

return ResponseEntity.ok(ResponseObject.builder()

.data(products)

.message("Get products successfully")

.status(HttpStatus.OK)

.build()

);

}

@DeleteMapping("/{id}")

@PreAuthorize("hasRole('ROLE\_ADMIN')")

@Operation(security = { @SecurityRequirement(name = "bearer-key") })

public ResponseEntity<ResponseObject> deleteProduct(@PathVariable long id) {

mauBangService.deleteMauBang(id);

return ResponseEntity.ok(ResponseObject.builder()

.data(null)

.message(String.format("Product with id = %d deleted successfully", id))

.status(HttpStatus.OK)

.build());

}

//update a product

@PutMapping("/{id}")

@PreAuthorize("hasRole('ROLE\_ADMIN')")

//@SecurityRequirement(name="bearer-key")

@Operation(security = { @SecurityRequirement(name = "bearer-key") })

public ResponseEntity<ResponseObject> updateProduct(

@PathVariable long id,

@RequestBody MauBangDTO productDTO) throws Exception {

MauBang updatedProduct = mauBangService.updateProduct(id, productDTO);

return ResponseEntity.ok(ResponseObject.builder()

.data(updatedProduct)

.message("Update product successfully")

.status(HttpStatus.OK)

.build());

}

}

### MauBangImageController

@RestController

@RequestMapping("${api.prefix}/maubang\_image")

//@Validated

//Dependency Injection

@RequiredArgsConstructor

public class MauBangImageController {

private final MauBangImageService productImageService;

private final MauBangService productService;

@DeleteMapping("/{id}")

@PreAuthorize("hasRole('ROLE\_ADMIN')")

public ResponseEntity<ResponseObject> delete(

@PathVariable Long id

) throws Exception {

MauBangImage productImage = productImageService.deleteProductImage(id);

if(productImage != null){

productService.deleteFile(productImage.getImageUrl());

}

return ResponseEntity.ok().body(

ResponseObject.builder()

.message("Delete product image successfully")

.data(productImage)

.status(HttpStatus.OK)

.build()

);

}

}

### OrderController

@RestController

@RequestMapping("${api.prefix}/orders")

@RequiredArgsConstructor

public class OrderController {

private final IOrderService orderService;

private final LocalizationUtils localizationUtils;

// Tạo đơn hàng mới

@PostMapping("")

@PreAuthorize("hasRole('ROLE\_ADMIN') or hasRole('ROLE\_USER')")

public ResponseEntity<ResponseObject> createOrder(

@Valid @RequestBody OrderDTO orderDTO,

BindingResult result

) throws Exception {

// Kiểm tra lỗi hợp lệ

if(result.hasErrors()) {

// Xử lý lỗi và trả về response

List<String> errorMessages = result.getFieldErrors()

.stream()

.map(FieldError::getDefaultMessage)

.toList();

return ResponseEntity.badRequest().body(

ResponseObject.builder()

.message(String.join(";", errorMessages))

.status(HttpStatus.BAD\_REQUEST)

.build());

}

// Tạo đơn hàng và trả về response

Order orderResponse = orderService.createOrder(orderDTO);

return ResponseEntity.ok(ResponseObject.builder()

.message("Insert order successfully")

.data(orderResponse)

.status(HttpStatus.OK)

.build());

}

// Lấy danh sách đơn hàng của một người dùng

@GetMapping("/user/{user\_id}") // Thêm biến đường dẫn "user\_id"

public ResponseEntity<ResponseObject> getOrders(@Valid @PathVariable("user\_id") Long userId) {

// Lấy danh sách đơn hàng của người dùng và trả về response

List<Order> orders = orderService.findByUserId(userId);

return ResponseEntity.ok(ResponseObject

.builder()

.message("Get list of orders successfully")

.data(orders)

.status(HttpStatus.OK)

.build());

}

// Lấy thông tin một đơn hàng

@GetMapping("/{id}")

public ResponseEntity<ResponseObject> getOrder(@Valid @PathVariable("id") Long orderId) {

// Lấy thông tin đơn hàng và trả về response

Order existingOrder = orderService.getOrder(orderId);

OrderResponse orderResponse = OrderResponse.fromOrder(existingOrder);

return ResponseEntity.ok(new ResponseObject(

"Get order successfully",

HttpStatus.OK,

orderResponse

));

}

// Cập nhật thông tin một đơn hàng

@PutMapping("/{id}")

@PreAuthorize("hasRole('ROLE\_ADMIN')")

public ResponseEntity<ResponseObject> updateOrder(

@Valid @PathVariable long id,

@Valid @RequestBody OrderDTO orderDTO) throws Exception {

// Cập nhật đơn hàng và trả về response

Order order = orderService.updateOrder(id, orderDTO);

return ResponseEntity.ok(new ResponseObject("Update order successfully", HttpStatus.OK, order));

}

// Xóa một đơn hàng

@DeleteMapping("/{id}")

@PreAuthorize("hasRole('ROLE\_ADMIN')")

public ResponseEntity<ResponseObject> deleteOrder(@Valid @PathVariable Long id) {

// Xóa đơn hàng và trả về response

orderService.deleteOrder(id);

String message = localizationUtils.getLocalizedMessage(

MessageKeys.DELETE\_ORDER\_SUCCESSFULLY, id);

return ResponseEntity.ok(

ResponseObject.builder()

.message(message)

.build()

);

}

// Tìm kiếm đơn hàng theo từ khóa

@GetMapping("/get-orders-by-keyword")

@PreAuthorize("hasRole('ROLE\_ADMIN')")

public ResponseEntity<ResponseObject> getOrdersByKeyword(

@RequestParam(defaultValue = "", required = false) String keyword,

@RequestParam(defaultValue = "0") int page,

@RequestParam(defaultValue = "10") int limit

) {

// Tạo Pageable từ thông tin trang và giới hạn

PageRequest pageRequest = PageRequest.of(

page, limit,

Sort.by("id").ascending()

);

// Lấy danh sách đơn hàng và trả về response

Page<OrderResponse> orderPage = orderService

.getOrdersByKeyword(keyword, pageRequest)

.map(OrderResponse::fromOrder);

int totalPages = orderPage.getTotalPages();

List<OrderResponse> orderResponses = orderPage.getContent();

OrderListResponse orderListResponse = OrderListResponse.builder()

.orders(orderResponses)

.totalPages(totalPages)

.build();

return ResponseEntity.ok().body(ResponseObject.builder()

.message("Get orders successfully")

.status(HttpStatus.OK)

.data(orderListResponse)

.build());

}

}

### OrderDetailController

@RestController

@RequestMapping("${api.prefix}/order\_details")

@RequiredArgsConstructor

public class OrderDetailController {

private final OrderDetailService orderDetailService;

private final LocalizationUtils localizationUtils;

//Thêm mới 1 order detail

@PostMapping("")

@PreAuthorize("hasRole('ROLE\_ADMIN') or hasRole('ROLE\_USER')")

public ResponseEntity<ResponseObject> createOrderDetail(

@Valid @RequestBody OrderDetailDTO orderDetailDTO) throws Exception {

// Tạo order detail mới và trả về response

OrderDetail newOrderDetail = orderDetailService.createOrderDetail(orderDetailDTO);

OrderDetailResponse orderDetailResponse = OrderDetailResponse.fromOrderDetail(newOrderDetail);

return ResponseEntity.ok().body(

ResponseObject.builder()

.message("Create order detail successfully")

.status(HttpStatus.CREATED)

.data(orderDetailResponse)

.build()

);

}

@GetMapping("/{id}")

@PreAuthorize("hasRole('ROLE\_ADMIN') or hasRole('ROLE\_USER')")

public ResponseEntity<?> getOrderDetail(

@Valid @PathVariable("id") Long id) throws DataNotFoundException {

// Lấy thông tin order detail theo id và trả về response

OrderDetail orderDetail = orderDetailService.getOrderDetail(id);

OrderDetailResponse orderDetailResponse = OrderDetailResponse.fromOrderDetail(orderDetail);

return ResponseEntity.ok().body(

ResponseObject.builder()

.message("Get order detail successfully")

.status(HttpStatus.OK)

.data(orderDetailResponse)

.build()

);

}

//lấy ra danh sách các order\_details của 1 order nào đó

@GetMapping("/order/{orderId}")

@PreAuthorize("hasRole('ROLE\_ADMIN') or hasRole('ROLE\_USER')")

public ResponseEntity<ResponseObject> getOrderDetails(

@Valid @PathVariable("orderId") Long orderId

) {

// Lấy danh sách order details theo order id và trả về response

List<OrderDetail> orderDetails = orderDetailService.findByOrderId(orderId);

List<OrderDetailResponse> orderDetailResponses = orderDetails

.stream()

.map(OrderDetailResponse::fromOrderDetail)

.toList();

return ResponseEntity.ok().body(

ResponseObject.builder()

.message("Get order details by orderId successfully")

.status(HttpStatus.OK)

.data(orderDetailResponses)

.build()

);

}

@PutMapping("/{id}")

@Operation(security = { @SecurityRequirement(name = "bearer-key") })

@PreAuthorize("hasRole('ROLE\_ADMIN') or hasRole('ROLE\_USER')")

public ResponseEntity<ResponseObject> updateOrderDetail(

@Valid @PathVariable("id") Long id,

@RequestBody OrderDetailDTO orderDetailDTO) throws DataNotFoundException, Exception {

// Cập nhật thông tin order detail và trả về response

OrderDetail orderDetail = orderDetailService.updateOrderDetail(id, orderDetailDTO);

return ResponseEntity.ok().body(ResponseObject

.builder()

.data(orderDetail)

.message("Update order detail successfully")

.status(HttpStatus.OK)

.build());

}

@DeleteMapping("/{id}")

@Operation(security = { @SecurityRequirement(name = "bearer-key") })

@PreAuthorize("hasRole('ROLE\_ADMIN') or hasRole('ROLE\_USER')")

public ResponseEntity<ResponseObject> deleteOrderDetail(

@Valid @PathVariable("id") Long id) {

// Xóa order detail và trả về response

orderDetailService.deleteById(id);

return ResponseEntity.ok()

.body(ResponseObject.builder()

.message(localizationUtils

.getLocalizedMessage(MessageKeys.DELETE\_ORDER\_DETAIL\_SUCCESSFULLY))

.build());

}

@GetMapping("/by-ids")

public ResponseEntity<ResponseObject> getProductsByIds(@RequestParam("ids") String ids) {

// Tách chuỗi ids thành một mảng các số nguyên

List<Long> productIds = Arrays.stream(ids.split(","))

.map(Long::parseLong)

.collect(Collectors.toList());

// Lấy danh sách order details theo ids và trả về response

List<OrderDetail> products = orderDetailService.finOrderDetails(productIds);

return ResponseEntity.ok(ResponseObject.builder()

.data(products)

.message("Get products successfully")

.status(HttpStatus.OK)

.build()

);

}

}

### ProductController

@RestController

@RequestMapping("${api.prefix}/products")

@RequiredArgsConstructor

public class ProductController {

private final ProductService productService;

@PostMapping("")

//POST http://localhost:8088/v1/api/products

public ResponseEntity<ResponseObject> createProduct(

@Valid @RequestBody ProductDTO productDTO,

BindingResult result

) throws Exception {

if(result.hasErrors()) {

List<String> errorMessages = result.getFieldErrors()

.stream()

.map(FieldError::getDefaultMessage)

.toList();

return ResponseEntity.badRequest().body(

ResponseObject.builder()

.message(String.join("; ", errorMessages))

.status(HttpStatus.BAD\_REQUEST)

.build()

);

}

// Tạo mới sản phẩm và trả về response

Product newProduct = productService.createProduct(productDTO);

return ResponseEntity.ok(

ResponseObject.builder()

.message("Create new product successfully")

.status(HttpStatus.CREATED)

.data(newProduct)

.build());

}

@GetMapping("/images/{imageName}")

public ResponseEntity<?> viewImage(@PathVariable String imageName) {

try {

java.nio.file.Path imagePath = Paths.get("uploads/"+imageName);

UrlResource resource = new UrlResource(imagePath.toUri());

if (resource.exists()) {

return ResponseEntity.ok()

.contentType(MediaType.IMAGE\_JPEG)

.body(resource);

} else {

return ResponseEntity.ok()

.contentType(MediaType.IMAGE\_JPEG)

.body(new UrlResource(Paths.get("uploads/notfound.jpeg").toUri()));

//return ResponseEntity.notFound().build();

}

} catch (Exception e) {

return ResponseEntity.notFound().build();

}

}

@GetMapping("/{id}")

public ResponseEntity<ResponseObject> getProductById(

@PathVariable("id") Long productId

) throws Exception {

// Lấy thông tin sản phẩm theo id và trả về response

Product existingProduct = productService.getProductById(productId);

return ResponseEntity.ok(ResponseObject.builder()

.data(ProductResponse.fromProduct(existingProduct))

.message("Get detail product successfully")

.status(HttpStatus.OK)

.build());

}

@GetMapping("/by-ids")

public ResponseEntity<ResponseObject> getProductsByIds(@RequestParam("ids") String ids) {

// Tách chuỗi ids thành một mảng các số nguyên

List<Long> productIds = Arrays.stream(ids.split(","))

.map(Long::parseLong)

.collect(Collectors.toList());

// Lấy danh sách sản phẩm theo ids và trả về response

List<Product> products = productService.findProductsByIds(productIds);

return ResponseEntity.ok(ResponseObject.builder()

.data(products)

.message("Get products successfully")

.status(HttpStatus.OK)

.build()

);

}

}

### RoleController

@RestController

@RequestMapping("${api.prefix}/roles")

@RequiredArgsConstructor

@CrossOrigin("\*")

public class RoleController {

private final RoleService roleService;

@GetMapping("")

public ResponseEntity<ResponseObject> getAllRoles() {

List<Role> roles = roleService.getAllRoles();

return ResponseEntity.ok().body(ResponseObject.builder()

.message("Get roles successfully")

.status(HttpStatus.OK)

.data(roles)

.build());

}

}

### UserController

@RestController

@RequestMapping("${api.prefix}/users")

@RequiredArgsConstructor

public class UserController {

private final IUserService userService;

private final LocalizationUtils localizationUtils;

private final ITokenService tokenService;

@GetMapping("")

@PreAuthorize("hasRole('ROLE\_ADMIN')")

public ResponseEntity<ResponseObject> getAllUser(

@RequestParam(defaultValue = "", required = false) String keyword,

@RequestParam(defaultValue = "0") int page,

@RequestParam(defaultValue = "10") int limit

) throws Exception{

// Tạo Pageable từ thông tin trang và giới hạn

PageRequest pageRequest = PageRequest.of(

page, limit,

//Sort.by("createdAt").descending()

Sort.by("id").ascending()

);

Page<UserResponse> userPage = userService.findAll(keyword, pageRequest)

.map(UserResponse::fromUser);

// Lấy tổng số trang

int totalPages = userPage.getTotalPages();

List<UserResponse> userResponses = userPage.getContent();

UserListResponse userListResponse = UserListResponse

.builder()

.users(userResponses)

.totalPages(totalPages)

.build();

return ResponseEntity.ok().body(ResponseObject.builder()

.message("Get user list successfully")

.status(HttpStatus.OK)

.data(userListResponse)

.build());

}

@PostMapping("/register")

//can we register an "admin" user ?

public ResponseEntity<ResponseObject> createUser(

@Valid @RequestBody UserDTO userDTO,

BindingResult result

) throws Exception {

if (result.hasErrors()) {

List<String> errorMessages = result.getFieldErrors()

.stream()

.map(FieldError::getDefaultMessage)

.toList();

return ResponseEntity.badRequest().body(ResponseObject.builder()

.status(HttpStatus.BAD\_REQUEST)

.data(null)

.message(errorMessages.toString())

.build());

}

if (!userDTO.getPassword().equals(userDTO.getRetypePassword())) {

//registerResponse.setMessage();

return ResponseEntity.badRequest().body(ResponseObject.builder()

.status(HttpStatus.BAD\_REQUEST)

.data(null)

.message(localizationUtils.getLocalizedMessage(MessageKeys.PASSWORD\_NOT\_MATCH))

.build());

}

User user = userService.createUser(userDTO);

return ResponseEntity.ok(ResponseObject.builder()

.status(HttpStatus.CREATED)

.data(UserResponse.fromUser(user))

.message("Đăng ký tài khoản thành công")

.build());

}

@PostMapping("/login")

public ResponseEntity<ResponseObject> login(

@Valid @RequestBody UserLoginDTO userLoginDTO,

HttpServletRequest request

) throws Exception {

// Kiểm tra thông tin đăng nhập và sinh token

String token = userService.login(

userLoginDTO.getPhoneNumber(),

userLoginDTO.getPassword(),

userLoginDTO.getRoleId() == null ? 1 : userLoginDTO.getRoleId()

);

String userAgent = request.getHeader("User-Agent");

User userDetail = userService.getUserDetailsFromToken(token);

Token jwtToken = tokenService.addToken(userDetail, token, isMobileDevice(userAgent));

LoginResponse loginResponse = LoginResponse.builder()

.message(localizationUtils.getLocalizedMessage(MessageKeys.LOGIN\_SUCCESSFULLY))

.token(jwtToken.getToken())

.tokenType(jwtToken.getTokenType())

.refreshToken(jwtToken.getRefreshToken())

.username(userDetail.getUsername())

.roles(userDetail.getAuthorities().stream().map(item -> item.getAuthority()).toList())

.id(userDetail.getId())

.build();

return ResponseEntity.ok().body(ResponseObject.builder()

.message("Login successfully")

.data(loginResponse)

.status(HttpStatus.OK)

.build());

}

@PostMapping("/refreshToken")

public ResponseEntity<ResponseObject> refreshToken(

@Valid @RequestBody RefreshTokenDTO refreshTokenDTO

) throws Exception {

User userDetail = userService.getUserDetailsFromRefreshToken(refreshTokenDTO.getRefreshToken());

Token jwtToken = tokenService.refreshToken(refreshTokenDTO.getRefreshToken(), userDetail);

LoginResponse loginResponse = LoginResponse.builder()

.message("Refresh token successfully")

.token(jwtToken.getToken())

.tokenType(jwtToken.getTokenType())

.refreshToken(jwtToken.getRefreshToken())

.username(userDetail.getUsername())

.roles(userDetail.getAuthorities().stream().map(item -> item.getAuthority()).toList())

.id(userDetail.getId()).build();

return ResponseEntity.ok().body(

ResponseObject.builder()

.data(loginResponse)

.message(loginResponse.getMessage())

.status(HttpStatus.OK)

.build());

}

private boolean isMobileDevice(String userAgent) {

// Kiểm tra User-Agent header để xác định thiết bị di động

// Ví dụ đơn giản:

return userAgent.toLowerCase().contains("mobile");

}

@PostMapping("/details")

@PreAuthorize("hasRole('ROLE\_ADMIN') or hasRole('ROLE\_USER')")

public ResponseEntity<ResponseObject> getUserDetails(

@RequestHeader("Authorization") String authorizationHeader

) throws Exception {

String extractedToken = authorizationHeader.substring(7); // Loại bỏ "Bearer " từ chuỗi token

User user = userService.getUserDetailsFromToken(extractedToken);

return ResponseEntity.ok().body(

ResponseObject.builder()

.message("Get user's detail successfully")

.data(UserResponse.fromUser(user))

.status(HttpStatus.OK)

.build()

);

}

@PutMapping("/details/{userId}")

@PreAuthorize("hasRole('ROLE\_ADMIN') or hasRole('ROLE\_USER')")

@Operation(security = { @SecurityRequirement(name = "bearer-key") })

public ResponseEntity<ResponseObject> updateUserDetails(

@PathVariable Long userId,

@RequestBody UpdateUserDTO updatedUserDTO,

@RequestHeader("Authorization") String authorizationHeader

) throws Exception{

String extractedToken = authorizationHeader.substring(7);

User user = userService.getUserDetailsFromToken(extractedToken);

// Ensure that the user making the request matches the user being updated

if (user.getId() != userId) {

return ResponseEntity.status(HttpStatus.FORBIDDEN).build();

}

User updatedUser = userService.updateUser(userId, updatedUserDTO);

return ResponseEntity.ok().body(

ResponseObject.builder()

.message("Update user detail successfully")

.data(UserResponse.fromUser(updatedUser))

.status(HttpStatus.OK)

.build()

);

}

@PutMapping("/block/{userId}/{active}")

@PreAuthorize("hasRole('ROLE\_ADMIN')")

public ResponseEntity<ResponseObject> blockOrEnable(

@Valid @PathVariable long userId,

@Valid @PathVariable int active

) throws Exception {

userService.blockOrEnable(userId, active > 0);

String message = active > 0 ? "Successfully enabled the user." : "Successfully blocked the user.";

return ResponseEntity.ok().body(ResponseObject.builder()

.message(message)

.status(HttpStatus.OK)

.data(null)

.build());

}

}

## **dtos**

### CartItemDTO

public class CartItemDTO {

@JsonProperty("product\_id")

private Long productId;

@JsonProperty("quantity")

private Integer quantity;

}

### MauBangDTO

public class MauBangDTO {

@NotBlank(message = "Title is required")

@Size(min = 3, max = 200, message = "Name must be between 3 and 200 characters")

private String name;

private int price = 230000;

}

### MauBangImageDTO

public class MauBangImageDTO {

@JsonProperty("maubang\_id")

@Min(value = 1, message = "MauBang's ID must be > 0")

private Long maubangid;

@Size(min = 5, max = 200, message = "Image's name")

@JsonProperty("image\_url")

private String imageUrl;

}

### OrderDetailDTO

public class OrderDetailDTO {

@JsonProperty("order\_id")

@Min(value=1, message = "Order's ID must be > 0")

private Long orderId;

private int price = 230000;

@Min(value=1, message = "Product's ID must be > 0")

@JsonProperty("product\_id")

private Long productId;

@Min(value=1, message = "number\_of\_products must be >= 1")

@JsonProperty("number\_of\_products")

private int numberOfProducts;

@Min(value=0, message = "total\_money must be >= 0")

@JsonProperty("total\_money")

private int totalMoney;

}

### OrderDTO

public class OrderDTO {

@JsonProperty("user\_id")

@Min(value = 1, message = "User's ID must be > 0")

private Long userId;

@JsonProperty("fullname")

private String fullName;

private String email;

@JsonProperty("phone\_number")

@NotBlank(message = "Phone number is required")

@Size(min = 5, message = "Phone number must be at least 5 characters")

private String phoneNumber;

@JsonProperty("status")

private String status;

private String address;

private String note;

@JsonProperty("total\_money")

@Min(value = 0, message = "Total money must be >= 0")

private Float totalMoney;

@JsonProperty("shipping\_method")

private String shippingMethod;

@JsonProperty("shipping\_address")

private String shippingAddress;

@JsonProperty("shipping\_date")

private LocalDate shippingDate;

@JsonProperty("payment\_method")

private String paymentMethod;

@JsonProperty("cart\_items")

private List<CartItemDTO> cartItems;

}

### OrderWithDetailsDTO

public class OrderWithDetailsDTO {

@JsonProperty("user\_id")

@Min(value = 1, message = "User's ID must be > 0")

private Long userId;

@JsonProperty("fullname")

private String fullName;

private String email;

@JsonProperty("phone\_number")

@NotBlank(message = "Phone number is required")

@Size(min = 5, message = "Phone number must be at least 5 characters")

private String phoneNumber;

private String address;

private String note;

@JsonProperty("total\_money")

@Min(value = 0, message = "Total money must be >= 0")

private Float totalMoney;

@JsonProperty("shipping\_method")

private String shippingMethod;

@JsonProperty("shipping\_address")

private String shippingAddress;

@JsonProperty("shipping\_date")

private LocalDate shippingDate;

@JsonProperty("payment\_method")

private String paymentMethod;

@JsonProperty("order\_details")

private List<OrderDetailDTO> orderDetailDTOS;

}

### ProductDTO

public class ProductDTO {

private String name;

private int price = 230000;

private int otpName;

private long maubangid;

private String matA;

private String matB;

private String bia;

}

### RefreshTokenDTO

public class RefreshTokenDTO {

@NotBlank

private String refreshToken;

}

### UpdateUserDTO

public class UpdateUserDTO {

@JsonProperty("fullname")

private String fullName;

@JsonProperty("phone\_number")

private String phoneNumber;

private String address;

@JsonProperty("password")

private String password;

@JsonProperty("retype\_password")

private String retypePassword;

@JsonProperty("date\_of\_birth")

private Date dateOfBirth;

@NotBlank(message = "Password cannot be blank")

private String password;

@JsonProperty("retype\_password")

private String retypePassword;

@JsonProperty("date\_of\_birth")

private Date dateOfBirth;

@NotNull(message = "Role ID is required")

@JsonProperty("role\_id")

//role admin not permitted

private Long roleId;

}

### UserLoginDTO

public class UserLoginDTO {

@JsonProperty("phone\_number")

@NotBlank(message = "Phone number is required")

private String phoneNumber;

@NotBlank(message = "Password cannot be blank")

private String password;

@Min(value = 1, message = "You must enter role's Id")

@JsonProperty("role\_id")

private Long roleId;

}}

### UserDTO

public class UserDTO {

@JsonProperty("fullname")

private String fullName;

@JsonProperty("phone\_number")

@NotBlank(message = "Phone number is required")

private String phoneNumber;

private String address;

}

## **responses**

### MauBang

* + 1. MauBangListResponse

public class MauBangListResponse {

private List<MauBangResponse> products;

private int totalPages;

}

* + 1. MauBangResponse

public class MauBangResponse extends BaseResponse {

private Long id;

private String name;

private String thumbnail;

// Thêm trường totalPages

private int totalPages;

@JsonProperty("maubang\_image")

private MauBangImage mauBangImage;

public static MauBangResponse fromProduct(MauBang product) {

MauBangResponse productResponse = MauBangResponse.builder()

.id(product.getId())

.name(product.getName())

.thumbnail(product.getThumbnail())

.mauBangImage(product.getImg())

.build();

productResponse.setCreatedAt(product.getCreatedAt());

productResponse.setUpdatedAt(product.getUpdatedAt());

return productResponse;

}

}

### order

* + 1. OrderDetailResponse

public class OrderDetailResponse {

private Long id;

@JsonProperty("order\_id")

private Long orderId;

@JsonProperty("product\_id")

private Long productId;

@JsonProperty("price")

private int price;

@JsonProperty("number\_of\_products")

private int numberOfProducts;

@JsonProperty("total\_money")

private int totalMoney;

public static OrderDetailResponse fromOrderDetail(OrderDetail orderDetail) {

return OrderDetailResponse

.builder()

.id(orderDetail.getId())

.orderId(orderDetail.getOrder().getId())

.productId(orderDetail.getProduct().getId())

.numberOfProducts(orderDetail.getNumberOfProducts())

.price(orderDetail.getPrice())

.totalMoney(orderDetail.getTotalMoney())

.build();

}

}

* + 1. OrderListResponse

public class OrderListResponse {

private List<OrderResponse> orders;

private int totalPages;

}

* + 1. OrderResponse

public class OrderResponse {

private Long id;

@JsonProperty("user\_id")

private Long userId;

@JsonProperty("fullname")

private String fullName;

@JsonProperty("phone\_number")

private String phoneNumber;

@JsonProperty("email")

private String email;

@JsonProperty("address")

private String address;

@JsonProperty("note")

private String note;

@JsonProperty("order\_date")

//@JsonFormat(pattern = "yyyy-MM-dd'T'HH:mm:ss.SSS'Z'", timezone =

"UTC")

private LocalDate orderDate;

@JsonProperty("status")

private String status;

@JsonProperty("total\_money")

private double totalMoney;

@JsonProperty("shipping\_method")

private String shippingMethod;

@JsonProperty("shipping\_address")

private String shippingAddress;

@JsonProperty("shipping\_date")

private LocalDate shippingDate;

@JsonProperty("payment\_method")

private String paymentMethod;

@JsonProperty("order\_details")

private List<OrderDetail> orderDetails;

public static OrderResponse fromOrder(Order order) {

OrderResponse orderResponse = OrderResponse

.builder()

.id(order.getId())

.userId(order.getUser().getId())

.fullName(order.getFullName())

.phoneNumber(order.getPhoneNumber())

.email(order.getEmail())

.address(order.getAddress())

.note(order.getNote())

.orderDate(order.getOrderDate())

.status(order.getStatus())

.totalMoney(order.getTotalMoney())

.shippingMethod(order.getShippingMethod())

.shippingAddress(order.getShippingAddress())

.shippingDate(order.getShippingDate())

.paymentMethod(order.getPaymentMethod())

.orderDetails(order.getOrderDetails())

.build();

return orderResponse;

}

}

### product

* + 1. ProductListResponse

public class ProductListResponse {

private List<ProductResponse> products;

private int totalPages;

}

* + 1. ProductResponse

public class ProductResponse extends BaseResponse {

private Long id;

private String name;

private int price;

private int otpName;

private String matA;

private String matB;

private long maubangId;

private String bia;

// Thêm trường totalPages

private int totalPages;

public static ProductResponse fromProduct(Product product) {

ProductResponse productResponse = ProductResponse.builder()

.id(product.getId())

.name(product.getName())

.price(product.getPrice())

.otpName(product.getOtpName())

.matA(product.getMatA())

.matB(product.getMatB())

.bia(product.getBia())

.maubangId(product.getMauBang().getId())

.build();

productResponse.setCreatedAt(product.getCreatedAt());

productResponse.setUpdatedAt(product.getUpdatedAt());

return productResponse;

}

}

### user

* + 1. LoginResponse

public class LoginResponse {

@JsonProperty("message")

private String message;

@JsonProperty("token")

private String token;

@JsonProperty("refresh\_token")

private String refreshToken;

private String tokenType = "Bearer";

//user's detail

private Long id;

private String username;

private List<String> roles;

}

* + 1. RegisterResponse

public class RegisterResponse {

@JsonProperty("message")

private String message;

@JsonProperty("user")

private User user;

}

* + 1. UserListResponse

public class UserListResponse {

private List<UserResponse> users;

private int totalPages;

}

* + 1. UserResponse

public class UserResponse {

@JsonProperty("id")

private Long id;

@JsonProperty("fullname")

private String fullName;

@JsonProperty("phone\_number")

private String phoneNumber;

@JsonProperty("address")

private String address;

@JsonProperty("is\_active")

private boolean active;

@JsonProperty("date\_of\_birth")

private Date dateOfBirth;

@JsonProperty("role")

private Role role;

public static UserResponse fromUser(User user) {

return UserResponse.builder()

.id(user.getId())

.fullName(user.getFullName())

.phoneNumber(user.getPhoneNumber())

.address(user.getAddress())

.active(user.isActive())

.dateOfBirth(user.getDateOfBirth())

.role(user.getRole())

.build();

}

}

### BaseResponse

public class BaseResponse {

@JsonProperty("created\_at")

@JsonFormat(pattern = "yyyy-MM-dd'T'HH:mm:ss", shape = JsonFormat.Shape.STRING)

private LocalDateTime createdAt;

@JsonFormat(pattern = "yyyy-MM-dd'T'HH:mm:ss", shape = JsonFormat.Shape.STRING)

@JsonProperty("updated\_at")

private LocalDateTime updatedAt;

}

### ResponseObject

public class ResponseObject {

@JsonProperty("message")

private String message;

@JsonProperty("status")

private HttpStatus status;

@JsonProperty("data")

private Object data;

}

## **exceptions**

### DataNotFoundException

public class DataNotFoundException extends Exception{

public DataNotFoundException(String message) {

super(message);

}

}

### ExpriredTokenException

public class ExpiredTokenException extends Exception{

public ExpiredTokenException(String message) {

super(message);

}

}

### InvalidParamException

public class InvalidParamException extends Exception{

public InvalidParamException(String message) {

super(message);

}

}

### InvalidPasswordException

public class InvalidPasswordException extends Exception{

public InvalidPasswordException(String message) {

super(message);

}

}

### PermissionDenyException

public class PermissionDenyException extends Exception{

public PermissionDenyException(String message) {

super(message);

}

}

## app.properties

spring.application.name=webbackend

spring.datasource.url=jdbc:mysql://localhost:3306/webbackend

spring.datasource.username=root

spring.datasource.password=

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.hibernate.ddl-auto=update

api.prefix=/api/v1

jwt.expiration=2592000

jwt.expiration-refresh-token=5184000

jwt.secretKey=TaqlmGv1iEDMRiFp/pHuID1+T84IABfuA0xXh4GhiUI=

# Đường dẫn đến tệp tin chứa thông điệp

spring.messages.basename=i18n/messages

# Mã hóa của tệp tin chứa thông điệp

spring.messages.encoding=UTF-8

# Ngôn ngữ mặc định

spring.messages.default-locale=en

# II Code Frontend

## interceptors

@Injectable()

// Đánh dấu lớp này có thể được tiêm phụ thuộc

export class TokenInterceptor implements HttpInterceptor {

// Inject dịch vụ TokenService để lấy token

constructor(private tokenService: TokenService) { }

// Phương thức này được gọi khi một yêu cầu HTTP được thực hiện

intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {

// Dòng này để kiểm tra khi debug, nên loại bỏ trong môi trường sản xuất

debugger;

// Lấy token từ TokenService

const token = this.tokenService.getToken();

// Nếu có token, clone yêu cầu và thêm tiêu đề Authorization

if (token) {

req = req.clone({

setHeaders: {

Authorization: `Bearer ${token}`, // Đặt tiêu đề Authorization với giá trị token

},

});

}

// Chuyển yêu cầu đã được xử lý cho handler tiếp theo trong chuỗi

return next.handle(req);

}

}

## guards

1. admin

@Injectable({

providedIn: 'root'

})

// Lớp AdminGuard bảo vệ các route dành riêng cho admin

export class AdminGuard {

userResponse?: UserResponse | null; // Biến lưu trữ thông tin phản hồi của người dùng

constructor(

private tokenService: TokenService, // Inject TokenService để quản lý token

private router: Router, // Inject Router để điều hướng

private userService: UserService // Inject UserService để lấy thông tin người dùng

) {}

// Phương thức canActivate kiểm tra quyền truy cập vào route

canActivate(next: ActivatedRouteSnapshot, state: RouterStateSnapshot): boolean {

const isTokenExpired = this.tokenService.isTokenExpired(); // Kiểm tra token hết hạn

const isUserIdValid = this.tokenService.getUserId() > 0; // Kiểm tra userId hợp lệ

this.userResponse = this.userService.getUserResponseFromLocalStorage(); // Lấy thông tin người dùng từ localStorage

const isAdmin = this.userResponse?.role.name == 'admin'; // Kiểm tra người dùng có phải admin không

debugger

if (!isTokenExpired && isUserIdValid && isAdmin) {

return true; // Cho phép truy cập nếu điều kiện thỏa mãn

} else {

// Nếu không thỏa mãn, điều hướng về trang login

this.router.navigate(['/login']);

return false; // Không cho phép truy cập

}

}

}

// Hàm AdminGuardFn sử dụng inject để gọi phương thức canActivate của AdminGuard

export const AdminGuardFn: CanActivateFn = (

next: ActivatedRouteSnapshot,

state: RouterStateSnapshot

): boolean => {

debugger

return inject(AdminGuard).canActivate(next, state); // Gọi canActivate và trả về kết quả

}

1. auth

@Injectable({

providedIn: 'root'

})

// Lớp AuthGuard bảo vệ các route cần xác thực

export class AuthGuard {

constructor(

private tokenService: TokenService, // Inject TokenService để quản lý token

private router: Router, // Inject Router để điều hướng

) {}

// Phương thức canActivate kiểm tra quyền truy cập vào route

canActivate(next: ActivatedRouteSnapshot, state: RouterStateSnapshot): boolean {

const isTokenExpired = this.tokenService.isTokenExpired(); // Kiểm tra token hết hạn

const isUserIdValid = this.tokenService.getUserId() > 0; // Kiểm tra userId hợp lệ

debugger

if (!isTokenExpired && isUserIdValid) {

return true; // Cho phép truy cập nếu điều kiện thỏa mãn

} else {

// Nếu không thỏa mãn, điều hướng về trang login

this.router.navigate(['/login']);

return false; // Không cho phép truy cập

}

}

}

// Sử dụng functional guard như sau:

export const AuthGuardFn: CanActivateFn = (next: ActivatedRouteSnapshot, state: RouterStateSnapshot): boolean => {

debugger

return inject(AuthGuard).canActivate(next, state); // Gọi canActivate và trả về kết quả

}

## routes

export const routes: Routes = [

// Định tuyến trang chủ

{ path: 'trangchu', component: TrangchuComponent },

// Định tuyến trang đăng nhập

{ path: 'login', component: LoginComponent },

// Định tuyến trang đăng ký

{ path: 'register', component: DangkiComponent },

// Định tuyến mặc định chuyển hướng tới trang chủ

{ path: '', redirectTo: "/trangchu", pathMatch: "full" },

// Định tuyến trang chọn bia ngoại

{ path: 'chonbiangoai', component: ChonbiangoaiComponent },

// Định tuyến trang thêm vào giỏ hàng với tham số id

{ path: 'themvaogio/:id', component: ThemvaogioComponent },

// Định tuyến trang thêm nhạc 2

{ path: 'themnhac2', component: Themnhac2Component },

// Định tuyến trang hóa đơn có bảo vệ AuthGuardFn

{ path: 'hoadon', component: HoadonComponent, canActivate: [AuthGuardFn] },

// Định tuyến trang thêm nhạc 1

{ path: 'themnhac1', component: Themnhac1Component },

// Định tuyến trang hồ sơ người dùng có bảo vệ AuthGuardFn

{ path: 'user-profile', component: UserProfileComponent, canActivate: [AuthGuardFn] },

// Định tuyến dành cho admin

{

path: 'admin',

component: AdminComponent,

canActivate: [AdminGuardFn],

children: [

// Định tuyến trang quản lý đơn hàng

{ path: 'orders', component: OrderAdminComponent },

// Định tuyến trang quản lý sản phẩm

{ path: 'products', component: ProductAdminComponent },

// Định tuyến chi tiết đơn hàng theo id

{ path: 'orders/:id', component: DetailOrderAdminComponent },

// Định tuyến cập nhật sản phẩm theo id

{ path: 'products/update/:id', component: UpdateProductAdminComponent },

// Định tuyến thêm mới sản phẩm

{ path: 'products/insert', component: InsertProductAdminComponent },

// Định tuyến quản lý người dùng

{ path: 'users', component: UserAdminComponent },

]

},

];

## service

1. cart

@Injectable({

providedIn: 'root'

})

// Dịch vụ CartService quản lý giỏ hàng

export class CartService {

private cart: Map<number, number> = new Map<number, number>(); // Dùng Map để lưu trữ giỏ hàng, key là id sản phẩm, value là số lượng

localStorage?: Storage;

constructor(@Inject(DOCUMENT) private document: Document) {

this.localStorage = document.defaultView?.localStorage; // Lấy đối tượng localStorage từ document

// Lấy dữ liệu giỏ hàng từ localStorage khi khởi tạo service

this.refreshCart();

}

// Phương thức làm mới giỏ hàng từ localStorage

public refreshCart() {

const storedCart = this.localStorage?.getItem(this.getCartKey());

if (storedCart) {

this.cart = new Map(JSON.parse(storedCart)); // Chuyển dữ liệu từ JSON thành Map

} else {

this.cart = new Map<number, number>(); // Khởi tạo giỏ hàng mới nếu không có dữ liệu trong localStorage

}

}

// Phương thức lấy key cho giỏ hàng từ user id

private getCartKey(): string {

const userResponseJSON = this.localStorage?.getItem('user');

const userResponse = JSON.parse(userResponseJSON!); // Parse JSON thành đối tượng

debugger

return `cart:${userResponse?.id ?? ''}`; // Trả về key giỏ hàng với id người dùng

}

// Phương thức thêm sản phẩm vào giỏ hàng

addToCart(productId: number, quantity: number = 1): void {

debugger

if (this.cart.has(productId)) {

// Nếu sản phẩm đã có trong giỏ hàng, tăng số lượng lên `quantity`

this.cart.set(productId, this.cart.get(productId)! + quantity);

} else {

// Nếu sản phẩm chưa có trong giỏ hàng, thêm sản phẩm vào với số lượng là `quantity`

this.cart.set(productId, quantity);

}

// Sau khi thay đổi giỏ hàng, lưu trữ nó vào localStorage

this.saveCartToLocalStorage();

}

// Phương thức lấy giỏ hàng hiện tại

getCart(): Map<number, number> {

return this.cart;

}

// Lưu trữ giỏ hàng vào localStorage

private saveCartToLocalStorage(): void {

debugger

this.localStorage?.setItem(this.getCartKey(), JSON.stringify(Array.from(this.cart.entries())));

}

// Phương thức đặt lại giỏ hàng và lưu vào localStorage

setCart(cart: Map<number, number>) {

this.cart = cart ?? new Map<number, number>();

this.saveCartToLocalStorage();

}

// Hàm xóa dữ liệu giỏ hàng và cập nhật localStorage

clearCart(): void {

this.cart.clear(); // Xóa toàn bộ dữ liệu trong giỏ hàng

this.saveCartToLocalStorage(); // Lưu giỏ hàng mới vào localStorage (trống)

}

}

1. http.ultil

/http';

@Injectable({

providedIn: 'root',

})

// Dịch vụ HttpUtilService để tạo tiêu đề HTTP

export class HttpUtilService {

// Phương thức tạo tiêu đề HTTP

createHeaders(): HttpHeaders {

return new HttpHeaders({

'Content-Type': 'application/json', // Đặt tiêu đề Content-Type là JSON

'Accept-Language': 'vi', // Đặt tiêu đề Accept-Language là tiếng Việt

});

}

}

1. maubang

@Injectable({

providedIn: 'root'

})

// Dịch vụ MauBangService để quản lý các thao tác liên quan đến sản phẩm

export class MauBangService {

private apiBaseUrl = environment.apiBaseUrl; // URL cơ sở của API

constructor(private http: HttpClient) { }

// Phương thức lấy danh sách sản phẩm với từ khóa tìm kiếm, trang và giới hạn số lượng

getProducts(keyword: string, page: number, limit: number): Observable<ApiResponse> {

const params = {

keyword: keyword,

page: page.toString(),

limit: limit.toString()

};

return this.http.get<ApiResponse>(`${this.apiBaseUrl}/maubangs`, { params });

}

// Phương thức lấy chi tiết sản phẩm theo ID

getDetailProduct(productId: number): Observable<ApiResponse> {

return this.http.get<ApiResponse>(`${this.apiBaseUrl}/maubangs/${productId}`);

}

// Phương thức xóa sản phẩm theo ID

deleteProduct(productId: number): Observable<ApiResponse> {

debugger

return this.http.delete<ApiResponse>(`${this.apiBaseUrl}/maubangs/${productId}`);

}

// Phương thức cập nhật sản phẩm theo ID và dữ liệu cập nhật

updateProduct(productId: number, updatedProduct: UpdateProductDTO): Observable<ApiResponse> {

return this.http.put<ApiResponse>(`${this.apiBaseUrl}/maubangs/${productId}`, updatedProduct);

}

// Phương thức thêm mới sản phẩm

insertProduct(insertProductDTO: InsertProductDTO): Observable<ApiResponse> {

return this.http.post<ApiResponse>(`${this.apiBaseUrl}/maubangs`, insertProductDTO);

}

// Phương thức tải lên hình ảnh cho sản phẩm

uploadImages(productId: number, files: File[]): Observable<ApiResponse> {

const formData = new FormData();

for (let i = 0; i < files.length; i++) {

formData.append('files', files[i]);

}

return this.http.post<ApiResponse>(`${this.apiBaseUrl}/maubangs/uploads/${productId}`, formData);

}

// Phương thức xóa hình ảnh sản phẩm theo ID

deleteProductImage(id: number): Observable<any> {

debugger

return this.http.delete<string>(`${this.apiBaseUrl}/maubang\_image/${id}`);

}

}

1. orderdetail

@Injectable({

providedIn: 'root',

})

// Dịch vụ OrderDetailService để quản lý các thao tác liên quan đến chi tiết đơn hàng

export class OrderDetailService {

private apiUrl = `${environment.apiBaseUrl}/order\_details`; // URL cơ sở của API cho chi tiết đơn hàng

constructor(private http: HttpClient) {}

// Phương thức đặt hàng với dữ liệu đơn hàng

placeOrder(orderData: OrderDetailDTO): Observable<ApiResponse> {

// Gửi yêu cầu đặt hàng

return this.http.post<ApiResponse>(this.apiUrl, orderData);

}

// Phương thức lấy đơn hàng theo ID

getOrderById(orderId: number): Observable<ApiResponse> {

const url = `${environment.apiBaseUrl}/orders/${orderId}`;

return this.http.get<ApiResponse>(url);

}

// Phương thức lấy sản phẩm theo danh sách ID sản phẩm

getProductsByIds(productIds: number[]): Observable<ApiResponse> {

const params = new HttpParams().set('ids', productIds.join(','));

return this.http.get<ApiResponse>(`${environment.apiBaseUrl}/order\_details/by-ids`, { params });

}

}

1. order

@Injectable({

providedIn: 'root',

})

// Dịch vụ OrderService để quản lý các thao tác liên quan đến đơn hàng

export class OrderService {

private apiUrl = `${environment.apiBaseUrl}/orders`; // URL cơ sở của API cho đơn hàng

private apiGetAllOrders = `${environment.apiBaseUrl}/orders/get-orders-by-keyword`; // URL API để lấy tất cả đơn hàng

constructor(private http: HttpClient) {}

// Phương thức đặt hàng với dữ liệu đơn hàng

placeOrder(orderData: OrderDTO): Observable<ApiResponse> {

// Gửi yêu cầu đặt hàng

return this.http.post<ApiResponse>(this.apiUrl, orderData);

}

// Phương thức lấy đơn hàng theo ID

getOrderById(orderId: number): Observable<ApiResponse> {

const url = `${environment.apiBaseUrl}/orders/${orderId}`;

return this.http.get<ApiResponse>(url);

}

// Phương thức lấy tất cả đơn hàng với từ khóa, trang và giới hạn số lượng đơn hàng trên mỗi trang

getAllOrders(keyword: string, page: number, limit: number): Observable<ApiResponse> {

const params = new HttpParams()

.set('keyword', keyword)

.set('page', page.toString())

.set('limit', limit.toString());

return this.http.get<ApiResponse>(this.apiGetAllOrders, { params });

}

// Phương thức cập nhật đơn hàng dựa trên ID đơn hàng và dữ liệu đơn hàng cập nhật

updateOrder(orderId: number, orderData: OrderDTO): Observable<ApiResponse> {

const url = `${environment.apiBaseUrl}/orders/${orderId}`;

return this.http.put<ApiResponse>(url, orderData);

}

// Phương thức xóa đơn hàng dựa trên ID đơn hàng

deleteOrder(orderId: number): Observable<ApiResponse> {

const url = `${environment.apiBaseUrl}/orders/${orderId}`;

return this.http.delete<ApiResponse>(url);

}

}

1. product

@Injectable({

providedIn: 'root'

})

// Dịch vụ ProductService để quản lý các thao tác liên quan đến sản phẩm

export class ProductService {

private apiBaseUrl = environment.apiBaseUrl; // URL cơ sở của API

localStorage?: Storage;

constructor(

private http: HttpClient,

@Inject(DOCUMENT) private document: Document

) {

this.localStorage = document.defaultView?.localStorage;

}

// Phương thức lấy thông tin sản phẩm dựa trên ID sản phẩm

getProductsById(productId: number): Observable<ApiResponse> {

return this.http.get<ApiResponse>(`${this.apiBaseUrl}/products/${productId}`);

}

// Phương thức lấy thông tin sản phẩm dựa trên danh sách ID sản phẩm

getProductsByIds(productIds: number[]): Observable<ApiResponse> {

const params = new HttpParams().set('ids', productIds.join(','));

return this.http.get<ApiResponse>(`${this.apiBaseUrl}/products/by-ids`, { params });

}

// Phương thức thêm mới một sản phẩm

insertProduct(insertCassetteDTO: InsertCassetteDTO): Observable<ApiResponse> {

// Thêm một sản phẩm mới

return this.http.post<ApiResponse>(`${this.apiBaseUrl}/products`, insertCassetteDTO);

}

// Phương thức lưu thông tin OTP, ID và tên vào Local Storage

saveIdOtpNameToLocal(id: number, otp: number, name: string): void {

this.localStorage?.setItem('otp', JSON.stringify(otp));

this.localStorage?.setItem('id', JSON.stringify(id));

this.localStorage?.setItem('name', JSON.stringify(name));

}

// Phương thức lưu thông tin mật A và mật B vào Local Storage

saveMatAMatBToLocal(matA: string, matB: string): void {

this.localStorage?.setItem('matA', JSON.stringify(matA));

this.localStorage?.setItem('matB', JSON.stringify(matB));

}

// Phương thức lấy ID từ Local Storage

getIdFromLocal(): number {

const mauBangId = Number(this.localStorage?.getItem('id'));

return mauBangId;

}

// Phương thức lấy OTP từ Local Storage

getOtpFromLocal(): number {

const otp = Number(this.localStorage?.getItem('otp'));

return otp;

}

// Phương thức lấy tên từ Local Storage

getNameFromLocal(): string {

const name = String(this.localStorage?.getItem('name'));

return name;

}

// Phương thức xóa dữ liệu sản phẩm từ Local Storage

removeProductFromLocalStorage(): void {

try {

// Xóa dữ liệu sản phẩm từ local storage bằng cách sử dụng key

this.localStorage?.removeItem('id');

this.localStorage?.removeItem('otp');

this.localStorage?.removeItem('name');

this.localStorage?.removeItem('matA');

this.localStorage?.removeItem('matB');

console.log('Product data removed from local storage.');

} catch (error) {

console.error('Error removing product data from local storage:', error);

// Xử lý lỗi khi cần thiết

}

}

}

1. role

@Injectable({

providedIn: 'root'

})

// Dịch vụ RoleService để quản lý các thao tác liên quan đến vai trò

export class RoleService {

private apiGetRoles = `${environment.apiBaseUrl}/roles`; // URL API để lấy danh sách vai trò

constructor(private http: HttpClient) {}

// Phương thức lấy danh sách vai trò

getRoles(): Observable<ApiResponse> {

return this.http.get<ApiResponse>(this.apiGetRoles);

}

}

1. token

@Injectable({

providedIn: 'root',

})

// Dịch vụ TokenService để quản lý token và các thao tác liên quan

export class TokenService {

private readonly TOKEN\_KEY = 'access\_token'; // Key để lưu trữ token trong localStorage

private jwtHelperService = new JwtHelperService(); // Đối tượng dùng để giải mã và kiểm tra token

localStorage?: Storage; // Đối tượng lưu trữ localStorage

constructor(@Inject(DOCUMENT) private document: Document) {

this.localStorage = document.defaultView?.localStorage;

}

// Phương thức lấy token từ localStorage

getToken(): string {

return this.localStorage?.getItem(this.TOKEN\_KEY) ?? '';

}

// Phương thức thiết lập token vào localStorage

setToken(token: string): void {

this.localStorage?.setItem(this.TOKEN\_KEY, token);

}

// Phương thức lấy ID người dùng từ token

getUserId(): number {

const token = this.getToken();

if (!token) {

return 0;

}

const userObject = this.jwtHelperService.decodeToken(token);

return 'userId' in userObject ? parseInt(userObject['userId']) : 0;

}

// Phương thức xóa token khỏi localStorage

removeToken(): void {

this.localStorage?.removeItem(this.TOKEN\_KEY);

}

// Phương thức kiểm tra xem token có hết hạn không

isTokenExpired(): boolean {

if (this.getToken() == null) {

return false;

}

return this.jwtHelperService.isTokenExpired(this.getToken()!);

}

}

1. user

@Injectable({

providedIn: 'root'

})

// Dịch vụ UserService để quản lý các thao tác liên quan đến người dùng

export class UserService {

private apiRegister = `${environment.apiBaseUrl}/users/register`; // URL API để đăng ký người dùng

private apiLogin = `${environment.apiBaseUrl}/users/login`; // URL API để đăng nhập

private apiUserDetail = `${environment.apiBaseUrl}/users/details`; // URL API để lấy thông tin chi tiết người dùng

private httpUtilService = inject(HttpUtilService); // Dịch vụ HttpUtilService để tạo header cho các yêu cầu HTTP

localStorage?: Storage; // Đối tượng lưu trữ localStorage

private apiConfig = {

headers: this.httpUtilService.createHeaders(), // Cấu hình header cho các yêu cầu HTTP

};

constructor(

@Inject(DOCUMENT) private document: Document,

private http: HttpClient

) {

this.localStorage = document.defaultView?.localStorage;

}

// Phương thức đăng ký người dùng

register(registerDTO: RegisterDTO): Observable<ApiResponse> {

return this.http.post<ApiResponse>(this.apiRegister, registerDTO, this.apiConfig);

}

// Phương thức đăng nhập

login(loginDTO: LoginDTO): Observable<ApiResponse> {

return this.http.post<ApiResponse>(this.apiLogin, loginDTO, this.apiConfig);

}

// Phương thức lấy thông tin chi tiết người dùng

getUserDetail(token: string): Observable<ApiResponse> {

const headers = new HttpHeaders({

'Content-Type': 'application/json',

Authorization: `Bearer ${token}`

});

return this.http.post<ApiResponse>(this.apiUserDetail, {}, { headers });

}

// Phương thức cập nhật thông tin người dùng

updateUserDetail(token: string, updateUserDTO: UpdateUserDTO): Observable<ApiResponse> {

let userResponse = this.getUserResponseFromLocalStorage();

return this.http.put<ApiResponse>(`${this.apiUserDetail}/${userResponse?.id}`, updateUserDTO, {

headers: new HttpHeaders({

'Content-Type': 'application/json',

Authorization: `Bearer ${token}`

})

})

}

// Phương thức lưu thông tin người dùng vào localStorage

saveUserResponseToLocalStorage(userResponse?: UserResponse) {

try {

if (userResponse == null || !userResponse) {

return;

}

const userResponseJSON = JSON.stringify(userResponse);

this.localStorage?.setItem('user', userResponseJSON);

console.log('User response saved to local storage.');

} catch (error) {

console.error('Error saving user response to local storage:', error);

}

}

// Phương thức lấy thông tin người dùng từ localStorage

getUserResponseFromLocalStorage(): UserResponse | null {

try {

const userResponseJSON = this.localStorage?.getItem('user');

if (userResponseJSON == null || userResponseJSON == undefined) {

return null;

}

const userResponse = JSON.parse(userResponseJSON!);

console.log('User response retrieved from local storage.');

return userResponse;

} catch (error) {

console.error('Error retrieving user response from local storage:', error);

return null;

}

}

// Phương thức xóa thông tin người dùng khỏi localStorage

removeUserFromLocalStorage(): void {

try {

this.localStorage?.removeItem('user');

console.log('User data removed from local storage.');

} catch (error) {

console.error('Error removing user data from local storage:', error);

}

}

// Phương thức lấy danh sách người dùng

getUsers(params: { page: number, limit: number, keyword: string }): Observable<ApiResponse> {

const url = `${environment.apiBaseUrl}/users`;

return this.http.get<ApiResponse>(url, { params: params });

}

// Phương thức thay đổi trạng thái người dùng

toggleUserStatus(params: { userId: number, enable: boolean }): Observable<ApiResponse> {

const url = `${environment.apiBaseUrl}/users/block/${params.userId}/${params.enable ? '1' : '0'}`;

return this.http.put<ApiResponse>(url, null, this.apiConfig);

}

}

## Header

* 1. html

<div class="header">

<div class="frame-9949">

<div class="rectangle-1"></div>

<div class="spring-2024-collection">Cassette</div>

</div>

<div class="frame-9950">

<div class="rectangle-2"></div>

<button class="rectangle-5" title="trangchu" [routerLink]="['/trangchu']"></button>

<div class="menu"></div>

<button class="shopping-cart" title="giohang" [routerLink]="['/hoadon']"></button>

<!-- Hiển thị nút đăng xuất và thông tin người dùng nếu có -->

<ng-container \*ngIf="userResponse">

<li>

<span><button class="account-circle" [routerLink]="['/user-profile']"></button></span>

<span></span><button class="dang-xuat" (click)="dangxuat()">Đăng xuất</button>

</li>

</ng-container>

<!-- Hiển thị nút đăng nhập nếu không có thông tin người dùng -->

<ng-container \*ngIf="!userResponse">

<button class="account-circle" [routerLink]="['/login']"></button>

</ng-container>

</div>

</div>

* 1. ts

@Component({

selector: 'app-header',

templateUrl: './header.component.html',

styleUrls: ['./header.component.css'],

})

export class HeaderComponent implements OnInit {

userResponse?: UserResponse | null; // Biến lưu trữ thông tin người dùng

isPopoverOpen = false; // Biến xác định trạng thái của popover

activeNavItem: number = 0; // Biến lưu trữ index của mục hiện tại được chọn trong menu

constructor(

private dialogRef: MatDialog, // Dịch vụ để mở dialog

private userService: UserService, // Dịch vụ để quản lý thông tin người dùng

private tokenService: TokenService, // Dịch vụ để quản lý token

private productService: ProductService, // Dịch vụ để quản lý sản phẩm

private router: Router // Dịch vụ để điều hướng đến các route khác trong ứng dụng

) {}

ngOnInit() {

// Khi component được khởi tạo, lấy thông tin người dùng từ localStorage

this.userResponse = this.userService.getUserResponseFromLocalStorage();

}

// Phương thức đăng xuất người dùng

dangxuat(): void {

// Xóa thông tin người dùng và token khỏi localStorage

this.userService.removeUserFromLocalStorage();

this.tokenService.removeToken();

// Xóa thông tin sản phẩm khỏi localStorage

this.productService.removeProductFromLocalStorage();

// Cập nhật lại thông tin người dùng

this.userResponse = this.userService.getUserResponseFromLocalStorage();

}

// Phương thức để thiết lập mục hiện tại được chọn trong menu

setActiveNavItem(index: number) {

this.activeNavItem = index;

}

}

## Trangchu

* 1. html

<div class="trang-ch-ch-a-ng-nh-p">

<!-- Header Component -->

<app-header></app-header>

<!-- Main Content -->

<div class="main">

<div class="main-2">

<div class="rectangle-35"></div>

<div class="t-vi-t-l-n-b-c-th-b-ng-m-nh-c">

TỰ VIẾT LÊN BỨC THƯ BẰNG ÂM NHẠC

</div>

<div class="th-a-s-c-ch-n-outfit-cho-b-ng">

THỎA SỨC CHỌN OUTFIT CHO BĂNG

</div>

<div class="frame-9948">

<div class="rectangle-16"></div>

<!-- Button for HƯỚNG DẪN -->

<button class="component-5" (click)="openHuongDan()">

<div class="rectangle-15"></div>

<div class="h-ng-d-n">HƯỚNG DẪN</div>

</button>

<!-- Button for TẠO MIXTAPE -->

<button class="component-52" [routerLink]="['/chonbiangoai']">

<div class="rectangle-15"></div>

<div class="h-ng-d-n">TẠO MIXTAPE</div>

</button>

<!-- Description -->

<div class="thu-b-ng-cassette-v-i-ngu-n-l-c-c-b-i-h-t-b-n-t-ch-n">

Thu băng cassette với nguồn là các bài hát bạn tự chọn

</div>

<div class="ch-n-nh-c-thu-b-ng">CHỌN NHẠC THU BĂNG</div>

</div>

<div class="rectangle-33"></div>

</div>

<div class="main-1">

<div class="frame-9951">

<div class="rectangle-34"></div>

<div class="rectangle-7"></div>

</div>

<div class="ph-ng-mixtape">PHÒNG MIXTAPE</div>

<div class="b-n-s-vi-t-g-l-n-m-t-b-c-th-b-ng-m-thanh">

Bạn sẽ viết gì lên một bức thư bằng âm thanh

</div>

<div class="frame-9953">

<div class="rectangle-8"></div>

<div class="frame-9952">

<div class="vector">

<!-- Button for TẠO MIXTAPE NGAY -->

<button class="component-4" [routerLink]="['/chonbiangoai']">

<div class="rectangle-152"></div>

<div class="t-o-mixtape-ngay">TẠO MIXTAPE NGAY</div>

</button>

</div>

</div>

<div class="vector-1">

<div class="m-t-b-ng">Một băng</div>

<div class="mixtape-l-g">MIXTAPE LÀ GÌ?</div>

</div>

</div>

</div>

</div>

</div>

* 1. ts

import { Component } from '@angular/core';

import { MatDialog } from '@angular/material/dialog';

import { RouterModule } from '@angular/router';

import { FooterComponent } from '../footer/footer.component';

import { HeaderComponent } from '../header/header.component';

import { HuongdanComponent } from '../huongdan/huongdan.component';

@Component({

selector: 'app-trangchu',

standalone: true,

imports: [HeaderComponent,FooterComponent,RouterModule],

templateUrl: './trangchu.component.html',

styleUrl: './trangchu.component.css'

})

export class TrangchuComponent {

isSlidePanelOpen = false;

constructor(private dialogRef: MatDialog){}

openHuongDan(){

this.dialogRef.open(HuongdanComponent);

}

}

## Chonngoaihinh

* 1. html

<div class="chon-bia-ngoai-main">

<app-header></app-header>

<div class="main">

<app-header-mixtape></app-header-mixtape>

<div class="main-container">

<span class="bang-se-trong-nhu">BĂNG SẼ TRÔNG NHƯ NÀY</span>

@for(product of products; track product.id){

@if(product.id == selectedMB){

<img [src]="product.url" class="rectangle">

}

}

<div class="line"></div>

<div class="rectangle-1"></div>

<div class="radio">

<div class="state-default">

<div class="form-check active" style="margin: auto; padding-bottom: 25px;">

<input checked

class="form-check-input" type="radio" name="flexRadioDefault" id="flexRadioDefault1"

[value]="'2'"

[checked]="isSelectedOption(2)"

[(ngModel)]="selectedOption" >

</div>

<div class="frame-2">

<span class="do-tui-minh-viet">Do tụi mình viết , kí cả tặng</span>

</div>

</div>

</div>

<span class="ai-se-viet-ten">AI SẼ VIẾT TÊN VÀ TRACKLIST?</span

><span class="dat-ten-cho-mixtape">ĐẶT TÊN CHO MIXTAPE:</span>

<div class="text-inputs-placeholder">

<div class="input-group mb-3">

<input

type="text"

class="form-control"

placeholder="Vi Du: AKA"

aria-label="Recipient's username"

aria-describedby="basic-addon2"

[(ngModel)]="name" >

</div>

</div>

<span class="label-viet-len"

>(Tên mixtape sẽ được viết lên label dán trên băng)</span

><span class="tracklist-viet-len"

>(Tracklist mixtape sẽ được viết lên label dán trên băng)</span

>

<div class="radio-picked">

<div class="form-check" style="margin: auto; padding-bottom: 25px;">

<input

class="form-check-input"

type="radio"

name="flexRadioDefault"

id="flexRadioDefault1"

[value]="'1'"

[checked]="isSelectedOption(1)"

[(ngModel)]="selectedOption"

/>

</div>

<span class="write-yourself">Để mình tự viết</span>

</div>

<div class="frame-4">

<button class="button-newest" (click)="save(selectedMB,selectedOption,name)" >

<span class="label-5">CHỌN NHẠC BỎ VÔ</span>

<div class="chevron-right">

<div class="chevron-right-6"><div class="vector"></div></div>

</div>

</button>

<div class="rectangle-7"></div>

</div>

<div class="frame-8">

<span class="mau-bang">MẪU BĂNG</span>

<table>

<tbody>

<tr>

@for(product of products; track product.id){

<td>

<button (click)="selectMB(product.id)"><img [src]="product.url" alt="Product Image" class="ellipse"></button>

</td>

}

</tr>

</tbody>

</table>

</div>

</div>

</div>

</div>

* 1. ts

@Component({

selector: 'app-chonbiangoai',

standalone: true,

imports: [HeaderComponent,FooterComponent,HeaderMixtapeComponent,RouterModule,FormsModule],

templateUrl: './chonbiangoai.component.html',

styleUrl: './chonbiangoai.component.css'

})

export class ChonbiangoaiComponent implements OnInit{

name:string='';

selectedOption: number = 1;

products: MauBang[] = [];

selectedMB:number=0;

currentPage: number = 0;

itemsPerPage: number = 12;

pages: number[] = [];

totalPages:number = 0;

visiblePages: number[] = [];

keyword:string = "";

constructor(

private maubangService: MauBangService,

private productService: ProductService,

private router: Router,

) {

}

ngOnInit() {

this.getProducts(this.keyword, this.currentPage, this.itemsPerPage);

this.productService.saveIdOtpNametoLocal(this.selectedMB,this.selectedOption,this.name)

}

getProducts(keyword: string, page: number, limit: number) {

debugger;

this.maubangService.getProducts(keyword, page, limit).subscribe({

next: (apiresponse: ApiResponse) => {

debugger;

const response = apiresponse.data;

response.products.forEach((product: MauBang) => {

product.url = `${environment.apiBaseUrl}/maubangs/images/${product.thumbnail}`;

});

this.products = response.products;

},

complete: () => {

debugger;

},

error: (error: HttpErrorResponse) => {

debugger;

console.error(error?.error?.message ?? '');

}

});

}

save(id:number,otp:number,name:string){

this.productService.saveIdOtpNametoLocal(id,otp,name);

this.router.navigate(['/themnhac2'])

}

selectMB(productId:number){

this.selectedMB = productId;

}

isSelectedOption(option: number): boolean {

return this.selectedOption === option;

}

}

## Themnhac

* 1. html

<div class="themnhac2-main">

<!-- Phần header -->

<app-header></app-header>

<div class="main">

<!-- Header Mixtape -->

<app-header-mixtape></app-header-mixtape>

<div class="frame-9966">

<div class="rectangle-28"></div>

<div class="frame-72">

<!-- Nút chọn playlist -->

<button class="active-left" [routerLink]="['/themnhac1']">

<div class="segmented-bg"></div>

<div class="label">Chọn playlist</div>

</button>

<!-- Nút tải âm thanh cá nhân -->

<button class="inactive-inner" [routerLink]="['/themnhac2']">

<div class="segmented-bg2"></div>

<div class="label2">Tải âm thanh của bạn</div>

</button>

</div>

<!-- Đường gạch phân cách -->

<div class="line-3"></div>

<!-- Nút thêm vào giỏ hàng -->

<div class="button-newest" [routerLink]="['/themvaogio']">

<div class="label3">Thêm vào giỏ</div>

<div class="chevron-right">

<div class="chevron-right2"></div>

</div>

</div>

<!-- Hiển thị playlist đã chọn -->

<div class="nh-c-b-n-ch-n-playlist">Nhạc bạn đã chọn: Playlist #$</div>

</div>

</div>

</div>

* 1. ts

@Component({

selector: 'app-themnhac1',

standalone: true,

imports: [HeaderComponent,FooterComponent,HeaderMixtapeComponent,RouterModule],

templateUrl: './themnhac1.component.html',

styleUrl: './themnhac1.component.css'

})

export class Themnhac1Component {

}

## Themvaogio

* 1. html

<div class="themvaogio-main">

<!-- Phần header -->

<app-header></app-header>

<div class="main">

<!-- Header Mixtape -->

<app-header-mixtape></app-header-mixtape>

<div class="main-container">

<!-- Nút Mixtape đã hoàn thành -->

<button class="mixtape-button">MIXTAPE ĐÃ HOÀN THÀNH</button>

<!-- Hình ảnh bìa -->

<img [src]="bia" class="rectangle">

<!-- Hiển thị nhạc đã chọn -->

<span class="selected-music">Nhạc bạn đã chọn: Nhac cua ban</span>

<!-- Phần chọn số lượng mixtape -->

<div class="component">

<button class="group-button">

<span class="span-1">

<input type="text" style="width: 130px;" [(ngModel)]="quantity">

</span>

<div class="rectangle-div"></div>

</button>

<span class="choose-quantity-span">Chọn số lượng mixtape bạn muốn phát hành:</span>

</div>

<!-- Nút mua ngay -->

<button class="newest-button" (click)="buyNow()">

<span class="buy-now">MUA NGAY</span>

<div class="chevron-right">

<div class="chevron-right-1"><div class="vector"></div></div>

</div>

</button>

<!-- Nút thêm vào giỏ hàng -->

<button class="newest-button-2" (click)="addToCart()" >

<span class="add-to-cart">THÊM VÀO GIỎ VÀ TIẾP TỤC MUA HÀNG</span>

<div class="chevron-right-3">

<div class="chevron-right-4"><div class="vector-5"></div></div>

</div>

</button>

<!-- Đường gạch phân cách -->

<div class="rectangle-6"></div>

</div>

</div>

</div>

* 1. ts

@Component({

selector: 'app-themvaogio',

standalone: true,

imports: [HeaderComponent,FooterComponent,HeaderMixtapeComponent, RouterModule,CommonModule, FormsModule],

templateUrl: './themvaogio.component.html',

styleUrl: './themvaogio.component.css'

})

export class ThemvaogioComponent implements OnInit {

product?: Product;

productId: number = 0;

quantity: number = 1;

bia:string='';

isPressedAddToCart:boolean = false;

constructor(

private productService: ProductService,

private maubangService: MauBangService,

private cartService: CartService,

private activatedRoute: ActivatedRoute,

private router: Router,

) {}

ngOnInit() {

// Lấy productId từ URL

const idParam = this.activatedRoute.snapshot.paramMap.get('id');

debugger

if (idParam !== null) {

this.productId = +idParam;

}

if (!isNaN(this.productId)) {

this.productService.getProductsById(this.productId).subscribe({

next: (apiResponse: ApiResponse) => {

// Lấy danh sách ảnh sản phẩm và thay đổi URL

const response = apiResponse.data

debugger

this.bia= `${environment.apiBaseUrl}/products/images/${response.bia}`;

debugger

this.product = response

},

error: (error: HttpErrorResponse) => {

console.error('Error fetching product:', error.message);

}

});

} else {

console.error('Invalid productId:', idParam);

}

}

addToCart(): void {

debugger

this.isPressedAddToCart = true;

if (this.product) {

this.cartService.addToCart(this.product.id, this.quantity);

this.router.navigate(['/trangchu']);

} else {

// Xử lý khi product là null

console.error('Không thể thêm sản phẩm vào giỏ hàng vì product là null.');

}

}

buyNow(): void {

if(this.isPressedAddToCart == false) {

this.addToCart();

}

this.router.navigate(['/hoadon']);

}

}

## hoadon

* 1. html

<div class="hoadon-main">

<div class="main">

<app-header></app-header>

<form [formGroup]="orderForm"> <!-- Thêm form và gán formGroup -->

<div class="row">

<div class="col-md-5">

<h2 class="product-header">Thông tin người nhận</h2>

<div class="mb-3">

<label for="fullname" class="form-label">Họ và tên</label>

<input type="text" formControlName="fullname" class="form-control" id="fullname" [class.is-invalid]="orderForm.get('fullname')!.invalid && orderForm.get('fullname')!.touched">

@if(orderForm.get('fullname')!.invalid && orderForm.get('fullname')!.touched) {

<div class="invalid-feedback">

Họ và tên là trường bắt buộc.

</div>

}

</div>

<div class="mb-3">

<label for="email" class="form-label">Email</label>

<input type="text" formControlName="email" class="form-control" id="email" [class.is-invalid]="orderForm.get('email')!.invalid && orderForm.get('email')!.touched">

@if(orderForm.get('email')!.invalid && orderForm.get('email')!.touched) {

<div class="invalid-feedback">

{{ orderForm.get('email')!.hasError('email') ? 'Email không hợp lệ' : 'Email là trường bắt buộc' }}

</div>

}

</div>

<div class="mb-3">

<label for="phone" class="form-label">Số điện thoại</label>

<input type="text" class="form-control" formControlName="phone\_number" placeholder="Số điện thoại" required id="phone" [class.is-invalid]="orderForm.get('phone\_number')!.invalid && orderForm.get('phone\_number')!.touched">

@if(orderForm.get('phone\_number')!.invalid && orderForm.get('phone\_number')!.touched) {

<div class="invalid-feedback">

Số điện thoại là trường bắt buộc và ít nhất 6 ký tự.

</div>

}

</div>

<div class="mb-3">

<label for="address" class="form-label">Địa chỉ</label>

<input type="text" class="form-control" formControlName="address" placeholder="Địa chỉ" required id="address" [class.is-invalid]="orderForm.get('address')!.invalid && orderForm.get('address')!.touched">

@if(orderForm.get('address')!.invalid && orderForm.get('address')!.touched) {

<div class="invalid-feedback">

Địa chỉ là trường bắt buộc và ít nhất 5 ký tự.

</div>

}

</div>

<div class="mb-3">

<label for="note" class="form-label">Ghi chú</label>

<input type="text" class="form-control" formControlName="note" placeholder="Ghi chú" required id="note">

</div>

<div class="mb-3">

<label for="shippingMethod">Phương thức vận chuyển</label>

<select class="form-control" id="shippingMethod" formControlName="shipping\_method">

<option value="express">Nhanh (Express)</option>

<option value="normal">Thường (Normal)</option>

</select>

</div>

<div class="mb-3">

<label for="paymentMethod">Phương thức thanh toán</label>

<select class="form-control" id="paymentMethod" formControlName="payment\_method">

<option value="cod">Thanh toán khi nhận hàng (COD)</option>

<option value="other">Thanh toán khác</option>

</select>

</div>

</div>

<div class="col-md-7">

<h2 class="product-order">Sản phẩm đã đặt hàng</h2>

<table>

<tbody>

<ng-container \*ngFor="let item of cartItems; let i = index">

<tr>

<td>

<div class="product-info">

<img [src]="item.product.bia" alt="Product Image" class="product-image">

<span class="product-name">{{ item.product.name }}</span>

</div>

</td>

<td>

<div class="product-quantity">

<div class="border-wrapper">

<button (click)="decreaseQuantity(i)">-</button>

{{ item.quantity }}

<button (click)="increaseQuantity(i)">+</button>

</div>

</div>

</td>

<td>{{ item.product.price | number:'1.2-2' }}</td>

<td>{{ (item.product.price \* item.quantity) | number:'1.2-2' }}</td>

<td>

<button class="btn

* 1. ts

@Component({

selector: 'app-hoadon',

standalone: true,

imports: [HeaderComponent, FooterComponent, RouterModule, ReactiveFormsModule, CommonModule],

templateUrl: './hoadon.component.html',

styleUrl: './hoadon.component.css'

})

export class HoadonComponent {

private cartService = inject(CartService);

private productService = inject(ProductService);

private orderService = inject(OrderService);

private tokenService = inject(TokenService);

private formBuilder = inject(FormBuilder);

private router = inject(Router);

orderForm: FormGroup; // Đối tượng FormGroup để quản lý dữ liệu của form

cartItems: { product: Product, quantity: number }[] = [];

totalAmount: number = 0; // Tổng tiền

couponApplied: boolean = false;

cart: Map<number, number> = new Map();

orderData: OrderDTO = {

user\_id: 0, // Thay bằng user\_id thích hợp

fullname: '', // Khởi tạo rỗng, sẽ được điền từ form

email: '', // Khởi tạo rỗng, sẽ được điền từ form

phone\_number: '', // Khởi tạo rỗng, sẽ được điền từ form

address: '', // Khởi tạo rỗng, sẽ được điền từ form

status: 'pending',

note: '', // Có thể thêm trường ghi chú nếu cần

total\_money: 0, // Sẽ được tính toán dựa trên giỏ hàng và mã giảm giá

payment\_method: 'cod', // Mặc định là thanh toán khi nhận hàng (COD)

shipping\_method: 'express', // Mặc định là vận chuyển nhanh (Express)

cart\_items: []

};

constructor(private dialogRef: MatDialog) {

// Tạo FormGroup và các FormControl tương ứng

this.orderForm = this.formBuilder.group({

fullname: ['', Validators.required], // fullname là FormControl bắt buộc

email: ['', [Validators.email]], // Sử dụng Validators.email cho kiểm tra định dạng email

phone\_number: ['', [Validators.required, Validators.minLength(6)]], // phone\_number bắt buộc và ít nhất 6 ký tự

address: ['', [Validators.required, Validators.minLength(5)]], // address bắt buộc và ít nhất 5 ký tự

note: [''],

couponCode: [''],

shipping\_method: ['express'],

payment\_method: ['cod']

});

}

ngOnInit(): void {

debugger

//this.cartService.clearCart();

this.orderData.user\_id = this.tokenService.getUserId();

// Lấy danh sách sản phẩm từ giỏ hàng

debugger

this.cart = this.cartService.getCart();

const productIds = Array.from(this.cart.keys()); // Chuyển danh sách ID từ Map giỏ hàng

// Gọi service để lấy thông tin sản phẩm dựa trên danh sách ID

debugger

if(productIds.length === 0) {

return;

}

this.productService.getProductsByIds(productIds).subscribe({

next: (apiResponse: ApiResponse) => {

debugger

const products: Product[] = apiResponse.data

// Lấy thông tin sản phẩm và số lượng từ danh sách sản phẩm và giỏ hàng

this.cartItems = productIds.map((productId) => {

debugger

const product = products.find((p) => p.id === productId);

if (product) {

product.bia = `${environment.apiBaseUrl}/products/images/${product.bia}`;

}

return {

product: product!,

quantity: this.cart.get(productId)!

};

});

console.log('haha');

},

complete: () => {

debugger;

this.calculateTotal()

},

error: (error: HttpErrorResponse) => {

debugger;

console.error(error?.error?.message ?? '');

}

});

## login

* 1. html

<div class="themnhac2-main">

<div class="main">

<app-header></app-header>

<div class="container">

<div class="row">

<div class="col-md-6 offset-md-3">

<div class="login-form mx-auto">

<h2 class="login-header">Đăng nhập</h2>

<div class="form-group">

<label for="phoneNumber">Phone</label>

<input type="text" class="form-control"

[(ngModel)]="phoneNumber"

(input)="onPhoneNumberChange()"

name="phone"

(focus)="loginForm.form.markAsTouched()"

id="phoneNumber">

<div class="divider-nospace"></div>

</div>

<div class="mt-5"></div>

<div class="form-group password-field">

<label for="password">Mật khẩu</label>

<input type="{{ showPassword ? 'text' : 'password' }}"

[(ngModel)]="password"

name="password"

placeholder="Ít nhất 3 ký tự"

class="form-control" id="password">

<i class="password-toggle fas" [ngClass]="showPassword ?

'fa-eye' : 'fa-eye-slash'" (click)="togglePassword()"></i>

<div class="divider-nospace"></div>

</div>

<div class="form-group">

<div class="form-check checkbox-text">

<span style="display: flex;">

<input type="checkbox"

class="form-check-input"

id="remember"

[(ngModel)]="rememberMe">

<label class="form-check-label text-start" for="remember">

Ghi nhớ đăng nhập</label>

</span>

<a href="#" class="register-link">Quên mật khẩu</a>

</div>

</div>

<div class="form-group">

<select class="form-control access-right"

[(ngModel)]="selectedRole">

<option selected disabled>Quyền đăng nhập</option>

@for(role of roles; track role.name) {

<option [ngValue]="role">

{{ role.name }}

</option>

}

</select>

</div>

<button

type="button"

(click)="login()"

class="login-button">Đăng nhập</button>

<div class="divider"></div>

<p class="text-center">Bạn chưa đăng ký? <span>

<button class="register-link"

(click)="createAccount()">Tạo tài khoản</button>

</span></p>

</div>

</div>

</div>

* 1. ts

@Component({

selector: 'app-login',

standalone: true,

imports: [FormsModule, CommonModule, HeaderComponent, FooterComponent, RouterModule],

templateUrl: './login.component.html',

styleUrl: './login.component.css'

})

export class LoginComponent implements OnInit {

@ViewChild('loginForm') loginForm!: NgForm;

phoneNumber: string = '';

password: string = '';

showPassword: boolean = false;

roles: Role[] = []; // Mảng roles

rememberMe: boolean = true;

selectedRole: Role | undefined; // Biến để lưu giá trị được chọn từ dropdown

userResponse?: UserResponse;

constructor(

private dialogRef: MatDialog,

private router: Router,

private activatedRoute: ActivatedRoute,

private userService: UserService,

private tokenService: TokenService,

private roleService: RoleService,

private cartService: CartService

) {}

ngOnInit(): void {

// Gọi API lấy danh sách roles và lưu vào biến roles

this.roleService.getRoles().subscribe({

next: (apiResponse: ApiResponse) => {

const roles = apiResponse.data;

this.roles = roles;

this.selectedRole = roles.length > 0 ? roles[0] : undefined;

},

complete: () => {},

error: (error: HttpErrorResponse) => {

console.error(error?.error?.message ?? '');

}

});

}

login() {

const loginDTO: LoginDTO = {

phone\_number: this.phoneNumber,

password: this.password,

role\_id: this.selectedRole?.id ?? 1

};

this.userService.login(loginDTO).subscribe({

next: (apiResponse: ApiResponse) => {

const { token } = apiResponse.data;

if (this.rememberMe) {

this.tokenService.setToken(token);

this.userService.getUserDetail(token).subscribe({

next: (apiResponse2: ApiResponse) => {

this.userResponse = {

...apiResponse2.data,

date\_of\_birth: new Date(apiResponse2.data.date\_of\_birth)

};

this.userService.saveUserResponseToLocalStorage(this.userResponse);

if (this.userResponse?.role.name == 'admin') {

this.router.navigate(['/admin']);

} else if (this.userResponse?.role.name == 'user') {

this.router.navigate(['/trangchu']);

}

},

complete: () => {

this.cartService.refreshCart();

},

error: (error: HttpErrorResponse) => {

console.error(error?.error?.message ?? '');

}

});

}

},

complete: () => {},

error: (error: HttpErrorResponse) => {

console.error(error?.error?.message ?? '');

}

});

}

togglePassword() {

this.showPassword = !this.showPassword;

}

createAccount() {

this.router.navigate(['/register']);

}

}

## dangki

* 1. html

<div class="chon-bia-ngoai-main">

<div class="container">

<form #registerForm="ngForm">

<div class="row">

<div class="col-md-6 offset-md-3">

<div class="register-form mx-auto">

<h2 class="register-header">Đăng ký</h2>

<div class="form-group">

<label for="phoneNumber">Phone's number</label>

<input type="text"

class="form-control"

[(ngModel)]="phoneNumber"

(input)="onPhoneNumberChange()"

name="phone"

(focus)="registerForm.form.markAsTouched()"

required id="phoneNumber">

<div class="divider-nospace"></div>

@if(phoneNumber.length < 6 && registerForm.form.touched) {

<p class="text-danger">Phone phải có ít nhất 6 ký tự</p>

}

</div>

<div class="mt-3"></div>

<div class="form-group password-field">

<label for="password">Mật khẩu</label>

<input type="password"

[(ngModel)]="password"

name="password"

placeholder="Ít nhất 3 ký tự" required

class="form-control" id="password">

<div class="divider-nospace"></div>

@if(password.length < 3 && registerForm.form.touched) {

<p class="text-danger">

Mật khẩu phải có ít nhất 3 ký tự

</p>

}

</div>

<div class="mt-3"></div>

<div class="form-group password-field">

<label for="retype-password">Gõ lại mật khẩu</label>

<input type="password"

placeholder="Ít nhất 3 ký tự"

[(ngModel)]="retypePassword"

name="retypePassword"

(input)="registerForm.form.markAsTouched(); checkPasswordsMatch()"

required

class="form-control" id="retype-password">

<div class="divider-nospace"></div>

@if(retypePassword.length < 3 && registerForm.form.touched) {

<p class="text-danger">

Mật khẩu phải có ít nhất 3 ký tự

</p>

}

@if(registerForm.controls['retypePassword'].hasError('passwordMismatch')

&& registerForm.form.touched) {

<p class="text-danger">

Mật khẩu không khớp

</p>

}

</div>

<div class="mt-3"></div>

<div class="form-group">

<label for="full-name">Họ và tên</label>

<input type="text"

[(ngModel)]="fullName"

name="fullName"

class="form-control" required

id="full-name">

<div class="divider-nospace"></div>

</div>

<div class="form-group">

<label for="date-of-birth">Ngày tháng năm sinh</label>

<!-- Thêm [(ngModel)] để thực hiện 2-way data binding -->

<input type="date"

class="form-control"

required

id="date-of-birth"

[(ngModel)]="dateOfBirth"

name="dateOfBirth"

(input)="registerForm.form.markAsTouched(); checkAge()">

<div class="divider-nospace"></div>

@if(registerForm.controls['dateOfBirth'].hasError('invalidAge')

&& registerForm.form.touched){

<p class="text-danger" >

Bạn chưa đủ 18 tuổi

</p>

}

</div>

<div class="mt-3"></div>

<div class="form-group">

<label for="address">Địa chỉ</label>

<input type="text"

[(ngModel)]="address"

name="address"

class="form-control"

id="address">

<div class="divider-nospace"></div>

</div>

<div class="form-group">

<div class="form-check checkbox-text">

<span style="display: flex;">

<input type="checkbox"

[(ngModel)]="isAccepted"

name="isAccepted"

class="form-check-input" id="agree">

<label class="form-check-label text-start" for="agree">

Tôi đồng ý với các điều khoản và điều kiện</label>

</span>

</div>

</div>

<button type="button" style="

flex-wrap: nowrap;

gap: 16px;

position: relative;

width: 300px;

height: 56px;

padding: 0 8px 0 8px;

cursor: pointer;

background: #be2623;

border: 1px solid #1370b9;

z-index: 14;

border-radius: 5px;"

(click)="register()"

class="register-button">Đăng ký</button>

<div class="divider"></div>

<p class="text-center">Bạn đã có tài khoản? <span>

<a class="register-link" [routerLink]="['/login']">Đăng nhập</a></span></p>

</div>

</div>

</div>

</form>

</div>

</div>

* 1. ts

@Component({

selector: 'app-dangki',

standalone: true,

imports: [FormsModule, RouterModule],

templateUrl: './dangki.component.html',

styleUrl: './dangki.component.css'

})

export class DangkiComponent {

@ViewChild('registerForm') registerForm!: NgForm;

// Khai báo các biến tương ứng với các trường dữ liệu trong form

phoneNumber: string;

password: string;

retypePassword: string;

fullName: string;

address: string;

isAccepted: boolean;

dateOfBirth: Date;

constructor(private router: Router, private userService: UserService, private dialogRef: MatDialog) {

this.phoneNumber = '';

this.password = '';

this.retypePassword = '';

this.fullName = '';

this.address = '';

this.isAccepted = true;

this.dateOfBirth = new Date();

this.dateOfBirth.setFullYear(this.dateOfBirth.getFullYear() - 18);

}

onPhoneNumberChange() {

console.log(`Phone typed: ${this.phoneNumber}`);

//how to validate ? phone must be at least 6 characters

}

register() {

const registerDTO: RegisterDTO = {

"fullname": this.fullName,

"phone\_number": this.phoneNumber,

"address": this.address,

"password": this.password,

"retype\_password": this.retypePassword,

"date\_of\_birth": this.dateOfBirth,

"role\_id": 2

};

this.userService.register(registerDTO).subscribe({

next: (apiResponse: ApiResponse) => {

const confirmation = window.confirm('Đăng ký thành công, mời bạn đăng nhập. Bấm "OK" để chuyển đến trang đăng nhập.');

if (confirmation) {

this.router.navigate(['/']);

}

},

complete: () => {},

error: (error: HttpErrorResponse) => {

console.error(error?.error?.message ?? '');

}

});

}

checkPasswordsMatch() {

if (this.password !== this.retypePassword) {

this.registerForm.form.controls['retypePassword'].setErrors({ 'passwordMismatch': true });

} else {

this.registerForm.form.controls['retypePassword'].setErrors(null);

}

}

checkAge() {

if (this.dateOfBirth) {

const today = new Date();

const birthDate = new Date(this.dateOfBirth);

let age = today.getFullYear() - birthDate.getFullYear();

const monthDiff = today.getMonth() - birthDate.getMonth();

if (monthDiff < 0 || (monthDiff === 0 && today.getDate() < birthDate.getDate())) {

age--;

}

if (age < 18) {

this.registerForm.form.controls['dateOfBirth'].setErrors({ 'invalidAge': true });

} else {

this.registerForm.form.controls['dateOfBirth'].setErrors(null);

}

}

}

closeAll() {

this.dialogRef.closeAll();

}

}

## user-profile

* 1. html

<div class="hoadon-main">

<div class="main">

<app-header></app-header>

<form [formGroup]="userProfileForm">

<div class="row">

<div class="col-md-3 mx-auto">

<h2 class="product-header">Thông tin người dùng</h2>

<div class="mb-3">

<label for="fullname" class="form-label">Họ và tên</label>

<input type="text"

formControlName="fullname"

class="form-control"

id="fullname"

[class.is-invalid]="userProfileForm.get('fullname')!.invalid && userProfileForm.get('fullname')!.touched">

@if(userProfileForm.get('fullname')!.invalid &&

userProfileForm.get('fullname')!.touched) {

<div

class="invalid-feedback">

Họ và tên là trường bắt buộc.

</div>

}

</div>

<div class="mb-3">

<label for="address" class="form-label">Address</label>

<input type="text"

formControlName="address"

class="form-control"

id="address"

[class.is-invalid]="userProfileForm.get('address')!.invalid && userProfileForm.get('address')!.touched">

</div>

<div class="mb-3">

<label for="date\_of\_birth" class="form-label">Ngày sinh</label>

<input type="date"

formControlName="date\_of\_birth"

class="form-control"

id="date\_of\_birth" >

</div>

<div class="mb-3">

<label for="password" class="form-label">Mật khẩu</label>

<input type="password"

class="form-control"

formControlName="password"

placeholder="Mật khẩu"

required

id="password"

[class.is-invalid]="userProfileForm.get('password')!.invalid

&& userProfileForm.get('password')!.touched">

@if(userProfileForm.get('password')!.invalid

&& userProfileForm.get('password')!.touched) {

<div class="invalid-feedback">

Mật khẩu ít nhất 3 ký tự.

</div>

}

</div>

<div class="mb-3">

<label for="retype\_password" class="form-label">Nhập lại mật khẩu</label>

<input type="password"

class="form-control"

formControlName="retype\_password"

placeholder="Nhập lại mật khẩu"

required

id="retype\_password"

[class.is-invalid]="userProfileForm.get('retype\_password')!.invalid

&& userProfileForm.get('retype\_password')!.touched">

@if(userProfileForm.hasError('passwordMismatch')) {

<div class="invalid-feedback">

Mật khẩu không khớp.

</div>

}

</div>

<div class="text-center">

<button type="button"

(click)="save()"

class="btn btn-primary">Lưu thay đổi</button>

</div>

</div>

</div>

</form>

</div>

</div>

* 1. ts

@Component({

templateUrl: './user-profile.component.html',

styleUrls: ['./user-profile.component.css']

})

export class UserProfileComponent implements OnInit {

userResponse?: UserResponse;

userProfileForm: FormGroup;

token: string = '';

constructor(

private formBuilder: FormBuilder,

private activatedRoute: ActivatedRoute,

private userService: UserService,

private router: Router,

private tokenService: TokenService,

) {

// Khởi tạo form user profile với các trường và kiểm tra hợp lệ

this.userProfileForm = this.formBuilder.group({

fullname: [''],

address: ['', [Validators.minLength(3)]],

password: ['', [Validators.minLength(3)]],

retype\_password: ['', [Validators.minLength(3)]],

date\_of\_birth: [new Date()],

}, {

validators: this.passwordMatchValidator // Hàm validator tùy chỉnh để kiểm tra mật khẩu trùng khớp

});

}

ngOnInit(): void {

// Lấy token của người dùng từ token service

this.token = this.tokenService.getToken();

// Lấy chi tiết người dùng bằng cách sử dụng token

this.userService.getUserDetail(this.token).subscribe({

next: (response: any) => {

// Điền thông tin người dùng vào form

this.userResponse = {

...response,

date\_of\_birth: new Date(response.date\_of\_birth),

};

this.userProfileForm.patchValue({

fullname: this.userResponse?.fullname ?? '',

address: this.userResponse?.address ?? '',

date\_of\_birth: this.userResponse?.date\_of\_birth.toISOString().substring(0, 10),

});

this.userService.saveUserResponseToLocalStorage(this.userResponse);

},

error: (error: HttpErrorResponse) => {

console.error(error?.error?.message ?? '');

}

})

}

// Hàm validator tùy chỉnh để kiểm tra mật khẩu và mật khẩu gõ lại trùng khớp

passwordMatchValidator(): ValidatorFn {

return (formGroup: AbstractControl): ValidationErrors | null => {

const password = formGroup.get('password')?.value;

const retypedPassword = formGroup.get('retype\_password')?.value;

if (password !== retypedPassword) {

return { passwordMismatch: true };

}

return null;

};

}

// Phương thức lưu thay đổi thông tin người dùng

save(): void {

if (this.userProfileForm.valid) {

// Tạo DTO để cập nhật thông tin người dùng từ giá trị của form

const updateUserDTO: UpdateUserDTO = {

fullname: this.userProfileForm.get('fullname')?.value,

address: this.userProfileForm.get('address')?.value,

password: this.userProfileForm.get('password')?.value,

retype\_password: this.userProfileForm.get('retype\_password')?.value,

date\_of\_birth: this.userProfileForm.get('date\_of\_birth')?.value

};

// Gọi service để cập nhật thông tin người dùng

this.userService.updateUserDetail(this.token, updateUserDTO)

.subscribe({

next: (response: any) => {

// Xóa thông tin người dùng từ local storage và chuyển hướng về trang đăng nhập

this.userService.removeUserFromLocalStorage();

this.tokenService.removeToken();

this.router.navigate(['/login']);

},

error: (error: HttpErrorResponse) => {

console.error(error?.error?.message ?? '');

}

});

} else {

// Nếu form không hợp lệ, kiểm tra xem có lỗi mật khẩu không khớp không

if (this.userProfileForm.hasError('passwordMismatch')) {

console.error('Mật khẩu và mật khẩu gõ lại chưa chính xác')

}

}

}

}

## admin-dashboard

* 1. html

<div class="container-fluid">

<header class="header bg-success text-white d-flex justify-content-between">

<h1 class="header-title">Dashboard Header</h1>

<!-- Logout Button -->

<button class="btn btn-danger btn-sm" (click)="logout()">Logout</button>

</header>

<div class="row">

<nav id="sidebar" class="col-md-2 col-lg-2 d-md-block bg-light sidebar">

<div class="position-sticky">

<ul class="nav flex-column">

<li class="nav-item">

<a class="nav-link"

(click)="showAdminComponent('orders')"

[routerLink]="['/admin/orders']"

routerLinkActive="active">

<i class="fab fa-first-order icon"></i>Orders

</a>

</li>

<li class="nav-item">

<a class="nav-link"

(click)="showAdminComponent('products')"

[routerLink]="['/admin/products']"

routerLinkActive="active">

<i class="fab fa-codepen icon"></i>Products

</a>

</li>

<li class="nav-item">

<a class="nav-link"

(click)="showAdminComponent('users')"

[routerLink]="['/admin/users']"

routerLinkActive="active">

<i class="fab fa-codepen icon"></i>Users

</a>

</li>

</ul>

</div>

</nav>

<div class="col-md-10 col-lg-10 overflow-auto">

<router-outlet></router-outlet>

</div>

</div>

</div>

* 1. ts

@Component({

selector: 'app-admin',

templateUrl: './admin.component.html',

styleUrls: [

'./admin.component.scss',

],

standalone: true,

imports: [

CommonModule,

RouterModule,

],

})

export class AdminComponent implements OnInit {

//adminComponent: string = 'orders';

userResponse?:UserResponse | null;

private userService = inject(UserService);

private tokenService = inject(TokenService);

private router = inject(Router);

ngOnInit() {

this.userResponse = this.userService.getUserResponseFromLocalStorage();

// Default router

debugger

if (this.router.url === '/admin') {

this.router.navigate(['/admin/orders']);

}

}

logout() {

this.userService.removeUserFromLocalStorage();

this.tokenService.removeToken();

this.userResponse = this.userService.getUserResponseFromLocalStorage();

this.router.navigate(['/']);

}

showAdminComponent(componentName: string): void {

debugger

if (componentName === 'orders') {

this.router.navigate(['/admin/orders']);

} else if (componentName === 'products') {

this.router.navigate(['/admin/products']);

} else if (componentName === 'users') {

this.router.navigate(['/admin/users']);

}

}

}

/\*\*

npm install --save font-awesome

angular.json:

"styles": [

"node\_modules/font-awesome/css/font-awesome.min.css"

],

\*/

## admin-orders

* 1. html

<div class="container-fluid">

<h1>Orders Admin Page</h1>

<!--Đoạn này có 1 ô searchbox bên trái, 1 combobox bên phải -->

<div class="row mb-3">

<div class="col-md-8">

<input type="text"

class="form-control search-input" placeholder="Tìm đơn hàng"

[(ngModel)]="keyword">

</div>

<div class="col-md-4">

<button class="btn btn-primary" (click)="searchOrders()">Tìm kiếm</button>

</div>

</div>

<table class="table">

<thead>

<tr>

<th>ID</th>

<th>User ID</th>

<th>Full Name</th>

<th>Email</th>

<th>Phone Number</th>

<th>Address</th>

<th>Note</th>

<th>Order Date</th>

<th>Status</th>

<th>Total Money</th>

<!--

<th>Shipping Method</th>

<th>Shipping Address</th>

<th>Shipping Date</th>

<th>Payment Method</th>

-->

<th>Action</th>

</tr>

</thead>

<tbody>

@for(order of orders; track order.id){

<tr>

<td>

{{ order.id }}</td>

<td>{{ order.user\_id }}</td>

<td>{{ order.fullname }}</td>

<td>{{ order.email }}</td>

<td>{{ order.phone\_number }}</td>

<td>{{ order.address }}</td>

<td>{{ order.note }}</td>

<td>{{ order.order\_date | date }}</td>

<td>{{ order.status }}</td>

<td>{{ order.total\_money }}</td>

<!--

<td>{{ order.shipping\_method }}</td>

<td>{{ order.shipping\_address }}</td>

<td>{{ order.shipping\_date | date }}</td>

<td>{{ order.payment\_method }}</td>

-->

<td>

<div style="display: flex; gap: 10px;">

<button

(click)="viewDetails(order)"

class="btn btn-primary">

View Details

</button>

<button class="btn btn-danger"

(click)="deleteOrder(order.id)"

>Delete</button>

</div>

</td>

</tr>

}

</tbody>

</table>

<div class="d-flex justify-content-center">

<nav aria-label="Page navigation">

<ul class="pagination">

@if(currentPage > 1) {

<li class="page-item" >

<a class="page-link"

(click)="onPageChange(1)">First</a>

</li>

<li class="page-item" >

<a class="page-link"

(click)="onPageChange(currentPage - 1)">Previous</a>

</li>

}

@for(page of visiblePages; track page) {

<ng-container>

<li class="page-item" [ngClass]="{'active': page === currentPage}">

<a class="page-link" (click)="onPageChange(page)">{{ page }}</a>

</li>

</ng-container>

}

@if(currentPage < totalPages) {

<li class="page-item">

<a class="page-link" (click)="onPageChange(currentPage + 1)">Next</a>

</li>

<li class="page-item">

<a class="page-link" (click)="onPageChange(totalPages)">Last</a>

</li>

}

</ul>

</nav>

</div>

</div>

* 1. ts

@Component({

selector: 'app-order-admin',

templateUrl: './order.admin.component.html',

styleUrls: ['./order.admin.component.scss'],

standalone: true,

imports: [

CommonModule,

FormsModule,

]

})

export class OrderAdminComponent implements OnInit{

orders: OrderResponse[] = [];

currentPage: number = 0;

itemsPerPage: number = 12;

pages: number[] = [];

totalPages:number = 0;

keyword:string = "";

visiblePages: number[] = [];

localStorage?:Storage;

constructor(

private orderService: OrderService,

private router: Router,

private route: ActivatedRoute,

private location: Location,

@Inject(DOCUMENT) private document: Document

) {

this.localStorage = document.defaultView?.localStorage;

}

ngOnInit(): void {

debugger

this.currentPage = Number(this.localStorage?.getItem('currentOrderAdminPage')) || 0;

this.getAllOrders(this.keyword, this.currentPage, this.itemsPerPage);

}

searchOrders() {

this.currentPage = 0;

this.itemsPerPage = 12;

//Mediocre Iron Wallet

debugger

this.getAllOrders(this.keyword.trim(), this.currentPage, this.itemsPerPage);

}

getAllOrders(keyword: string, page: number, limit: number) {

debugger

this.orderService.getAllOrders(keyword, page, limit).subscribe({

next: (apiResponse: ApiResponse) => {

debugger

this.orders = apiResponse.data.orders;

this.totalPages = apiResponse.data.totalPages;

this.visiblePages = this.generateVisiblePageArray(this.currentPage, this.totalPages);

},

complete: () => {

debugger;

},

error: (error: HttpErrorResponse) => {

debugger;

console.error(error?.error?.message ?? '');

}

});

}

onPageChange(page: number) {

debugger;

this.currentPage = page < 0 ? 0 : page;

this.localStorage?.setItem('currentOrderAdminPage', String(this.currentPage));

this.getAllOrders(this.keyword, this.currentPage, this.itemsPerPage);

}

generateVisiblePageArray(currentPage: number, totalPages: number): number[] {

const maxVisiblePages = 5;

const halfVisiblePages = Math.floor(maxVisiblePages / 2);

let startPage = Math.max(currentPage - halfVisiblePages, 1);

let endPage = Math.min(startPage + maxVisiblePages - 1, totalPages);

if (endPage - startPage + 1 < maxVisiblePages) {

startPage = Math.max(endPage - maxVisiblePages + 1, 1);

}

const visiblePages: number[] = [];

for (let i = startPage; i <= endPage; i++) {

visiblePages.push(i);

}

return visiblePages;

}

deleteOrder(id:number) {

const confirmation = window

.confirm('Are you sure you want to delete this order?');

if (confirmation) {

debugger

this.orderService.deleteOrder(id).subscribe({

next: (response: ApiResponse) => {

debugger

location.reload();

},

complete: () => {

debugger;

},

error: (error: HttpErrorResponse) => {

debugger;

console.error(error?.error?.message ?? '');

}

});

}

}

viewDetails(order:OrderResponse) {

debugger

this.router.navigate(['/admin/orders', order.id]);

}

}

## admin-users

* 1. html

<h1>sss</h1>

<table class="table table-striped">

<thead>

<tr>

<th>ID</th>

<th>Full Name</th>

<th>Phone</th>

<th>Address</th>

<th>State</th>

<th>Role</th>

<th>Action</th>

</tr>

</thead>

<tbody>

@for(user of users; track user.id) {

<tr>

<td>{{ user.id }}</td>

<td>{{ user.fullname }}</td>

<td>{{ user.phone\_number }}</td>

<td>{{ user.address }}</td>

<td>{{ user.is\_active ? 'Active' : 'Inactive' }}</td>

<td>{{ user.role.name }}</td>

<td>

<button class="btn" [ngClass]="{

'btn-danger': user.is\_active,

'btn-success': !user.is\_active

}" (click)="toggleUserStatus(user)">

{{ user.is\_active ? 'Block User' : 'Unblock User' }}

</button>

</td>

</tr>

}

</tbody>

</table>

* 1. ts

@Component({

selector: 'app-user.admin',

templateUrl: './user.admin.component.html',

styleUrl: './user.admin.component.scss',

standalone: true,

imports: [

CommonModule,

FormsModule,

]

})

export class UserAdminComponent implements OnInit{

userService = inject(UserService);

router = inject(Router)

route = inject(ActivatedRoute);

users: UserResponse[] = [];

currentPage: number = 0;

itemsPerPage: number = 12;

pages: number[] = [];

totalPages:number = 0;

visiblePages: number[] = [];

keyword:string = "";

localStorage?:Storage;

constructor(

//private location: Location,

@Inject(DOCUMENT) private document: Document

) {

this.localStorage = document.defaultView?.localStorage;

}

ngOnInit(): void {

this.currentPage = Number(this.localStorage?.getItem('currentUserAdminPage')) || 0;

this.getUsers(this.keyword, this.currentPage, this.itemsPerPage);

}

searchUsers() {

this.currentPage = 0;

this.itemsPerPage = 12;

this.getUsers(this.keyword.trim(), this.currentPage, this.itemsPerPage);

}

getUsers(keyword: string, page: number, limit: number) {

this.userService.getUsers({ keyword, page, limit }).subscribe({

next: (apiResponse: ApiResponse) => {

debugger

const response = apiResponse.data

this.users = response.users;

this.totalPages = response.totalPages;

this.visiblePages = this.generateVisiblePageArray(this.currentPage, this.totalPages);

},

complete: () => {

// Handle complete event

debugger

},

error: (error: HttpErrorResponse) => {

debugger;

console.error(error?.error?.message ?? '');

}

});

}

onPageChange(page: number) {

this.currentPage = page < 0 ? 0 : page;

this.localStorage?.setItem('currentUserAdminPage', String(this.currentPage));

this.getUsers(this.keyword, this.currentPage, this.itemsPerPage);

}

generateVisiblePageArray(currentPage: number, totalPages: number): number[] {

const maxVisiblePages = 5;

const halfVisiblePages = Math.floor(maxVisiblePages / 2);

let startPage = Math.max(currentPage - halfVisiblePages, 1);

let endPage = Math.min(startPage + maxVisiblePages - 1, totalPages);

if (endPage - startPage + 1 < maxVisiblePages) {

startPage = Math.max(endPage - maxVisiblePages + 1, 1);

}

return new Array(endPage - startPage + 1).fill(0)

.map((\_, index) => startPage + index);

}

// Hàm xử lý sự kiện khi thêm mới sản phẩm

insertUser() {

debugger

// Điều hướng đến trang detail-user với userId là tham số

this.router.navigate(['/admin/users/insert']);

}

// Hàm xử lý sự kiện khi sản phẩm được bấm vào

updateUser(userId: number) {

debugger

// Điều hướng đến trang detail-user với userId là tham số

this.router.navigate(['/admin/users/update', userId]);

}

toggleUserStatus(user: UserResponse) {

let confirmation: boolean;

if (user.is\_active) {

confirmation = window.confirm('Are you sure you want to block this user?');

} else {

confirmation = window.confirm('Are you sure you want to enable this user?');

}

if (confirmation) {

const params = {

userId: user.id,

enable: !user.is\_active

};

this.userService.toggleUserStatus(params).subscribe({

next: (response: any) => {

console.error('Block/unblock user successfully');

location.reload();

},

complete: () => {

// Handle complete event

},

error: (error: HttpErrorResponse) => {

debugger;

console.error(error?.error?.message ?? '');

}

});

}

}

}

## dtos

1. insert.cassette.dto.ts

import {

IsString,

IsNotEmpty,

IsPhoneNumber,

} from 'class-validator';

import { MauBang } from '../../model/maubang';

export class InsertCassetteDTO {

@IsString()

name: string;

@IsString()

matA:string;

@IsString()

matB:string;

otpName:number;

maubangid:number;

constructor(data: any) {

this.name = data.name;

this.matA = data.matA;

this.matB = data.matB;

this.otpName=data.otpName;

this.maubangid=data.maubangid;

}

}

1. cartitem.dto

import { IsNumber } from 'class-validator';

export class CartItemDTO {

@IsNumber()

product\_id: number;

@IsNumber()

quantity: number;

constructor(data: any) {

this.product\_id = data.product\_id;

this.quantity = data.quantity;

}

}

1. order-detail.dto

import { IsNumber,

IsString

} from 'class-validator';

export class OrderDetailDTO {

@IsNumber()

maubang\_id: number;

@IsNumber()

otpName: number;

@IsString()

matA:string;

@IsString()

matB:string;

@IsString()

name:string;

@IsNumber()

number\_product: number;

@IsNumber()

total\_money:number;

constructor(data: any) {

this.maubang\_id = data.maubang\_id;

this.otpName = data.otpName;

this.matA=data.matA;

this.matB=data.matB;

this.name=data.name;

this.number\_product=data.number\_product;

this.total\_money=data.total\_money

}

}

1. oder.dto

import { IsString,

IsNotEmpty,

IsPhoneNumber,

IsNumber, ArrayMinSize,

ValidateNested,

Length

} from 'class-validator';

import { Type } from 'class-transformer';

import { CartItemDTO } from './cart.item.dto';

export class OrderDTO {

user\_id: number;

fullname: string;

email: string;

phone\_number: string;

address: string;

status: string;

note: string;

total\_money?: number;

shipping\_method: string;

order\_date?: Date;

payment\_method: string;

cart\_items: { product\_id: number, quantity: number}[]; // Thêm cart\_items để lưu thông tin giỏ hàng

constructor(data: any) {

this.user\_id = data.user\_id;

this.fullname = data.fullname;

this.email = data.email;

this.status = data.status;

this.phone\_number = data.phone\_number;

this.address = data.address;

this.note = data.note;

this.order\_date = data.order\_date;

this.total\_money = data.total\_money;

this.shipping\_method = data.shipping\_method;

this.payment\_method = data.payment\_method;

this.cart\_items = data.cart\_items;

}

}

1. insert.product.dto

import {

IsString,

IsNotEmpty,

IsPhoneNumber,

} from 'class-validator';

export class InsertProductDTO {

@IsPhoneNumber()

name: string;

images: File[] = [];

constructor(data: any) {

this.name = data.name;

}

}

1. update.product.dto

import {

IsString,

IsNotEmpty,

IsPhoneNumber,

} from 'class-validator';

export class UpdateProductDTO {

@IsPhoneNumber()

name: string;

constructor(data: any) {

this.name = data.name;

}

}

1. login.dto

import {

IsString,

IsNotEmpty,

IsPhoneNumber,

IsDate

} from 'class-validator';

export class LoginDTO {

@IsPhoneNumber()

phone\_number: string;

@IsString()

@IsNotEmpty()

password: string;

role\_id: number;

constructor(data: any) {

this.phone\_number = data.phone\_number;

this.password = data.password;

this.role\_id = data.role\_id

}

}

1. register.dto

import {

IsString,

IsNotEmpty,

IsPhoneNumber,

IsDate

} from 'class-validator';

export class RegisterDTO {

@IsString()

fullname: string;

@IsPhoneNumber()

phone\_number: string;

@IsString()

@IsNotEmpty()

address: string;

@IsString()

@IsNotEmpty()

password: string;

@IsString()

@IsNotEmpty()

retype\_password: string;

@IsDate()

date\_of\_birth: Date;

role\_id: number = 1;

constructor(data: any) {

this.fullname = data.fullname;

this.phone\_number = data.phone\_number;

this.address = data.address;

this.password = data.password;

this.retype\_password = data.retype\_password;

this.date\_of\_birth = data.date\_of\_birth;

this.role\_id = data.role\_id || 1;

}

}

1. update.user.dto

export class UpdateUserDTO {

fullname: string;

address: string;

password: string;

retype\_password: string;

date\_of\_birth: Date;

constructor(data: any) {

this.fullname = data.fullname;

this.address = data.address;

this.password = data.password;

this.retype\_password = data.retype\_password;

this.date\_of\_birth = data.date\_of\_birth;

}

}

## responses

1. order.response

import { OrderDetail } from "../../model/order.detail";

export interface OrderResponse {

id: number;

user\_id: number;

fullname: string;

email: string;

phone\_number: string;

address: string;

note: string;

order\_date: Date; // Dạng chuỗi ISO 8601

status: string;

total\_money: number;

shipping\_method: string;

shipping\_address: string;

shipping\_date: Date; // Dạng chuỗi ISO 8601

payment\_method: string;

order\_details: OrderDetail[]; // Đảm bảo có một interface OrderDetail tương ứng

}

1. login.reponse

export interface LoginResponse {

message: string;

token: string;

}

1. user.response

import { Role } from "../../model/role";

export interface UserResponse {

id: number;

fullname: string;

phone\_number: string;

address:string;

is\_active: boolean;

date\_of\_birth: Date;

role: Role;

}

## models

1. maubang.ts

import { MauBangImage } from "./maubangimage";

export interface MauBang {

id: number;

name: string;

thumbnail: string;

url: string;

product\_images: MauBangImage[];

}

1. maubangimage.ts

export interface MauBangImage {

id: number;

image\_url: string;

}

1. order-details.ts

import {Order} from './order'

import { Product } from './product';

export interface OrderDetail {

id: number;

order: Order;

product:Product;

price: number;

number\_of\_products: number;

total\_money: number;

}

1. order.ts

import {OrderDetail} from './order.detail'

export interface Order {

id: number;

user\_id: number;

fullname: string;

email: string;

phone\_number: string;

address: string;

note: string;

order\_date: Date;

status: string;

total\_money: number;

shipping\_method: string;

shipping\_address: string;

shipping\_date: Date;

tracking\_number: string;

payment\_method: string;

active: boolean;

order\_details: OrderDetail[];

}

1. product.ts

export interface Product {

id: number;

name: string;

price: number;

otpName: number;

matA:string;

matB:string;

maubang\_id:number

bia:string;

}

1. role.ts

export interface Role {

id: number;

name: string;

}

1. user.ts

import { Role } from "./role";

export interface User {

id: number;

fullname: string;

phone\_number: string;

address: string;

password: string;

active: boolean;

date\_of\_birth: Date;

role: Role;

}

C. HƯỚNG DẪN SỬ DỤNG

# I Hướng dẫn sử dụng phần user

## Đăng nhập



1. Click vào button Account

A screenshot of a computer

Description automatically generated

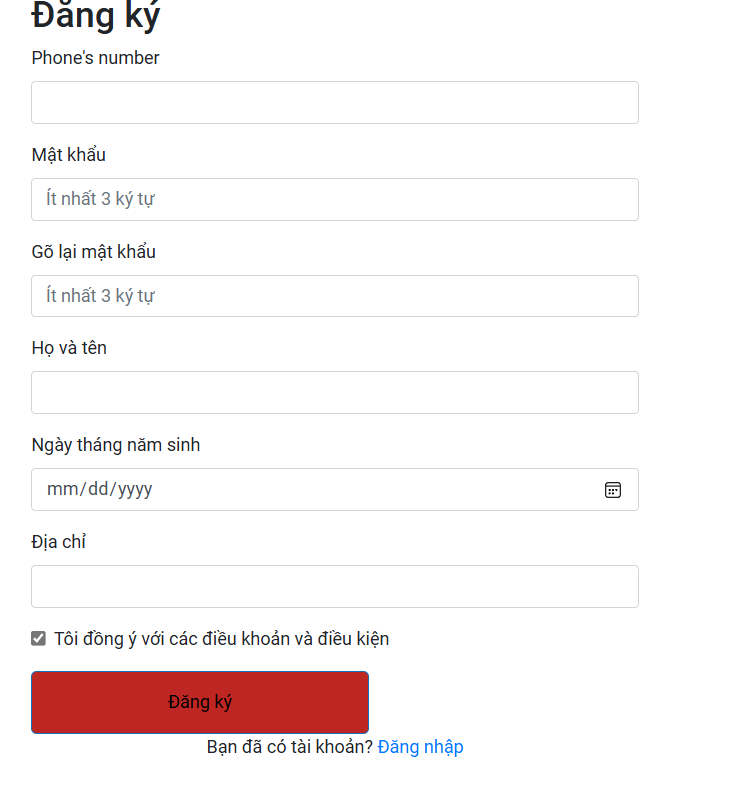
1. Điền tài khoản mật khẩu và Chọn quyền đăng nhập User
2. Nhấn button Đăng nhập

## Đăng kí

A screenshot of a computer

Description automatically generated

1. Nhấn button Tạo tài khoản



1. Điền form tạo tài khỏan và nhấn Đăng kí

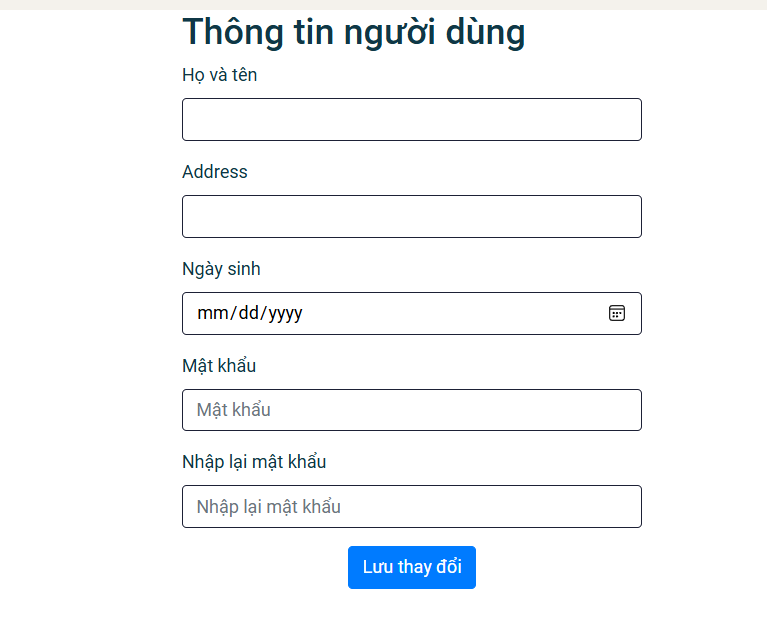
## Đăng xuất

A black arrow with black text

Description automatically generated

1. Click vào button Đăng xuất

## Chỉnh sửa thông tin cá nhân



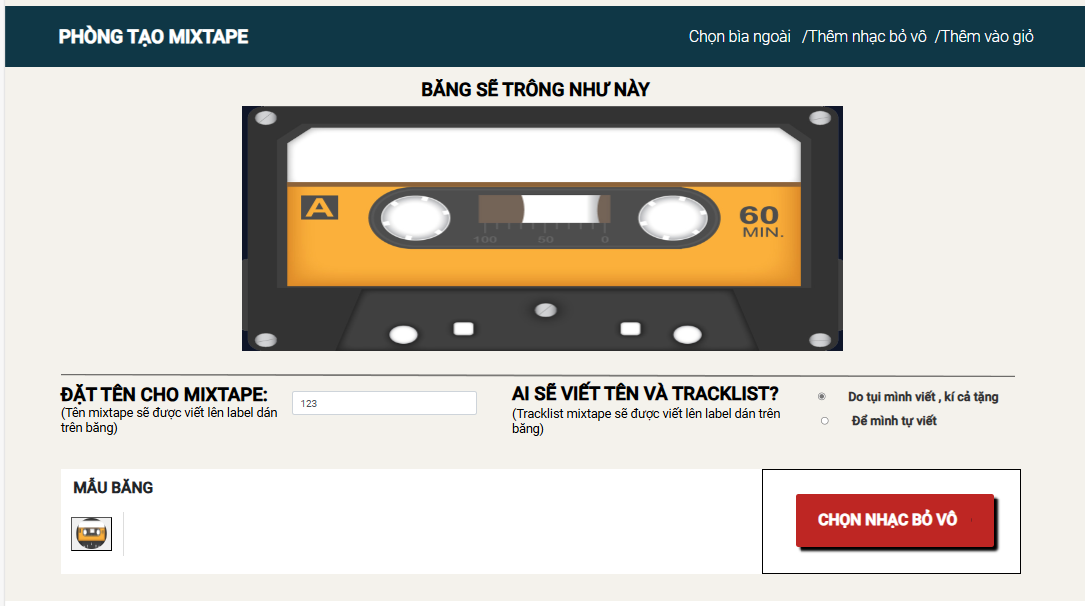
Sau khi đã đăng nhập:

1. Click vào button Acount:
2. Điền form thông tin người dùng
3. Click button Lưu

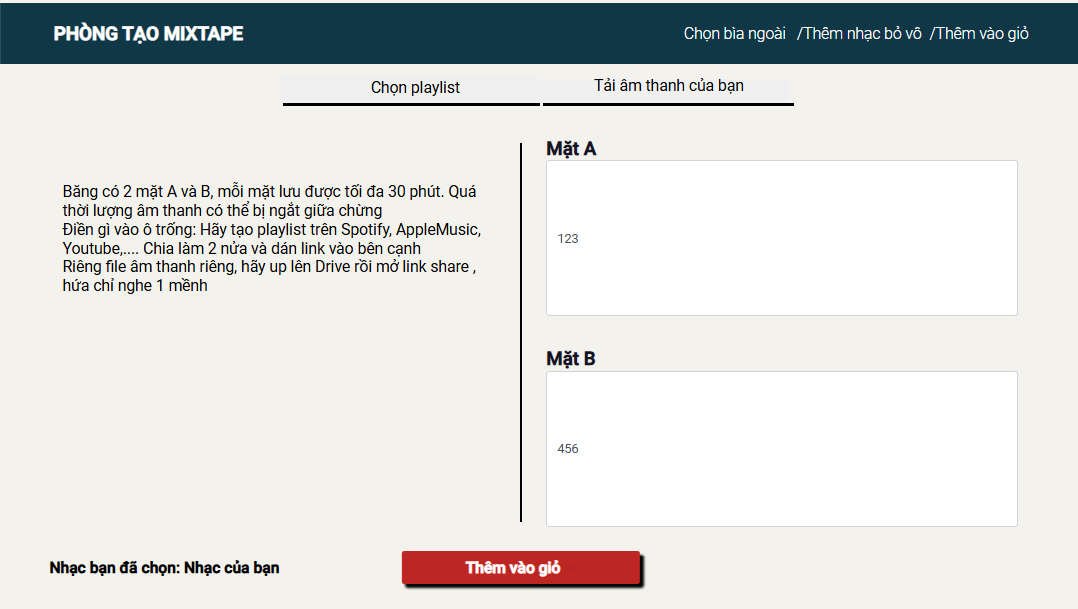
## Tạo cassette



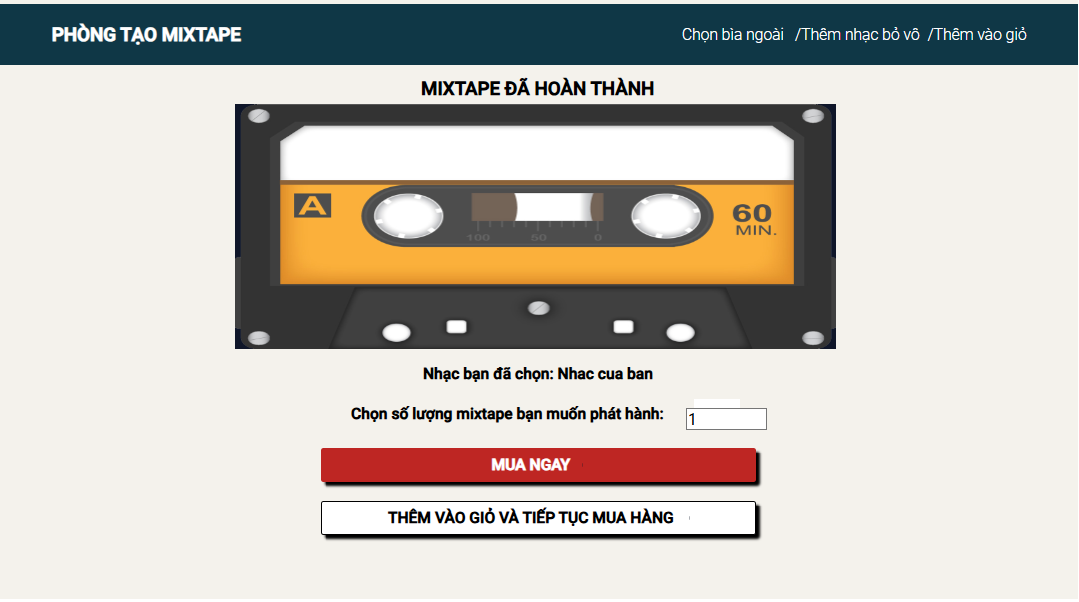
1. Nhấn button Tạo mixtape



1. Điền tên , chọn mẫu băng , chọn ai sẽ viết tên => Click button Chọn nhạc bỏ vô



1. Điền mặt A,mặt B => Click vào button Thêm vào giỏ



## Thêm giỏ hàng

A cassette tape with a black border

Description automatically generated

1. Thêm số lượng mixtape muốn làm
2. Nhấn thêm vào giỏ và tiếp tục mua hàng

## Xem giỏ hàng

A screenshot of a computer

Description automatically generated (1) Click button giỏ hàng

A screenshot of a computer

Description automatically generated

(2) Hiển thị các sản phẩm đã đặt

## Tạo đơn hàng

A screenshot of a computer

Description automatically generated

1. Điền form đặt hàng
2. Nhấn đặt hàng

# III Hướng dẫn sử dụng phần admin

## Đăng nhập

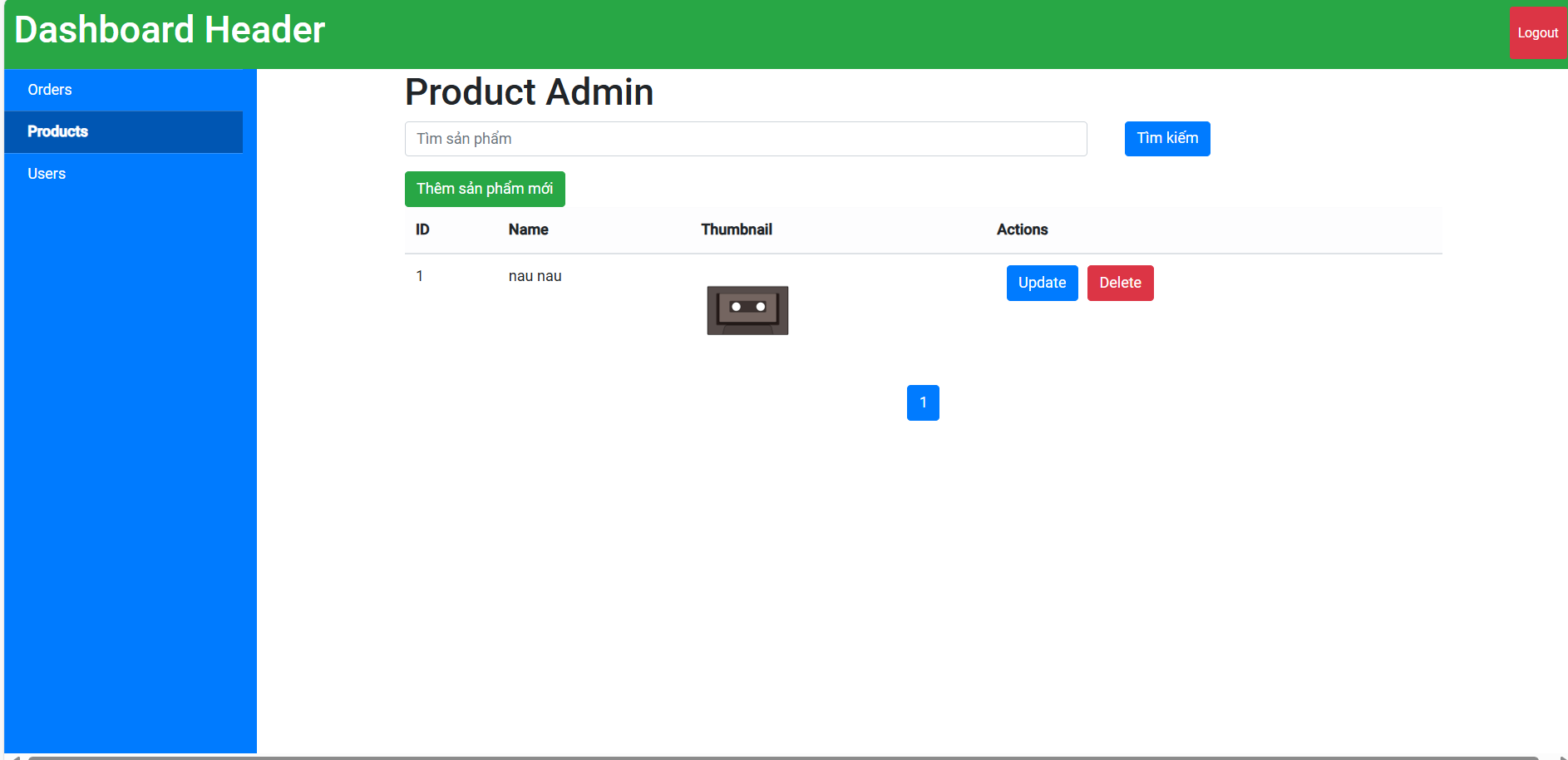
Tham khảo chức năng Đăng nhập user

## Đăng xuất

A red and green square with white text

Description automatically generated(1) Click button Logout

## Quản lý mẫu băng



1. Click button Products

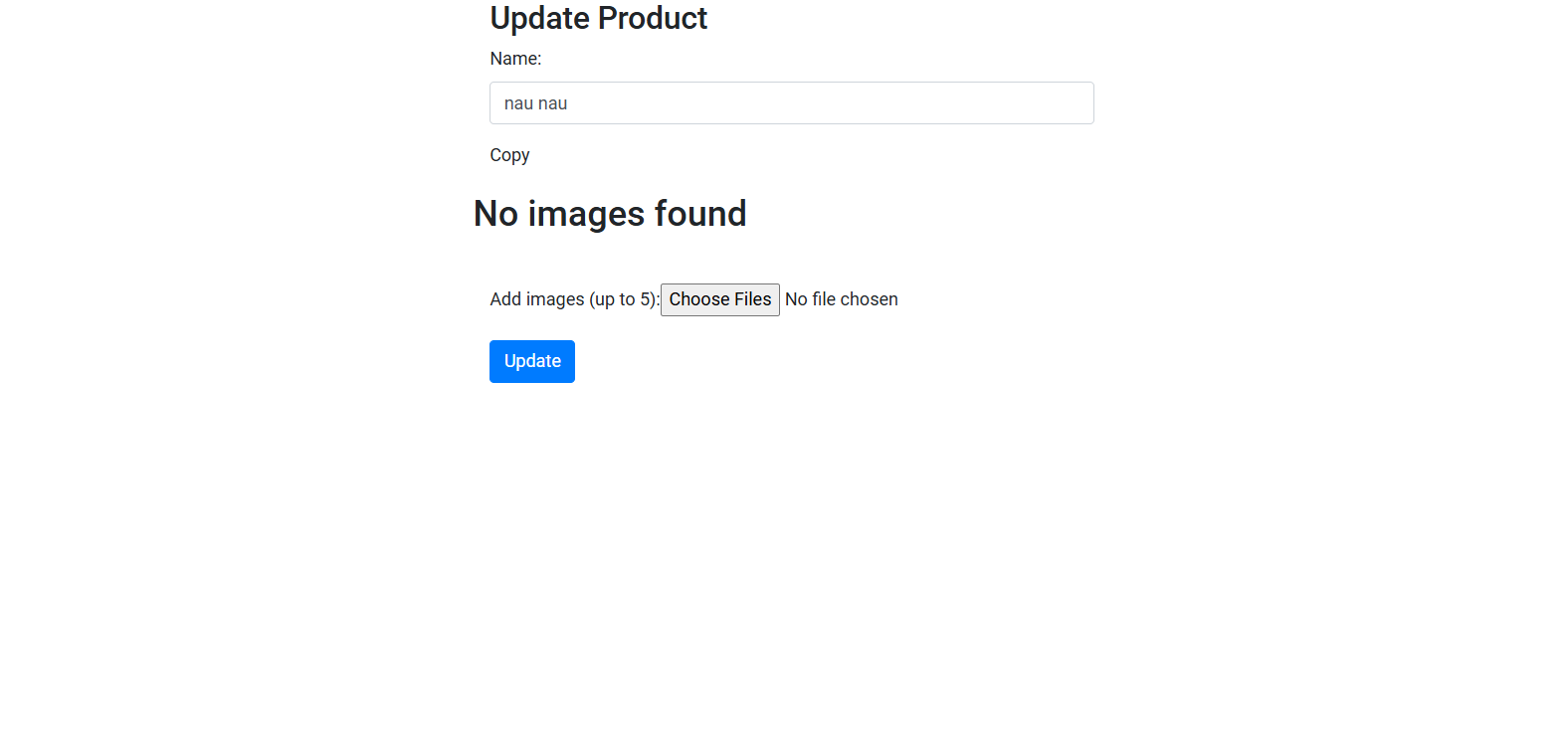
### 3.1 Thêm mới

A screenshot of a computer

Description automatically generated

1. Click button thêm sản phẩm
2. Điền thông tin sản phẩm
3. Nhấn Insert

### 3.2 Cập nhập



1. Click button Update
2. Điền thông tin sản phẩm
3. Nhấn Update

### 3.3 Xóa

A screenshot of a computer

Description automatically generated

1. Click button Delete
2. Nhấn Ok để xóa sản phẩm

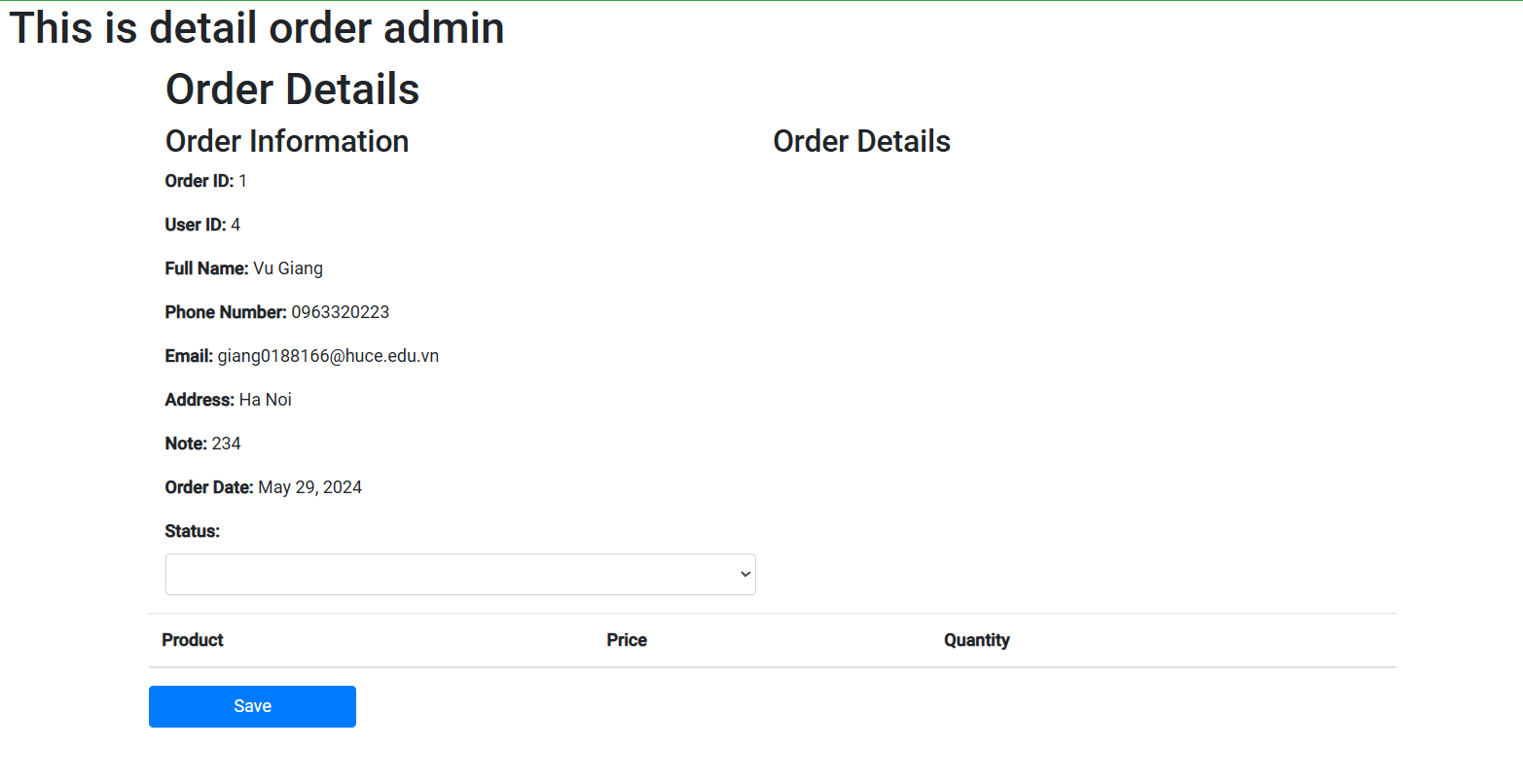
## Quản lý đơn hàng

A screenshot of a computer

Description automatically generated

1. Click button Order

### 4.1 Xem đơn hàng



1. Click button View Details
2. Chỉnh sửa trạng thái đơn hàng
3. Click Save

### 4.2 Xóa đơn hàng

A screenshot of a computer

Description automatically generated

1. Click button Delete
2. Nhấn Ok để xóa đơn hàng

## Báo cáo thống kê

N/A

## Quản lý user

A screenshot of a computer

Description automatically generated

1. Click button Users

### 6.1 Block user

A screenshot of a computer

Description automatically generated

1. Click button Block
2. Nhấn Ok để block user