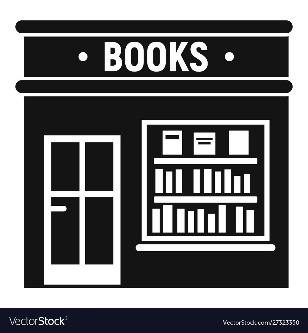
**Graphical user interface, text

Description automatically generated**





**BÁO CÁO DỰ ÁN**

***Website bán sách CityZen***

**Giảng viên hướng dẫn :** Thầy Lê Anh THú

**Môn học** **:** Java 5

**Lớp**   **:** IT16302

**Ngành :** Ứng dụng phần mềm

**Nhóm thực hiện :** Nhóm 1

Diệp Ông Đại Quang

Trần Gia Bảo

Trần Thanh Tùng

Nguyễn Hoàng Nhi

Ngô Ngọc Tài

Contents

[1. Giới thiệu dự án 3](#_Toc106697882)

[1.1. Giới thiệu CityZen & Website 3](#_Toc106697883)

[1.2. Yêu cầu của cửa hàng 3](#_Toc106697884)

[1.2.1. Yêu cầu về chức năng nghiệp vụ: 4](#_Toc106697885)

[1.2.2. Yêu cầu về bảo mật 4](#_Toc106697886)

[1.2.3. Yêu cầu về công nghệ: 4](#_Toc106697887)

[1.3. Lập kế hoạch dự án 4](#_Toc106697888)

[1.4. Bảng phân chia công việc 5](#_Toc106697889)

[2. Phân tích yêu cầu khách hàng 6](#_Toc106697890)

[2.1. Sơ đồ Use case 6](#_Toc106697891)

[2.2. Đặc tả yêu cầu hệ thống 8](#_Toc106697892)

[2.2.1. Quản lý hoá đơn 8](#_Toc106697893)

[2.2.2. Quản lý sản phẩm 8](#_Toc106697894)

[2.2.3. Quản lý tài khoản 9](#_Toc106697895)

[2.2.4. Chức năng nghiệp vụ mua hàng 9](#_Toc106697896)

[2.3. Mô hình ứng dụng triển khai 10](#_Toc106697897)

[2.4. Yêu cầu hệ thống 10](#_Toc106697898)

[3. Thiết kế ứng dụng 11](#_Toc106697899)

[3.1. Thiết kế cơ sở dữ liệu: 11](#_Toc106697900)

[3.1.1. Thực thể Categories: 11](#_Toc106697901)

[3.1.2. Thực thể Products 12](#_Toc106697902)

[3.1.3. Thực thể Accounts 13](#_Toc106697903)

[3.1.4. Thực thể Orders 13](#_Toc106697904)

[3.1.5. Thực thể Orders\_Detail 14](#_Toc106697905)

[3.1.6. Thực thể Verify\_Accounts 14](#_Toc106697906)

[3.2. Thiết kế giao diện 15](#_Toc106697907)

[3.2.1. Trang đăng nhập 15](#_Toc106697908)

[3.2.2. Trang đăng ký 16](#_Toc106697909)

[3.2.3. Trang quên mật khẩu 16](#_Toc106697910)

[3.2.4. Trang nhập mã xác nhận và đặt mật khẩu mới 17](#_Toc106697911)

[3.2.5. Trang chủ 18](#_Toc106697912)

[3.2.6. Trang cập nhật tài khoản 23](#_Toc106697913)

[3.2.7. Trang chi tiết sản phẩm 23](#_Toc106697914)

[3.2.8. Trang giỏ hàng 24](#_Toc106697915)

[3.2.9. Trang quản lý website dành cho quản trị viên 24](#_Toc106697916)

[3.2.10. Trang lịch sử mua hàng 27](#_Toc106697917)

[4. Thực hiện dự án 28](#_Toc106697918)

[4.1. Tạo giao diện với HTML, CSS, Javascript, Bootstrap: 28](#_Toc106697919)

[4.1.1. Trang đăng nhập: 28](#_Toc106697920)

[4.1.2. Trang đăng ký: 28](#_Toc106697921)

[4.1.3. Trang quên mật khẩu 29](#_Toc106697922)

[4.1.4. Trang nhập mã xác nhận và đặt lại mật khẩu 29](#_Toc106697923)

[4.1.5. Trang chủ 30](#_Toc106697924)

[4.1.6. Trang cập nhật tài khoản 35](#_Toc106697925)

[4.1.7. Trang chi tiết sản phẩm 36](#_Toc106697926)

[4.1.8. Trang giỏ hàng 36](#_Toc106697927)

[4.1.9. Trang quản lý website dành cho quản trị viên 38](#_Toc106697928)

[4.1.10. Trang lịch sử mua hàng 40](#_Toc106697929)

[4.2. Tạo cơ sở dữ liệu với SQL Server 41](#_Toc106697930)

[4.2.1. Tạo bảng Categories 41](#_Toc106697931)

[4.2.2. Tạo bảng Products 42](#_Toc106697932)

[4.2.3. Tạo bảng Accounts 42](#_Toc106697933)

[4.2.4. Tạo bảng Orders 42](#_Toc106697934)

[4.2.5. Tạo bảng Orders\_Detail 43](#_Toc106697935)

[4.2.6. Tạo bảng Verify\_Accounts 43](#_Toc106697936)

[4.3. Tạo dự án web với Spring Boot 43](#_Toc106697937)

[4.3.1. Khai báo thư viện vào pom.xml 43](#_Toc106697938)

[4.3.2. Tuỳ chỉnh application.properties 45](#_Toc106697939)

[4.3.3. Cấu trúc ứng dụng 45](#_Toc106697940)

[4.3.4. Lập trình tiles để tổ chức layout website 47](#_Toc106697941)

[4.3.5. Lập trình Interceptor để bảo mật và phân quyền người dùng cho website 48](#_Toc106697942)

[4.3.6. Lập trình các thực thể ánh xạ xuống cơ sở dữ liệu 49](#_Toc106697943)

[4.3.7. Lập trình các lớp tiện ích assignment.util 53](#_Toc106697944)

[4.3.8. Lập trình các lớp Service xử lý nghiệp vụ 55](#_Toc106697945)

[4.3.9. Lập trình xử lý giỏ hàng 57](#_Toc106697946)

[4.3.10. Lập trình Controller xử lý Request từ Client Web Browser 57](#_Toc106697947)

[5. Tổng kết 61](#_Toc106697948)

[5.1. Thuận lợi 61](#_Toc106697949)

[5.2. Khó khăn 61](#_Toc106697950)

[5.3. Kinh nghiệm đạt được 61](#_Toc106697951)

[5.4. Hướng phát triển 61](#_Toc106697952)

[6. Lời kết 61](#_Toc106697953)

# Giới thiệu dự án

## Giới thiệu TCO Shop & Website

CityZen là của hàng bán sách online đa năng đồng thời bán các dụng cụ vẽ, thời trang và các vật dụng đời với đủ các thể loại khác nhau. Và được sự kết nối với các nhà xuất bản lớn. Gồm các hạng mục như sau:

* Sách – Truyện tranh
* Dụng cụ vẽ - VPP
* Bảng vẽ - Phụ kiện số
* Vật dụng đời sống
* Quà tặng – Đồ chơi
* Thời Trang
* Các sản phẩm khác có liên quan đến các danh mục trên.

Website lập ra với mục đích quản lý sản phẩm đồng thời bán cho các khách hàng ở xa. Và đồng thời mở rộng quy mô ra toàn quốc

## Yêu cầu của cửa hàng

* Nhầm giải quyết các nhu cầu của của hàng, qua các phương pháp nghiệp vụ và thương thảo với cửa hàng thì chúng tôi đã có các chức năng nghiệp vụ như sau

### Yêu cầu về chức năng nghiệp vụ:

#### Về quản lý (chỉ dành cho quản trị viên)

* Quản lý sản phẩm.
* Lịch sử mua hàng của người dùng (hoá đơn, hoá đơn chi tiết).
* Quản lý tài khoản (admin và người dùng website).
* Toàn bộ các chức năng nghiệp vụ khác của website .

#### Về thống kê

* Số lượng sản phẩm bán được.
* Doanh thu của cửa hàng.
* Sản phẩm tồn kho.

#### Về nghiệp vụ bán hàng

* Hiển thị sản phẩm.
* Xem chi tiết từng mặt hàng.
* Xem hàng hoá theo từng danh mục.
* Thêm sản phẩm vào giỏ hàng.
* Thay đổi số lượng sản phẩm trong giỏ.
* Xoá sản phẩm ra khỏi giỏ.
* Hiển thị giỏ hàng chi tiết.
* Đặt sản phẩm (thanh toán).
* Xem lịch sử thanh toán (hoá đơn đã mua).
* Xem chi tiết từng hoá đơn.

### Yêu cầu về bảo mật

* Đăng nhập mới được phép đặt hàng không đăng nhập chỉ xem được sản phẩm, xem lịch sử mua sản phẩm, cập nhật tài khoản, đổi mật khẩu.
* Menu người dùng gồm các mục: tài khoản, đổi mật khẩu, đăng xuất, lịch sử mua hàng.
* Menu admin gồm các mục: tài khoản, đổi mật khẩu, đăng xuất, lịch sử mua hàng, quản lí website, thống kê.
* Trang quản lí website chỉ cho phép admin sử dụng.
* Đăng ký và xác nhận tài khoản qua email.
* Quên mật khẩu.

### Yêu cầu về công nghệ:

* Ứng dụng phải được viết bằng ngôn ngữ lập trình Java, sử dụng công nghệ Spring Boot và Hibernate.
* Yêu cầu phiên bản Java tối thiểu: JDK 1.8.
* Hệ quản trị cơ sở dữ liệu tối thiểu: SQL Server 2012.

## Lập kế hoạch dự án

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TT | HẠNG MỤC | BẮT ĐẦU | KẾT THÚC | KẾT QUẢ |
| 1 | Giới thiệu dự án | 21/05/2022 | 22/05/2022 | Hoàn thành |
| 1.1 | Giới thiệu TCO Shop & website | 21/05/2022 | 22/05/2022 | Hoàn thành |
| 1.2 | Yêu cầu cửa hàng | 21/05/2022 | 22/05/2022 | Hoàn thành |
| 1.3 | Lập kế hoạch dự án | 21/05/2022 | 22/05/2022 | Hoàn thành |
| 1.4 | Phân chia công việc | 21/05/2022 | 22/05/2022 | Hoàn thành |
| 2 | Phân tích yêu cầu khách hàng | 22/05/2022 | 25/05/2022 | Hoàn thành |
| 2.1 | Use case | 22/05/2022 | 22/05/2022 | Hoàn thành |
| 2.2 | Đặc tả yêu cầu hệ thống | 22/05/2022 | 22/05/2022 | Hoàn thành |
| 2.3 | Mô hình ứng dụng triển khai | 22/05/2022 | 22/05/2022 | Hoàn thành |
| 2.4 | Yêu cầu hệ thống | 22/05/2022 | 22/05/2022 | Hoàn thành |
| 3 | Thiết kế ứng dụng | 25/05/2022 | 30/05/2022 | Hoàn thành |
| 3.1 | Thiết kế cơ sở dữ liệu |  |  |  |
| 3.2 | Thiết kế giao diện |  |  |  |
| 4 | Thực hiện dự án | 30/05/2022 | 20/06/2022 | Hoàn thành |
| 4.1 | Tạo giao diện |  |  |  |
| 4.2 | Tạo cơ sở dữ liệu |  |  |  |
| 4.3 | Tạo dự án web |  |  |  |
| 5 | Tổng kết | 20/06/2022 | 21/06/2022 | Hoàn thành |
| 6 | Lời kết | 20/06/2022 | 21/06/2022 | Hoàn thành |

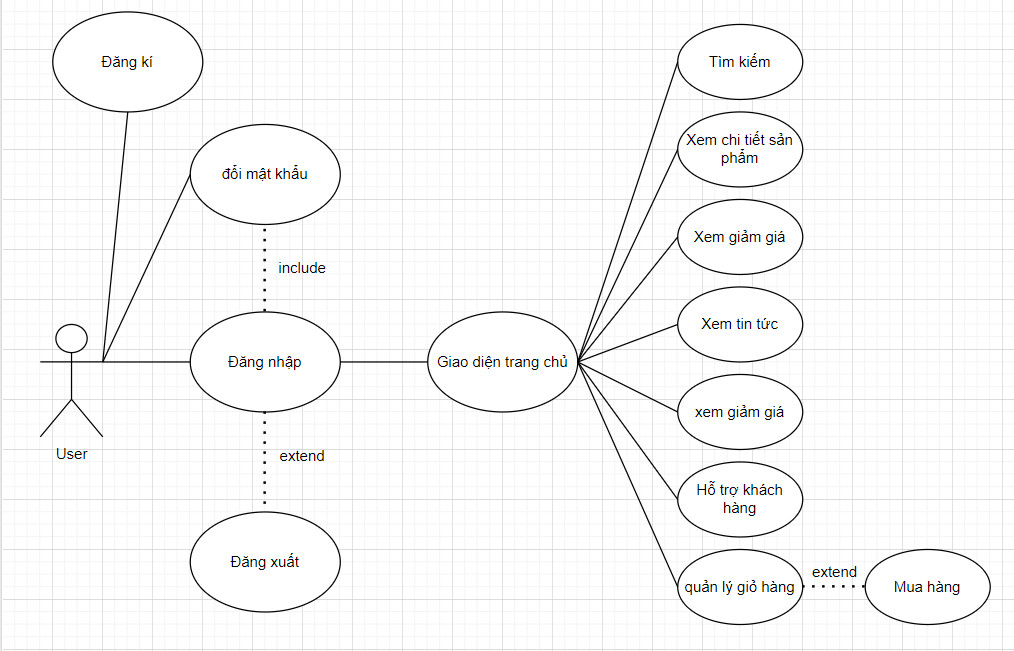
## Bảng phân chia công việc

|  |  |  |
| --- | --- | --- |
| TT | Tên công việc | Thành viên |
| 1 | Thiết kế cơ sở dữ liệu | Nguyễn Hoàng Nhi |
| 2 | Thiết kế giao diện ứng dụng | Ngô Ngọc Tài  Trần Thanh Tùng |
| 3 | Lập trình giao diện | Ngô Ngọc Tài |
| 4 | Lập trình các lớp tiện ích | Diệp Ông Đại Quang  Trần Gia Bảo |
| 5 | Lập trình thực thể ánh xạ cơ sở dữ liệu | Diệp Ông Đại Quang |
| 6 | Lập trình tiles tổ chức layout | Trần Gia Bảo  Diệp Ông Đại Quang |
| 7 | Lập trình xử lý giỏ hàng | Diệp Ông Đại Quang |
| 8 | Lập trình các lớp Service nghiệp vụ | Trần Thanh Tùng  Trần Gia Bảo  Ngô ngọc Tài |
| 9 | Lập trình controller | Diệp Ông Đại Quang  Trần Gia Bảo  Trần Thanh Tùng  Nguyễn Hoàng Nhi  Ngô Ngọc Tài |
| 10 | Viết báo cáo | Nguyễn Hoàng Nhi  Trần Thanh Tùng |

# Phân tích yêu cầu khách hàng

## Sơ đồ Use case

* User



* Admin



## Đặc tả yêu cầu hệ thống

### Quản lý hoá đơn

* **Mô tả chức năng:**

Hóa đơn sẽ xuất hiện sau khi người mua đặt hàng. Thông tin trong hóa đơn gồm có mã hóa đơn, người mua, ngày tạo, địa chỉ.

* **Đối tượng sử dụng:**

Những người đã đăng nhập.

### Quản lý sản phẩm

* **Mô tả chức năng:**

Quản lý sản phẩm sẽ hiện khi người đăng nhập là admin và bấm vào trang admin/sanpham. Thông tin của mỗi sản phẩm gồm mã sản phẩm, tên sản phẩm, hình ảnh, ngày tạo, số lượng, giá. Trong quản lý sản phẩm sẽ có các chức năng thêm, xóa và sửa sản phẩm.

* **Đối tượng sử dụng:**

Người đăng nhập phải là admin

### Quản lý tài khoản

* **Mô tả chức năng**

Quản lý sản phẩm sẽ hiện khi người đăng nhập là admin và bấm vào trang admin/taikhoan. Thông tin của mỗi tài khoản bao gồm mã người dùng, tên người dùng, địa chỉ, email. Trong quản lý người dùng sẽ có các chức năng thêm, xóa, sửa người dùng

* **Đối tượng sử dụng:**

Người đăng nhập phải là admin

### Chức năng nghiệp vụ mua hàng

* **Mô tả chức năng:**

Chức năng mua hàng sẽ hiện ra khi người dùng bấm vào nút mua hàng trên mỗi sản phẩm. Giỏ hàng sẽ hiện ra và người dùng có thể thay đổi số lượng mặc hàng muốn mua.

* **Đối tượng sử dụng:**

Những người đã đăng nhập

## Mô hình ứng dụng triển khai

* Dự án được triển khai theo mô hình MVC:

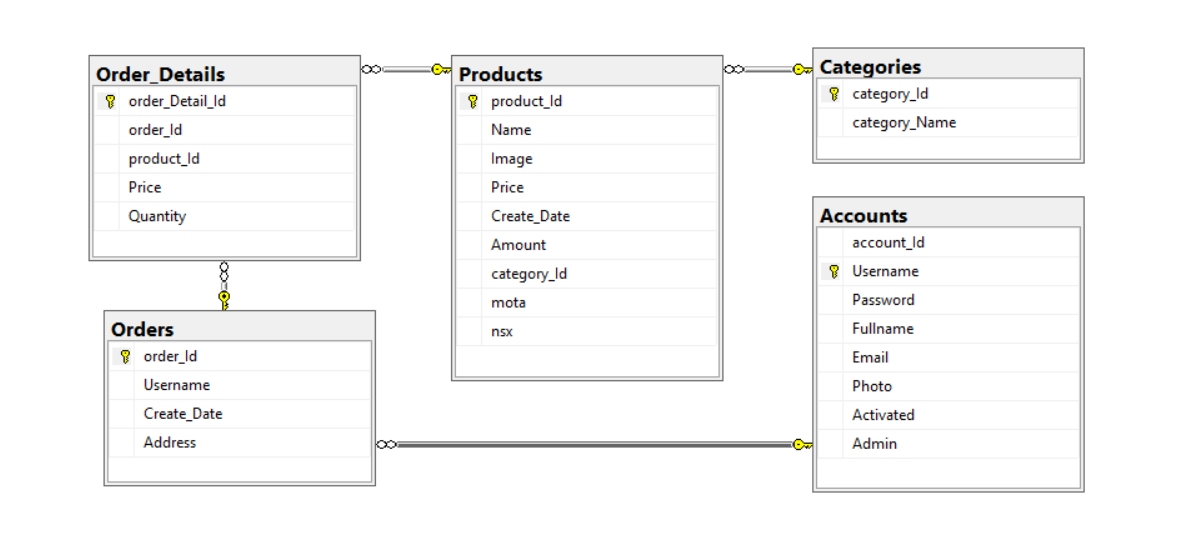
## Yêu cầu hệ thống

Vì là một website nên mọi người chỉ cần các công cụ để truy cập vào miền của trang web là có thể sử dụng

# Thiết kế ứng dụng

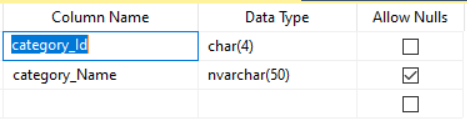
## Thiết kế cơ sở dữ liệu:

Mô tả các mối quan hệ

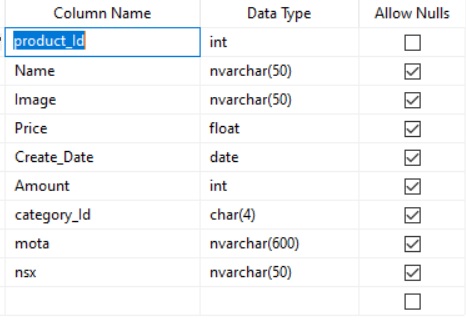


Chi tiết thực thể:

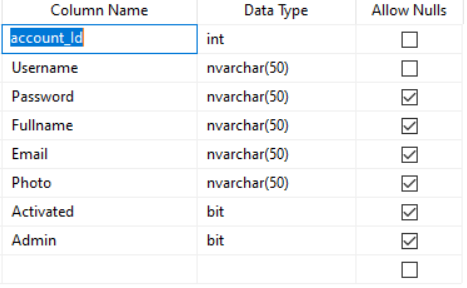
### Thực thể Categories:



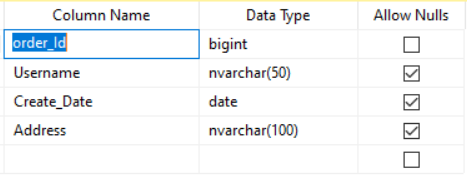
### Thực thể Products



### Thực thể Accounts



### Thực thể Orders

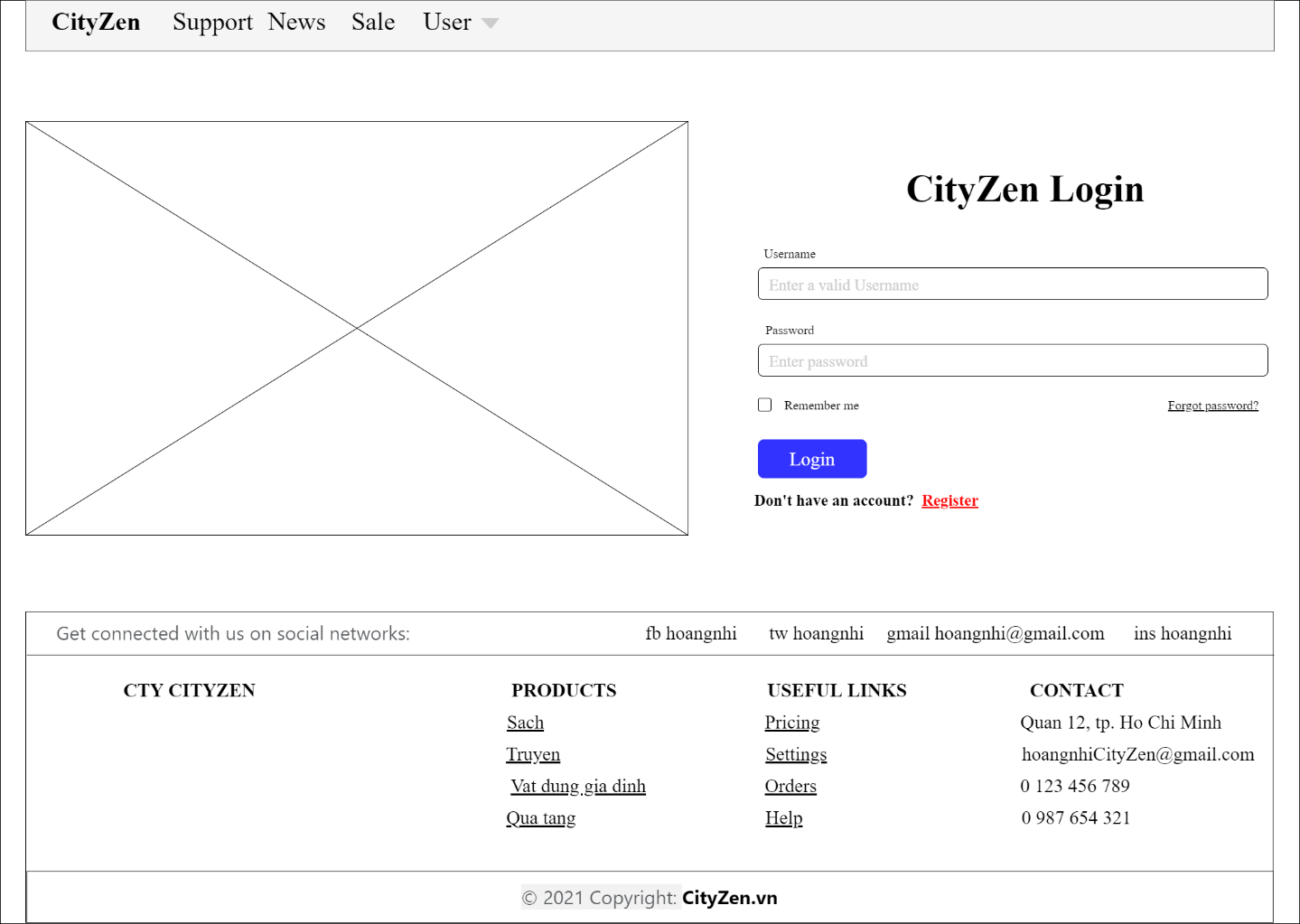


### Thực thể Orders\_Detail

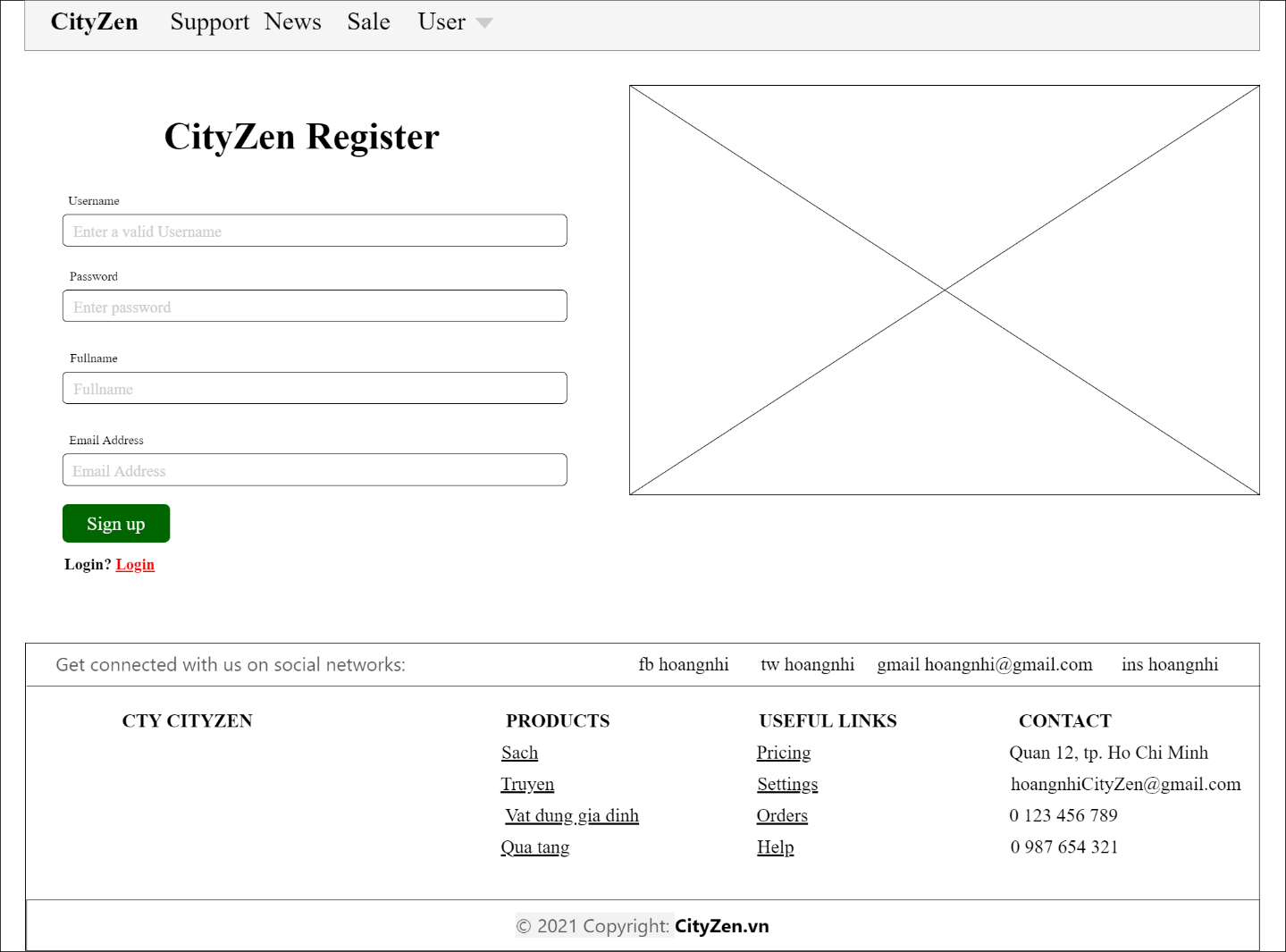
### 

## Thiết kế giao diện

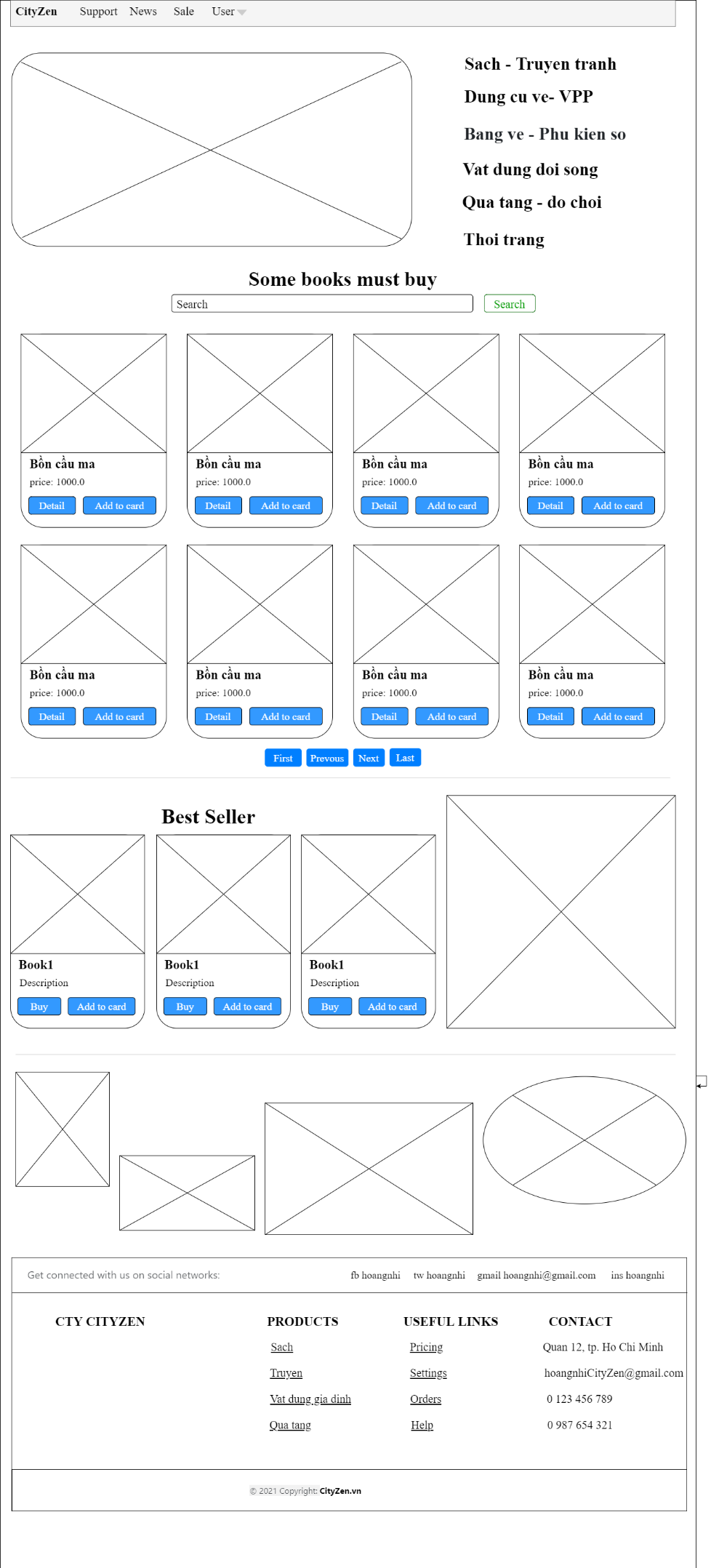
### Trang đăng nhập



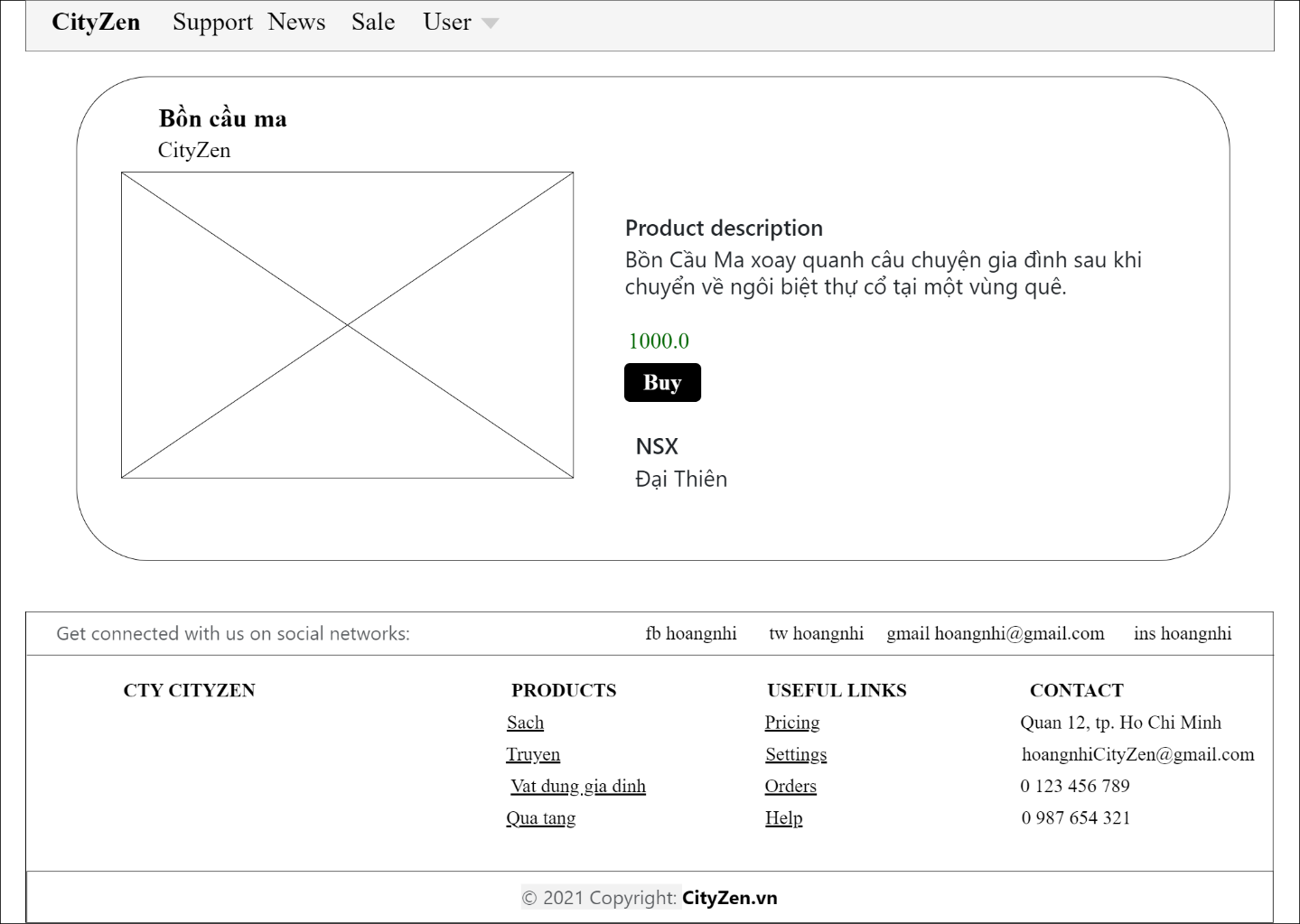
### Trang đăng ký



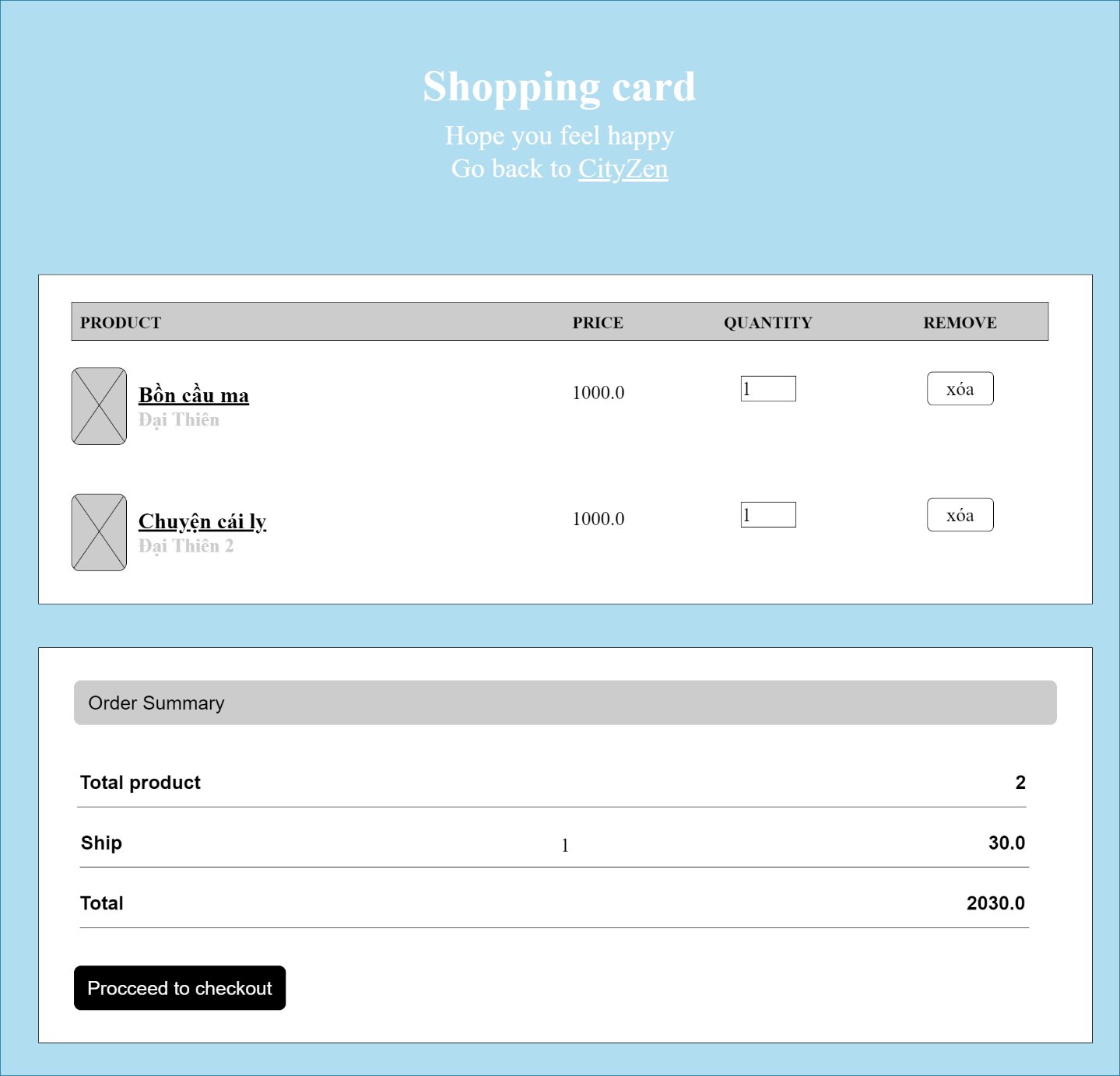
### Trang chủ



### Trang chi tiết sản phẩm



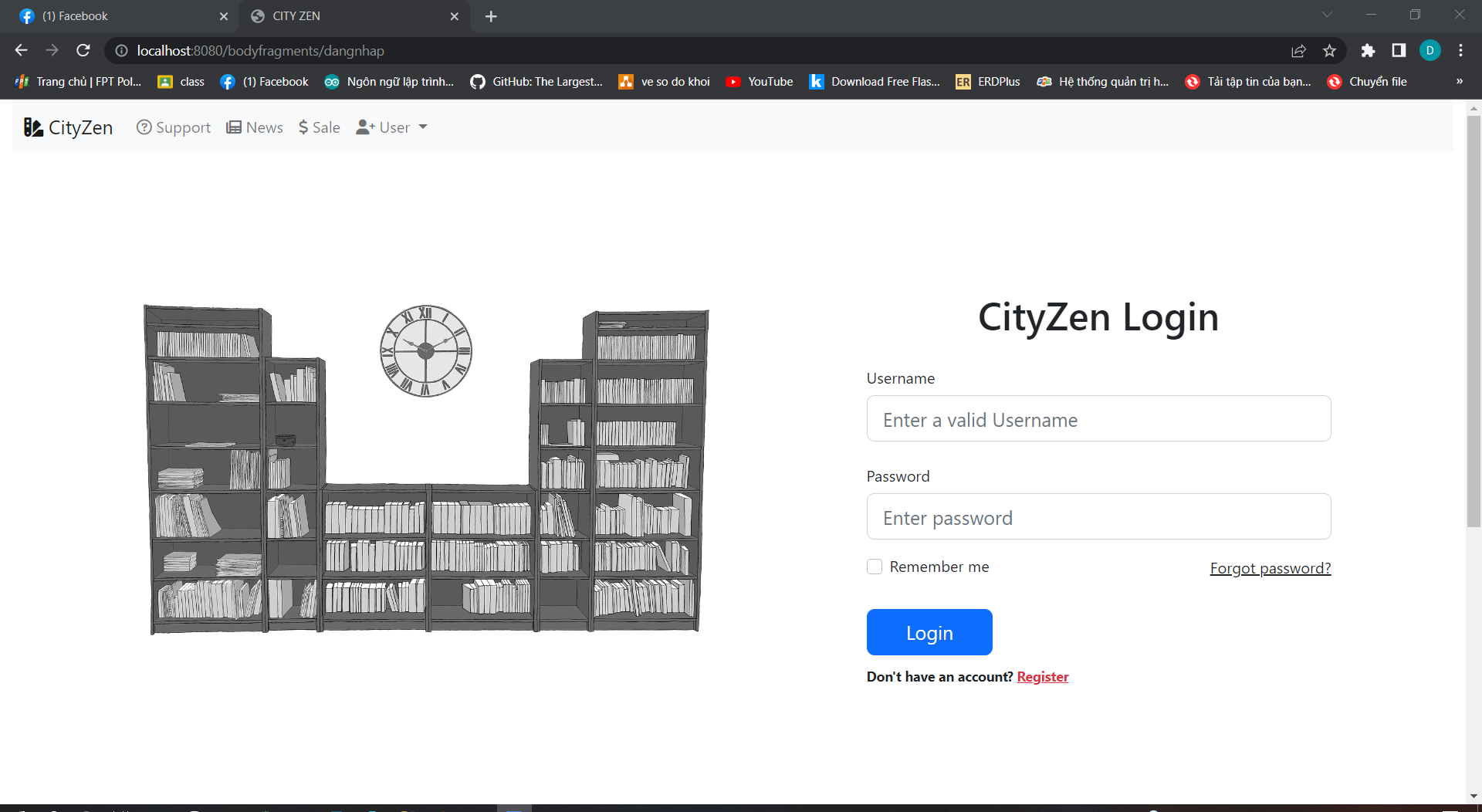
### Trang giỏ hàng



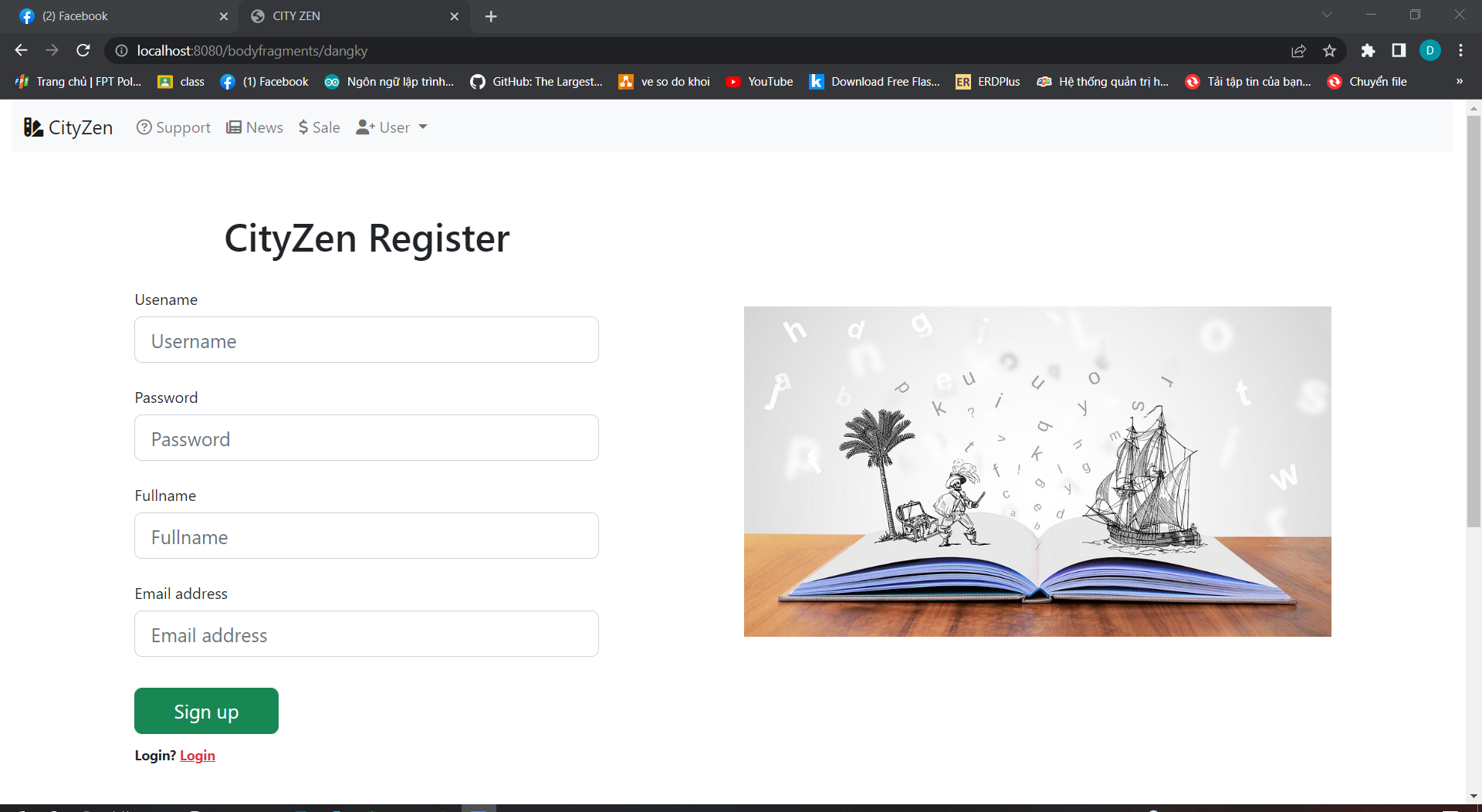
# Thực hiện dự án

## Tạo giao diện với JSP, CSS, Javascript, Bootstrap:

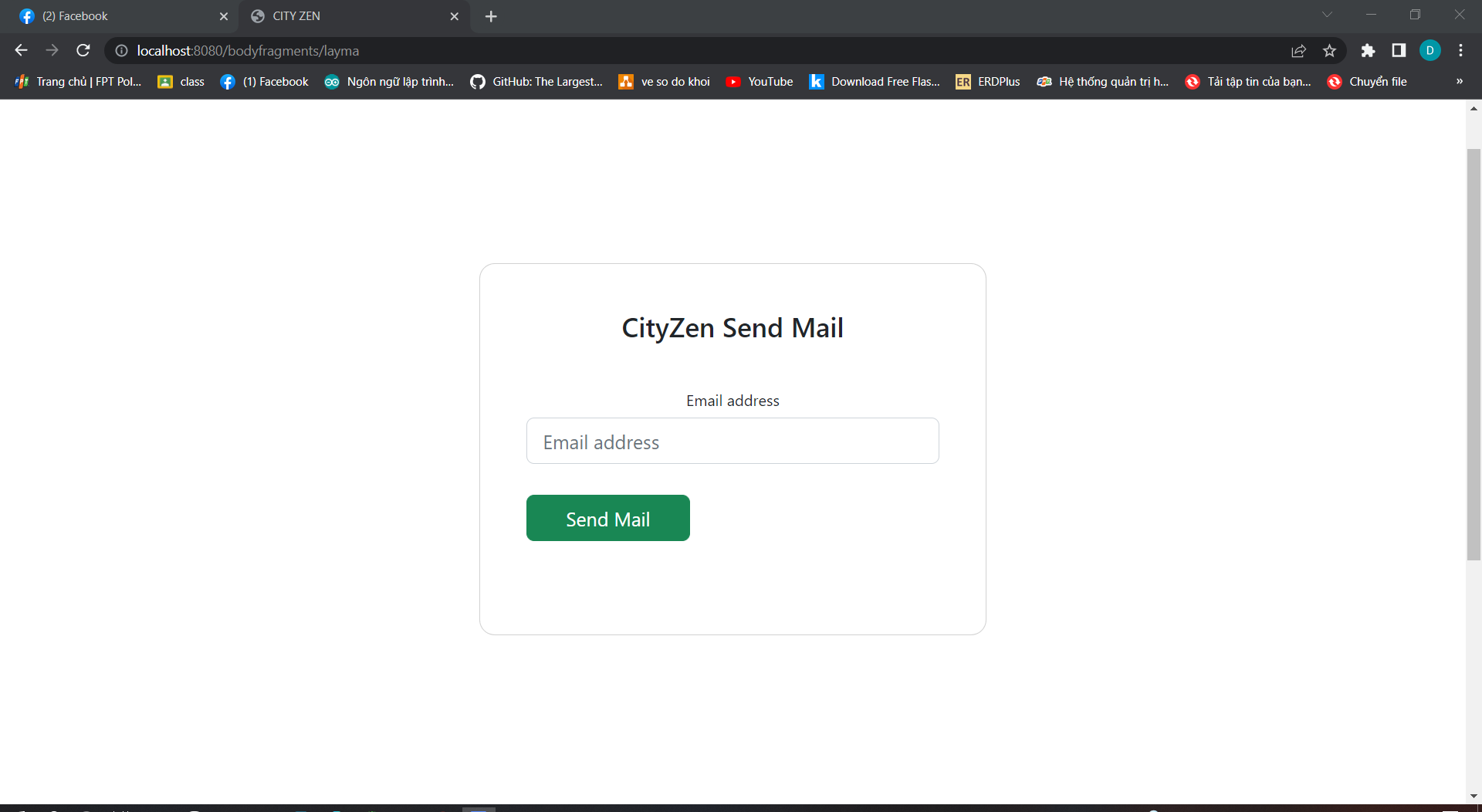
### Trang đăng nhập:



### Trang đăng ký:

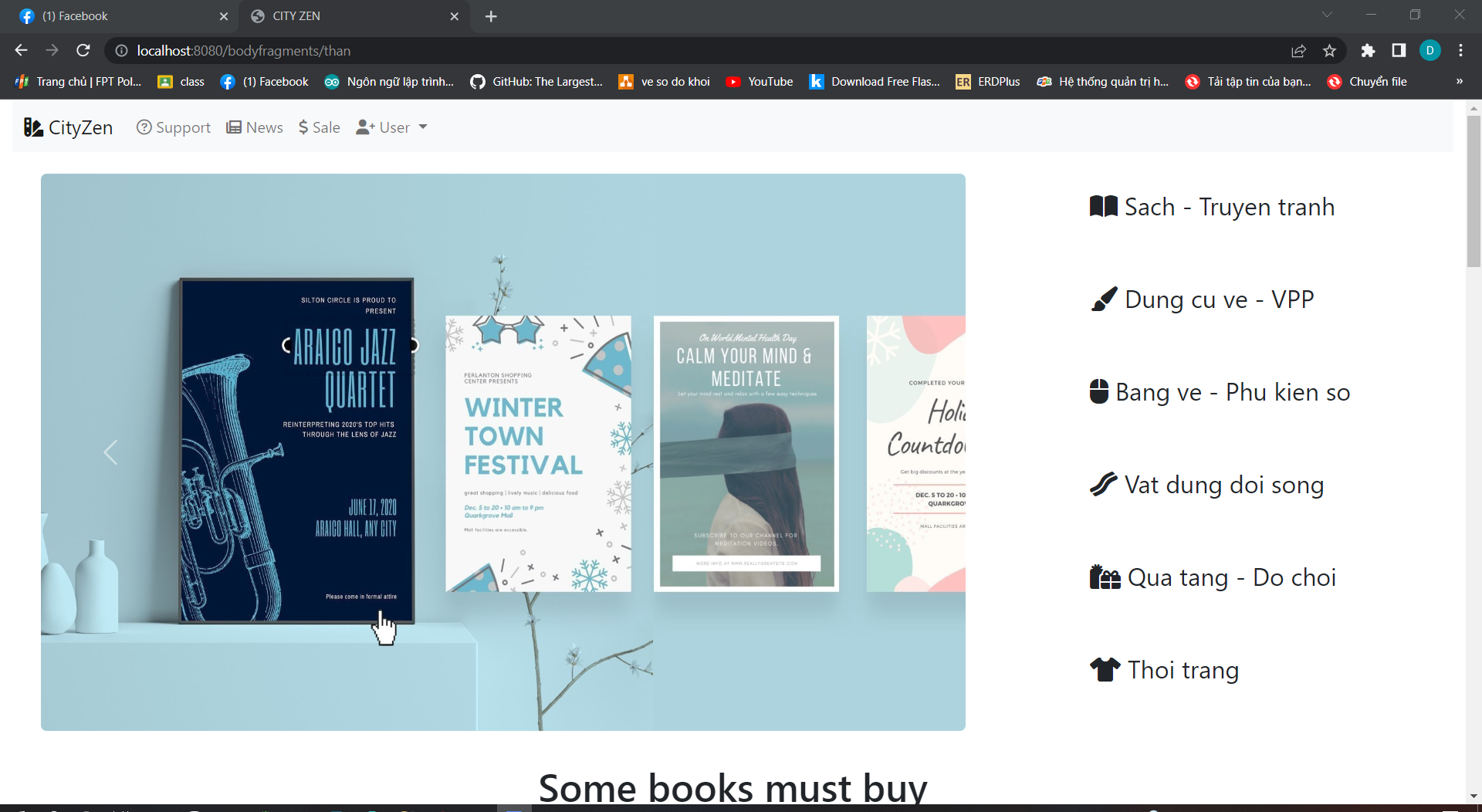


### Trang nhập mã xác nhận và đặt lại mật khẩu

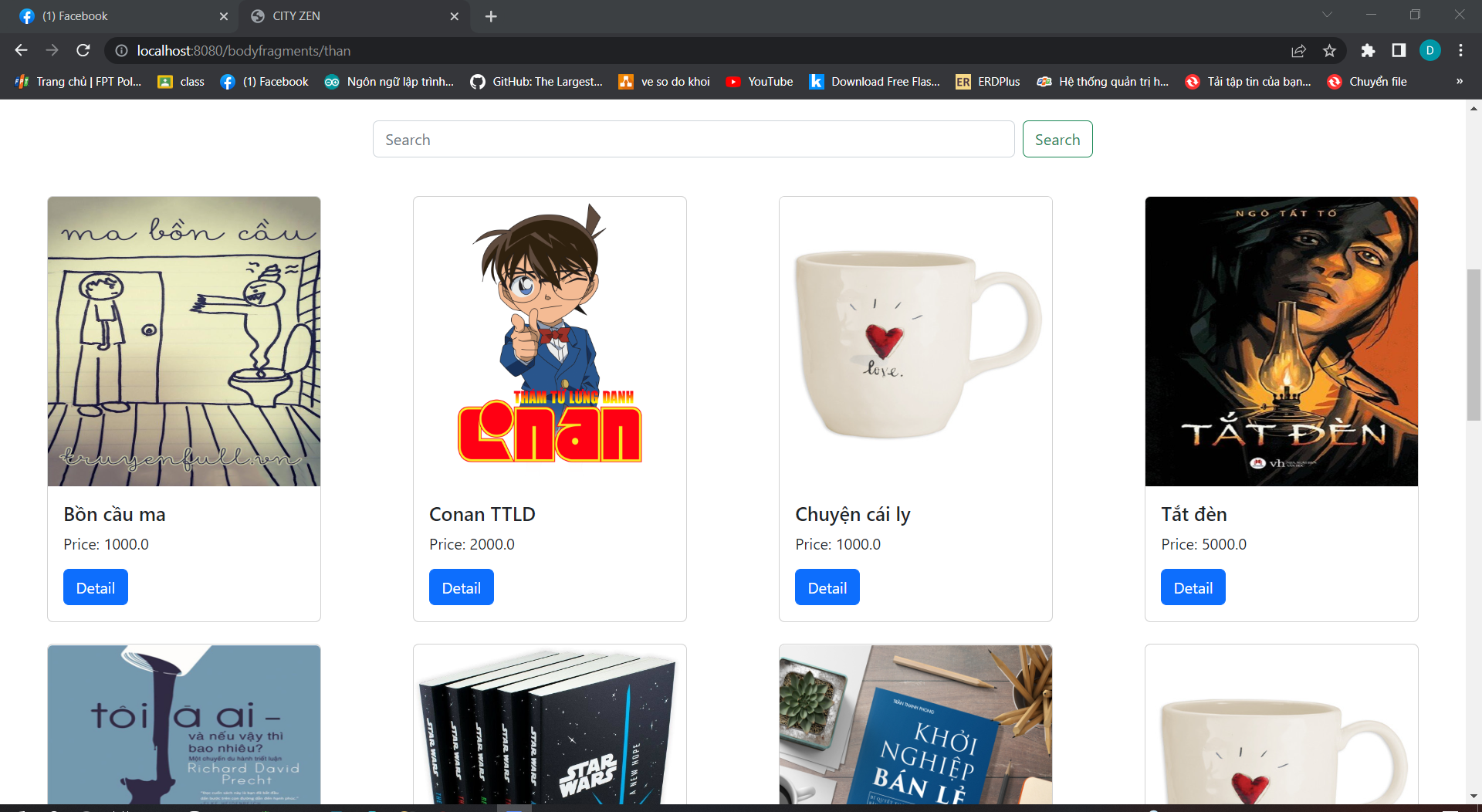


### Trang chủ

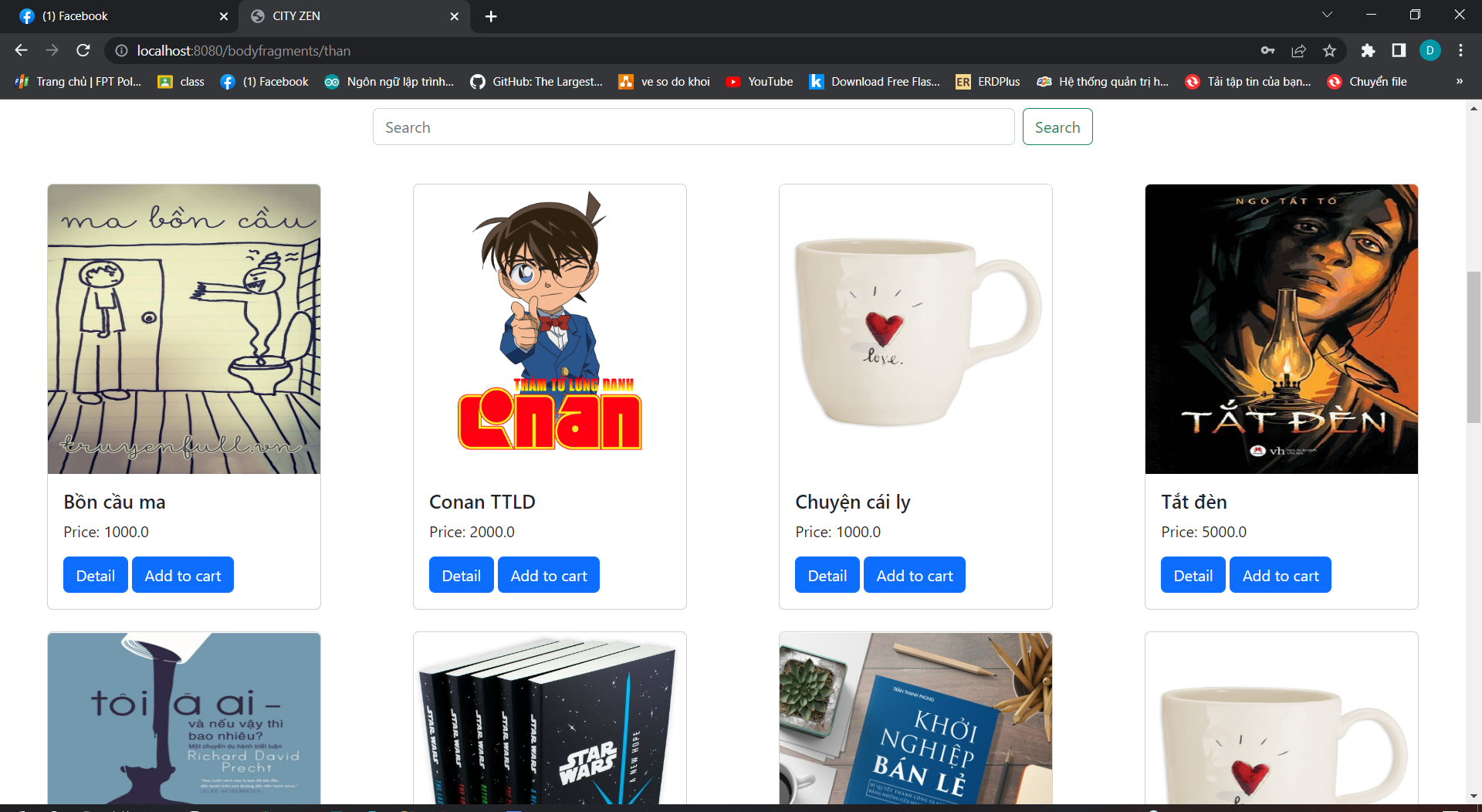
* Thanh danh mục sản phẩm



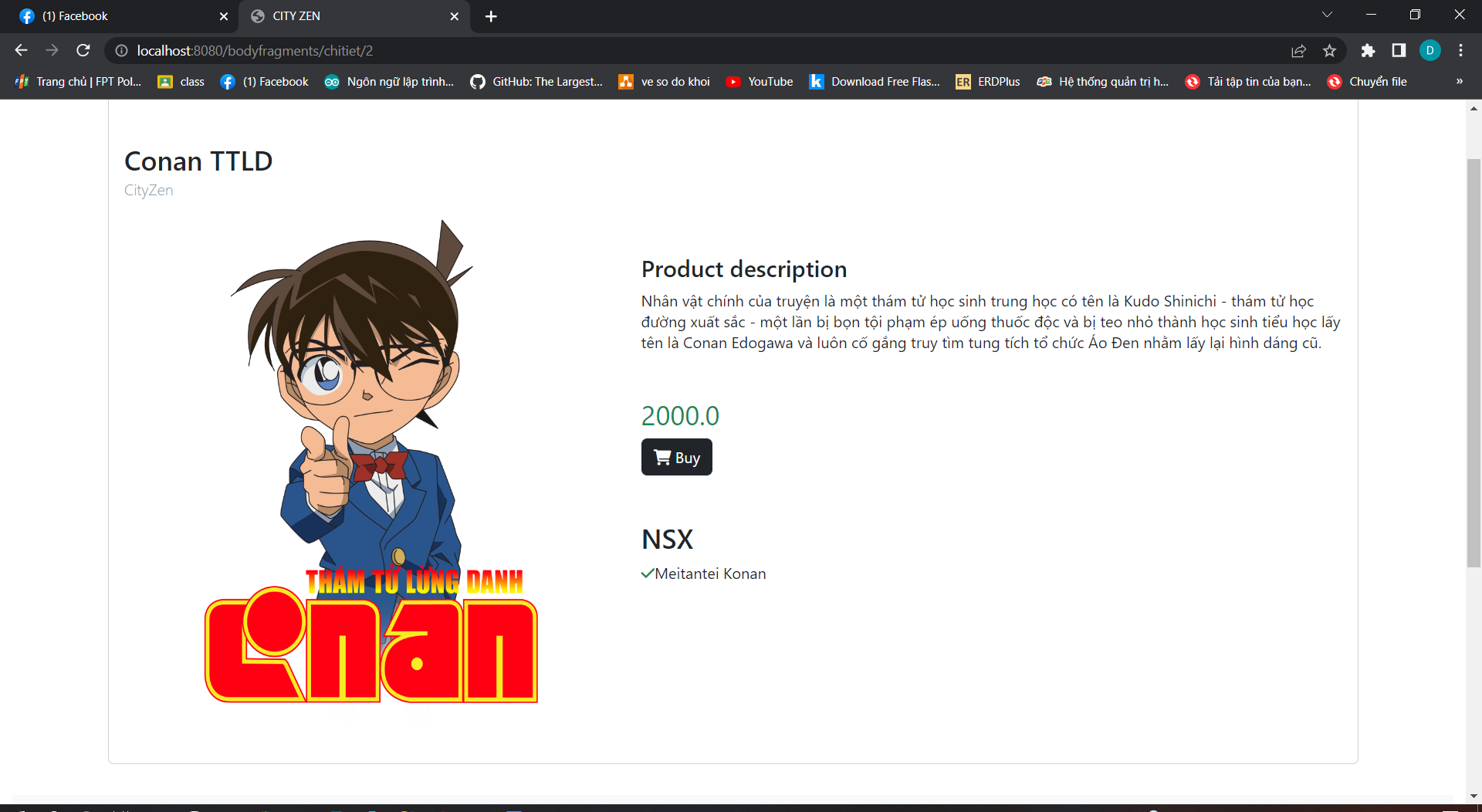
* Người dùng khi chưa đăng nhập



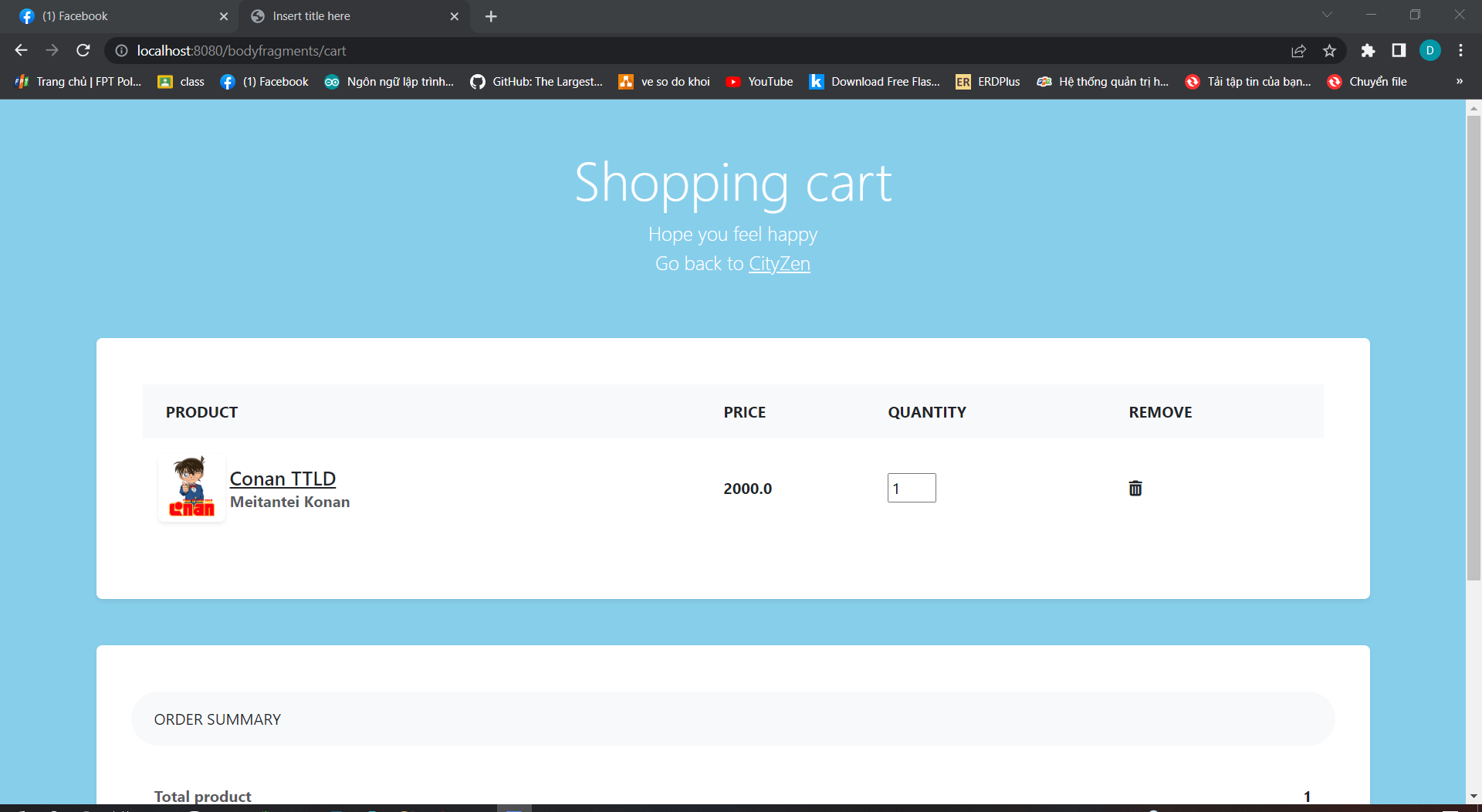
* Người dùng khi đăng nhập với vai trò khách hàng



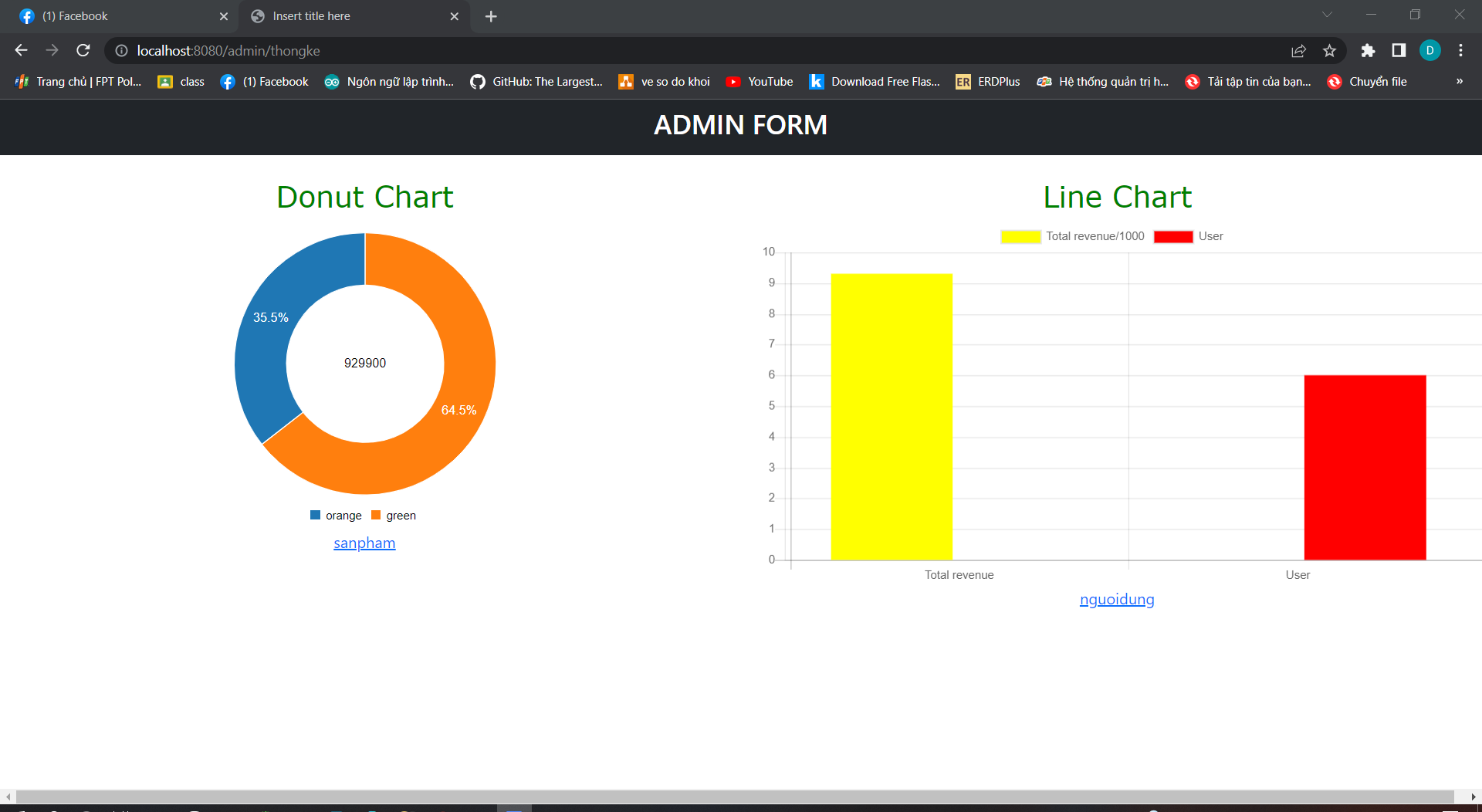
### Trang chi tiết sản phẩm

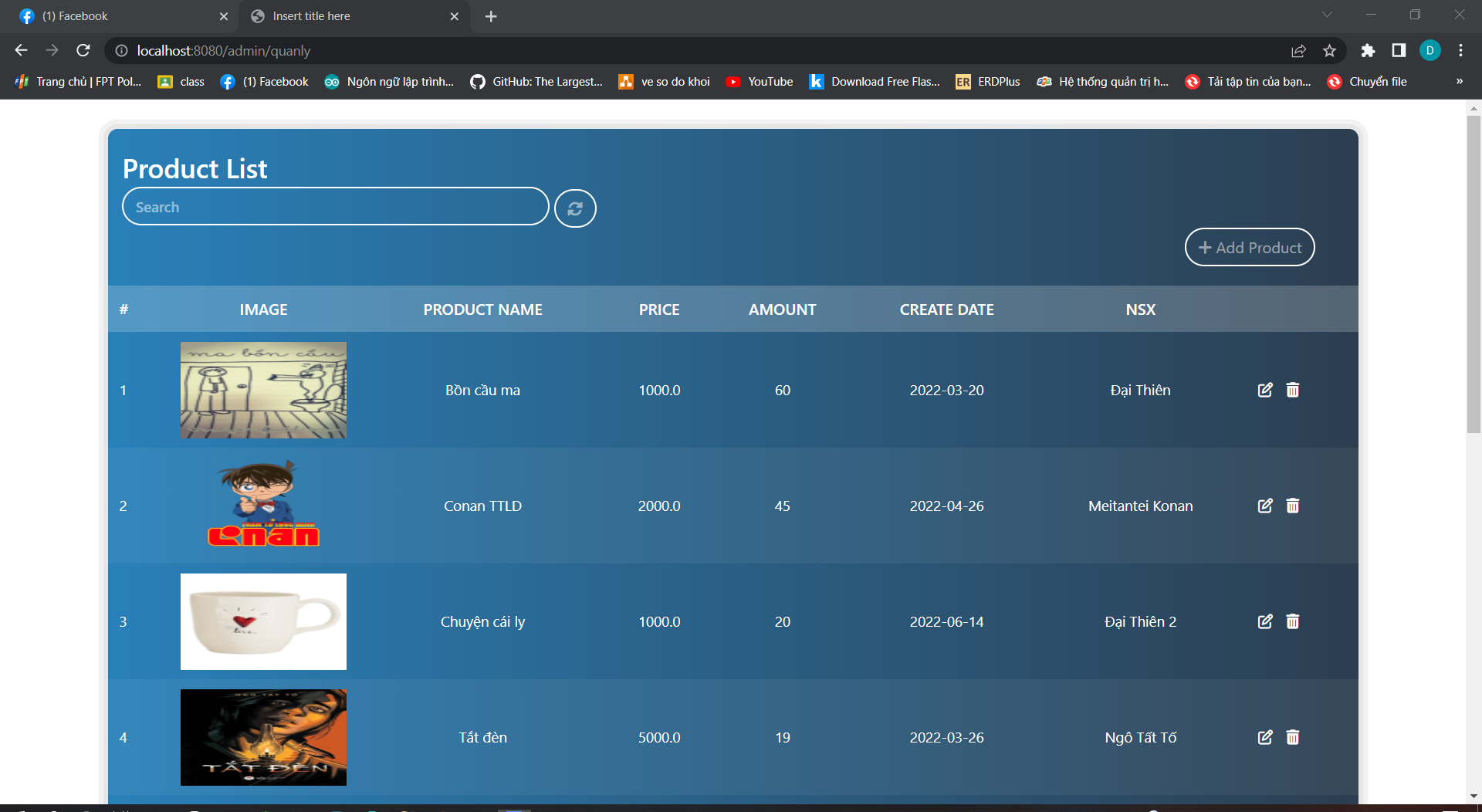


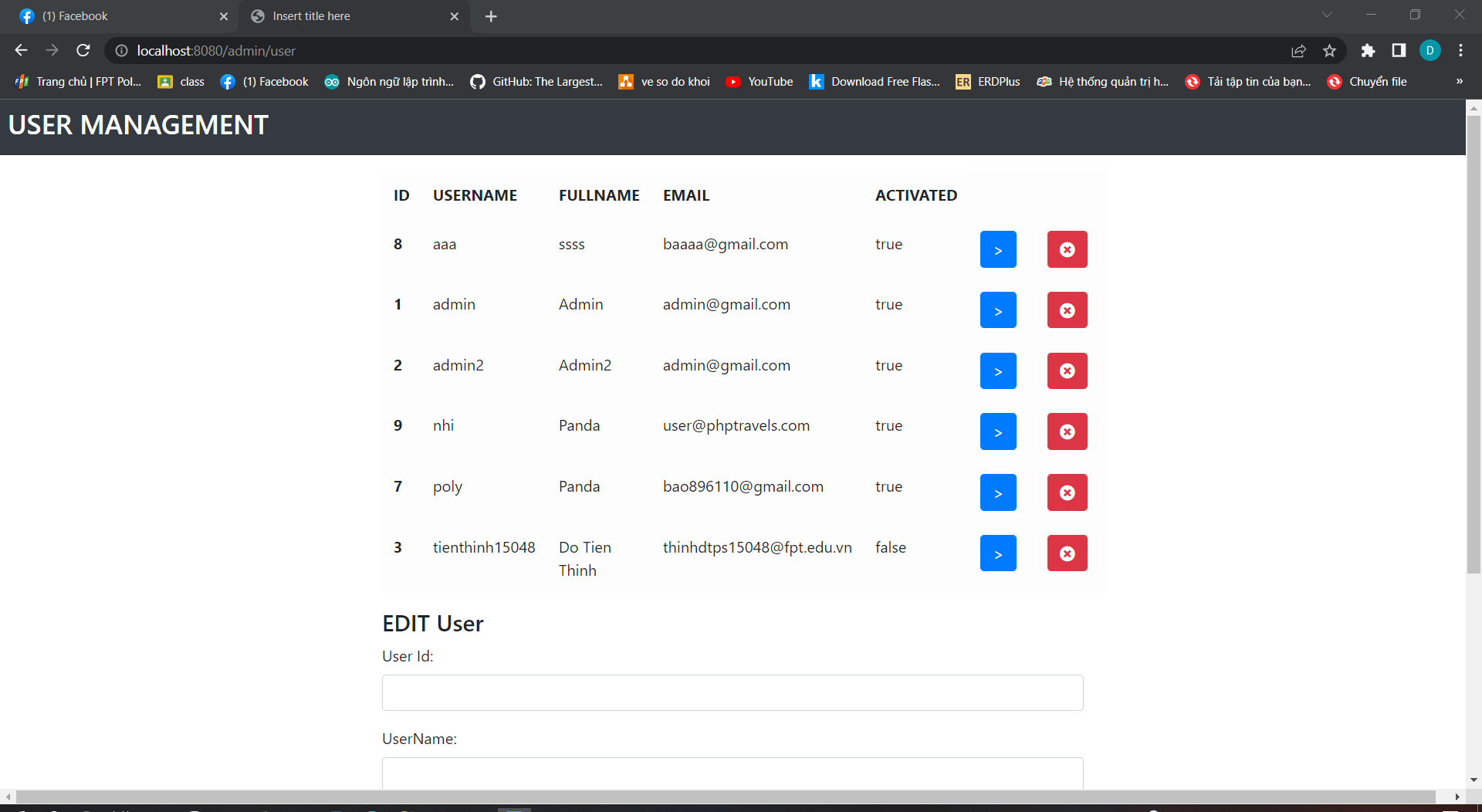
### Trang giỏ hàng



### Trang quản lý website dành cho quản trị viên







## Tạo cơ sở dữ liệu với SQL Server

Dưới đây là đoạn mã dùng để tạo cơ sở dữ liệu:

### Tạo bảng Categories

create table Categories(

category\_Id char(4),

category\_Name nvarchar(50),

Primary key(category\_Id)

### )

### Tạo bảng Products

create table Products(

product\_Id int identity(1,1),

Name nvarchar(50),

Image nvarchar(50),

Price float,

Create\_Date date,

Amount int,

category\_Id char(4),

mota nvarchar(600),

nsx nvarchar(50),

Primary key(product\_Id),

constraint fk\_Products\_CategoryId

foreign key(category\_Id) references Categories(category\_Id)

)

### Tạo bảng Accounts

create table Accounts(

account\_Id int identity(1,1),

Username nvarchar(50),

Password nvarchar(50),

Fullname nvarchar(50),

Email nvarchar(50),

Photo nvarchar(50),

Activated bit,

Admin bit,

Primary key(Username)

)

### Tạo bảng Orders

create table Orders(

order\_Id bigint identity(1,1),

Username nvarchar(50),

Create\_Date date,

Address nvarchar(100),

Primary key (order\_Id),

constraint fk\_Orders\_Username

foreign key (Username) references Accounts(Username)

)

### Tạo bảng Orders\_Detail

create table Order\_Details(

order\_Detail\_Id bigint identity(1,1),

order\_Id bigint,

product\_Id int,

Price float,

Quantity int,

Primary key (order\_Detail\_Id),

constraint fk\_OrderDetails\_orderId

foreign key (order\_Id) references Orders(order\_Id),

constraint fk\_OrderDetails\_productId

foreign key (product\_Id) references Products(product\_Id)

)

## Tạo dự án web với Spring Boot

### Khai báo thư viện vào pom.xml

#### Thư viện xây dựng trang web áp dụng mô hình MVC

#### Thư viện tích hợp tomcat server

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-tomcat</artifactId>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>org.apache.tomcat.embed</groupId>

<artifactId>tomcat-embed-jasper</artifactId>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

</dependency>

#### Thư viện kiểm thử ứng dụng web

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

#### Thư viện giúp tổ chức giao diện layout website

<!-- Tiles API -->

<!-- http://mvnrepository.com/artifact/org.apache.tiles/tiles-api -->

<dependency>

<groupId>org.apache.tiles</groupId>

<artifactId>tiles-api</artifactId>

<version>3.0.8</version>

</dependency>

<!-- Tiles Core -->

<!-- http://mvnrepository.com/artifact/org.apache.tiles/tiles-core -->

<dependency>

<groupId>org.apache.tiles</groupId>

<artifactId>tiles-core</artifactId>

<version>3.0.8</version>

</dependency>

<!-- Tiles Servlet -->

<!-- http://mvnrepository.com/artifact/org.apache.tiles/tiles-servlet -->

<dependency>

<groupId>org.apache.tiles</groupId>

<artifactId>tiles-servlet</artifactId>

<version>3.0.8</version>

</dependency>

<!-- Tiles JSP -->

<!-- http://mvnrepository.com/artifact/org.apache.tiles/tiles-jsp -->

<dependency>

<groupId>org.apache.tiles</groupId>

<artifactId>tiles-jsp</artifactId>

<version>3.0.8</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.tiles/tiles-request-api -->

<dependency>

<groupId>org.apache.tiles</groupId>

<artifactId>tiles-request-api</artifactId>

<version>1.0.6</version>

</dependency>

#### Thư viện hỗ trợ JSTL

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

</dependency>

#### Thư viện hỗ trợ truy cập cơ sở dữ liệu SQL Server

<dependency>

<groupId>com.microsoft.sqlserver</groupId>

<artifactId>mssql-jdbc</artifactId>

<version>6.2.2.jre8</version>

</dependency>

<dependency>

<groupId>com.microsoft.sqlserver</groupId>

<artifactId>mssql-jdbc</artifactId>

<scope>runtime</scope>

</dependency>

#### Thư viện Spring hỗ trợ sử dụng JPA với Hibernate

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-validator</artifactId>

<version>5.1.3.Final</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

#### Thư viện Spring hỗ trợ gửi email:

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-mail</artifactId>

</dependency>

### Tuỳ chỉnh application.properties

#### Tuỳ chỉnh để ứng dụng Spring dùng file .jsp từ webroot 🡪 WEB-INF 🡪 views để hiển thị

spring.mvc.view.prefix=/WEB-INF/

spring.mvc.view.suffix=.jsp

#### Tuỳ chỉnh truy cập cơ sở dữ liệu

spring.datasource.url=jdbc:sqlserver://localhost;database=J5Shop

spring.datasource.username=sa

spring.datasource.password=songlong

spring.datasource.driverClassName=com.microsoft.sqlserver.jdbc.SQLServerDriver

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.SQLServer2012Dialect

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=none

#### Tuỳ chỉnh gửi email

spring.mail.host=smtp.gmail.com

spring.mail.port=587

spring.mail.username=quangdodps15144@fpt.edu.vn

spring.mail.password=cbzywzlafctfvsfj

spring.mail.properties.mail.smpt.auth=true

spring.mail.properties.mail.smtp.starttls.enable=true

### Cấu trúc ứng dụng

### Lập trình tiles để tổ chức layout website

* Lập trình tiles.xml để tạo layout website:

<tiles-definitions>

<!-- Base Define -->

<definition name=*"base.definition"*

template=*"/WEB-INF/layouts/classic.jsp"*>

<put-attribute name=*"header"* value=*"/WEB-INF/basefragments/thanhmenu.jsp"* />

<put-attribute name=*"body"* value=*""* />

<put-attribute name=*"footer"* value=*"/WEB-INF/basefragments/chan.jsp"* />

</definition>

<!-- body -->

<definition name=*"\*/\*"* extends=*"base.definition"*>

<put-attribute name=*"title"* value=*"CITY ZEN"* />

<put-attribute name=*"body"* value=*"/WEB-INF/{1}/{2}.jsp"* />

</definition>

</tiles-definitions>

* Tạo TilesConfig trong Package com.assignment để nạp tiles.xml vào ứng dụng

@Configuration

**public** **class** TilesConfig {

@Bean(name = "viewResolver")

**public** ViewResolver getViewResolver() {

UrlBasedViewResolver viewResolver = **new** UrlBasedViewResolver();

viewResolver.setViewClass(TilesView.**class**);

**return** viewResolver;

}

@Bean(name = "tilesConfigurer")

**public** TilesConfigurer getTilesConfigurer() {

TilesConfigurer tilesConfigurer = **new** TilesConfigurer();

tilesConfigurer.setDefinitions("/WEB-INF/tiles.xml");

**return** tilesConfigurer;

}

}

### Lập trình Interceptor để bảo mật và phân quyền người dùng cho website

* Tạo Class AuthorizeInterceptor để kiểm tra đăng nhập & quyền truy cập trang quản trị

@Service

**public** **class** AuthInterceptor **implements** HandlerInterceptor {

@Autowired

SessionService session;

@Override

**public** **boolean** preHandle(HttpServletRequest request,

HttpServletResponse response, Object handler) **throws** Exception {

String uri = request.getRequestURI();

Account user = session.getAttribute("user");

String error = "";

**if**(user == **null**) {

error = "Please login!";

response.sendRedirect("/bodyfragments/dangnhap?error=" + error);

**return** **false**;

}

**else** **if**(!user.isAdmin() && uri.startsWith("/admin")) {

error = "Access denied!";

response.sendRedirect("/bodyfragments/dangnhap?error=" + error);

**return** **false**;

}

**return** **true**;

}

}

* Tạo Class InterceptorConfig trong package com.assignment để nạp Class AuthorizeInterceptor vừa tạo

@Configuration

**public** **class** AuthInterConfig **implements** WebMvcConfigurer {

@Autowired

AuthInterceptor auth;

@Override

**public** **void** addInterceptors(InterceptorRegistry registry) {

registry.addInterceptor(auth).addPathPatterns("/admin/\*\*")

.excludePathPatterns("/bodyfragments/\*\*");

}

}

### Lập trình các thực thể ánh xạ xuống cơ sở dữ liệu

#### Class Account

@Entity

@Table(name = "Accounts")

**public** **class** Account **implements** Serializable {

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

@Id

@Column(name = "accountId")

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

**private** Integer id;

@Column(name = "Username")

**private** String username;

@Column(name = "Password")

**private** String password;

@Column(name = "Fullname")

**private** String fullname;

@Column(name = "Email")

**private** String email;

@Column(name = "Photo")

**private** String photo;

@Column(name = "Activated")

**private** **boolean** activated;

@Column(name = "Admin")

**private** **boolean** admin;

@OneToMany(mappedBy = "account")

**private** List<Order> orders;

* Và các getters/setters/constructors ở phía dưới.

**public** Account() {

**super**();

}

**public** Account(Integer id, String username, String password, String fullname, String email, String photo,

**boolean** activated, **boolean** admin, List<Order> orders) {

**super**();

**this**.id = id;

**this**.username = username;

**this**.password = password;

**this**.fullname = fullname;

**this**.email = email;

**this**.photo = photo;

**this**.activated = activated;

**this**.admin = admin;

**this**.orders = orders;

}

**public** Integer getId() {

**return** id;

}

**public** **void** setId(Integer id) {

**this**.id = id;

}

**public** String getUsername() {

**return** username;

}

**public** **void** setUsername(String username) {

**this**.username = username;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

**public** String getFullname() {

**return** fullname;

}

**public** **void** setFullname(String fullname) {

**this**.fullname = fullname;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** String getPhoto() {

**return** photo;

}

**public** **void** setPhoto(String photo) {

**this**.photo = photo;

}

**public** **boolean** isActivated() {

**return** activated;

}

**public** **void** setActivated(**boolean** activated) {

**this**.activated = activated;

}

**public** **boolean** isAdmin() {

**return** admin;

}

**public** **void** setAdmin(**boolean** admin) {

**this**.admin = admin;

}

**public** List<Order> getOrders() {

**return** orders;

}

**public** **void** setOrders(List<Order> orders) {

**this**.orders = orders;

}

**public** **static** **long** getSerialversionuid() {

**return** ***serialVersionUID***;

}

#### Class Category

@Entity

@Table(name = "Categories")

**public** **class** Category **implements** Serializable {

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

@Id

@Column(name = "categoryId")

**private** String id;

@Column(name = "categoryName")

**private** String name;

@OneToMany(mappedBy = "category")

**private** List<Product> products;

* Và các getters/setters/constructors ở phía dưới.

**public** Category() {

**super**();

}

**public** Category(String id, String name) {

**super**();

**this**.id = id;

**this**.name = name;

}

**public** Category(String id, String name, List<Product> products) {

**super**();

**this**.id = id;

**this**.name = name;

**this**.products = products;

}

**public** String getId() {

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** List<Product> getProducts() {

**return** products;

}

**public** **void** setProducts(List<Product> products) {

**this**.products = products;

}

#### Class Orders

@Entity

@Table(name = "Orders")

**public** **class** Order **implements** Serializable {

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

@Id

@Column(name = "orderId")

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

**private** **long** id;

@Column(name = "CreateDate")

@Temporal(TemporalType.***DATE***)

**private** Date createDate = **new** Date();

@Column(name = "Address")

**private** String address;

@ManyToOne

@JoinColumn(name = "Username")

Account account;

@OneToMany(mappedBy = "order")

**private** List<OrderDetail> orderDetails;

* Và các getters/setters/constructors ở phía dưới.

**public** Order() {

**super**();

}

**public** Order(**long** id, Date createDate, String address, Account account, List<OrderDetail> orderDetails) {

**super**();

**this**.id = id;

**this**.createDate = createDate;

**this**.address = address;

**this**.account = account;

**this**.orderDetails = orderDetails;

}

**public** **long** getId() {

**return** id;

}

**public** **void** setId(**long** id) {

**this**.id = id;

}

**public** Date getCreateDate() {

**return** createDate;

}

**public** **void** setCreateDate(Date createDate) {

**this**.createDate = createDate;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

**public** Account getAccount() {

**return** account;

}

**public** **void** setAccount(Account account) {

**this**.account = account;

}

**public** List<OrderDetail> getOrderDetails() {

**return** orderDetails;

}

**public** **void** setOrderDetails(List<OrderDetail> orderDetails) {

**this**.orderDetails = orderDetails;

}

**public** **static** **long** getSerialversionuid() {

**return** ***serialVersionUID***;

}

#### Class Orders\_Detail

@Entity

@Table(name = "Order\_Details")

**public** **class** OrderDetail **implements** Serializable {

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

@Id

@Column(name = "orderDetailId", unique = **true**)

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

**private** **long** id;

@Column(name = "Price")

**private** **double** price;

@Column(name = "Quantity")

**private** **int** quantity;

@ManyToOne

@JoinColumn(name = "orderId")

**private** Order order;

@ManyToOne

@JoinColumn(name = "productId")

**private** Product product;

* Và các getters/setters/constructors ở phía dưới.

**public** OrderDetail() {

**super**();

}

**public** OrderDetail(**long** id, **double** price, **int** quantity, Order order, Product product) {

**super**();

**this**.id = id;

**this**.price = price;

**this**.quantity = quantity;

**this**.order = order;

**this**.product = product;

}

**public** **long** getId() {

**return** id;

}

**public** **void** setId(**long** id) {

**this**.id = id;

}

**public** **double** getPrice() {

**return** price;

}

**public** **void** setPrice(**double** price) {

**this**.price = price;

}

**public** **int** getQuantity() {

**return** quantity;

}

**public** **void** setQuantity(**int** quantity) {

**this**.quantity = quantity;

}

**public** Order getOrder() {

**return** order;

}

**public** **void** setOrder(Order order) {

**this**.order = order;

}

**public** Product getProduct() {

**return** product;

}

**public** **void** setProduct(Product product) {

**this**.product = product;

}

**public** **static** **long** getSerialversionuid() {

**return** ***serialVersionUID***;

}

#### Class Product

@Entity

@Table(name = "Products")

**public** **class** Product **implements** Serializable {

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

//strategy name hibernate5: name\_column\_id.

@Id

@Column(name = "productId")

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

**private** Integer id;

@Column(name = "Name")

**private** String name;

@Column(name = "Image")

**private** String image;

@Column(name = "Price")

**private** Double price;

@Column(name = "Amount")

**private** **int** amount;

@Column(name = "CreateDate")

@Temporal(TemporalType.***DATE***)

@DateTimeFormat(pattern = "MM-dd-yyyy")

**private** Date createDate = **new** Date();

@ManyToOne

@JoinColumn(name = "category\_Id")

**private** Category category;

@Column(name = "mota")

**private** String mota;

@Column(name = "nsx")

**private** String nsx;

@OneToMany(mappedBy = "product")

**private** List<OrderDetail> orderDetails;

* Và các getters/setters/constructors ở phía dưới.

**public** Product() {

**super**();

}

**public** Product(Integer id, String name, String image, Double price, Integer amount, Date createDate, String mota, String nsx) {

**super**();

**this**.id = id;

**this**.name = name;

**this**.image = image;

**this**.price = price;

**this**.createDate = createDate;

**this**.amount = amount;

**this**.mota = mota;

**this**.nsx = nsx;

}

**public** Product(Integer id, String name, String image, Double price, Integer amount, Date createDate,

Category category, String mota, String nsx, List<OrderDetail> orderDetails) {

**super**();

**this**.id = id;

**this**.name = name;

**this**.image = image;

**this**.price = price;

**this**.createDate = createDate;

**this**.category = category;

**this**.orderDetails = orderDetails;

**this**.amount = amount;

**this**.mota = mota;

**this**.nsx = nsx;

}

**public** Integer getId() {

**return** id;

}

**public** **void** setId(Integer id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getImage() {

**return** image;

}

**public** **void** setImage(String image) {

**this**.image = image;

}

**public** Double getPrice() {

**return** price;

}

**public** **void** setPrice(Double price) {

**this**.price = price;

}

**public** Date getCreateDate() {

**return** createDate;

}

**public** **void** setCreateDate(Date createDate) {

**this**.createDate = createDate;

}

**public** Category getCategory() {

**return** category;

}

**public** **void** setCategory(Category category) {

**this**.category = category;

}

**public** **int** getAmount() {

**return** amount;

}

**public** **void** setAmount(**int** amount) {

**this**.amount = amount;

}

**public** List<OrderDetail> getOrderDetails() {

**return** orderDetails;

}

**public** **void** setOrderDetails(List<OrderDetail> orderDetails) {

**this**.orderDetails = orderDetails;

}

**public** String getMota() {

**return** mota;

}

**public** **void** setMota(String mota) {

**this**.mota = mota;

}

**public** String getNsx() {

**return** nsx;

}

**public** **void** setNsx(String nsx) {

**this**.nsx = nsx;

}

### Lập trình các lớp tiện ích assignment.util

#### Lớp tiện ích MailUtil

@Data

@NoArgsConstructor

@AllArgsConstructor

**public** **class** MailInfo {

String from = "Quang<quangdodps15144@fpt.edu.vn>";

String to;

String subject;

String body;

**public** MailInfo(String to, String subject, String body) {

**this**.to = to;

**this**.subject = subject;

**this**.body = body;

}

}

### Lập trình các lớp Service xử lý nghiệp vụ

#### Lập trình truy vấn cơ sở dữ liệu

* Các class truy vấn dữ liệu đều là Interface được kế thừa từ Class JpaRepository

#### Lập trình gửi mail

@Controller

**public** **class** MailerController {

@Autowired

MailerService mailer;

@Autowired

AccountDAO dao;

@GetMapping("/bodyfragments/guima")

**public** String mailViewer() {

**return** "bodyfragments/quUser";

}

@PostMapping("/user/passwordMail")

**public** String send(Model model, @RequestParam String txtTo) {

**if** (txtTo.equals("")) {

model.addAttribute("message", "Không được để trống email");

**return** "bodyfragments/layma";

} **else** {

**try** {

Account account = dao.findByEmail(txtTo);

mailer.queue(txtTo, "Mật khẩu","Mật khẩu của tài khoản: "+ account.getUsername()+" là: "+ account.getPassword());

model.addAttribute("message", "Mật khẩu của bạn đã được gửi qua email!");

System.***out***.println("Đã gửi");

**return** "redirect:/bodyfragments/dangnhap";

} **catch** (Exception e) {

model.addAttribute("message", "Không có tài khoản đăng ký email này");

**return** "bodyfragments/layma";

}

}

}

}

##### Xây dựng interface MailService

**public** **interface** MailerService {

/\*\*

\* Gửi email

\* **@param** mail thông tin email

\* **@throws** MessagingException lỗi gửi email

\*/

**void** send(MailInfo mail) **throws** MessagingException;

/\*\*

\* Gửi email đơn giản

\* **@param** to email người nhận

\* **@param** subject tiêu đề email

\* **@param** body nội dung email

\* **@throws** MessagingException lỗi gửi email

\*/

**void** send(String to, String subject, String body) **throws** MessagingException;

**void** queue(MailInfo mail);

/\*\*

\* Tạo MailInfo và xếp vào hàng đợi

\* **@param** to email người nhận

\* **@param** subject tiêu đề email

\* **@param** body nội dung email

\*/

**void** queue(String to, String subject, String body);

}

##### Xậy dựng MailServiceImplement interface MailService

@Service

**public** **class** MailServiceImpl **implements** MailerService{

@Autowired

JavaMailSender sender;

List<MailInfo> list = **new** ArrayList<>();

@Override

**public** **void** send(MailInfo mail) **throws** MessagingException {

MimeMessage message = sender.createMimeMessage();

// Sử dụng Helper để thiết lập các thông tin cần thiết cho message

MimeMessageHelper helper = **new** MimeMessageHelper(message, **true**, "utf-8");

helper.setFrom(mail.getFrom());

helper.setTo(mail.getTo());

helper.setSubject(mail.getSubject());

helper.setText(mail.getBody(), **true**);

helper.setReplyTo(mail.getFrom());

// Gửi message đến SMTP server

sender.send(message);

}

@Override

**public** **void** send(String to, String subject, String body) **throws** MessagingException {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** queue(MailInfo mail) {

// **TODO** Auto-generated method stub

list.add(mail);

}

@Override

**public** **void** queue(String to, String subject, String body) {

// **TODO** Auto-generated method stub

queue(**new** MailInfo(to, subject, body));

}

/\* MailInfo từ hàng đợi và gửi đi (5 giây sẽ kiểm tra và gửi một lần) \*/

@Scheduled(fixedDelay = 5000)

**public** **void** run() {

**while**(!list.isEmpty()) {

MailInfo mail = list.remove(0);

**try** {

**this**.send(mail);

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

}

### Lập trình xử lý giỏ hàng

@Controller

**public** **class** ShoppingCartController {

@Autowired

ShoppingCartService cart;

@RequestMapping("/bodyfragments/cart")

**public** String view(Model md) {

md.addAttribute("cartsl", cart);

md.addAttribute("Count", cart.getCount());

md.addAttribute("Amount", cart.getAmount());

**return** "/bodyfragments/cart";

}

@RequestMapping("/bodyfragments/cart/them/{id}")

**public** String add(@PathVariable("id") Integer id) {

cart.add(id);

**return** "redirect:/bodyfragments/cart";

}

@RequestMapping("/bodyfragments/cart/xoa/{id}")

**public** String remove(@PathVariable("id") Integer id) {

cart.remove(id);

**return** "redirect:/bodyfragments/cart";

}

@RequestMapping("/bodyfragments/cart/sua/{id}")

**public** String update(@PathVariable("id") Integer id, @RequestParam("qty") Integer qty) {

cart.update(id, qty);

**return** "redirect:/bodyfragments/cart";

}

@RequestMapping("/bodyfragments/cart/trang")

**public** String clear() {

cart.clear();

**return** "redirect:/bodyfragments/cart";

}

}

#### Xây dựng interface ShoppingCartService

**public** **interface** ShoppingCartService {

Product add(Integer id);

**void** remove(Integer id);

**void** delete(Integer id);

Product update(Integer id, **int** qty);

**void** clear();

Collection<Product> getItems();

**int** getCount();

**double** getAmount();

#### }

#### Lập trình ShoppingCartServiceImplement implement ShoppingCartService

@SessionScope

@Service

**public** **class** ShoppingCartServiceImpl **implements** ShoppingCartService{

@Autowired

ProductDAO pdD;

Map<Integer, Product> map = **new** HashMap<>();

@Override

**public** Product add(Integer id) {

Product it = map.get(id);

**if**(it == **null**) {

it = pdD.findID(id);

it.setAmount(1);

map.put(id, it);

}**else** {

it.setAmount(it.getAmount() +1);

map.put(id, it);

}

**return** it;

}

@Override

**public** **void** remove(Integer id) {

map.remove(id);

}

@Override

**public** **void** delete(Integer id) {

pdD.deleteID(id);

}

@Override

**public** Product update(Integer id, **int** qty) {

Product it = map.get(id);

it.setAmount(qty);

map.put(id, it);

**return** it;

}

@Override

**public** **void** clear() {

map.clear();

}

@Override

**public** Collection<Product> getItems(){

Collection<Product> collection = **new** ArrayList<Product>();

Iterator<Map.Entry<Integer, Product>> iterator = map.entrySet().iterator();

**while**(iterator.hasNext()) {

Map.Entry<Integer, Product> entry = iterator.next();

Product item = entry.getValue();

collection.add(item);

}

**return** collection;

}

@Override

**public** **int** getCount() {

**return** map.values().stream().mapToInt(item -> item.getAmount()).sum();

}

@Override

**public** **double** getAmount() {

**return** map.values().stream().mapToDouble(item -> item.getPrice()\*item.getAmount()).sum();

}

}

### Lập trình Controller xử lý Request từ Client Web Browser

#### AccountController

@Controller

**public** **class** AccountController {

@Autowired

AccountDAO dao;

@Autowired

SessionService session;

@Autowired

**private** UserRepository repo;

@GetMapping("/bodyfragments/dangky")

**public** String dangky() {

**return** "bodyfragments/dangky";

}

@GetMapping("/bodyfragments/dangnhap")

**public** String dangnhap() {

**return** "bodyfragments/dangnhap";

}

@PostMapping("/dangky")

**public** String dangky(@RequestParam("username") String username, @RequestParam("password") String password,

@RequestParam("fullname") String fullname, @RequestParam("email") String email) {

Account account = **new** Account();

account.setUsername(username);

account.setPassword(password);

account.setFullname(fullname);

account.setEmail(email);

account.setActivated(**true**);

repo.save(account);

**return** "bodyfragments/thongbao";

}

@PostMapping("/dangnhap")

**public** String dangnhap(Model model, @RequestParam("username") String username,

@RequestParam("password") String password) {

**try** {

Account user = dao.getOne(username);

**if** (!user.getPassword().equals(password)) {

model.addAttribute("message", "Invalid password");

} **else** {

String uri = session.getAttribute("security-uri");

**if** (uri != **null**) {

**return** "redirect:" + uri;

} **else** {

**if** (user.isActivated()) {

**if** (user.isAdmin()) {

model.addAttribute("message", "Login succeed");

session.setAttribute("user", user);

**return** "redirect:/admin/user";

} **else** {

model.addAttribute("message", "Login succeed");

session.setAttribute("user", user);

**return** "redirect:/bodyfragments/than";

}

} **else** {

model.addAttribute("message",

"Login failed , Your account has been banned. Contact ADMIN to unlock your account");

}

}

}

} **catch** (Exception e) {

model.addAttribute("message", "User does not exist");

}

**return** "bodyfragments/dangnhap";

}

@RequestMapping("/dangxuat")

**public** String logout() {

session.removeAttribute("user");

**return** "redirect:/bodyfragments/dangnhap";

}

}

#### CartController

@Controller

**public** **class** ShoppingCartController {

@Autowired

ShoppingCartService cart;

@RequestMapping("/bodyfragments/cart")

**public** String view(Model md) {

md.addAttribute("cartsl", cart);

md.addAttribute("Count", cart.getCount());

md.addAttribute("Amount", cart.getAmount());

**return** "/bodyfragments/cart";

}

@RequestMapping("/bodyfragments/cart/them/{id}")

**public** String add(@PathVariable("id") Integer id) {

cart.add(id);

**return** "redirect:/bodyfragments/cart";

}

@RequestMapping("/bodyfragments/cart/xoa/{id}")

**public** String remove(@PathVariable("id") Integer id) {

cart.remove(id);

**return** "redirect:/bodyfragments/cart";

}

@RequestMapping("/bodyfragments/cart/sua/{id}")

**public** String update(@PathVariable("id") Integer id, @RequestParam("qty") Integer qty) {

cart.update(id, qty);

**return** "redirect:/bodyfragments/cart";

}

@RequestMapping("/bodyfragments/cart/trang")

**public** String clear() {

cart.clear();

**return** "redirect:/bodyfragments/cart";

}

}

#### HomeController

@Controller

**public** **class** MainController {

@Autowired

ProductDAO productDAO;

@Autowired

SessionService sessionService;

// chay bang cai nay truoc

@RequestMapping("/bodyfragments/than")

**public** String homePage(Model model, @RequestParam("keyword") Optional<String> keyword,

@RequestParam("p") Optional<Integer> p) {

String sessionKeyword = sessionService.getAttribute("keyword");

**if**(sessionKeyword == **null**) {

sessionKeyword = "";

}

String kword = keyword.orElse(sessionKeyword);

sessionService.setAttribute("keyword", kword);

Pageable pageable = PageRequest.*of*(p.orElse(0), 8);

Page<Product> page = productDAO.findAllByNameLike("%"+kword+"%", pageable);

model.addAttribute("page", page);

**return** "bodyfragments/than";

}

@RequestMapping("/bodyfragments/hotro")

**public** String hotro(Model model) {

**return** "bodyfragments/hotro";

}

@RequestMapping("/bodyfragments/tintuc")

**public** String tintuc(Model model) {

**return** "bodyfragments/tintuc";

}

@RequestMapping("/bodyfragments/giamgia")

**public** String giamgia(Model model) {

**return** "bodyfragments/giamgia";

}

// chung nao ong lam form dangnhap.jsp thi de no vao file bodyfragments la chay dc

@RequestMapping("/bodyfragments/dangnhap")

**public** String dangnhap(Model model) {

**return** "bodyfragments/dangnhap";

}

@RequestMapping("/bodyfragments/dangky")

**public** String dangky(Model model) {

**return** "bodyfragments/dangky";

}

@RequestMapping("/bodyfragments/doimatkhau")

**public** String doimatkhau(Model model) {

**return** "bodyfragments/doimatkhau";

}

@RequestMapping("/bodyfragments/layma")

**public** String guimail(Model model) {

**return** "bodyfragments/layma";

}

}

# Tổng kết

## Thuận lợi

* Các thành viên trong nhóm luôn cố gắn hoàn thành, giúp đỡ fix bug
* Được giảng viên chỉ đạo và hướng dẫn xây dựng chức năng

## Khó khăn

* Nhiều bug khó hiểu không thể tự giải quyết
* Giai đoạn đầu còn nhiều bắt trắc vì tiếp cận công cụ sử lý mã nguồn mới

## Kinh nghiệm đạt được

* Đã biết cách thêm, xóa, sửa khi sử dụng spring boot
* Biết tạo các trang layout để chạy trên website
* Thành thục hơn khi sử dụng jsp, css, js và chạy miền bằng spring boot

## Hướng phát triển

* Cần thêm đầu báo mới
* Công cụ giúp giảm giá
* Mua được nhiều mặc hàng một lần
* Có thể cho đọc sách một đoạn đầu khi bấm vào chi tiết của mỗi cuốn sách

# Lời kết

* Báo cáo trên là toàn bộ tài liệu về ứng dụng website bán đồ điện từ online TCO Shop. Với sự giúp đỡ nhiệt tình, tận tâm đến từ thầy Lê Anh Tú. Mặc dù website có thể còn lỗi và thiếu những chức năng thêm nhưng nhóm chúng em đã hoàn thành dự án bằng hết khả năng của mình. Nhóm chúng em xin chân thành cảm ơn thầy!

**Nhóm sinh viên thực hiện**

Diệp Ông Đại Quang

Trần Gia Bảo

Trần Thanh Tùng

Ngô Ngọc Tài

Nguyễn Hoàng Nhi