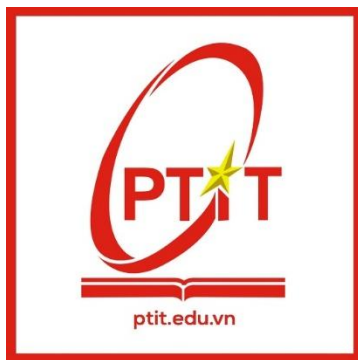


HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG CƠ SỞ TẠI TP. HCM
KHOA CÔNG NGHỆ THÔNG TIN 2



BÁO CÁO ĐỊNH KỲ LẦN 1
THỰC TẬP TỐT NGHIỆP ĐẠI HỌC

Đề tài: Xây dựng website bán sách trực tuyến và tại cửa hàng
có hỗ trợ tư vấn bằng AI Agent

Giảng viên hướng dẫn: Thầy Nguyễn Trung Hiếu
Sinh viên thực hiện: Nguyễn Ngọc Thiên Phúc - N21DCCN066
Trần Thị Thùy Ngân – N21DCCN055
Lớp: D21CQCNP01-N
Khóa: 2021 – 2026
Ngành: Công nghệ thông tin
Hệ: Đại học Chính Quy

TP. Hồ Chí Minh, ngày 16 tháng 07 năm 2025

PHIẾU GIAO ĐỀ TÀI

LỜI CẢM ƠN

Trong thời đại số, nhu cầu xây dựng các hệ thống phần mềm ngày càng cao, đòi hỏi quy trình thiết kế, lập trình và triển khai cần được thực hiện một cách bài bản để đáp ứng yêu cầu thực tiễn. Để đáp ứng xu thế này, nhóm chúng em đã thực hiện đề tài “Xây dựng website bán sách trực tuyến và tại cửa hàng có hỗ trợ tư vấn bằng AI Agent”.

Trong quá trình thực hiện đề tài, nhóm chúng em không chỉ củng cố kiến thức về thiết kế hệ thống và mô hình hóa nghiệp vụ, mà còn rèn luyện kỹ năng lập trình và triển khai phần mềm theo hướng chuyên nghiệp và thực tiễn hơn.

Chúng em xin chân thành cảm ơn thầy Nguyễn Trung Hiếu đã tận tình hướng dẫn, truyền đạt những kiến thức quý báu và luôn hỗ trợ, giải đáp mọi thắc mắc trong suốt quá trình làm đề tài. Dù đã nỗ lực hết mình, nhưng do kiến thức và kinh nghiệm còn hạn chế, nhóm em khó tránh khỏi những thiếu sót trong quá trình thực hiện. Rất mong thầy có thể góp ý, chỉ bảo để nhóm em hoàn thiện sản phẩm tốt hơn.

MỤC LỤC

Phiếu giao đề tài.....	1
Lời cảm ơn.....	2
Mục lục	3
Danh mục các bảng	5
Danh mục các hình ảnh.....	6
Chương 1: Giới thiệu đề tài	7
1. Tên đề tài: “Xây dựng website bán sách trực tuyến và tại cửa hàng có hỗ trợ tư vấn bằng AI Agent”	7
2. Lý do chọn đề tài:	7
3. Mục tiêu nghiên cứu:.....	7
4. Cơ sở lý thuyết:	7
Chương 2: Phân tích, thiết kế hệ thống.....	11
1. Yêu cầu từ stake-holders (người sử dụng):	11
2. Mô tả hệ thống bằng ngôn ngữ tự nhiên:	12
3. Công nghệ và tài nguyên sử dụng:	12
4. Sơ đồ diagram:	13
5. Từ điển dữ liệu:	13
Chương 3: Xây dựng và phát triển hệ thống	17
1. Xây dựng các API quan trọng:	17
2. Giao diện:	24
Kết luận	31

1. Nhận xét về điểm mạnh, điểm yếu trong cách tiến hành đồ án:	31
2. Khả năng cải tiến cách tiến hành đồ án:.....	31
Tài liệu tham khảo	32

DANH MỤC CÁC BẢNG

Bảng 1: Bảng Category	14
Bảng 2: Bảng Author	14
Bảng 3: Bảng Publisher	14
Bảng 4: Bảng Book	15
Bảng 5: Bảng Customer.....	15
Bảng 6: Bảng Staff	15
Bảng 7: Bảng Promotion	16
Bảng 8: Bảng Order	16
Bảng 9: Bảng OrderItem	16

DANH MỤC CÁC HÌNH ẢNH

Hình 1: Mô hình MVC	8
Hình 2: ASP.NET	10
Hình 3: Sơ đồ Diagram.....	13
Hình 4: Giao diện đăng ký tài khoản nhân viên	24
Hình 5: Giao diện đăng nhập nhân viên	24
Hình 6: Giao diện quản lý thẻ loại.....	25
Hình 7: Giao diện quản lý tác giả.....	25
Hình 8: Giao diện quản lý nhà xuất bản	26
Hình 9: Giao diện quản lý sách	26
Hình 10: Giao diện quản lý khách hàng	27
Hình 11: Giao diện quản lý khuyến mãi.....	27
Hình 12: Giao diện quản lý hóa đơn.....	28
Hình 13: Giao diện trang chủ	28
Hình 14: Giao diện danh sách sách	29
Hình 15: Giao diện gợi ý sách	29
Hình 16: Giao diện sách liên quan	30
Hình 17: Giao diện chi tiết sách	30

CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI

1. Tên đề tài: “Xây dựng website bán sách trực tuyến và tại cửa hàng có hỗ trợ tư vấn bằng AI Agent”

2. Lý do chọn đề tài:

Trong thời đại chuyển đổi số, việc kết hợp giữa mô hình kinh doanh truyền thống và thương mại điện tử ngày càng trở nên phổ biến. Các cửa hàng sách không chỉ cần tối ưu hóa quy trình bán hàng tại quầy mà còn phải mở rộng kênh bán hàng trực tuyến nhằm tiếp cận nhiều khách hàng hơn. Ngoài ra, ứng dụng trí tuệ nhân tạo (AI) trong việc tư vấn chọn sách đang là xu hướng hiện đại, giúp nâng cao trải nghiệm người dùng. Vì vậy, việc xây dựng một hệ thống website bán sách tích hợp cả bán hàng tại cửa hàng và trực tuyến, đồng thời hỗ trợ tư vấn sách bằng AI Agent, là một giải pháp toàn diện, phù hợp với nhu cầu thực tiễn và xu hướng công nghệ hiện nay.

3. Mục tiêu nghiên cứu:

- Tìm hiểu và phân tích quy trình nghiệp vụ thực tế tại cửa hàng sách.
- Tìm hiểu và phân tích quy trình hoạt động tại cửa hàng sách truyền thống cũng như hành vi người dùng trên kênh trực tuyến.
- Thiết kế hệ thống backend theo kiến trúc RESTful API với ASP.NET Core, áp dụng Repository Design Pattern và các nguyên tắc phát triển phần mềm hiện đại.
- Xây dựng cơ sở dữ liệu quan hệ trên nền tảng SQL Server để phục vụ cho cả hai hình thức bán hàng.
- Xây dựng giao diện frontend bằng ReactJS, gồm giao diện dành cho nhân viên tại cửa hàng và giao diện dành cho khách hàng mua sắm trực tuyến.
- Tích hợp AI Agent để tư vấn chọn sách phù hợp dựa trên hành vi xem và mua hàng của người dùng.
- Triển khai và kiểm thử các chức năng quản lý sách, khuyến mãi, khách hàng, đơn hàng và thanh toán, đảm bảo hệ thống hoạt động hiệu quả và thân thiện với người dùng.

4. Cơ sở lý thuyết:

- **Kiến trúc phần mềm:**

+ Mô hình MVC:

MVC (Model-View-Controller) là một mẫu kiến trúc phần mềm được sử dụng rộng rãi trong việc phát triển ứng dụng để tạo ra các giao diện người dùng trực quan và có khả năng tương tác cao. MVC chia một ứng dụng thành 3 phần chính và mỗi phần có một vai trò riêng biệt:

1. Model đại diện cho dữ liệu và quy tắc nghiệp vụ của ứng dụng.
2. View chịu trách nhiệm hiển thị dữ liệu cho người dùng một cách trực quan và tương tác.
3. Controller đóng vai trò là cầu nối giữa Model và View, xử lý các yêu cầu từ người dùng và cập nhật giao diện tương ứng.



Hình 1: Mô hình MVC

Các thành phần:

- Model là lớp đại diện cho dữ liệu của ứng dụng. Nó có thể là một cơ sở dữ liệu, một file cấu hình hoặc một đối tượng phức tạp. Model chịu trách nhiệm lưu trữ, truy xuất và cập nhật dữ liệu.
- View là lớp giao diện người dùng. Nó hiển thị dữ liệu từ Model cho người dùng và cho phép người dùng tương tác với ứng dụng. View thường được xây dựng bằng các ngôn ngữ template như HTML, JSP hoặc React.

- Controller là lớp điều khiển luồng của ứng dụng. Nó nhận các yêu cầu từ người dùng, cập nhật Model và chọn View phù hợp để hiển thị.

+ **RESTful API:**

RESTful API là một tiêu chuẩn dùng trong việc thiết kế API cho các ứng dụng web (thiết kế Web services) để tiện cho việc quản lý các resource. Nó chú trọng vào tài nguyên hệ thống (tệp văn bản, ảnh, âm thanh, video, hoặc dữ liệu động...), bao gồm các trạng thái tài nguyên được định dạng và được truyền tải qua HTTP.

API (Application Programming Interface) là một tập các quy tắc và cơ chế mà theo đó, một ứng dụng hay một thành phần sẽ tương tác với một ứng dụng hay thành phần khác. API có thể trả về dữ liệu mà bạn cần cho ứng dụng của mình ở những kiểu dữ liệu phổ biến như [JSON](#) hay XML.

REST (REpresentational State Transfer) là một dạng chuyển đổi cấu trúc dữ liệu, một kiểu kiến trúc để viết API. Nó sử dụng phương thức HTTP đơn giản để tạo cho giao tiếp giữa các máy. Vì vậy, thay vì sử dụng một URL cho việc xử lý một số thông tin người dùng, REST gửi một yêu cầu HTTP như GET, POST, DELETE, vv đến một URL để xử lý dữ liệu.

RESTful API là một tiêu chuẩn dùng trong việc thiết kế các API cho các ứng dụng web để quản lý các resource. RESTful là một trong những kiểu thiết kế API được sử dụng phổ biến ngày nay để cho các ứng dụng (web, mobile...) khác nhau giao tiếp với nhau.

Chức năng quan trọng nhất của **REST** là quy định cách sử dụng các HTTP method (như GET, POST, PUT, DELETE...) và cách định dạng các URL cho ứng dụng web để quản các resource. RESTful không quy định logic code ứng dụng và không giới hạn bởi ngôn ngữ lập trình ứng dụng, bất kỳ ngôn ngữ hoặc framework nào cũng có thể sử dụng để thiết kế một **RESTful API**.

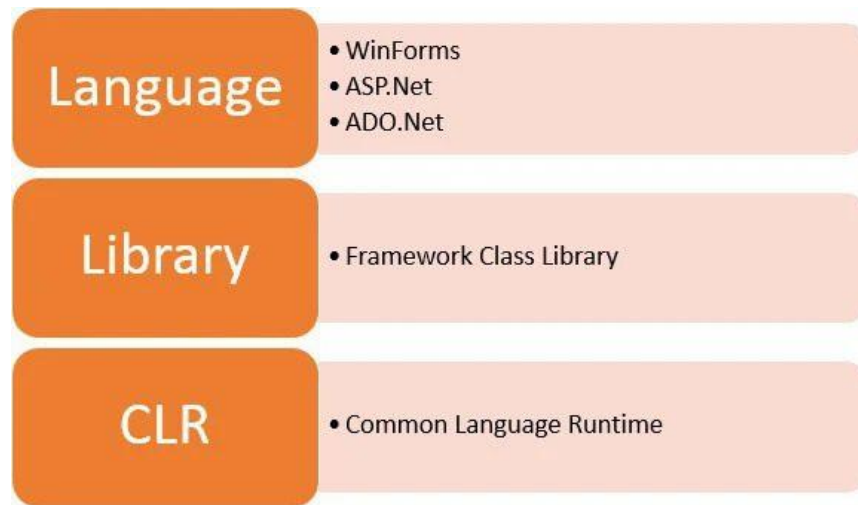
– **Công nghệ sử dụng:**

+ **ASP.NET:**

ASP.NET là một framework mã nguồn mở phía server được phát triển bởi [Microsoft](#), nhằm mục đích xây dựng các ứng dụng web động và dịch vụ web. Nó

cho phép các lập trình viên tạo ra các trang web và ứng dụng sử dụng HTML5, CSS, và JavaScript. Được giới thiệu lần đầu tiên vào đầu thế kỷ 21, ASP.NET đã nhanh chóng trở thành một công cụ quan trọng trong việc phát triển ứng dụng web, đặc biệt là trong môi trường doanh nghiệp.

Thành phần:



Hình 2: ASP.NET

+ ReactJS:

React.js là một thư viện Javascript đang nổi lên trong những năm gần đây với xu hướng Single Page Application. Trong khi những framework khác cố gắng hướng đến một mô hình MVC hoàn thiện thì React nổi bật với sự đơn giản và dễ dàng phối hợp với những thư viện Javascript khác. Nếu như AngularJS là một Framework cho phép nhúng code javascript trong code html thông qua các attribute như ng-model, ng-repeat...thì với react là một library cho phép nhúng code html trong code javascript nhờ vào JSX, bạn có thể dễ dàng lồng các đoạn HTML vào trong JS. Tích hợp giữa javascript và HTML vào trong JSX làm cho các component dễ hiểu hơn

CHƯƠNG 2: PHÂN TÍCH, THIẾT KẾ HỆ THỐNG

1. Yêu cầu từ stake-holders (người sử dụng):

Đối tượng 1: Nhân viên cửa hàng sách

Mong muốn & nhu cầu:

- Hiểu và ghi nhận quy trình vận hành hiện tại tại cửa hàng sách, bao gồm: tiếp nhận đơn hàng, kiểm tra tồn kho, áp dụng khuyến mãi, xử lý thanh toán và xuất hóa đơn.
- Giảm thiểu công việc giấy tờ thủ công: Tránh ghi chép bằng tay, tính toán thủ công, dễ sai sót.
- Tăng tốc độ phục vụ khách hàng tại quầy, đặc biệt vào giờ cao điểm hoặc các dịp khuyến mãi.
- Hỗ trợ tìm kiếm và kiểm tra thông tin sách nhanh chóng, bao gồm tên sách, tồn kho, giá bán, khuyến mãi đang áp dụng.
- Quản lý danh mục sách, tác giả, nhà xuất bản, nhân viên và khách hàng một cách rõ ràng, có phân quyền truy cập phù hợp.
- Theo dõi và thống kê đơn hàng đã bán, báo cáo doanh thu theo ngày/tháng/quý.

Yêu cầu cụ thể được đề xuất:

- Giao diện đơn giản, dễ sử dụng cho nhân viên không chuyên CNTT.
- Có thể tạo nhanh đơn hàng từ danh sách sách đã chọn.
- Hệ thống gợi ý khuyến mãi phù hợp với giỏ hàng hiện tại (nếu có).
- Chức năng tìm kiếm, lọc sách theo tên, mã ISBN, tác giả, danh mục.
- Hỗ trợ khôi phục mật khẩu nếu quên.

Đối tượng 2: Người mua sách trực tuyến

Mong muốn & nhu cầu:

- Giao diện mua sắm thân thiện, dễ sử dụng trên cả máy tính và thiết bị di động.
- Duyệt sách theo danh mục, tìm kiếm nhanh theo từ khóa, tác giả, nhà xuất bản, đánh giá.
- Xem thông tin chi tiết sách, giá bán, khuyến mãi đang áp dụng.

- Thêm sách vào giỏ hàng, đặt mua và thanh toán nhanh chóng, hỗ trợ nhiều phương thức thanh toán.
- Theo dõi đơn hàng, xem lịch sử mua hàng, hủy đơn nếu cần.
- Tư vấn chọn sách phù hợp dựa trên hành vi đã xem hoặc mua trước đó – hỗ trợ bởi AI Agent.
- Đăng ký tài khoản, quản lý thông tin cá nhân, đổi mật khẩu hoặc khôi phục nếu quên.

Yêu cầu cụ thể được đề xuất:

- Hệ thống AI có thể gợi ý sách liên quan hoặc phù hợp theo sở thích.
- Gửi email xác nhận đơn hàng và trạng thái giao hàng định kỳ.
- Giao diện responsive, tương thích với điện thoại và máy tính bảng.
- Hỗ trợ đăng ký và đăng nhập an toàn qua email, có xác thực.

2. Mô tả hệ thống bằng ngôn ngữ tự nhiên:

Hệ thống được xây dựng nhằm hỗ trợ cả hai hình thức bán sách tại cửa hàng và trực tuyến. Người dùng có thể đăng ký, đăng nhập, đổi mật khẩu, và lấy lại mật khẩu qua email. Nhân viên có thể quản lý danh mục sách, tác giả, nhà xuất bản, khách hàng, khuyến mãi, và nhân viên. Khi khách mua tại cửa hàng, nhân viên có thể tạo đơn hàng tại quầy, áp dụng khuyến mãi (nếu có) và tiến hành thanh toán.

Đối với khách hàng trực tuyến, hệ thống cung cấp website giúp duyệt và tìm kiếm sách, thêm sản phẩm vào giỏ hàng, đặt hàng và thanh toán. Ngoài ra, hệ thống tích hợp AI Agent nhằm tư vấn chọn sách phù hợp với hành vi mua sắm và sở thích của người dùng, góp phần nâng cao trải nghiệm cá nhân hóa.

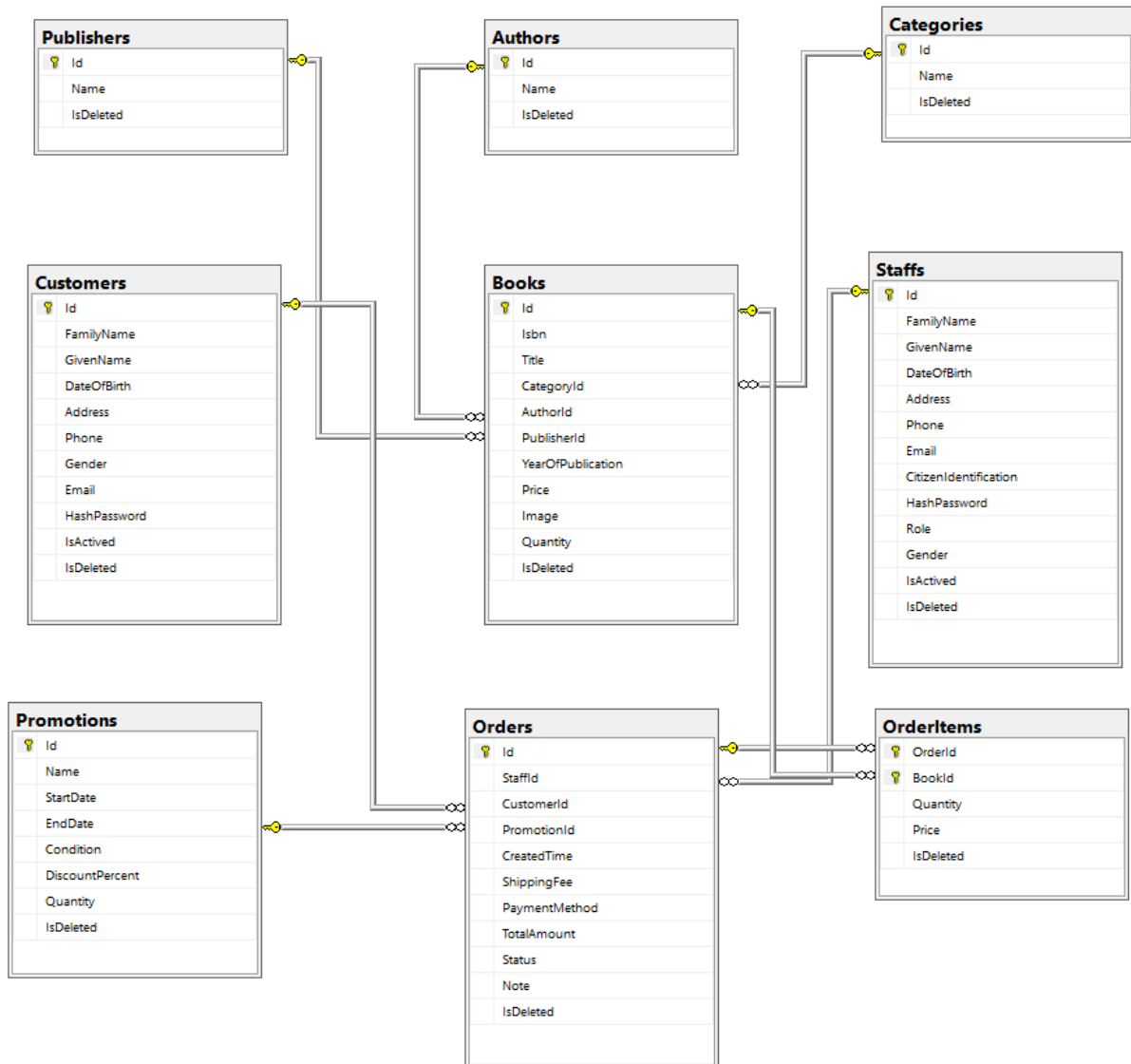
Tất cả các chức năng backend được xây dựng dưới dạng RESTful API và được frontend sử dụng thông qua ReactJS.

3. Công nghệ và tài nguyên sử dụng:

- **Backend:** ASP.NET Core MVC
- **Pattern:** Repository Design Pattern + Entity Framework Core
- **Frontend:** ReactJS

- **Cơ sở dữ liệu:** Microsoft SQL Server
- **AI Agent:** GPT-4 API
- **IDE:** Visual Studio 2022 (backend), Visual Studio Code (frontend)
- **Công cụ khác:** Postman (kiểm thử API), GitHub (quản lý mã nguồn), Docker (triển khai)

4. Sơ đồ diagram:



Hình 3: Sơ đồ Diagram

5. Từ điển dữ liệu:

- **Bảng Category:**

Category				
ID	Attribute	Type	Constraint	Note
1	id	uniqueidentifier	PK, DEFAULT = NEWID()	ID, tự động generate khi new create
2	name	nvarchar(100)	UK, NOT NULL	Tên
3	isDeleted	bit	NOT NULL, DEFAULT = 0	Trạng thái xóa, mặc định là false

Bảng 1: Bảng Category

– **Bảng Author:**

Author				
ID	Attribute	Type	Constraint	Note
1	id	uniqueidentifier	PK, DEFAULT = NEWID()	ID, tự động generate khi new create
2	name	nvarchar(100)	NOT NULL	Tên (có thể trùng, đã có trường hợp trùng bút danh)
3	isDeleted	bit	NOT NULL, DEFAULT = 0	Trạng thái xóa, mặc định là false

Bảng 2: Bảng Author

– **Bảng Publisher:**

Publisher				
ID	Attribute	Type	Constraint	Note
1	id	uniqueidentifier	PK, DEFAULT = NEWID()	ID, tự động generate khi new create
2	name	nvarchar(100)	UK, NOT NULL	Tên
3	isDeleted	bit	NOT NULL, DEFAULT = 0	Trạng thái xóa, mặc định là false

Bảng 3: Bảng Publisher

– **Bảng Book:**

Book				
ID	Attribute	Type	Constraint	Note
1	id	uniqueidentifier	PK, DEFAULT = NEWID()	ID, tự động generate khi new create
2	isbn	char(13)	UK, NOT NULL	Mã isbn
3	title	nvarchar(100)	NOT NULL	Tên
4	categoryId	uniqueidentifier	FK, NOT NULL	Mã thể loại
5	authorId	uniqueidentifier	FK, NOT NULL	Mã tác giả
6	publisherId	uniqueidentifier	FK, NOT NULL	Mã nhà xuất bản
7	yearOfPublication	smallint	NOT NULL, > 1500	Năm xuất bản
8	price	decimal(8, 0)	NOT NULL, > 1000	Giá tiền ở thời điểm hiện tại
9	image	varchar(max)		URL hình ảnh (lưu ở server, đừng lấy link onl)
10	quantity	int	NOT NULL, >= 0	Số lượng tồn

11	isDeleted	bit	NOT NULL, DEFAULT = 0	Trạng thái xóa, mặc định là false
----	-----------	-----	--------------------------	-----------------------------------

Bảng 4: Bảng Book

– **Bảng Customer:**

Customer				
ID	Attribute	Type	Constraint	Note
1	id	uniqueidentifier	PK, DEFAULT = NEWID()	ID, tự động generate khi new create
2	familyName	nvarchar(70)	NOT NULL	Họ và tên đệm
3	givenName	nvarchar(30)	NOT NULL	Tên
4	dateOfBirth	date	NOT NULL	Ngày sinh
5	address	nvarchar(50)	NOT NULL	Địa chỉ
6	phone	char(10)	UK, NOT NULL	Số điện thoại
7	gender	bit	NOT NULL, DEFAULT = 0	Giới tính, 0 = nam, 1 = nữ
8	email	varchar(50)	UK	Địa chỉ mail
9	hashPassword	varchar(255)		Mật khẩu mã hóa
10	isActive	bit	NOT NULL, DEFAULT = 0	Trạng thái kích hoạt tài khoản, 0 = chưa, 1 = rồi
11	isDeleted	bit	NOT NULL, DEFAULT = 0	Trạng thái xóa, mặc định là false

Bảng 5: Bảng Customer

– **Bảng Staff:**

Staff				
ID	Attribute	Type	Constraint	Note
1	id	uniqueidentifier	PK, DEFAULT = NEWID()	ID, tự động generate khi new create
2	familyName	nvarchar(70)	NOT NULL	Họ và tên đệm
3	givenName	nvarchar(30)	NOT NULL	Tên
4	dateOfBirth	date	NOT NULL	Ngày sinh
5	address	nvarchar(50)	NOT NULL	Địa chỉ
6	phone	char(10)	UK, NOT NULL	Số điện thoại
7	email	varchar(50)	UK, NOT NULL	Địa chỉ mail
8	citizenIdentification	char(12)	UK, NOT NULL	Mã định danh
9	hashPassword	varchar(255)	NOT NULL	Mật khẩu mã hóa
10	role	bit	NOT NULL, DEFAULT = 0	Quyền, 0 = staff, 1 = admin
11	gender	bit	NOT NULL, DEFAULT = 0	Giới tính, 0 = nam, 1 = nữ
12	isActive	bit	NOT NULL, DEFAULT = 0	Trạng thái kích hoạt tài khoản, 0 = chưa, 1 = rồi
13	isDeleted	bit	NOT NULL, DEFAULT = 0	Trạng thái xóa, mặc định là false

Bảng 6: Bảng Staff

– **Bảng Promotion:**

Promotion				
ID	Attribute	Type	Constraint	Note
1	id	uniqueidentifier	PK, DEFAULT = NEWID()	ID, tự động generate khi new create
2	name	nvarchar(100)	UK, NOT NULL	Tên, không được trùng
3	startDate	datetime2	NOT NULL	Ngày bắt đầu
4	endDate	datetime2	NOT NULL, > startDate	Ngày kết thúc
5	condition	decimal(8, 0)	NOT NULL, > 1000	Điều kiện tối thiểu
6	discountPercent	decimal(3, 2)	NOT NULL, > 0.0	Phần trăm khuyến mãi
7	quantity	smallint	NOT NULL, >= 0	Số lượng
8	isDeleted	bit	NOT NULL, DEFAULT = 0	Trạng thái xóa, mặc định là false

Bảng 7: Bảng Promotion

– **Bảng Order:**

Order				
ID	Attribute	Type	Constraint	Note
1	id	uniqueidentifier	PK, DEFAULT = NEWID()	ID, tự động generate khi new create
2	staffId	uniqueidentifier	FK	Mã nhân viên (cho phép null nếu khách mua onl)
3	customerId	uniqueidentifier	FK, NOT NULL	Mã khách hàng
4	promotionId	uniqueidentifier	FK	Mã khuyến mãi
5	createdTime	datetime2	NOT NULL, DEFAULT = SYSDATETIME()	Thời gian tạo hóa đơn
6	shippingFee	decimal(11, 3)	DEFAULT = 0.0	Tiền ship (nếu mua online), còn mua tại store thì ship = 0
7	paymentMethod	nvarchar(50)	NOT NULL	Phương thức thanh toán
8	totalAmount	decimal(11, 3)	NOT NULL	Tổng tiền khách phải trả
9	status	nvarchar(50)	NOT NULL	Trạng thái hóa đơn
10	note	nvarchar(MAX)		Ghi chú hóa đơn (khi phát sinh vấn đề)
11	isDeleted	bit	NOT NULL, DEFAULT = 0	Trạng thái xóa, mặc định là false

Bảng 8: Bảng Order

– **Bảng OrderItem:**

OrderItem				
ID	Attribute	Type	Constraint	Note
1	orderId	uniqueidentifier	PK, FK	Mã hóa đơn
2	bookId	uniqueidentifier		Mã sách
3	quantity	smallint	NOT NULL, > 0	Số lượng
4	price	decimal(8, 0)	NOT NULL, > 1000	Giá (ở thời điểm bán)
5	isDeleted	bit	NOT NULL, DEFAULT = 0	Trạng thái xóa, mặc định là false

Bảng 9: Bảng OrderItem

CHƯƠNG 3: XÂY DỰNG VÀ PHÁT TRIỂN HỆ THỐNG

1. Xây dựng các API quan trọng:

- API đăng ký tài khoản nhân viên:

```
public async Task<bool> RegisterAsync(RegisterDTO registerDTO)
{
    var existingByEmail = await
_authRepository.GetByEmailAsync(registerDTO.Email);
    var existingByPhone = await
_authRepository.GetByPhoneAsync(registerDTO.Phone);
    var existingByCitizenIdentification = await
_authRepository.GetByCitizenIdentificationAsync(registerDTO.CitizenIdentification);

    if (existingByEmail != null)
        throw new InvalidOperationException("Email đã được sử dụng.");

    if (existingByPhone != null)
        throw new InvalidOperationException("Số điện thoại đã được sử
dụng.");

    if (existingByCitizenIdentification != null)
        throw new InvalidOperationException("CCCD đã được sử dụng.");

    if (registerDTO.Password != registerDTO.ConfirmPassword)
        throw new ArgumentException("Mật khẩu xác nhận không khớp với mật
khẩu.");

    if (!IsOver18(registerDTO.DateOfBirth))
        throw new ArgumentException("Bạn phải đủ 18 tuổi trở lên.");

    string hashedPassword =
BCrypt.Net.BCrypt.HashPassword(registerDTO.Password);

    var staff = new Staff
    {
        FamilyName = registerDTO.FamilyName,
        GivenName = registerDTO.GivenName,
        DateOfBirth = registerDTO.DateOfBirth,
        Address = registerDTO.Address,
        Phone = registerDTO.Phone,
        Email = registerDTO.Email,
        CitizenIdentification = registerDTO.CitizenIdentification,
        HashPassword = hashedPassword,
        Gender = registerDTO.Gender,
        Role = false,
        IsActivated = false
    };

    await _authRepository.AddSync(staff);

    var saved = await _authRepository.SaveChangesAsync();

    if (saved == false)
        return false;

    var token = GenerateActivationToken(staff.Id);
```

```

        var activationLink =
            $"http://localhost:5208/api/auth/activate?token={token}";

        await _emailService.SendEmailAsync(staff.Email, "Kích hoạt tài khoản",
            $"Nhấn vào liên kết sau để kích hoạt tài khoản: <a href='{activationLink}'>Xác minh tài khoản</a>");

        Console.WriteLine($"Gửi email xác nhận đến: {staff.Email} thành công.");

        return true;
    }

    public async Task<bool> ActivateAccountAsync(string token)
    {
        var jwtKey = Environment.GetEnvironmentVariable("JWT_SECRET_KEY") ??
            _config["Jwt:Key"];

        var handler = new JwtSecurityTokenHandler();

        var tokenValidationParameters = new TokenValidationParameters
        {
            ValidateIssuerSigningKey = true,
            IssuerSigningKey = new
                SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtKey)),
            ValidateIssuer = false,
            ValidateAudience = false,
            ValidateLifetime = true
        };

        var principal = handler.ValidateToken(token, tokenValidationParameters,
            out var validatedToken);
        var staffIdClaim = principal.FindFirst("staffId");
        if (staffIdClaim == null) return false;

        var staffId = Guid.Parse(staffIdClaim.Value);
        var staff = await _authRepository.GetByIdAsync(staffId);
        if (staff == null) return false;

        if (staff.IsActivated) return false;

        staff.IsActivated = true;
        return await _authRepository.SaveChangesAsync();
    }

    private string GenerateActivationToken(Guid staffId)
    {
        var jwtKey = Environment.GetEnvironmentVariable("JWT_SECRET_KEY")
            ?? _config["Jwt:Key"]
            ?? throw new Exception("JWT key is missing");

        var jwtIssuer = _config["Jwt:Issuer"];
        var jwtAudience = _config["Jwt:Audience"];
        var expireMinutes = int.Parse(_config["Jwt:ExpireMinutes"] ?? "60");

        var securityKey = new
            SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtKey));
        var credentials = new SigningCredentials(securityKey,
            SecurityAlgorithms.HmacSha256);
    }

```

```

        var claims = new[] { new Claim("staffId", staffId.ToString()) };

        var token = new JwtSecurityToken(
            issuer: jwtIssuer,
            audience: jwtAudience,
            claims: claims,
            expires: DateTime.UtcNow.AddMinutes(expireMinutes),
            signingCredentials: credentials
        );

        return new JwtSecurityTokenHandler().WriteToken(token);
    }
}

```

– API quản lý sách:

```

public async Task<bool> AddAsync(BookCreateDTO bookCreateDTO)
{
    if (string.IsNullOrWhiteSpace(bookCreateDTO.Isbn))
        throw new ArgumentException("Mã ISBN là bắt buộc.");

    if (string.IsNullOrWhiteSpace(bookCreateDTO.Title))
        throw new ArgumentException("Tiêu đề là bắt buộc.");

    await ValidateForeignKeysAsync(bookCreateDTO);

    if (bookCreateDTO.YearOfPublication <= 1500)
        throw new ArgumentException("Năm xuất bản phải lớn hơn 1500.");

    if (bookCreateDTO.Price <= 1000)
        throw new ArgumentException("Giá phải lớn hơn 1000.");

    if (string.IsNullOrWhiteSpace(bookCreateDTO.Image))
        throw new ArgumentException("Hình ảnh không được để trống hoặc null.");

    if (bookCreateDTO.Quantity < 0)
        throw new ArgumentException("Số lượng phải lớn hơn hoặc bằng 0.");

    var existingBook = await _bookRepository.GetByIsbnAsync(bookCreateDTO.Isbn);
    if (existingBook != null)
        throw new InvalidOperationException("Đã có một cuốn sách có cùng mã
ISBN.");

    var book = new Book
    {
        Isbn = bookCreateDTO.Isbn,
        Title = bookCreateDTO.Title,
        CategoryId = bookCreateDTO.CategoryId,
        AuthorId = bookCreateDTO.AuthorId,
        PublisherId = bookCreateDTO.PublisherId,
        YearOfPublication = bookCreateDTO.YearOfPublication,
        Price = bookCreateDTO.Price,
        Image = bookCreateDTO.Image,
        Quantity = bookCreateDTO.Quantity
    };

    await _bookRepository.AddAsync(book);
    return await _bookRepository.SaveChangesAsync();
}

```

```

public async Task<bool> UpdateAsync(Guid id, BookUpdateDTO bookUpdateDTO)
{
    var category = await
_categoryRepository.GetByIdAsync(bookUpdateDTO.CategoryId);
    if (category == null || category.IsDeleted)
        throw new ArgumentException("Danh mục không hợp lệ hoặc đã bị xóa.");

    var author = await _authorRepository.GetByIdAsync(bookUpdateDTO.AuthorId);
    if (author == null || author.IsDeleted)
        throw new ArgumentException("Tác giả không hợp lệ hoặc đã bị xóa.");

    var publisher = await
_publisherRepository.GetByIdAsync(bookUpdateDTO.PublisherId);
    if (publisher == null || publisher.IsDeleted)
        throw new ArgumentException("Nhà xuất bản không hợp lệ hoặc đã bị xóa.");

    var existingBook = await _bookRepository.GetByIdAsync(id);
    if (existingBook == null)
        throw new KeyNotFoundException($"Không tìm thấy sách có id '{id}'.");

    if (bookUpdateDTO.YearOfPublication <= 1500)
        throw new ArgumentException("Năm xuất bản phải lớn hơn 1500.");

    if (bookUpdateDTO.Price <= 1000)
        throw new ArgumentException("Giá phải lớn hơn 1000.");

    if (string.IsNullOrEmpty(bookUpdateDTO.Image))
        throw new ArgumentException("Hình ảnh không được để trống hoặc null.");

    if (bookUpdateDTO.Quantity < 0)
        throw new ArgumentException("Số lượng phải lớn hơn hoặc bằng 0.");

    var duplicateBook = await _bookRepository.GetByIsbnAsync(bookUpdateDTO.Isbn);
    if (duplicateBook != null && duplicateBook.Id != id)
        throw new InvalidOperationException("Đã có một cuốn sách có cùng mã
ISBN.");

    existingBook.Isbn = bookUpdateDTO.Isbn;
    existingBook.Title = bookUpdateDTO.Title;
    existingBook.CategoryId = bookUpdateDTO.CategoryId;
    existingBook.AuthorId = bookUpdateDTO.AuthorId;
    existingBook.PublisherId = bookUpdateDTO.PublisherId;
    existingBook.YearOfPublication = bookUpdateDTO.YearOfPublication;
    existingBook.Price = bookUpdateDTO.Price;
    existingBook.Image = bookUpdateDTO.Image;
    existingBook.Quantity = bookUpdateDTO.Quantity;
    existingBook.IsDeleted = bookUpdateDTO.IsDeleted;

    _bookRepository.Update(existingBook);
    return await _bookRepository.SaveChangesAsync();
}

public async Task<bool> DeleteAsync(Guid id)
{
    var existingBook = await _bookRepository.GetByIdAsync(id);

    if (existingBook == null)
        throw new KeyNotFoundException($"Không tìm thấy sách có id '{id}'.");
}

```

```

        if (existingBook.IsDeleted == true)
            throw new InvalidOperationException($"Sách có id {id} đã bị xóa.");

        _bookRepository.Delete(existingBook);

        return await _bookRepository.SaveChangesAsync();
    }

    private async Task ValidateForeignKeysAsync(BookCreateDTO dto)
    {
        var category = await _categoryRepository.GetByIdAsync(dto.CategoryId);
        if (category == null || category.IsDeleted)
            throw new ArgumentException("Danh mục không hợp lệ hoặc đã bị xóa.");

        var author = await _authorRepository.GetByIdAsync(dto.AuthorId);
        if (author == null || author.IsDeleted)
            throw new ArgumentException("Tác giả không hợp lệ hoặc đã bị xóa.");

        var publisher = await _publisherRepository.GetByIdAsync(dto.PublisherId);
        if (publisher == null || publisher.IsDeleted)
            throw new ArgumentException("Nhà xuất bản không hợp lệ hoặc đã bị xóa.");
    }

```

- API đăng nhập nhân viên:

```

public async Task<String?> LoginAsync(LoginDTO loginDTO)
{
    if (string.IsNullOrEmpty(loginDTO.Email))
        throw new ArgumentException("Vui lòng nhập email và mật khẩu để đăng nhập.");

    var staff = await _authRepository.GetByEmailAsync(loginDTO.Email);

    if (staff == null)
        throw new UnauthorizedAccessException("Tài khoản không hợp lệ, vui lòng kiểm tra email hoặc tạo tài khoản mới.");

    if (staff.IsDeleted)
        throw new UnauthorizedAccessException("Tài khoản của bạn đã bị xóa. Vui lòng liên hệ quản trị viên.");

    if (!staff.IsActive)
        throw new UnauthorizedAccessException("Tài khoản của bạn chưa được kích hoạt. Vui lòng kiểm tra email để kích hoạt.");

    bool isValidPassword = BCrypt.Net.BCrypt.Verify(loginDTO.Password, staff.HashPassword);

    if (!isValidPassword)
        throw new UnauthorizedAccessException("Mật khẩu không đúng. Vui lòng thử lại.");

    var token = GenerateJwtToken(staff);

    return token;
}

private string GenerateJwtToken(Staff user)

```

```

{
    var jwtKey = Environment.GetEnvironmentVariable("JWT_SECRET_KEY") ??
    _config["Jwt:Key"];
    var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtKey));
    var credentials = new SigningCredentials(securityKey,
    SecurityAlgorithms.HmacSha256);

    var claims = new[]
    {
        new Claim(JwtRegisteredClaimNames.Sub, user.Email),
        new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString()),
        new Claim("staffId", user.Id.ToString()),
        new Claim("email", user.Email),
        new Claim(ClaimTypes.Role, user.Role ? "Admin" : "Staff")
    };

    var token = new JwtSecurityToken(
        issuer: _config["Jwt:Issuer"],
        audience: _config["Jwt:Audience"],
        claims: claims,
        expires: DateTime.UtcNow.AddMinutes(int.Parse(_config["Jwt:ExpireMinutes"]
        ?? "60")),
        signingCredentials: credentials
    );

    return new JwtSecurityTokenHandler().WriteToken(token);
}

```

– API bán hàng tại cửa hàng:

```

public async Task<bool> AddAsync(OrderCreateDTO orderCreateDTO, ClaimsPrincipal
user)
{
    if (orderCreateDTO.Items == null || !orderCreateDTO.Items.Any())
        throw new ArgumentException("Đơn hàng phải có ít nhất một sản phẩm.");

    var customer = await
    _orderRepository.GetCustomerByIdAsync(orderCreateDTO.CustomerId);
    if (customer == null || customer.IsDeleted)
        throw new ArgumentException("Khách hàng không hợp lệ hoặc đã bị xóa.");

    var order = new Order
    {
        Id = Guid.NewGuid(),
        StaffId = CurrentUserHelper.GetStaffId(user),
        CustomerId = orderCreateDTO.CustomerId,
        PromotionId = orderCreateDTO.PromotionId,
        CreatedTime = DateTime.Now,
        Status = true
    };

    decimal subTotal = 0;

    foreach (var item in orderCreateDTO.Items)
    {
        var book = await _orderRepository.GetBookByIdAsync(item.BookId)
        ?? throw new KeyNotFoundException($"Không tìm thấy sách với ID
        '{item.BookId}'.");
    }
}

```

```

        if (book.Quantity < item.Quantity)
            throw new InvalidOperationException($"Số lượng sách '{book.Title}'
không đủ để bán.");

        if (book.IsDeleted)
            throw new InvalidOperationException($"Sách '{book.Title}' đã bị xóa.");

        var orderItem = new OrderItem
        {
            OrderId = order.Id,
            BookId = item.BookId,
            Quantity = item.Quantity,
            Price = book.Price
        };

        book.Quantity -= item.Quantity;

        subTotal += (orderItem.Price * orderItem.Quantity);
        order.OrderItems.Add(orderItem);
    }

    order.SubTotalAmount = subTotal;

    if (order.PromotionId.HasValue)
    {
        var promotion = await
_orderRepository.GetPromotionByIdAsync(order.PromotionId.Value)
            ?? throw new ArgumentException("Không tìm thấy khuyến mãi.");

        if (promotion.IsDeleted)
            throw new InvalidOperationException($"Khuyến mãi '{promotion.Name}' đã
bị xóa.");

        if (DateTime.Now < promotion.StartDate)
            throw new InvalidOperationException("Khuyến mãi chưa bắt đầu.");

        if (DateTime.Now > promotion.EndDate)
            throw new InvalidOperationException("Khuyến mãi đã hết hạn.");

        if (promotion.Quantity <= 0)
            throw new InvalidOperationException("Khuyến mãi đã hết lượt sử dụng.");

        if (subTotal < promotion.Condition)
            throw new InvalidOperationException("Đơn hàng không đủ điều kiện áp
dụng khuyến mãi.");

        order.PromotionAmount = subTotal * promotion.DiscountPercent;
        promotion.Quantity--;
    }

    order.TotalAmount = subTotal - order.PromotionAmount;

    await _orderRepository.AddAsync(order);

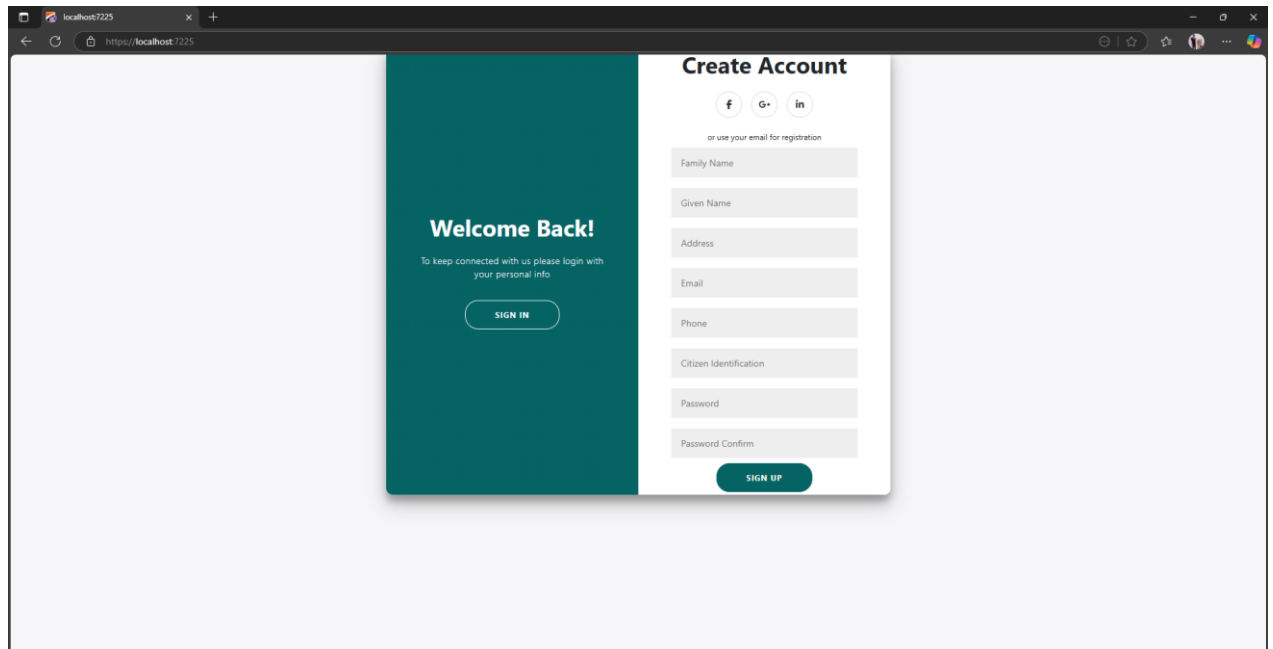
    return await _orderRepository.SaveChangesAsync();
}

```


2. Giao diện:

2.1. Giao diện dành cho nhân viên:

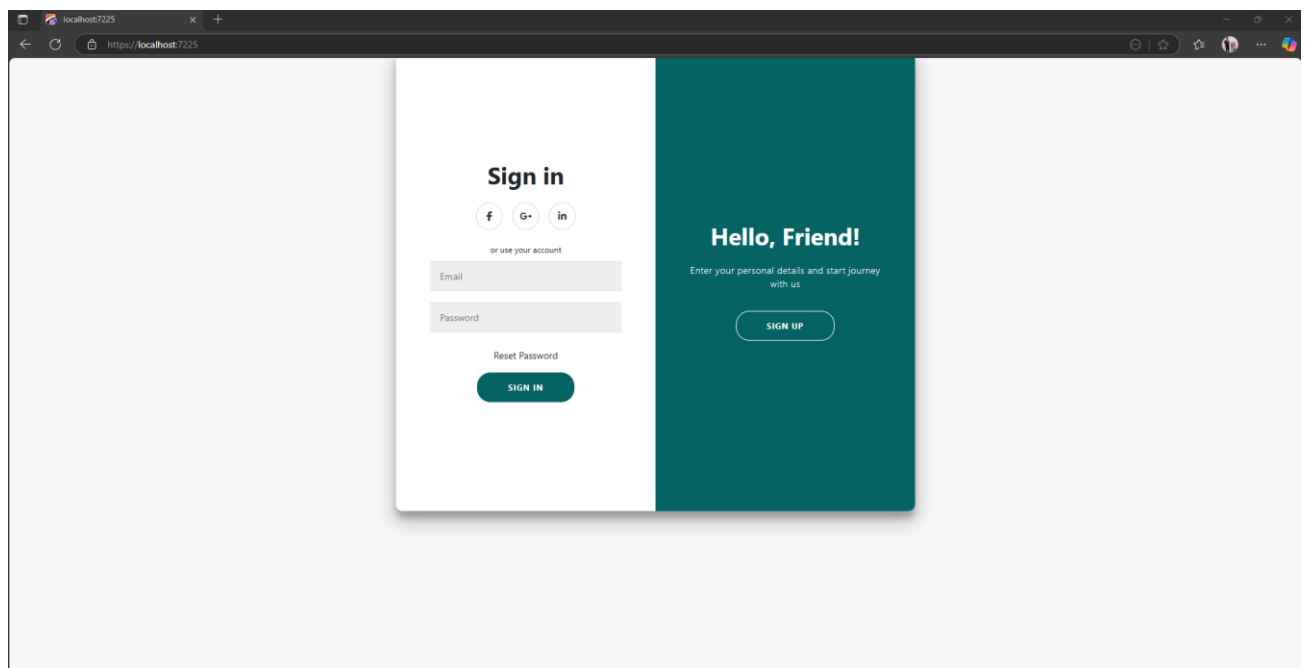
– Đăng ký tài khoản:



The screenshot shows a web browser window with the URL `https://localhost:7225`. The page features a dark teal background on the left with the text "Welcome Back!" and a "SIGN IN" button. On the right, there is a white "Create Account" form. The form includes social media login options (Facebook, Google, LinkedIn) and a section for email registration. The registration fields are: Family Name, Given Name, Address, Email, Phone, Citizen Identification, Password, and Password Confirm. A "SIGN UP" button is located at the bottom right of the form.

Hình 4: Giao diện đăng ký tài khoản nhân viên

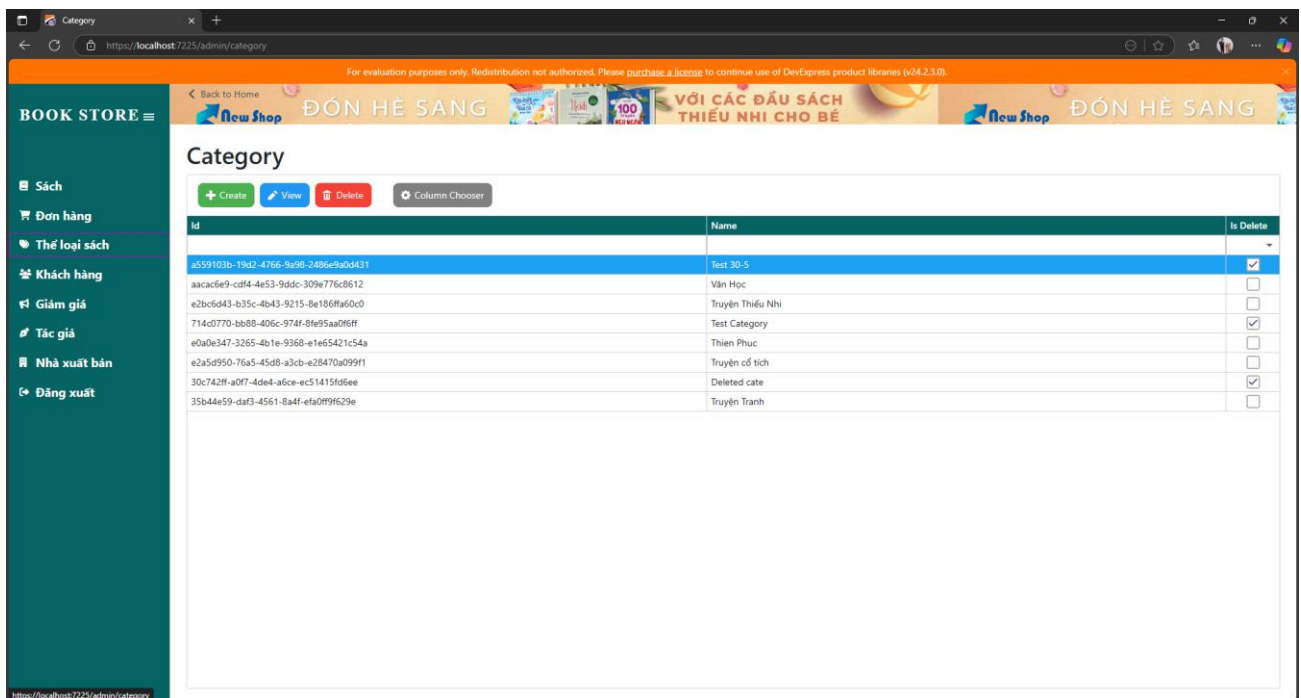
– Đăng nhập:



The screenshot shows a web browser window with the URL `https://localhost:7225`. The page features a dark teal background on the right with the text "Hello, Friend!" and a "SIGN UP" button. On the left, there is a white "Sign in" form. The form includes social media login options (Facebook, Google, LinkedIn) and a section for account login. The login fields are: Email and Password. There is also a "Reset Password" link and a "SIGN IN" button at the bottom left of the form.

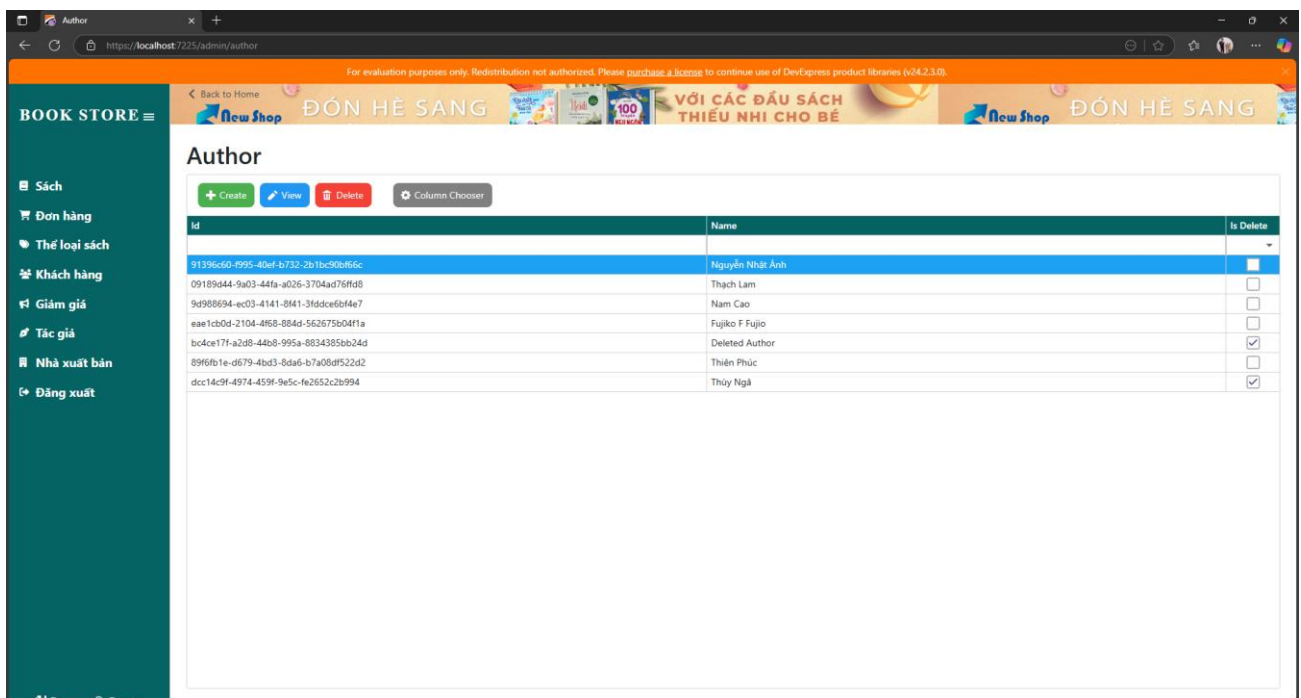
Hình 5: Giao diện đăng nhập nhân viên

– Quản lý thể loại:



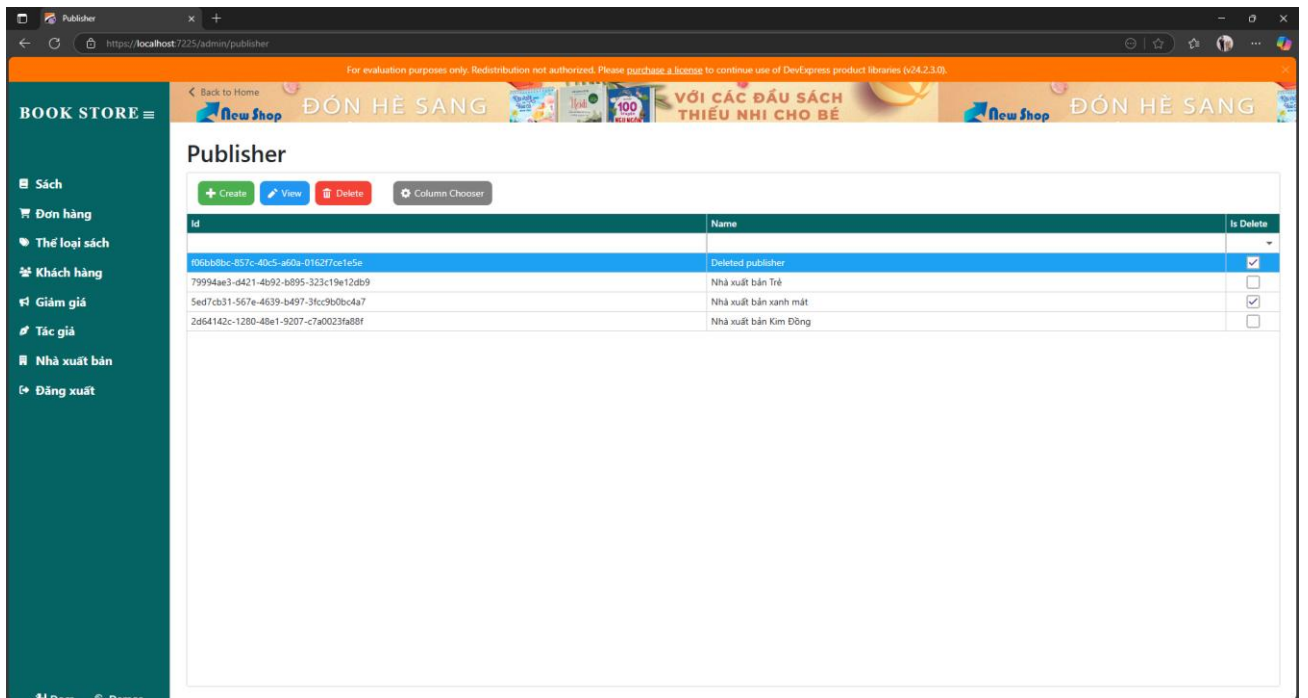
Hình 6: Giao diện quản lý thể loại

– Quản lý tác giả:



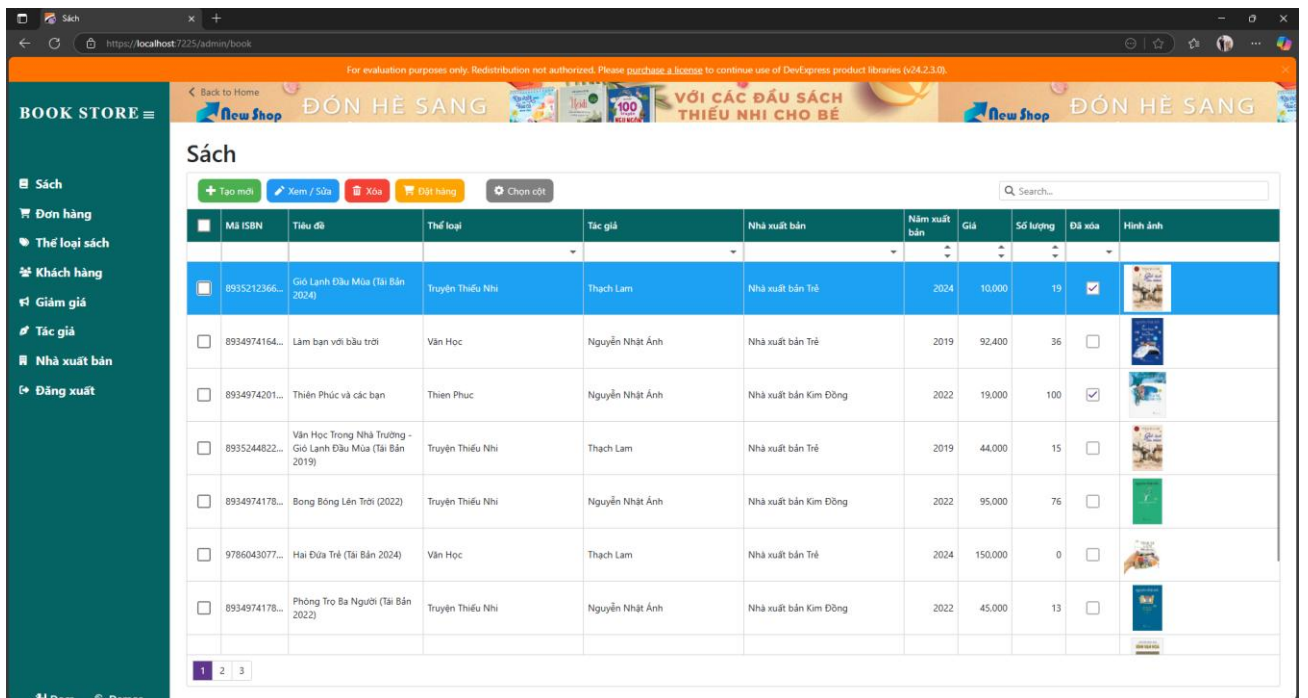
Hình 7: Giao diện quản lý tác giả

– Quản lý nhà xuất bản:



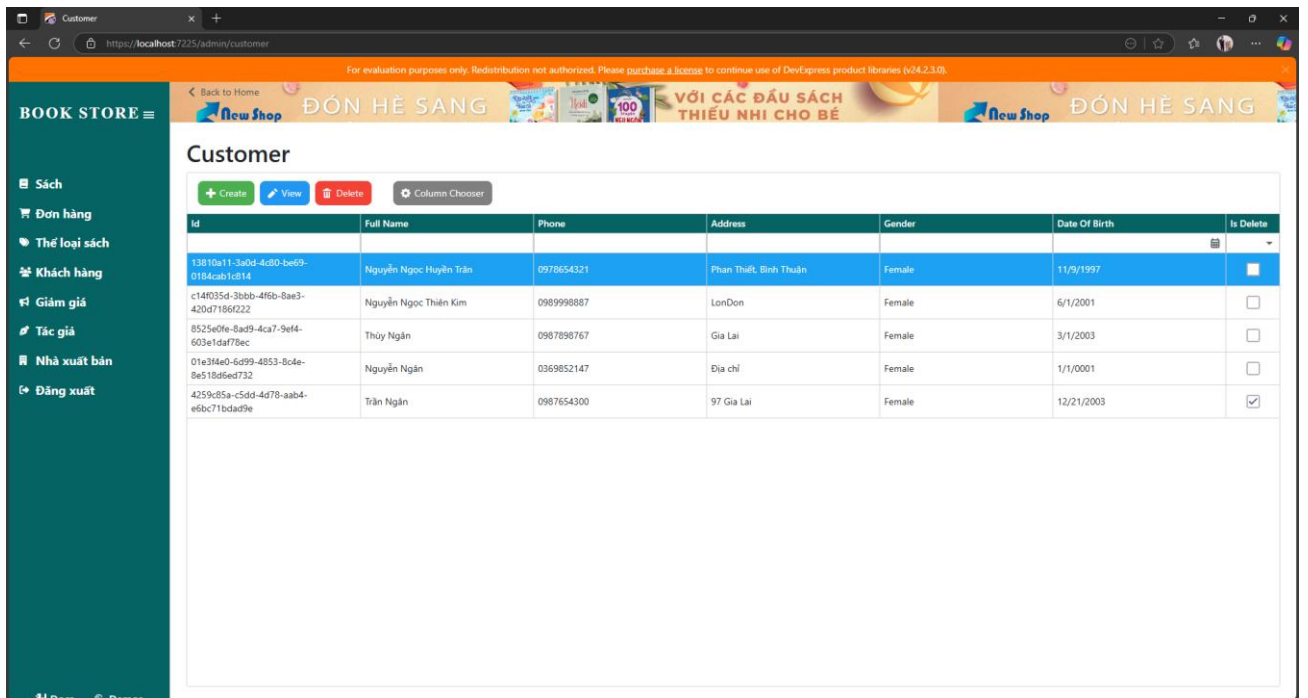
Hình 8: Giao diện quản lý nhà xuất bản

– Quản lý sách:



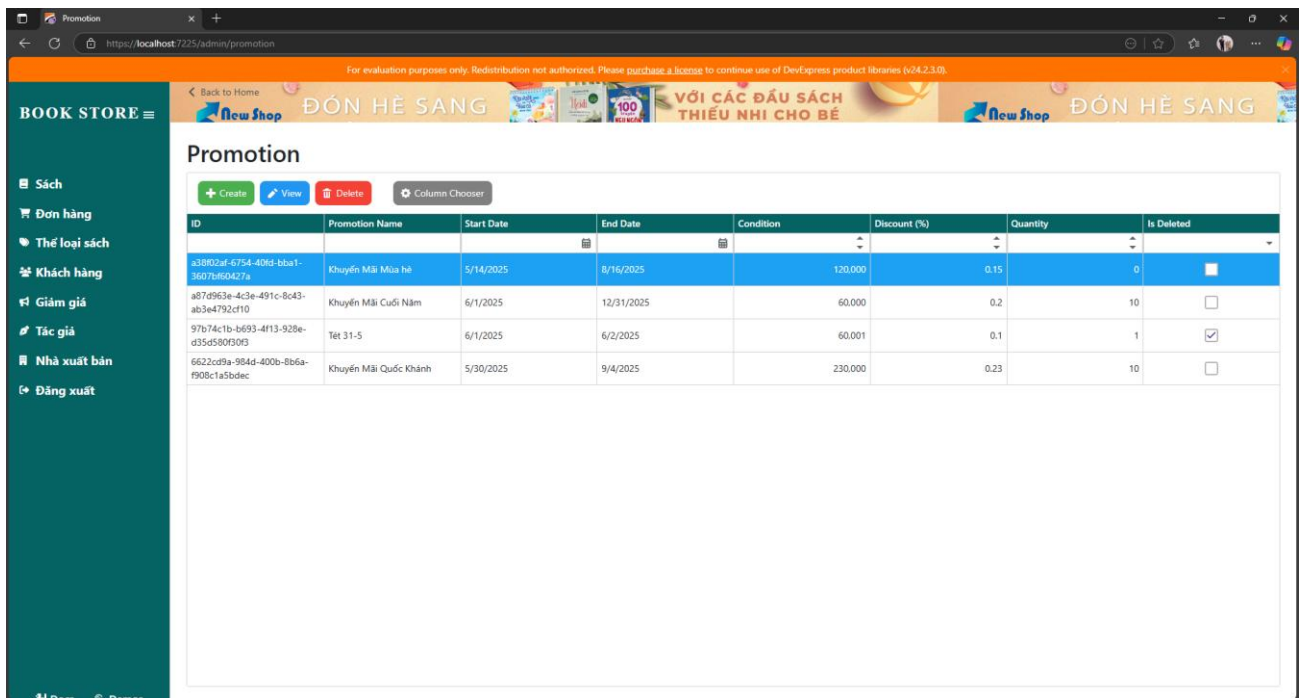
Hình 9: Giao diện quản lý sách

- Quản lý khách hàng:



Hình 10: Giao diện quản lý khách hàng

- Quản lý khuyến mãi:



Hình 11: Giao diện quản lý khuyến mãi

- Quản lý hóa đơn:

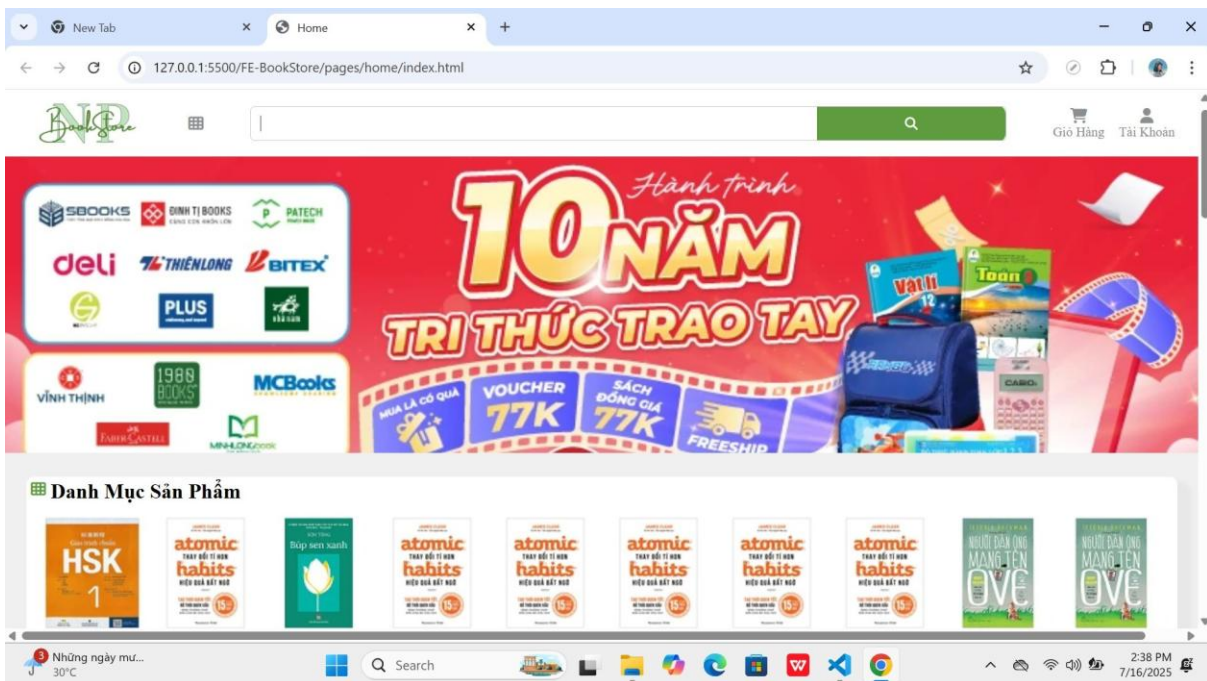
Order

Staff	Customer	Promotion	Created Time	Sub-Total	Promotion Amount	Total	Status	Note	Is Deleted
Nguyễn Ngọc Thiến Phúc	Thủy Ngân		6/20/2025	110.700	0	110.700	Completed		<input checked="" type="checkbox"/>
Nguyễn Ngọc Thiến Phúc	Thủy Ngân		6/20/2025	110.700	0	110.700	Completed		<input type="checkbox"/>
Nguyễn Ngọc Thiến Phúc	Nguyễn Ngọc Huyền Trân		6/1/2025	44.000	0	44.000	Completed		<input type="checkbox"/>
Thiến Phúc 2	Nguyễn Ngọc Huyền Trân		6/1/2025	92.400	0	92.400	Completed		<input type="checkbox"/>
Nguyễn Ngọc Thiến Phúc	Nguyễn Ngọc Huyền Trân	Khuyến Mãi Mùa hè	5/29/2025	92.400	0	92.400	Issue		<input type="checkbox"/>
Nguyễn Ngọc Thiến Phúc	Trần Ngân		5/14/2025	200.000	30.000	170.000	Completed	Sách bị rách	<input type="checkbox"/>
Thiến Phúc 2	Nguyễn Ngọc Huyền Trân	Khuyến Mãi Mùa hè	6/1/2025	150.000	22.500	127.500	Completed		<input type="checkbox"/>
Nguyễn Ngọc Thiến Phúc	Nguyễn Ngọc Huyền Trân	Khuyến Mãi Mùa hè	6/1/2025	722.100	108.315	613.785	Completed		<input type="checkbox"/>
PTIT Nguyễn Ngọc Thiến Phúc	Nguyễn Ngọc Huyền Trân	Khuyến Mãi Mùa hè	6/1/2025	176.000	26.400	149.600	Completed		<input type="checkbox"/>
Nguyễn Ngọc Thiến Phúc	Nguyễn Ngọc Huyền Trân	Khuyến Mãi Cuối Năm	5/31/2025	238.000	0	238.000	Completed		<input type="checkbox"/>

Hình 12: Giao diện quản lý hóa đơn

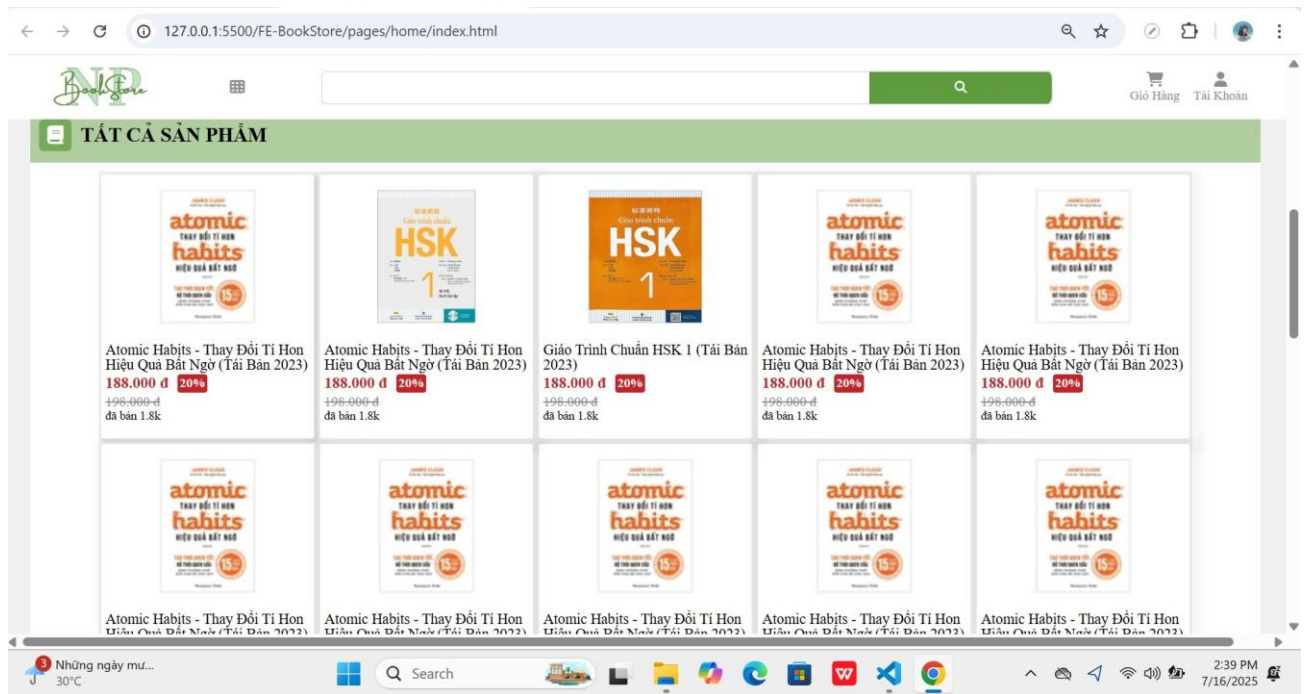
2.2. Giao diện dành cho khách hàng:

- Trang chủ:



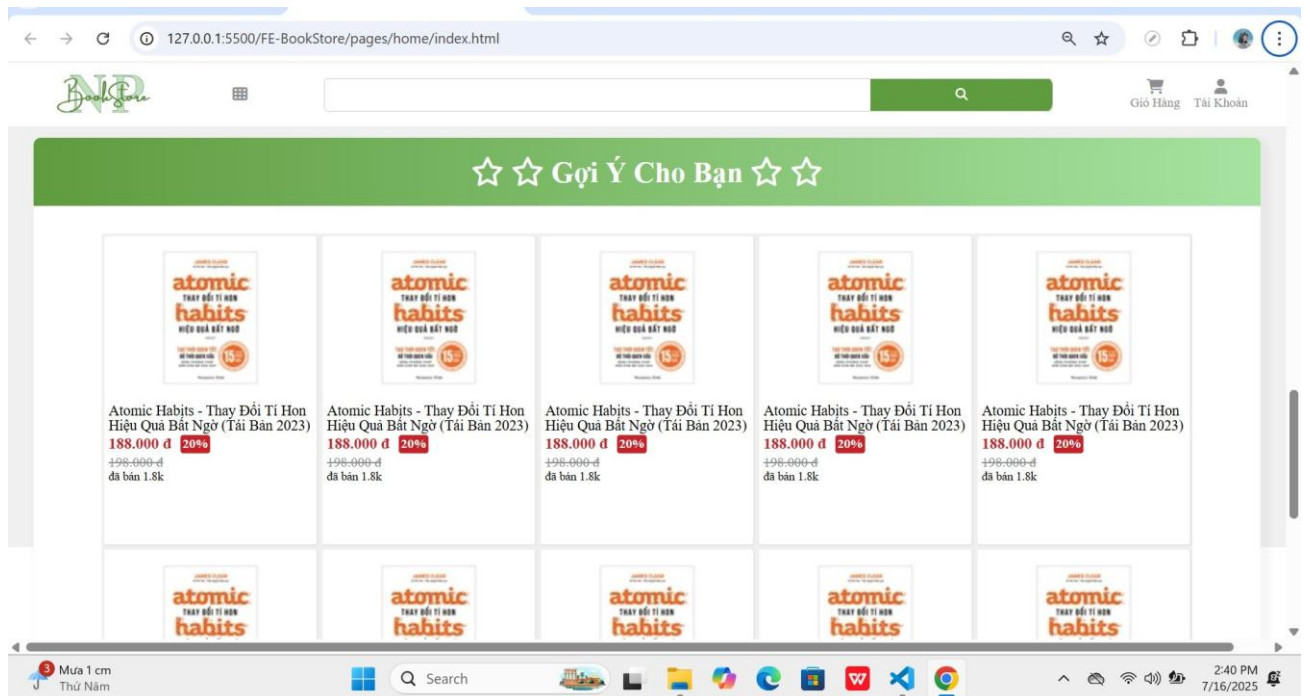
Hình 13: Giao diện trang chủ

– Danh sách sách:



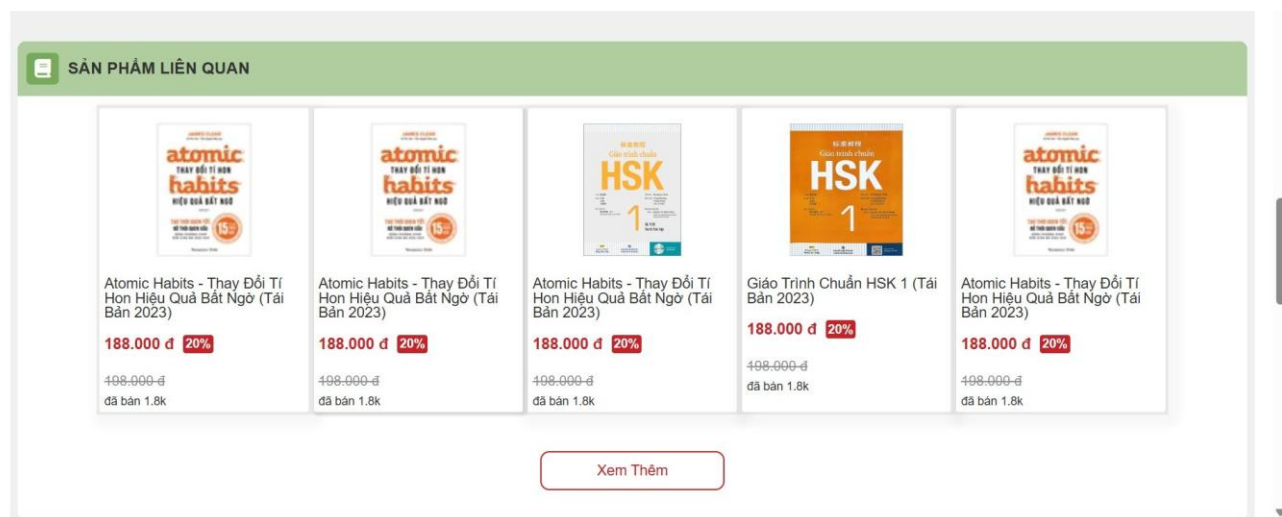
Hình 14: Giao diện danh sách sách

– Gợi ý:



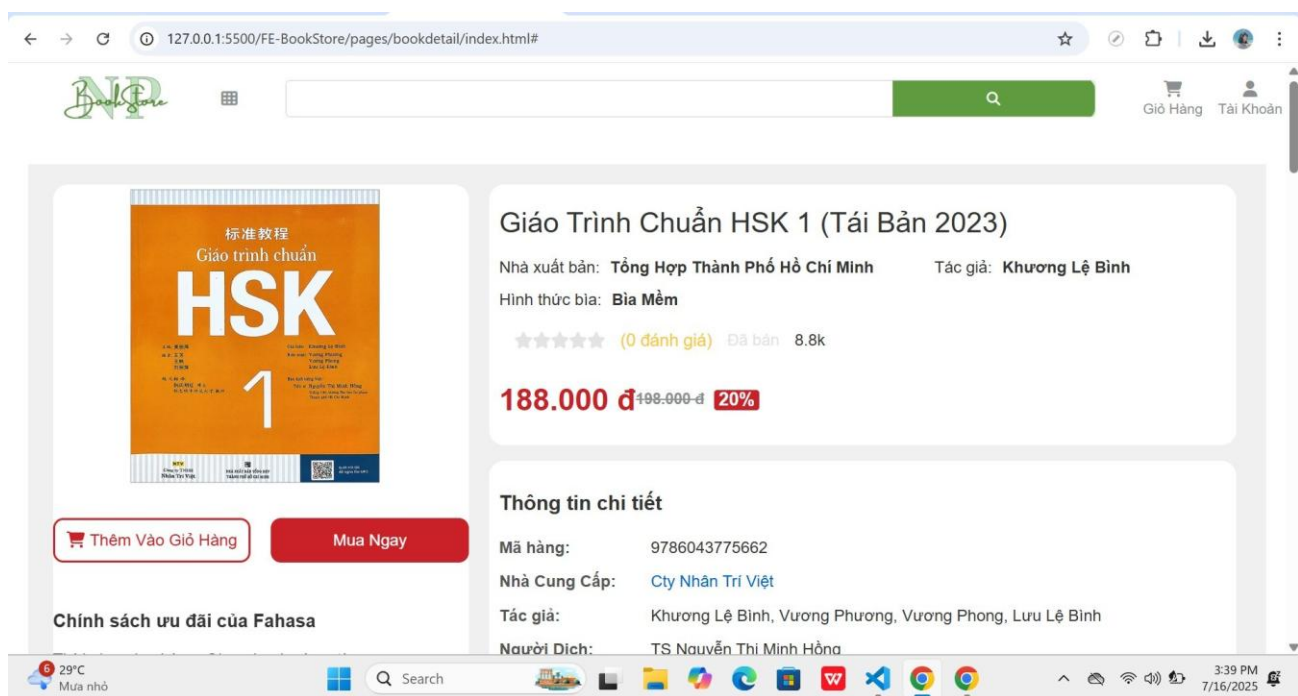
Hình 15: Giao diện gợi ý sách

– Sản phẩm liên quan:



Hình 16: Giao diện sách liên quan

– Chi tiết sách:



Hình 17: Giao diện chi tiết sách

KẾT LUẬN

1. Nhận xét về điểm mạnh, điểm yếu trong cách tiến hành đồ án:

– Điểm mạnh:

- + **Thiết kế kiến trúc rõ ràng:** Ứng dụng được phát triển theo mô hình tách lớp (Controller - Service - Repository), giúp dễ bảo trì và mở rộng.
- + **Sử dụng công nghệ hiện đại:** Kết hợp ASP.NET Core cho backend và ReactJS cho frontend giúp tăng hiệu suất phát triển, dễ tương tác và hỗ trợ nhiều tính năng UI nâng cao.
- + **Tổ chức mã nguồn khoa học:** Tách biệt rõ các chức năng như Models, DTOs, Configurations, Middlewares, giúp project dễ đọc và dễ quản lý.

– Điểm yếu:

- + **Triển khai còn hạn chế:** Dự án chưa tích hợp CI/CD hoặc chưa triển khai thực tế trên môi trường cloud/public test.

2. Khả năng cải tiến cách tiến hành đồ án:

- **Triển khai môi trường staging:** Tích hợp Docker + GitHub Actions hoặc sử dụng Azure/Render để triển khai ứng dụng trên môi trường test public cho người dùng nội bộ dùng thử.
- **Mở rộng thêm chức năng quản trị:** Bổ sung chức năng báo cáo doanh thu, thống kê tồn kho, giúp sát với yêu cầu thực tế.
- **Tích hợp mã vạch và in hóa đơn:** Để ứng dụng thực sự sử dụng được tại cửa hàng, nên kết hợp máy quét barcode, máy in hóa đơn, v.v.
- **Phát triển thêm ứng dụng di động:** Có thể phát triển app Flutter hoặc .NET MAUI để nhân viên có thể thao tác trên điện thoại/tablet.

TÀI LIỆU THAM KHẢO

- [1] C. L. V. Tiến, "vietnix," [Online]. Available: <https://vietnix.vn/tim-hieu-mo-hinh-mvc-la-gi/>. [Accessed 16 06 2025].
- [2] TopDev, "TopDev," [Online]. Available: <https://topdev.vn/blog/restful-api-la-gi/>. [Accessed 16 06 2025].
- [3] TopDev, "TopDev," [Online]. Available: <https://topdev.vn/blog/aspnet-la-gi/>. [Accessed 16 6 2025].
- [4] D. K. Toan, "Viblo," [Online]. Available: <https://viblo.asia/p/gioi-thieu-ve-reactjs-phan-i-cac-khai-niem-co-ban-V3m5WzjblO7>. [Accessed 17 07 2025].