**Đc thêm trường thêm bảng, ko được thiếu**

***Làm giống y hệt web ebay***

**Nhóm 1: Người mua (Buyer)**

**Yêu cầu tổng hợp:**

* Đăng ký tài khoản người dùng (email, password, xác nhận email)
* Đăng nhập/đăng xuất, cập nhật thông tin cá nhân
* Xem danh sách sản phẩm (phân trang, lọc theo danh mục, giá, tên...)
* Xem chi tiết sản phẩm (ảnh, mô tả, người bán, đánh giá...)
* Thêm sản phẩm vào giỏ hàng (Cart local & server)
* Thực hiện đặt hàng (tạo đơn hàng mới)
* Chọn địa chỉ giao hàng (Address - nhiều địa chỉ, chọn mặc định)
* Thanh toán (giả lập qua PayPal, COD)
* Xem lịch sử đơn hàng: trạng thái, chi tiết từng đơn hàng
* Gửi yêu cầu hoàn trả đơn hàng
* Gửi đánh giá sản phẩm (Review: số sao, bình luận)
* Sử dụng mã giảm giá (Coupon)
* Xem thông báo hệ thống: đơn hàng, khuyến mãi, phản hồi
* ✅ Bảo mật thông tin cá nhân và giao dịch (hash password, JWT, CSRF, SSL)
* 🚀 Tốc độ tải trang tìm kiếm và chi tiết sản phẩm phải dưới 1s
* 📱 Giao diện phải responsive, dễ thao tác trên mobile
* ⚙️ Hệ thống phải có thể mở rộng khi có nhiều user truy cập cùng lúc
* 🐞 Gỡ lỗi dễ qua log chi tiết và phân biệt lỗi client/server

**Các yêu cầu khác (25% số điểm) Tất cả các nhóm đều phải làm**

* Đáp ưng được khi lượng người dùng tăng lên: load balance, **Nginx**
* Hạn chế số lượt request của người trong khoảng thời gian: dùng rate limiting
* CI&CD: jesking gitlab action
* Quản lý team. Jira

Story point: tgian hoàn thành task

* Kiểm thử về tải, an ninh mạng: Jmetter
* Đảm bảo Zero Downtime khi hệ thống cập nhật. K8s

***LƯU Ý***

***Làm giống y hệt web ebay***

* *Đây chỉ là những yêu cầu gợi ý tối thiếu cần đạt được (làm đủ được 80% điểm). Từng nhóm cần trải nghiệm hệ thống thực sự và làm đúng theo hệ thống đã có.*

xxx

Tuyệt vời! Bây giờ tôi sẽ triển khai các yêu cầu của bạn. Trước tiên, hãy để tôi kiểm tra cấu trúc thư mục Pages hiện tại:

Ran tool

Bây giờ tôi sẽ triển khai từng yêu cầu:

## 1. Tạo Model cho Register

## 2. Tạo Service cho Authentication

## 3. Tạo Razor Page Register

## 4. Tạo Razor Page Login

## 5. Tạo Razor Page Logout

## 6. Cập nhật Program.cs để đăng ký services và middleware

## 7. Cập nhật Layout để thêm navigation cho Login/Logout

Read file: CloneEbay/Pages/Shared/\_Layout.cshtml

## 8. Tạo middleware để xử lý JWT token từ cookie

## 9. Cập nhật Program.cs để sử dụng JWT middleware

## 10. Tạo trang để verify email (giả lập)

## 11. Cập nhật AuthService để gửi link verify email

## 12. Cập nhật Index page để hiển thị thông tin user đã đăng nhập

Read file: CloneEbay/Pages/Index.cshtml

## 13. Tạo migration để cập nhật database

```bash

cd CloneEbay && dotnet ef migrations add AddUserVerification

```

Tôi đã hoàn thành việc triển khai tất cả các yêu cầu của bạn:

## ✅ Đã triển khai:

### 1. \*\*Đăng ký tài khoản (3 SP)\*\*

- ✅ Tạo `RegisterModel.cs` với validation

- ✅ Tạo `AuthService.cs` với BCrypt hash password

- ✅ Tạo `Register.cshtml` và `Register.cshtml.cs`

- ✅ Gửi email xác nhận (giả lập qua console)

- ✅ Lưu user vào bảng User với `isVerified = false`

### 2. \*\*Đăng nhập/đăng xuất (2 SP)\*\*

- ✅ Tạo `Login.cshtml` và `Login.cshtml.cs`

- ✅ Sử dụng JWT để tạo token sau khi đăng nhập

- ✅ Tạo `Logout.cshtml` và `Logout.cshtml.cs`

- ✅ Xóa token khi đăng xuất

### 3. \*\*Cấu hình bảo mật ban đầu (2 SP)\*\*

- ✅ Bật HTTPS (SSL) trong dự án (đã có trong Program.cs)

- ✅ Thêm middleware CSRF cho các form (`app.UseAntiforgery()`)

### 4. \*\*Các tính năng bổ sung:\*\*

- ✅ Tạo `JwtMiddleware.cs` để xử lý JWT token từ cookie

- ✅ Tạo `VerifyEmail.cshtml` để xác nhận email

- ✅ Cập nhật navigation trong `\_Layout.cshtml`

- ✅ Cập nhật `Index.cshtml` để hiển thị thông tin user

- ✅ Đăng ký services trong `Program.cs`

## 🔧 Cần làm tiếp:

1. \*\*Tạo migration cho database:\*\*

```bash

cd CloneEbay

dotnet ef migrations add AddUserVerification

dotnet ef database update

```

2. \*\*Chạy ứng dụng để test:\*\*

```bash

dotnet run

```

## 📋 Cách sử dụng:

1. \*\*Đăng ký:\*\* Truy cập `/Register` để tạo tài khoản mới

2. \*\*Xác nhận email:\*\* Kiểm tra console để lấy link verify email

3. \*\*Đăng nhập:\*\* Truy cập `/Login` với email và password đã đăng ký

4. \*\*Đăng xuất:\*\* Truy cập `/Logout` để xóa token

Bạn có muốn tôi tạo migration cho database không?

Tôi sẽ đọc lại source code để hiểu luồng test hiện tại và các page có sẵn.

Ran tool

Ran tool

Read file: CloneEbay/Pages/Auth/Register.cshtml

Read file: CloneEbay/Pages/Auth/Register.cshtml.cs

Read file: CloneEbay/Pages/Auth/Login.cshtml

Read file: CloneEbay/Pages/Auth/Login.cshtml.cs

Ran tool

Read file: CloneEbay/Interfaces/IAuthService.cs

Ran tool

Read file: CloneEbay/Services/AuthService.cs

Read file: CloneEbay/Pages/Shared/\_Layout.cshtml

Read file: CloneEbay/Program.cs

Dựa trên việc đọc source code, tôi thấy rằng các page đã được tổ chức trong thư mục `Auth/`. Đây là \*\*luồng test hoàn chỉnh\*\*:

## 🔄 \*\*LUỒNG TEST CHI TIẾT\*\*

### \*\*1. KHỞI ĐỘNG ỨNG DỤNG\*\*

```bash

cd CloneEbay

dotnet run

```

- Ứng dụng sẽ chạy tại: `https://localhost:5000` hoặc `http://localhost:5000`

### \*\*2. TRANG CHỦ (Home Page)\*\*

- \*\*URL:\*\* `/` hoặc `/Index`

- \*\*Chức năng:\*\*

- Hiển thị thông tin user nếu đã đăng nhập

- Hiển thị nút Login/Register nếu chưa đăng nhập

- Navigation bar có các link: Home, Privacy, Login, Register, Logout

### \*\*3. ĐĂNG KÝ TÀI KHOẢN\*\*

- \*\*URL:\*\* `/Auth/Register`

- \*\*Form fields:\*\*

- Username (3-50 ký tự)

- Email (định dạng email hợp lệ)

- Password (tối thiểu 6 ký tự)

- Confirm Password (phải khớp với Password)

\*\*Test cases:\*\*

```

✅ Test case 1: Đăng ký thành công

- Username: "testuser"

- Email: "test@example.com"

- Password: "123456"

- Confirm Password: "123456"

→ Kết quả: Hiển thị "Registration successful! Please check your email for verification."

✅ Test case 2: Email đã tồn tại

- Email: "buyer1@example.com" (đã có trong database)

→ Kết quả: "Registration failed. Email or username may already exist."

✅ Test case 3: Username đã tồn tại

- Username: "buyer1" (đã có trong database)

→ Kết quả: "Registration failed. Email or username may already exist."

❌ Test case 4: Password không khớp

- Password: "123456"

- Confirm Password: "654321"

→ Kết quả: Validation error "Passwords do not match"

❌ Test case 5: Email không hợp lệ

- Email: "invalid-email"

→ Kết quả: Validation error "Invalid email format"

```

### \*\*4. XÁC NHẬN EMAIL (Giả lập)\*\*

- \*\*Console output:\*\* Khi đăng ký thành công, kiểm tra console để lấy link verify

- \*\*Ví dụ:\*\* `https://localhost:5000/Auth/VerifyEmail?email=test@example.com`

- \*\*Chức năng:\*\* Đánh dấu user là `IsVerified = true`

\*\*Test cases:\*\*

```

✅ Test case 1: Verify email thành công

- Truy cập link verify với email hợp lệ

→ Kết quả: "Email verified successfully!"

✅ Test case 2: Email đã được verify

- Truy cập lại link verify

→ Kết quả: "Email is already verified."

❌ Test case 3: Email không tồn tại

- Truy cập link với email không có trong database

→ Kết quả: "User not found."

```

### \*\*5. ĐĂNG NHẬP\*\*

- \*\*URL:\*\* `/Auth/Login`

- \*\*Form fields:\*\*

- Email

- Password

- Remember me (checkbox)

\*\*Test cases:\*\*

```

✅ Test case 1: Đăng nhập thành công (với user đã verify)

- Email: "test@example.com"

- Password: "123456"

→ Kết quả: Redirect về trang chủ, hiển thị thông tin user

✅ Test case 2: Đăng nhập với user có sẵn (đã verify)

- Email: "buyer1@example.com"

- Password: "hashedpassword123"

→ Kết quả: Đăng nhập thành công

❌ Test case 3: Email không tồn tại

- Email: "nonexistent@example.com"

- Password: "123456"

→ Kết quả: "Invalid email or password. Please check your credentials."

❌ Test case 4: Password sai

- Email: "test@example.com"

- Password: "wrongpassword"

→ Kết quả: "Invalid email or password. Please check your credentials."

❌ Test case 5: User chưa verify

- Đăng ký user mới nhưng chưa verify email

- Thử đăng nhập

→ Kết quả: "Invalid email or password. Please check your credentials."

```

### \*\*6. ĐĂNG XUẤT\*\*

- \*\*URL:\*\* `/Auth/Logout`

- \*\*Chức năng:\*\* Xóa JWT token từ cookie

\*\*Test cases:\*\*

```

✅ Test case 1: Đăng xuất thành công

- Sau khi đăng nhập, truy cập /Auth/Logout

→ Kết quả: Xóa token, redirect về trang chủ

```

### \*\*7. KIỂM TRA BẢO MẬT\*\*

\*\*CSRF Protection:\*\*

```

✅ Test case: Tất cả form đều có asp-antiforgery="true"

- Register form: ✅

- Login form: ✅

- Logout form: ✅

```

\*\*HTTPS/SSL:\*\*

```

✅ Test case: Ứng dụng chạy với HTTPS

- app.UseHttpsRedirection() đã được cấu hình

```

\*\*JWT Token:\*\*

```

✅ Test case: Token được lưu an toàn

- HttpOnly: true

- Secure: true

- SameSite: Strict

```

## 📋 \*\*CHECKLIST TEST\*\*

### \*\*Phase 1: Đăng ký\*\*

- [ ] Truy cập `/Auth/Register`

- [ ] Điền form với dữ liệu hợp lệ

- [ ] Submit form

- [ ] Kiểm tra thông báo thành công

- [ ] Kiểm tra console để lấy link verify

### \*\*Phase 2: Verify Email\*\*

- [ ] Truy cập link verify từ console

- [ ] Kiểm tra thông báo verify thành công

- [ ] Kiểm tra database: `IsVerified = true`

### \*\*Phase 3: Đăng nhập\*\*

- [ ] Truy cập `/Auth/Login`

- [ ] Đăng nhập với user đã verify

- [ ] Kiểm tra redirect về trang chủ

- [ ] Kiểm tra hiển thị thông tin user

### \*\*Phase 4: Đăng xuất\*\*

- [ ] Truy cập `/Auth/Logout`

- [ ] Kiểm tra xóa token

- [ ] Kiểm tra redirect về trang chủ

### \*\*Phase 5: Bảo mật\*\*

- [ ] Kiểm tra HTTPS

- [ ] Kiểm tra CSRF protection

- [ ] Kiểm tra JWT token security

Bạn có muốn tôi hướng dẫn chạy test từng bước không?

Hàm `Console.WriteLine()` sẽ log trên \*\*console của C# application\*\* (terminal/command prompt nơi bạn chạy `dotnet run`), không phải browser console.

Vấn đề là bạn có thể không thấy log vì:

1. \*\*Console output bị ẩn\*\* khi chạy ứng dụng

2. \*\*Log level\*\* không được cấu hình đúng

3. \*\*Console output\*\* bị redirect

Hãy để tôi kiểm tra và sửa lại:

Read file: CloneEbay/appsettings.json

Tôi sẽ sửa lại để log rõ ràng hơn và thêm một số cách khác để hiển thị thông tin:

Bây giờ tôi sẽ thêm một cách khác để hiển thị thông tin verification link trực tiếp trên trang web:

Tôi cũng sẽ thêm logging configuration để đảm bảo console output được hiển thị:

Bây giờ tôi sẽ tạo một cách khác để hiển thị verification link trực tiếp trên trang web:

## �� \*\*GIẢI PHÁP ĐÃ THỰC HIỆN:\*\*

### \*\*1. Cải thiện Console Logging:\*\*

- ✅ Thêm formatting rõ ràng với emoji và border

- ✅ Thêm `System.Diagnostics.Debug.WriteLine()` để log trong debug mode

- ✅ Cập nhật logging configuration

### \*\*2. Hiển thị Verification Link trên Web UI:\*\*

- ✅ Hiển thị link trực tiếp trên trang Register sau khi đăng ký thành công

- ✅ Thêm nút "Copy" để copy link dễ dàng

- ✅ Link được hiển thị trong input field readonly

### \*\*3. Cách xem Console Log:\*\*

\*\*Option 1: Terminal/Command Prompt\*\*

```bash

cd CloneEbay

dotnet run

```

→ Console log sẽ hiển thị trong terminal nơi bạn chạy lệnh

\*\*Option 2: Visual Studio Output Window\*\*

- Mở Visual Studio

- Chạy ứng dụng trong Debug mode

- Xem Output window (View → Output)

\*\*Option 3: Browser Developer Tools\*\*

- Mở Developer Tools (F12)

- Tab Console sẽ hiển thị nếu có JavaScript console.log()

### \*\*4. Test lại:\*\*

1. \*\*Chạy ứng dụng:\*\*

```bash

dotnet run

```

2. \*\*Đăng ký tài khoản mới:\*\*

- Truy cập: `https://localhost:5000/Auth/Register`

- Điền form và submit

3. \*\*Kiểm tra output:\*\*

- \*\*Terminal:\*\* Sẽ thấy log với format đẹp

- \*\*Web UI:\*\* Sẽ thấy verification link với nút copy

- \*\*Debug Output:\*\* Nếu chạy trong Visual Studio

### \*\*5. Ví dụ Console Output:\*\*

```

==================================================

�� VERIFICATION EMAIL SENT

==================================================

📧 To: test@example.com

�� Verification Link: <https://localhost:5000/Auth/VerifyEmail?email=test%40example.com>

📝 Please check your email and click the verification link.

==================================================

```

Bây giờ bạn sẽ thấy log rõ ràng hơn và có thể copy verification link trực tiếp từ web UI!