**BỘ CÔNG THƯƠNG  
TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP.HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

o0o



**TÊN ĐỀ TÀI: QUẢN LÝ CỬA HÀNG THỜI TRANG**

**BÁO CÁO ĐỒ ÁN MÔN HỌC**

**HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU**

SINH VIÊN THỰC HIỆN:

1. 2001215807, Đỗ Quang Huy
2. 2001230363, Nguyễn Tuấn Khải
3. 2001230857, Nguyễn Tấn Tuấn
4. 2001231069, Trần Xuân Vỷ
5. 2044223142, Phạm Vũ Khôi Nguyên

**GVHD:  Trần Quang Bình**

Thành phố Hồ Chí Minh, tháng 11 năm 2025

Phân công công việc

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Họ và tên | Nội dung công việc | Ghi chú |
| 1 | Nguyễn Tấn Tuấn | Thiết kế word phần xây dựng ứng dụng nhóm QuanTriTaiKhoan, thiết kế powerpoint chinh, thiết kế powerpoint phần xây dựng ứng dụng nhóm QuanTriTaiKhoan | Hoàn thành |
| 2 | Trần Xuân Vỷ | Thiết kế word phần xây dựng ứng dụng nhóm QuanLyBrand ,thiết kế powerpoint phần xây dựng ứng dụng nhóm QuanLyBrand | Hoàn thành |
| 3 | Đỗ Quang Huy | Thiết kế word phần xây dựng ứng dụng nhóm QuanLySanPham ,thiết kế powerpoint phần xây dựng ứng dụng nhóm QuanLySanPham | Chưa Hoàn thành |
| 4 | Phạm Vũ Khôi Nguyên | Thiết kế word phần xây dựng ứng dụng nhóm QuanLyKhachHang ,thiết kế powerpoint phần xây dựng ứng dụng nhóm QuanLyKhachHang | Hoàn thành |
| 5 | Nguyễn Tuấn Khải | Thiết kế word phần xây dựng ứng dụng nhóm QuanLyDonHang ,thiết kế powerpoint phần xây dựng ứng dụng nhóm QuanLyDonHang | Chưa Hoàn thành |

MỤC LỤC

[LỜI NÓI ĐẦU 1](#_Toc215612124)

[CHƯƠNG 1. XÂY DỰNG CƠ SỞ DỮ LIỆU 2](#_Toc215612125)

[1.1. Lược đồ diagram 2](#_Toc215612126)

[1.2. Chi tiết và thuộc tính từng bảng 3](#_Toc215612127)

[CHƯƠNG 2. QUẢN TRỊ HỆ THỐNG 6](#_Toc215612128)

[2.1. Quản lý Tài khoản (Logins & Users) 6](#_Toc215612129)

[2.2. Chiến lược Phân quyền dựa trên Vai trò 6](#_Toc215612130)

[2.3. Chi tiết Quyền hạn (Permissions) 7](#_Toc215612131)

[CHƯƠNG 3. XÂY DỰNG ỨNG DỤNG 9](#_Toc215612132)

[3.1. Nhom\_QuanLySanPham 9](#_Toc215612133)

[3.2. Nhom\_QuanLyBrand 9](#_Toc215612134)

[3.2.1. Mô tả ứng dụng 9](#_Toc215612135)

[3.2.2. Các đối tượng xử lý và điều khiển 11](#_Toc215612136)

[3.3. Nhom\_QuanTriTaiKhoan 30](#_Toc215612137)

[3.3.1. Mô tả ứng dụng 30](#_Toc215612138)

[3.3.2. Các đối tượng xử lý và điều khiển: 30](#_Toc215612139)

[3.4. Nhom\_XuLyDonHang 69](#_Toc215612140)

[3.5. Nhom\_QuanLyKH 70](#_Toc215612141)

[3.5.1. Mô tả ứng dụng: 70](#_Toc215612142)

[3.3.2. Các đối tượng xử lý và điều khiển: 70](#_Toc215612143)

[TÀI LIỆU THAM KHẢO 121](#_Toc215612144)

# LỜI NÓI ĐẦU

Trong bối cảnh chuyển đổi số hiện nay, thương mại điện tử được xem là xu hướng tất yếu và là động lực quan trọng thúc đẩy sự phát triển kinh tế. Đối với ngành thời trang – lĩnh vực có tốc độ luân chuyển hàng hóa nhanh và nhu cầu quản lý đa dạng – việc ứng dụng công nghệ thông tin vào quy trình vận hành trở thành yêu cầu bắt buộc nhằm duy trì lợi thế cạnh tranh. Trên cơ sở đó, đề tài Xây dựng Cơ sở dữ liệu cho Hệ thống Bán hàng Thời trang được lựa chọn với mục tiêu đáp ứng xu hướng phát triển của thị trường, đồng thời đảm bảo mức độ phức tạp cần thiết để vận dụng kiến thức chuyên môn về thiết kế hệ thống.

Cơ sở dữ liệu (CSDL) được xác định là thành phần trung tâm, đóng vai trò như “trái tim” của toàn bộ hệ thống. Một CSDL được thiết kế hợp lý không chỉ đảm bảo khả năng lưu trữ, tìm kiếm và xử lý nghiệp vụ chính xác mà còn tạo nền tảng cho tính bảo mật, khả năng tích hợp và mở rộng trong tương lai. Phạm vi nghiên cứu tập trung vào việc giải quyết bài toán quản lý cho các cửa hàng thời trang quy mô vừa và nhỏ, hướng tới tối ưu hóa các quy trình cốt lõi như quản lý danh mục sản phẩm, kiểm soát tồn kho, xử lý đơn hàng và phân quyền chặt chẽ giữa các đối tượng sử dụng (Admin, Nhân viên, Khách hàng).

# CHƯƠNG 1. XÂY DỰNG CƠ SỞ DỮ LIỆU

## 1.1. Lược đồ diagram



## 1.2. Chi tiết và thuộc tính từng bảng

**Nhóm Người dùng & Khách hàng**

* **Users** (UserID, Name, Email, Password, Role, CreatedAt)
* **KhachHang** (MaKH, TenKH, Email, SoDT, BirthDayKH, GioiTinhKH, NgayDangKy, UserId)

**Nhóm Sản phẩm & Danh mục**

* **Category** (Category\_Id, CategoryName, CategorySlug, Description, CategoryImageUrl, IsActive, ParentCategory\_Id, Brand\_Id)
* **Brand** (Brand\_Id, Name, Phone, Address, CountryOfOrigin, Email)
* **Size** (Size\_Id, SizeName, SortOrder)
* **Color** (Color\_Id, ColorName, ColorCode)
* **Product** (Product\_Id, ProductName, ProductSlug, ShortDescription, FullDescription, Price, SalePrice, Quantity, IsActive, SoldCount, CreatedDate, Category\_Id, Brand\_Id, DefaultImage\_Id, Size\_Id, Color\_Id)
* **ProductImage** (Image\_Id, ImageUrl, IsDefault)
* **Product\_Image\_Mapping** (Product\_Id, Image\_Id)

**Nhóm Đơn hàng & Giao dịch**

* **Cart** (Cart\_Id, Quantity, CreatedDate, Product\_Id, MaKH)
* **Orders** (Order\_Id, OrderNo, Status, SubTotal, TotalAmount, ShippingAddress, OrderDate, IsCancel, MaKH, Payment\_Id, Coupon\_Id, ConfirmedDate, ShippedDate, DeliveredDate, CancelledDate, CancelledBy, CancelReason, InternalNote)
* **OrderDetail** (OrderDetail\_Id, Quantity, UnitPrice, TotalPrice, Order\_Id, Product\_Id)
* **Payment** (Payment\_Id, PaymentMethod, PaymentStatus, Amount, MaKH, CreatedDate, PaymentDate)
* **Coupon** (Coupon\_Id, CouponCode, DiscountType, DiscountValue, UsageLimitPerUser, IsActive, StartDate, ExpiryDate)
* **Coupon\_Usage** (Usage\_Id, Coupon\_Id, MaKH, UsedDate)

**Nhóm Kho hàng (Inventory)**

* **Warehouse** (Warehouse\_Id, WarehouseName)
* **WarehouseStock** (Stock\_Id, Quantity, Warehouse\_Id, Product\_Id)
* **StockTransaction** (Transaction\_Id, TransactionType, Quantity, PreviousQuantity, NewQuantity, UnitPrice, TotalValue, Reason, ReferenceNo, TransactionDate, CreatedBy, Stock\_Id, Order\_Id)

**Nhóm Đánh giá & Phản hồi**

* **ReviewProduct** (Review\_Id, Rating, Comment, IsApproved, CreatedDate, Product\_Id, MaKH, Order\_Id)
* **InternalRequests** (RequestID, RequestTitle, RequestContent, CreatedBy, CreatedDate, Status)
* **RequestCounter** (CounterID, TotalRequests)

**Nhóm Hệ thống & Log**

* **AuditLog** (LogID, ActionName, Description, PerformedBy, TargetUserEmail, ActionTime)
* **SystemLog** (LogID, Msg, LogTime)
* **Brand\_Log** (LogId, Brand\_Id, Action, LogTime)
* **sysdiagrams** (name, principal\_id, diagram\_id, version, definition)
* **\_\_EFMigrationsHistory** (MigrationId, ProductVersion)

# CHƯƠNG 2. QUẢN TRỊ HỆ THỐNG

## 2.1. Quản lý Tài khoản (Logins & Users)

Mô hình quản lý chia làm 2 cấp độ: Server Login (để kết nối vào Server) và Database User (để thao tác trong FashionShopDb)

**Tài khoản Quản trị viên (Super Admin)**

* Login: Nhom4
* User: Nhom4
* Quyền hạn: Được gán vào nhóm db\_owner. Đây là tài khoản có quyền cao nhất trong Database, được phép thực hiện mọi thao tác (DDL, DML, DCL) và quản lý các user khác

**Tài khoản Cá nhân (Individual Users)**

Hệ thống tạo các Login riêng biệt cho từng nhân viên/thành viên trong nhóm để đảm bảo tính định danh (biết rõ ai làm gì).

* Danh sách: Tuan, Huy, Khai, Nguyen, Vy.
* Mapping: Mỗi Login này được map thành một User tương ứng bên trong FashionShopDb.

## 2.2. Chiến lược Phân quyền dựa trên Vai trò

Thay vì cấp quyền trực tiếp cho từng User (rất khó quản lý khi nhân sự thay đổi), hệ thống áp dụng RBAC. Quyền hạn được cấp cho Role, sau đó User được thêm vào Role.

Hệ thống định nghĩa 5 nhóm quyền nghiệp vụ (Custom Database Roles) tương ứng với quy trình vận hành thực tế:

| **Tên Role (Nhóm)** | **Mô tả chức năng** | **Thành viên** |
| --- | --- | --- |
| **Nhom\_QuanLySanPham** | Chịu trách nhiệm về danh mục, sản phẩm, hình ảnh. | Huy |
| **Nhom\_QuanLyBrand** | Làm việc với nhà cung cấp/thương hiệu. | Vy |
| **Nhom\_QuanTriTaiKhoan** | Quản lý người dùng hệ thống, khách hàng, bảo mật. | Tuan |
| **Nhom\_XuLyDonHang** | Xử lý quy trình bán hàng, đơn hàng, báo cáo doanh thu. | Khai |
| **Nhom\_QuanLyKH** | Chăm sóc khách hàng (Customer Service). | Nguyen |

## 2.3. Chi tiết Quyền hạn (Permissions)

Dưới đây là chi tiết các quyền GRANT (Cấp) và REVOKE (Thu hồi) cho từng nhóm:

**Nhóm Quản lý Sản phẩm (Nhom\_QuanLySanPham)**

* Quyền dữ liệu (CRUD): Toàn quyền SELECT, INSERT, UPDATE, DELETE trên các bảng: Product, Category, Size, Color, ProductImage.
* Quyền tham chiếu: Chỉ được xem (SELECT) bảng Brand để gán thương hiệu cho sản phẩm (không được sửa Brand).
* Quyền thực thi (EXECUTE): Được chạy các Stored Procedures liên quan đến kiểm tra hàng hóa (sp\_GetAllSizes, sp\_HangThoiTrang\_CheckExists...).
* Quyền xem định nghĩa (VIEW DEFINITION): Được phép xem code SQL của các thủ tục trên để phục vụ debug.
* Hạn chế: Đã thực hiện REVOKE VIEW DEFINITION trên toàn bộ Schema để tránh việc nhân viên này xem trộm cấu trúc các bảng không liên quan.

**Nhóm Quản lý Thương hiệu (Nhom\_QuanLyBrand)**

* Quyền dữ liệu: Toàn quyền trên bảng Brand và Brand\_Log.
* Quyền tham chiếu: Chỉ xem bảng Product (để biết Brand cung cấp SP nào) và Users.
* Quyền nghiệp vụ: Được chạy các thủ tục đặc thù như AddNewBrand, UpdateBrandWithTransaction.

**Nhóm Quản trị Tài khoản (Nhom\_QuanTriTaiKhoan)**

* Quyền dữ liệu: Toàn quyền trên bảng Users (Hệ thống nhân viên). Đối với bảng KhachHang: Ban đầu cấp toàn quyền, sau đó dùng REVOKE để thu hồi quyền CRUD trực tiếp, ép buộc phải dùng qua Stored Procedure để đảm bảo tính toàn vẹn dữ liệu.
* Quyền nghiệp vụ:Chạy thủ tục đăng ký (sp\_RegisterUser...), đăng nhập (sp\_LoginUser).Chạy thủ tục Reset mật khẩu (sp\_ResetUserPassword, sp\_ChangeUserPassword).Ghi và xem nhật ký hệ thống (AuditLog).

**Nhóm Xử lý Đơn hàng (Nhom\_XuLyDonHang)**

* Quyền dữ liệu: Toàn quyền trên Orders, OrderDetail.
* Quyền tham chiếu: Chỉ xem thông tin KhachHang, Product để xác nhận đơn.
* Quyền nghiệp vụ (Báo cáo & Thống kê):Được chạy các thủ tục thống kê Dashboard (sp\_GetDashboardStatistics), xuất báo cáo doanh thu (sp\_ExportSalesReport).Cập nhật trạng thái đơn hàng (sp\_UpdateOrderStatus), hủy đơn (sp\_CancelOrder).Bảo mật: Đã REVOKE quyền sp\_CreateOrderFromCart (tạo đơn) và sp\_AddToCart vì nghiệp vụ này thường do Khách hàng tự thực hiện trên App, nhân viên kho chỉ xử lý đơn đã có.

**Nhóm Quản lý Khách hàng (Nhom\_QuanLyKH)**

* Quyền Thao tác (SELECT, INSERT, UPDATE): Bảng KhachHang: Để hỗ trợ khách đăng ký, cập nhật địa chỉ/SĐT. Không cấp quyền DELETE để bảo vệ dữ liệu lịch sử. Bảng InternalRequests: Để tạo và cập nhật trạng thái các yêu cầu hỗ trợ (Ticket).
* Quyền Duyệt (SELECT, UPDATE): Bảng ReviewProduct (Duyệt nội dung đánh giá sản phẩm).
* Quyền Tra cứu (SELECT Only): Bảng Product: Để tư vấn thông tin sản phẩm.

# CHƯƠNG 3. XÂY DỰNG ỨNG DỤNG

## 3.1. Nhom\_QuanLySanPham

## 3.2. Nhom\_QuanLyBrand

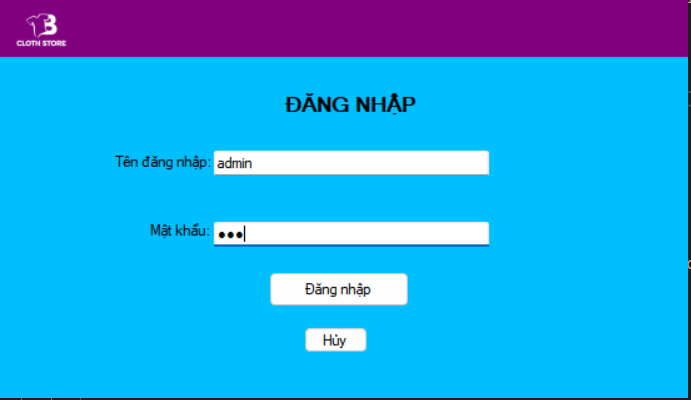
### 3.2.1. Mô tả ứng dụng

Ứng dụng Quản lý Nhà Cung Cấp được xây dựng bằng Windows Forms (WinForms) trên nền tảng C# .NET, hỗ trợ doanh nghiệp quản lý thông tin các nhà cung cấp một cách nhanh chóng và chính xác.

Chương trình gồm các chức năng chính:

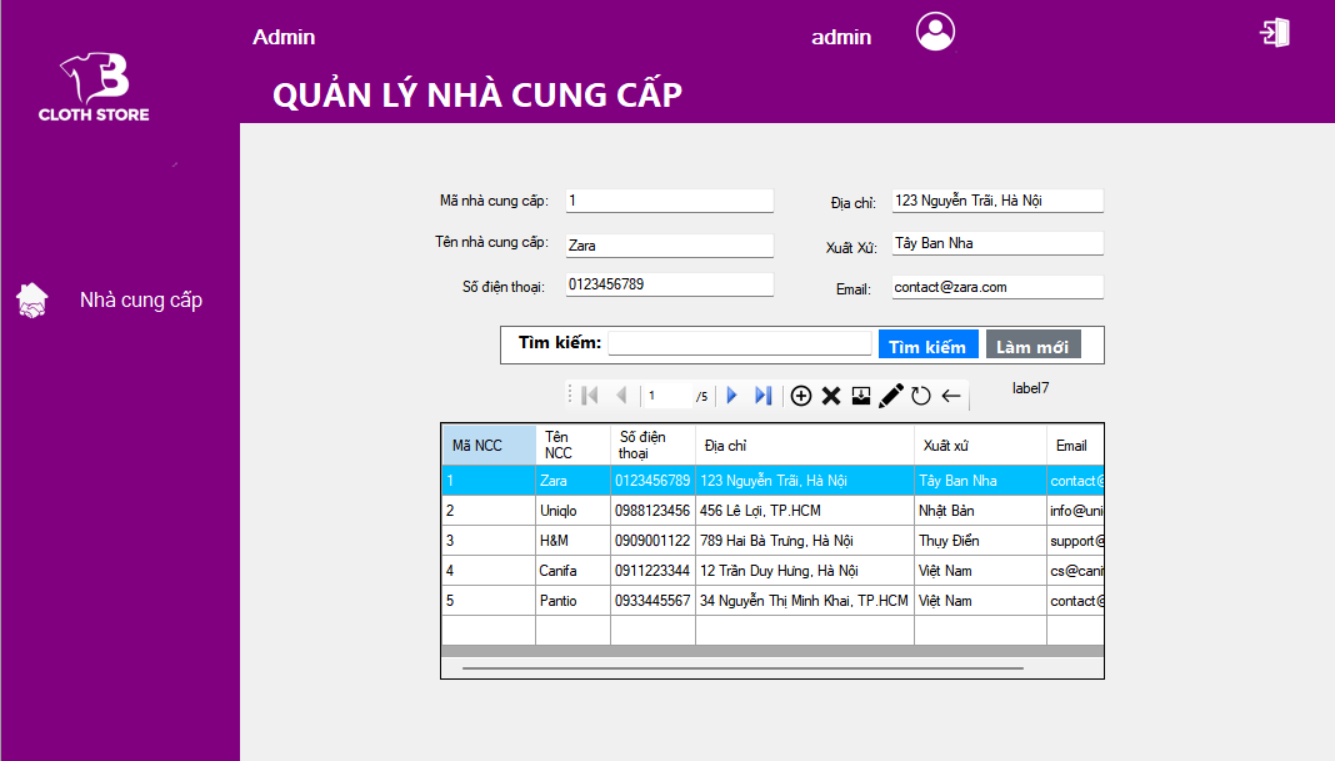
* Đăng nhập hệ thống
* Xem danh sách nhà cung cấp
* Thêm nhà cung cấp
* Sửa thông tin
* Xóa
* Tìm kiếm
* Điều hướng dữ liệu

**- Giao diện đăng nhập:**



**Mục đích:**

* Xác thực người dùng để cho phép truy cập vào phần quản trị .
* Chỉ những người có tài khoản quản trị hợp lệ mới được vào để quản lý toàn bộ cửa hàng trực tuyến.
* **Giao diện chính:**



**Mục đích:** Quản lý toàn bộ các nhà cung cấp/sỉ mà cửa hàng đang nhập hàng về để bán lẻ.

### 3.2.2. Các đối tượng xử lý và điều khiển

**--PROCEDURE**

**--Kiểm tra mã và tên nhà cung cấp**

CREATE PROCEDURE AddNewBrand

@Brand\_Id INT,

@Name NVARCHAR(250),

@Phone VARCHAR(50),

@Address NVARCHAR(100),

@CountryOfOrigin NVARCHAR(100),

@Email NVARCHAR(100)

AS

BEGIN

IF EXISTS (SELECT 1 FROM Brand WHERE Brand\_Id = @Brand\_Id OR Name = @Name)

BEGIN

RAISERROR('Mã hoặc tên nhà cung cấp đã tồn tại.', 16, 1)

RETURN

END

INSERT INTO Brand (Brand\_Id, Name, Phone, Address, CountryOfOrigin, Email)

VALUES (@Brand\_Id, @Name, @Phone, @Address, @CountryOfOrigin, @Email)

END

**Mục đích chính:** Thêm một nhà cung cấp mới vào hệ thống một cách an toàn, không cho phép trùng mã hoặc trùng tên.

**Cách hoạt động:**

1. Kiểm Tra Điều Kiện Tồn Tại

Procedure sử dụng câu lệnh IF EXISTS (SELECT 1 FROM Brand WHERE Brand\_Id = @Brand\_Id OR Name = @Name) để kiểm tra xem đã có bản ghi nào trong bảng Brand có Brand\_Id trùng với giá trị @Brand\_Id được truyền vào hoặc có Name trùng với giá trị @Name được truyền vào hay không.

- Nếu Tồn Tại: Procedure sẽ dừng thực thi và báo lỗi bằng lệnh RAISERROR('Mã hoặc tên nhà cung cấp đã tồn tại.', 16, 1). Điều này ngăn chặn việc thêm bản ghi trùng lặp, đảm bảo tính toàn vẹn của dữ liệu (duy nhất cho ID và Tên).

- Nếu Không Tồn Tại: Procedure tiếp tục đến bước 2.

2. Thêm Dữ Liệu

Procedure sử dụng câu lệnh INSERT INTO Brand (...) VALUES (...) để thêm một bản ghi mới với các thông tin chi tiết của nhà cung cấp (@Brand\_Id, @Name, @Phone, @Address, @CountryOfOrigin, @Email) vào bảng Brand.

**Kiểm thử:**

* Thêm hợp lệ:

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, hàng

Nội dung do AI tạo ra có thể không chính xác.

* Thêm không hợp lệ:

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, hàng

Nội dung do AI tạo ra có thể không chính xác.

* Hiển thị trong ứng dụng:

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Trang web

Nội dung do AI tạo ra có thể không chính xác.

**--.Xóa nhà cung cấp**

CREATE PROCEDURE DeleteBrand

@Brand\_Id INT

AS

BEGIN

SET NOCOUNT ON;

IF EXISTS (SELECT 1 FROM Product WHERE Brand\_Id = @Brand\_Id)

BEGIN

RAISERROR('Không thể xóa nhà cung cấp này vì đang có sản phẩm thuộc nhà cung cấp.', 16, 1);

RETURN;

END

DELETE FROM Brand WHERE Brand\_Id = @Brand\_Id;

IF @@ROWCOUNT > 0

PRINT 'Xóa nhà cung cấp thành công!';

ELSE

RAISERROR('Không tìm thấy nhà cung cấp để xóa.', 16, 1);

END

GO

**Mục đích chính:**

Mục đích chính của Stored Procedure DeleteBrand là xóa một nhà cung cấp khỏi bảng Brand dựa trên @Brand\_Id được cung cấp, đồng thời ngăn chặn việc xóa nếu nhà cung cấp đó hiện đang có sản phẩm (Product) liên kết.

**Cách hoạt động:**

1. Thiết lập môi trường và kiểm tra ràng buộc

SET NOCOUNT ON;: Thiết lập này ngăn chặn việc trả về số lượng hàng bị ảnh hưởng sau mỗi câu lệnh (như DELETE), giúp giảm lưu lượng mạng và tối ưu hóa hiệu suất, đặc biệt trong các ứng dụng.

Kiểm tra ràng buộc:

IF EXISTS (SELECT 1 FROM Product WHERE Brand\_Id = @Brand\_Id): Procedure kiểm tra xem có bất kỳ bản ghi nào trong bảng Product đang tham chiếu đến @Brand\_Id cần xóa hay không. Điều này nhằm duy trì tính toàn vẹn tham chiếu (Referential Integrity).

Nếu Tồn Tại (có sản phẩm):

Procedure dừng thực thi và báo lỗi bằng lệnh RAISERROR('Không thể xóa nhà cung cấp này vì đang có sản phẩm thuộc nhà cung cấp.', 16, 1);.

RETURN;: Thoát khỏi procedure.

2. Thực hiện thao tác xóa

Nếu Không Tồn Tại (không có sản phẩm nào liên kết):

DELETE FROM Brand WHERE Brand\_Id = @Brand\_Id;: Thực hiện lệnh xóa nhà cung cấp khỏi bảng Brand có mã trùng khớp.

3. Kiểm tra kết quả và thông báo

IF @@ROWCOUNT > 0: Biến hệ thống @@ROWCOUNT chứa số lượng hàng bị ảnh hưởng bởi câu lệnh DML (DELETE) ngay trước đó. Procedure kiểm tra xem có hàng nào đã được xóa thành công hay không.

Nếu @@ROWCOUNT > 0 (Xóa thành công):

PRINT 'Xóa nhà cung cấp thành công!';: In thông báo xác nhận việc xóa thành công.

Nếu @@ROWCOUNT = 0 (Không tìm thấy ID):

RAISERROR('Không tìm thấy nhà cung cấp để xóa.', 16, 1);: Báo lỗi nếu không tìm thấy nhà cung cấp nào có @Brand\_Id tương ứng để xóa (mặc dù không có lỗi nghiêm trọng, nhưng thông báo cho người dùng biết thao tác xóa không xảy ra).

**Kiểm thử**:

* Xóa hợp lệ:

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

* Xóa không hợp lệ:

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

* Hiển thị trong ứng dụng:

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Trang web

Nội dung do AI tạo ra có thể không chính xác.

**--TRIGGER**

**--Kiểm tra Email**

CREATE TRIGGER trg\_CheckEmailFormat

ON Brand

FOR INSERT, UPDATE

AS

BEGIN

SET NOCOUNT ON;

IF EXISTS (

SELECT 1

FROM inserted

WHERE Email NOT LIKE '\_%@\_%.\_%'

)

BEGIN

RAISERROR('Email không đúng định dạng!', 16, 1);

ROLLBACK TRANSACTION;

END

END

**Mục đích chính:**

Mục đích chính của Trigger trg\_CheckEmailFormat là kiểm tra định dạng của trường Email trước khi một bản ghi mới được thêm (INSERT) hoặc một bản ghi hiện có được cập nhật (UPDATE) trong bảng Brand. Trigger này nhằm đảm bảo rằng địa chỉ Email được cung cấp có cấu trúc cơ bản của một Email hợp lệ.

**Cách hoạt động:**

1. Kích hoạt và Thiết lập

Sự kiện kích hoạt: Trigger được kích hoạt sau khi bất kỳ lệnh INSERT hoặc UPDATE nào được thực hiện trên bảng Brand.

SET NOCOUNT ON;: Ngăn chặn việc trả về thông báo số lượng hàng bị ảnh hưởng.

Bảng inserted: Trigger sử dụng bảng ảo inserted, chứa tất cả các hàng dữ liệu mới được thêm vào hoặc được sửa đổi.

2. Kiểm tra Định dạng Email

Cơ chế kiểm tra:

Trigger sử dụng câu lệnh IF EXISTS (SELECT 1 FROM inserted WHERE Email NOT LIKE '\_%@\_%.\_%') để quét tất cả các hàng trong bảng inserted.

Mẫu kiểm tra (LIKE '\_%@\_%.\_%'): Mẫu này kiểm tra các điều kiện tối thiểu sau cho trường Email:

Phải bắt đầu bằng ít nhất một ký tự (\_).

Phải chứa ký tự @.

Phải có ít nhất một ký tự sau @ (\_%).

Phải chứa ký tự . (dấu chấm).

Phải có ít nhất một ký tự sau dấu chấm (\_%).

3. Xử lý Lỗi

Nếu Định dạng Sai:

Nếu tìm thấy bất kỳ hàng nào trong inserted có Email không khớp với mẫu đã định:

RAISERROR('Email không đúng định dạng!', 16, 1);: Một lỗi nghiêm trọng (mức 16) sẽ được gửi đến người dùng hoặc ứng dụng.

ROLLBACK TRANSACTION;: Lệnh quan trọng này sẽ hoàn tác toàn bộ giao dịch (cả INSERT hoặc UPDATE ban đầu) đã kích hoạt trigger. Điều này ngăn chặn dữ liệu không hợp lệ được lưu vào bảng Brand.

**Kiểm thử:**

* Thêm/Sửa Email dạng đúng:

Ảnh có chứa văn bản, ảnh chụp màn hình, hàng, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

* Thêm/Sửa Email dạng sai:

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, Trang web

Nội dung do AI tạo ra có thể không chính xác.

* Hiển thị trong ứng dụng:

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Trang web

Nội dung do AI tạo ra có thể không chính xác.

**--TRIGGER**

**--Số điện thoại phải bắt đầu bằng 0 và chỉ chứa số (độ dài 10-11 chữ số)**

CREATE OR ALTER TRIGGER trg\_CheckPhoneFormat

ON Brand

FOR INSERT, UPDATE

AS

BEGIN

SET NOCOUNT ON;

IF EXISTS (

SELECT 1

FROM inserted i

WHERE

(i.Phone IS NOT NULL AND LTRIM(RTRIM(i.Phone)) != '')

AND

(

i.Phone NOT LIKE '0%'

OR

i.Phone NOT LIKE '%[0-9]%'

OR PATINDEX('%[^0-9]%', i.Phone) > 0

OR

-- Độ dài không hợp lệ

LEN(REPLACE(i.Phone, ' ', '')) NOT BETWEEN 10 AND 11

)

)

BEGIN

RAISERROR('Số điện thoại không hợp lệ!

- Phải bắt đầu bằng số 0

- Chỉ được chứa chữ số (có thể có dấu cách)

- Độ dài từ 10 đến 11 chữ số', 16, 1);

ROLLBACK TRANSACTION;

RETURN;

END

IF UPDATE(Phone)

BEGIN

UPDATE Brand

SET Phone = LTRIM(RTRIM(

REPLACE(REPLACE(REPLACE(REPLACE(i.Phone, ' ', ''), '-', ''), '.', ''), '(', '')

))

FROM inserted i

WHERE Brand.Brand\_Id = i.Brand\_Id;

END

END

GO

**Mục đích chính:**

Kiểm tra tính hợp lệ của trường Phone (phải bắt đầu bằng '0', chỉ chứa chữ số, và có độ dài 10-11 chữ số) khi thêm hoặc cập nhật bản ghi trong bảng Brand.

Chuẩn hóa số điện thoại bằng cách xóa các ký tự không phải là số (như dấu cách, dấu gạch ngang, dấu chấm, dấu ngoặc) nếu việc kiểm tra định dạng thành công.

**Cách hoạt động:**

1. Kích hoạt và Thiết lập

Sự kiện kích hoạt: Trigger được kích hoạt sau khi bất kỳ lệnh INSERT hoặc UPDATE nào được thực hiện trên bảng Brand.

SET NOCOUNT ON;: Ngăn chặn việc trả về số lượng hàng bị ảnh hưởng.

Bảng inserted: Chứa dữ liệu số điện thoại mới hoặc đã được sửa đổi.

2. Kiểm tra Điều kiện Hợp lệ

(i.Phone IS NOT NULL AND LTRIM(RTRIM(i.Phone)) != ''): Đảm bảo trường Phone không rỗng và không chỉ chứa dấu cách.

i.Phone NOT LIKE '0%': Kiểm tra xem số điện thoại có bắt đầu bằng '0' hay không. Nếu không, điều kiện này đúng (là lỗi).

i.Phone NOT LIKE '%[0-9]%' OR PATINDEX('%[^0-9]%', i.Phone) > 0:

Đây là cơ chế kép để kiểm tra ký tự .

LEN(REPLACE(i.Phone, ' ', '')) NOT BETWEEN 10 AND 11: Trước tiên, loại bỏ các dấu cách khỏi chuỗi (REPLACE(i.Phone, ' ', '')), sau đó kiểm tra độ dài phải nằm trong khoảng 10 đến 11 ký tự số.

3. Chuẩn hóa Dữ liệu

IF UPDATE(Phone): Chỉ thực hiện chuẩn hóa nếu cột Phone là cột đang được cập nhật/thêm vào.

Thao tác Chuẩn hóa:

Một lệnh UPDATE được thực hiện trực tiếp trên bảng Brand (sau khi các kiểm tra ở bước 2 đã vượt qua).

Mục tiêu: Xóa tất cả các ký tự định dạng (như , -, ., () khỏi chuỗi số điện thoại.

Cơ chế: Sử dụng hàm REPLACE lồng nhau để loại bỏ các ký tự này, sau đó dùng LTRIM(RTRIM(...)) để xóa dấu cách ở hai đầu.

Kết quả: Dữ liệu cuối cùng được lưu vào cột Phone sẽ là một chuỗi chỉ gồm 10 hoặc 11 chữ số, bắt đầu bằng '0' (ví dụ: 0901234567).

**Kiểm thử:**

- Thêm hợp lệ:

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, hàng

Nội dung do AI tạo ra có thể không chính xác.

- Thêm không hợp lệ:

Ảnh có chứa văn bản, đồ điện tử, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

- Hiển thị trong ứng dụng:

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Trang web

Nội dung do AI tạo ra có thể không chính xác.

**--TRANSACTION**

**--Cập nhật nhiều bảng, rollback khi lỗi**

-- Bảng Brand\_Log

CREATE TABLE Brand\_Log (

LogId INT IDENTITY PRIMARY KEY,

Brand\_Id INT,

Action NVARCHAR(100),

LogTime DATETIME

)

drop table Brand\_Log

--Cập nhật bảng với Transaction

CREATE OR ALTER PROCEDURE UpdateBrandWithTransaction

@Brand\_Id INT,

@Name NVARCHAR(100),

@Phone NVARCHAR(20),

@Address NVARCHAR(200),

@CountryOfOrigin NVARCHAR(50),

@Email NVARCHAR(100)

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

BEGIN TRANSACTION;

UPDATE Brand

SET Name = @Name,

Phone = @Phone,

Address = @Address,

CountryOfOrigin = @CountryOfOrigin,

Email = @Email

WHERE Brand\_Id = @Brand\_Id;

IF @@ROWCOUNT = 0

BEGIN

ROLLBACK TRANSACTION;

THROW 51000, N'Không tìm thấy Brand với Id này.', 1;

RETURN;

END

INSERT INTO Brand\_Log (Brand\_Id, Action, LogTime)

VALUES (@Brand\_Id, N'Cập nhật thông tin', GETDATE());

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION;

THROW;

END CATCH

END

**Mục đích chính:**

Cập nhật thông tin chi tiết của một nhà cung cấp (Brand) trong bảng Brand.

Ghi lại hành động cập nhật này vào bảng Brand\_Log.

Đảm bảo rằng cả hai hành động (Cập nhật và Ghi Log) đều thành công hoặc thất bại cùng nhau thông qua việc sử dụng Transaction (giao dịch), duy trì nguyên tắc Tính nguyên tố của ACID.

**Cách hoạt động:**

1. Khối BEGIN TRY (Thực hiện Giao dịch)

BEGIN TRANSACTION;: Bắt đầu một giao dịch mới. Tất cả các thao tác DML (như UPDATE, INSERT) từ thời điểm này trở đi được coi là một đơn vị công việc duy nhất.

Thao tác 1: Cập nhật dữ liệu Brand

UPDATE Brand

SET Name = @Name, ...

WHERE Brand\_Id = @Brand\_Id;

Thông tin nhà cung cấp được cập nhật trong bảng Brand.

Kiểm tra Bản ghi bị Ảnh hưởng:

IF @@ROWCOUNT = 0

BEGIN

ROLLBACK TRANSACTION;

THROW 51000, N'Không tìm thấy Brand với Id này.', 1;

RETURN;

END

Nếu UPDATE không tìm thấy Brand nào (@@ROWCOUNT = 0), nó sẽ:

ROLLBACK TRANSACTION;: Hủy bỏ giao dịch. (Mặc dù lệnh UPDATE không thực sự làm gì, việc hủy bỏ là một bước an toàn).

THROW: Báo lỗi tùy chỉnh cho người dùng biết không tìm thấy bản ghi.

Thao tác 2: Ghi Log (Log Action)

INSERT INTO Brand\_Log (Brand\_Id, Action, LogTime)

VALUES (@Brand\_Id, N'Cập nhật thông tin', GETDATE());

Nếu cập nhật thành công, thông tin giao dịch sẽ được ghi vào bảng Brand\_Log.

COMMIT TRANSACTION;: Nếu cả hai thao tác (UPDATE và INSERT) đều thực hiện mà không có lỗi, giao dịch sẽ được xác nhận và các thay đổi được ghi vĩnh viễn vào cơ sở dữ liệu.

2. Khối BEGIN CATCH (Xử lý Lỗi)

Khối này được kích hoạt nếu bất kỳ lỗi nào xảy ra trong khối BEGIN TRY (ví dụ: lỗi ràng buộc khóa ngoại, lỗi định dạng dữ liệu, hoặc lỗi THROW đã được tạo).

IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION;:

@@TRANCOUNT là biến hệ thống đếm số lượng giao dịch đang mở.

Nếu có lỗi xảy ra và một giao dịch vẫn đang mở, lệnh này sẽ hoàn tác (Rollback) giao dịch đó. Điều này đảm bảo rằng nếu bất kỳ phần nào của giao dịch thất bại, toàn bộ thay đổi (cập nhật Brand và ghi Log) sẽ bị hủy bỏ, dữ liệu được trả về trạng thái trước khi thực thi procedure.

THROW;: Báo lại lỗi đã xảy ra trong khối TRY cho người gọi .

**Kiểm thử:**

* Gọi không có lỗi:

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Nội dung do AI tạo ra có thể không chính xác.

* Gọi khi có lỗi: Bảng Brand\_Log không ghi và dữ liệu trong bảng Brand giữ nguyên.

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Nội dung do AI tạo ra có thể không chính xác.

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Nội dung do AI tạo ra có thể không chính xác.

## 3.3. Nhom\_QuanTriTaiKhoan

### 3.3.1. Mô tả ứng dụng

Ứng dụng được phát triển là hệ thống quản lý tài khoản người dùng được xây dựng trên nền tảng **ASP.NET** và **SQL Server**. Hệ thống cung cấp các chức năng chính bao gồm: đăng nhập, đăng ký, đăng xuất, khôi phục mật khẩu, cập nhật hồ sơ cá nhân và thay đổi mật khẩu. Đối với người dùng có vai trò quản trị viên (Admin), hệ thống cho phép xem danh sách toàn bộ tài khoản, thêm mới và chỉnh sửa thông tin tài khoản. Ngoài ra, ứng dụng còn tích hợp tính năng ghi nhật ký thay đổi mật khẩu (Audit Logs) nhằm theo dõi các hoạt động và nâng cao tính bảo mật. Về mặt công nghệ, ứng dụng sử dụng ASP.NET kết hợp với Entity Framework để thực hiện kết nối cơ sở dữ liệu, các thao tác CRUD (Create, Read, Update, Delete) và xác thực người dùng. Toàn bộ quy trình từ đăng nhập, quản lý tài khoản đến phân quyền người dùng đều đã được triển khai và kiểm thử đầy đủ.

### 3.3.2. Các đối tượng xử lý và điều khiển:

**[dbo].[sp\_ChangeUserPassword]**

CREATE PROCEDURE sp\_ChangeUserPassword

@UserID INT,

@NewPasswordHash NVARCHAR(255),

@Result INT OUTPUT

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

UPDATE Users

SET Password = @NewPasswordHash

WHERE UserID = @UserID;

IF EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'[dbo].[AuditLog]'))

BEGIN

DECLARE @UserEmail NVARCHAR(100);

SELECT @UserEmail = Email FROM Users WHERE UserID = @UserID;

INSERT INTO AuditLog (ActionName, Description, PerformedBy, TargetUserEmail, ActionTime)

VALUES (

'CHANGE\_PASSWORD',

N'Người dùng tự thay đổi mật khẩu cá nhân',

@UserEmail,

@UserEmail,

GETDATE()

);

END

SET @Result = 0;

END TRY

BEGIN CATCH

SET @Result = -1;

END CATCH

END

* **Chức năng**: sp\_ChangeUserPassword là Stored Procedure dùng để thay đổi mật khẩu người dùng. Bao gồm: Cập nhật mật khẩu mới vào bảng Users Và Ghi lại lịch sử thay đổi mật khẩu vào bảng AuditLog.  
  Lưu thông tin gồm: hành động, mô tả, người thực hiện, email đích và thời gian.

Thủ tục trả về kết quả qua biến OUTPUT:

* 0 → Thành công
* -1 → Lỗi trong quá trình xử lý
* **Cách hoạt động:**

Bước 1: Kiểm tra tham số

Nhận vào:

* @UserID → ID của người muốn đổi mật khẩu
* @NewPasswordHash → mật khẩu mới đã được *hash*
* @Result → biến OUTPUT báo kết quả

Bước 2: Bắt đầu TRY/CATCH

* Đảm bảo khi xảy ra lỗi sẽ chuyển sang CATCH và trả về -1.

Bước 3: Cập nhật mật khẩu

UPDATE Users

SET Password = @NewPasswordHash

WHERE UserID = @UserID;

Bước 4: Kiểm tra xem bảng AuditLog có tồn tại

* IF EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'[dbo].[AuditLog]'))

Bước 5: Lấy email người dùng

* SELECT @UserEmail = Email FROM Users WHERE UserID = @UserID;

Bước 6: Ghi lịch sử Audit Log

* INSERT INTO AuditLog (...)
* VALUES (...)

Bước 7: Trả kết quả

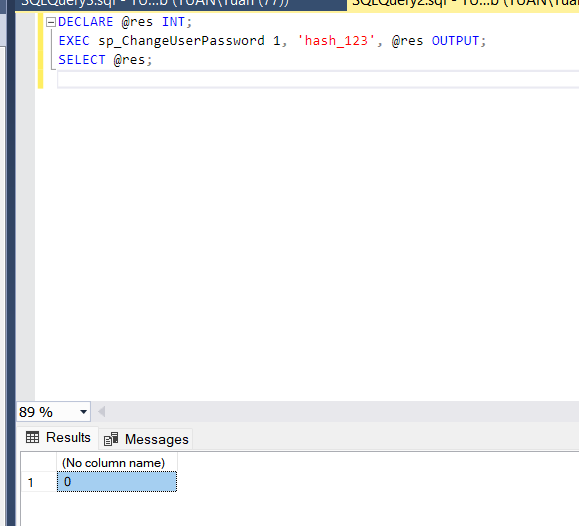
* Nếu không có lỗi → @Result = 0
* Nếu có lỗi → CATCH → @Result = -1
* **Kiểm thử:**

**Test 1 – Đổi mật khẩu hợp lệ:**

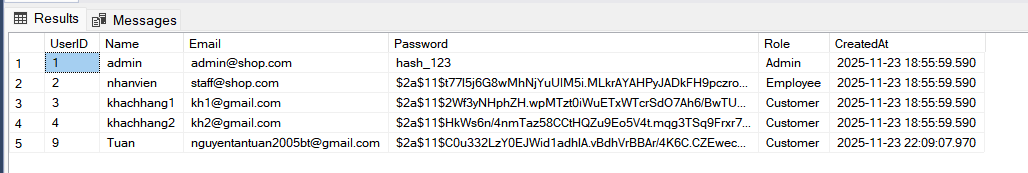
DECLARE @res INT;

EXEC sp\_ChangeUserPassword 1, 'hash\_123', @res OUTPUT;

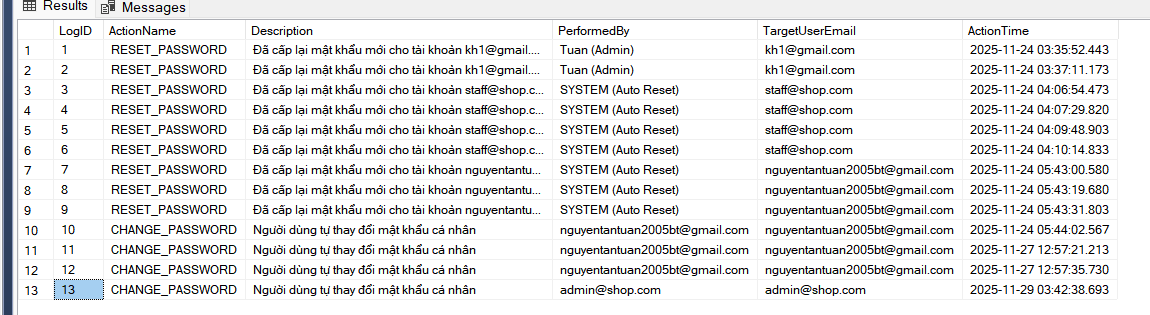
SELECT @res;



Output = 0, chạy thành công



Password của UserID = 1 thay đổi thành “hash\_123”



Lưu 1 bản ghi mới vào AuditLog

**Test 2 - UserID không tồn tại:**

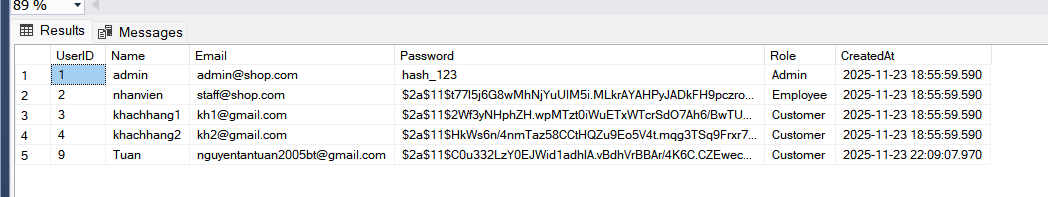
DECLARE @res INT;

EXEC sp\_ChangeUserPassword 9999, '123', @res OUTPUT;

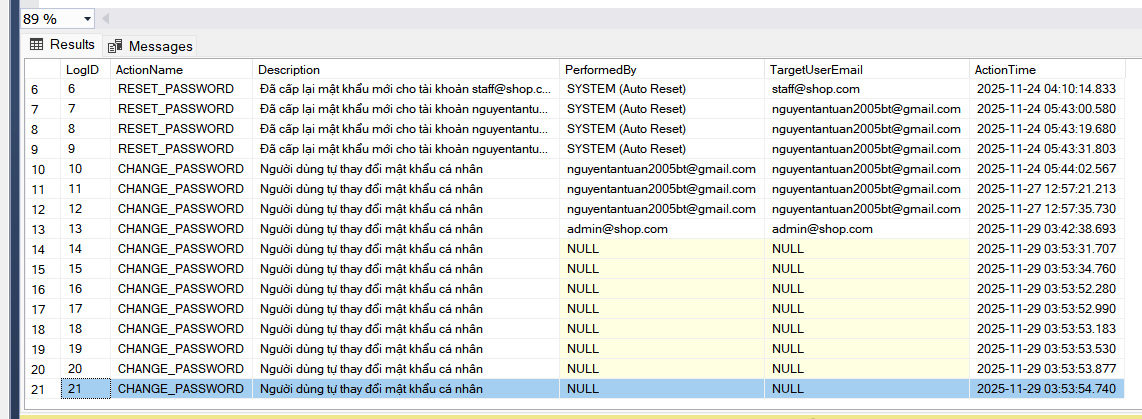
SELECT @res;

****

Output = 0, chạy thành công



Không có UserID = 999, không cập nhật bảng Users

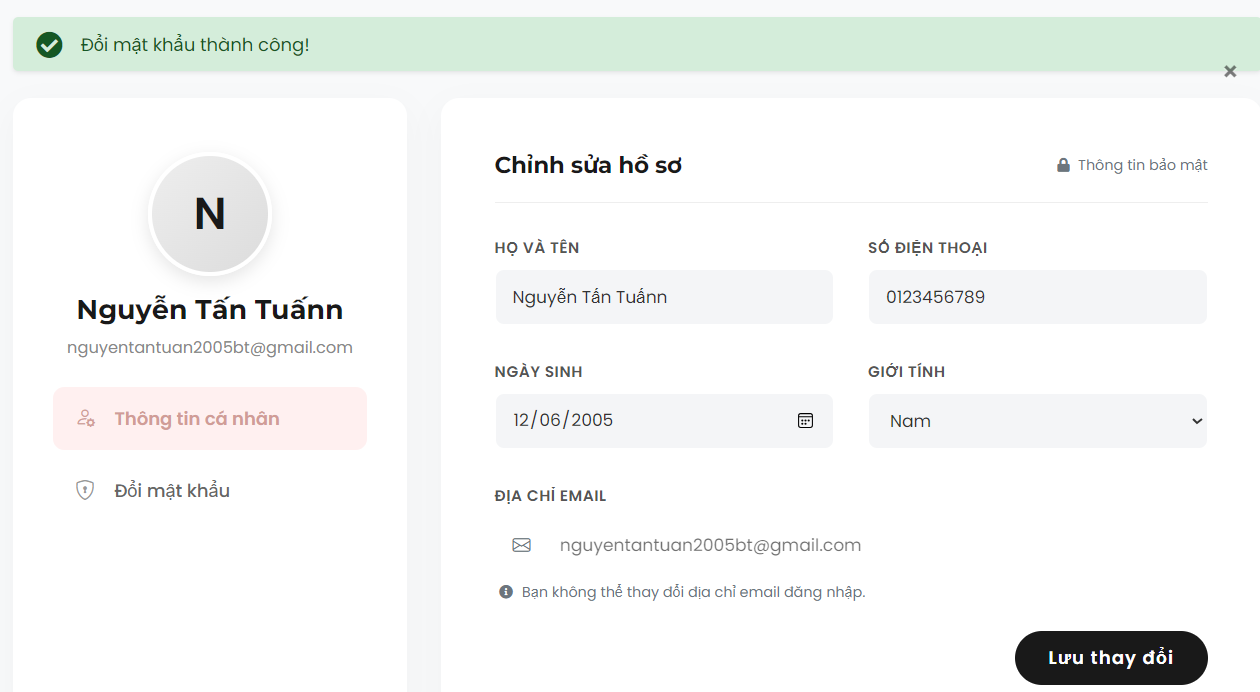


AuditLog báo null

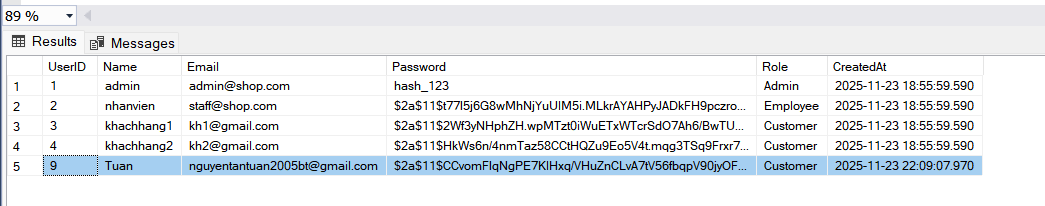
**Test 3 – đổi mật khẩu từ ứng dụng**



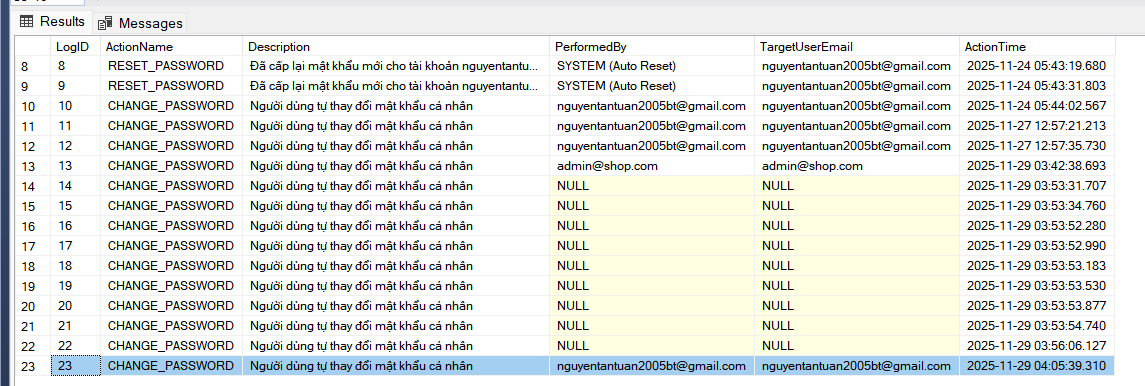
Đăng nhập tài khoản khách hàng và truy cập vào mục đổi mật khẩu, nhập mật khẩu hiện tại và xác nhận mật khẩu mới



Hệ thống xác nhận đổi mật khẩu thành công



Bảng user cập nhật mật khẩu mới dã Hash



Bảng AuditLog lưu 1 bản ghi mói

**[dbo].[sp\_LoginUser]**

CREATE PROCEDURE sp\_LoginUser

@Email NVARCHAR(100),

@Result INT OUTPUT,

@UserID INT OUTPUT,

@Name NVARCHAR(100) OUTPUT,

@Role NVARCHAR(50) OUTPUT,

@Password NVARCHAR(255) OUTPUT

AS

BEGIN

SET NOCOUNT ON;

SELECT

@UserID = UserID,

@Name = Name,

@Role = Role,

@Password = Password

FROM Users

WHERE Email = @Email;

IF @UserID IS NULL

BEGIN

SET @Result = 1;

RETURN;

END

SET @Result = 0;

END

* **Chức năng:** sp\_LoginUser là Stored Procedure thực hiện kiểm tra thông tin đăng nhập dựa trên email người dùng. Bao gồm: Kiểm tra sự tồn tại của Email trong bảng Users. Nếu có, trả về các thông tin liên quan: UserID, Name, Role, Password (đã băm).

Ứng dụng C# sẽ tự đối chiếu mật khẩu người dùng nhập với mật khẩu hash trả về.

Trả kết quả qua biến OUTPUT:

* + 0 → Tìm thấy user
  + 1 → Không tìm thấy email
  + -1 → Lỗi
* **Cách hoạt động:**

Bước 1: Nhận tham số đầu vào

* @Email → email người dùng nhập.

**Bước 2: Truy vấn bảng Users**

SELECT

@UserID = UserID,

@Name = Name,

@Role = Role,

@Password = Password

FROM Users

WHERE Email = @Email;

**Bước 3: Kiểm tra có tìm thấy User hay không**

* Nếu @UserID IS NULL → không tồn tại email → trả về @Result = 1.

**Bước 4: Trả dữ liệu khi tìm thấy**

* Gán @Result = 0.
* Trả về các thông tin:  
  UserID, Name, Role, Password (hash để C# verify).
* **Kiểm thử:**

**Test 1 – Email hợp lệ:**

DECLARE @res INT, @id INT, @name NVARCHAR(100), @role NVARCHAR(50), @pass NVARCHAR(255);

EXEC sp\_LoginUser

'nguyentantuan2005bt@gmail.com',

@res OUTPUT,

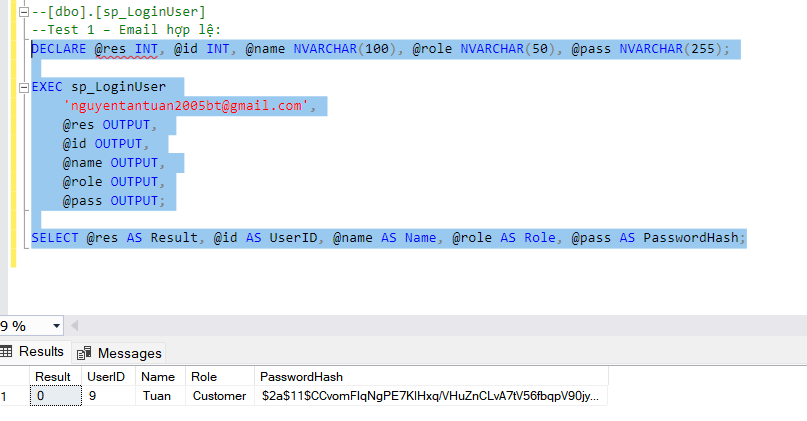
@id OUTPUT,

@name OUTPUT,

@role OUTPUT,

@pass OUTPUT;

SELECT @res AS Result, @id AS UserID, @name AS Name, @role AS Role, @pass AS PasswordHash;

****

Trả về kết quả Output = 0 tìm thấy User với UserID = 9 với Email là [nguyentantuan2005bt@gmail.com](mailto:nguyentantuan2005bt@gmail.com)

**Test 2 – Email không tồn tại**

DECLARE @res INT, @id INT, @name NVARCHAR(100), @role NVARCHAR(50), @pass NVARCHAR(255);

EXEC sp\_LoginUser

'abc@xyz.com',

@res OUTPUT,

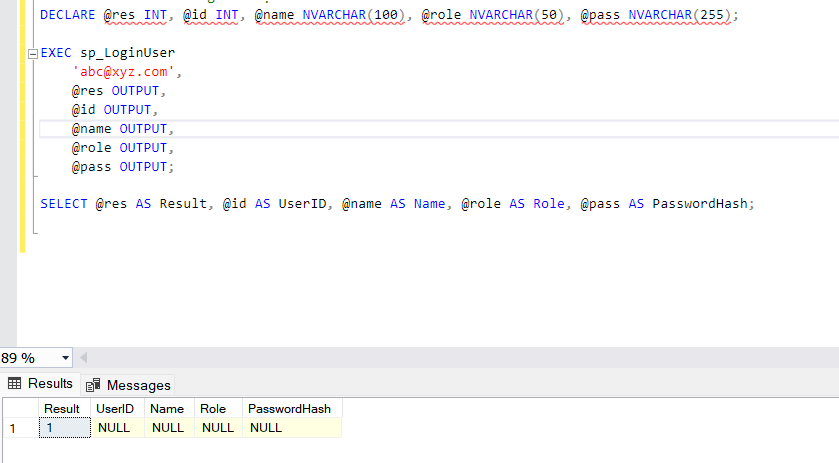
@id OUTPUT,

@name OUTPUT,

@role OUTPUT,

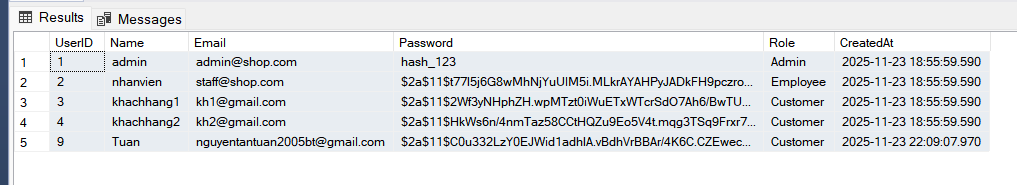
@pass OUTPUT;

SELECT @res AS Result, @id AS UserID, @name AS Name, @role AS Role, @pass AS PasswordHash;

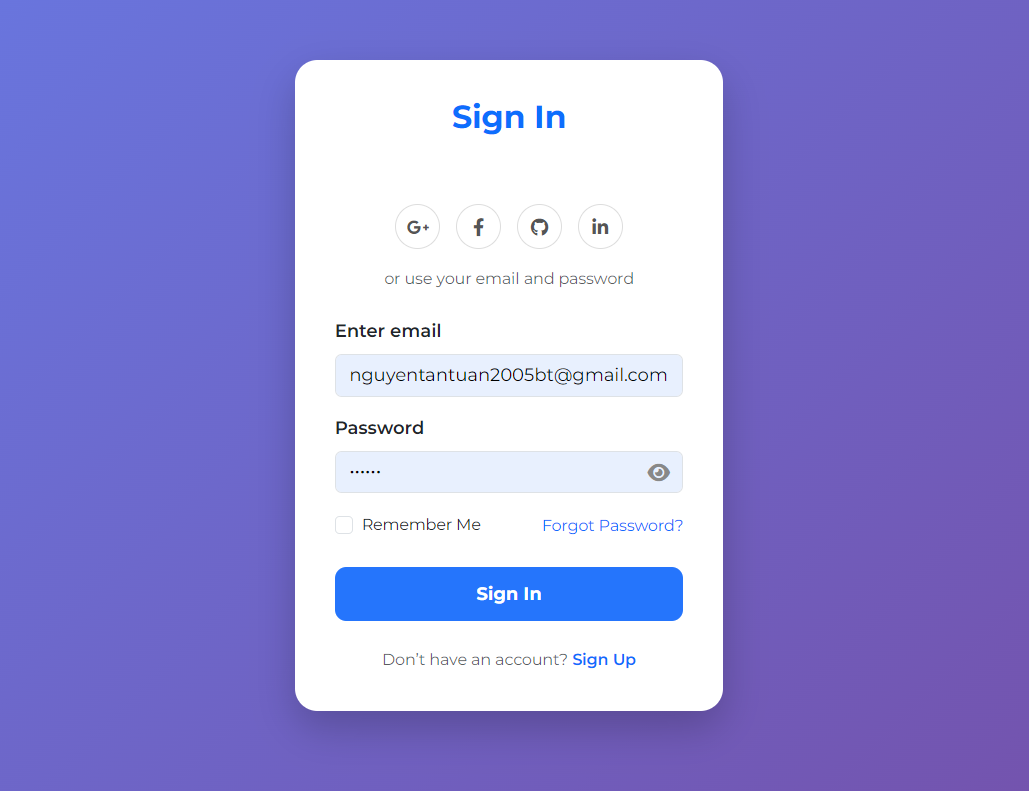


Trả về kết quả Output = 1 thể hiện Email không tồn tại

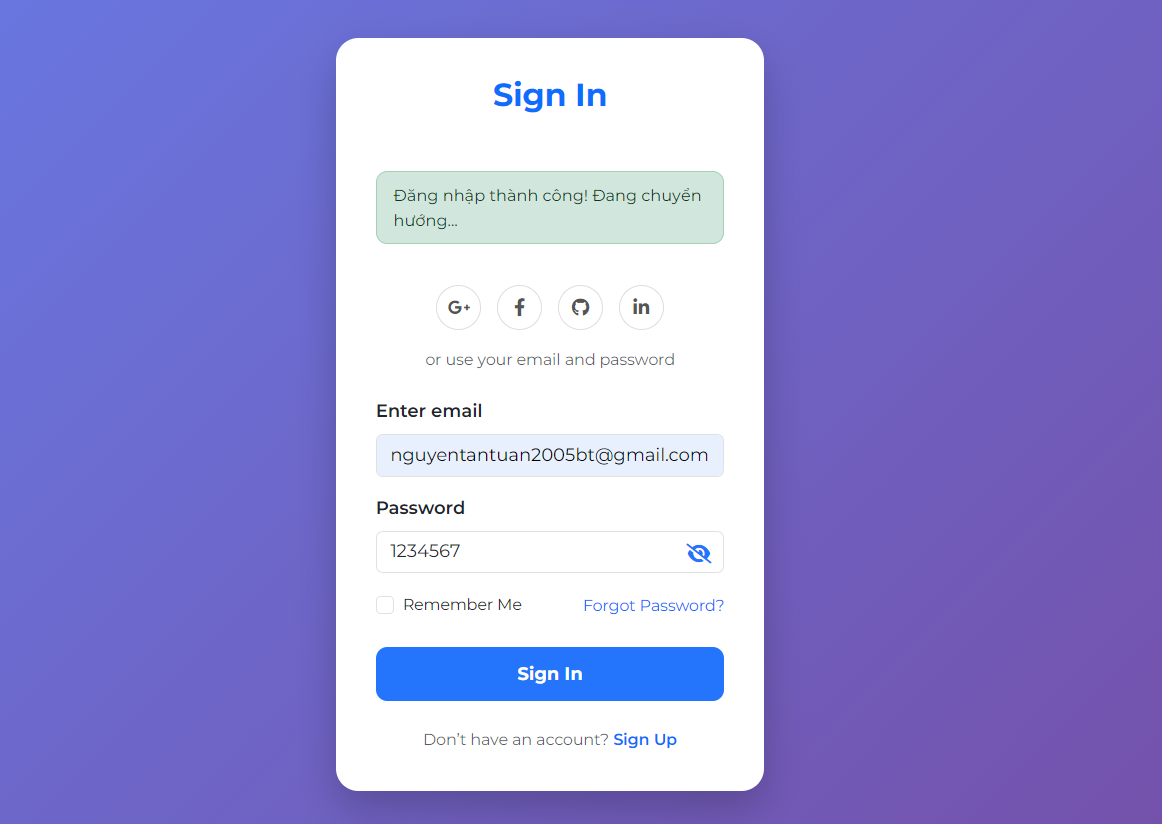
**Test 3 – Kiểm tra đăng nhập từ ứng dụng**

****

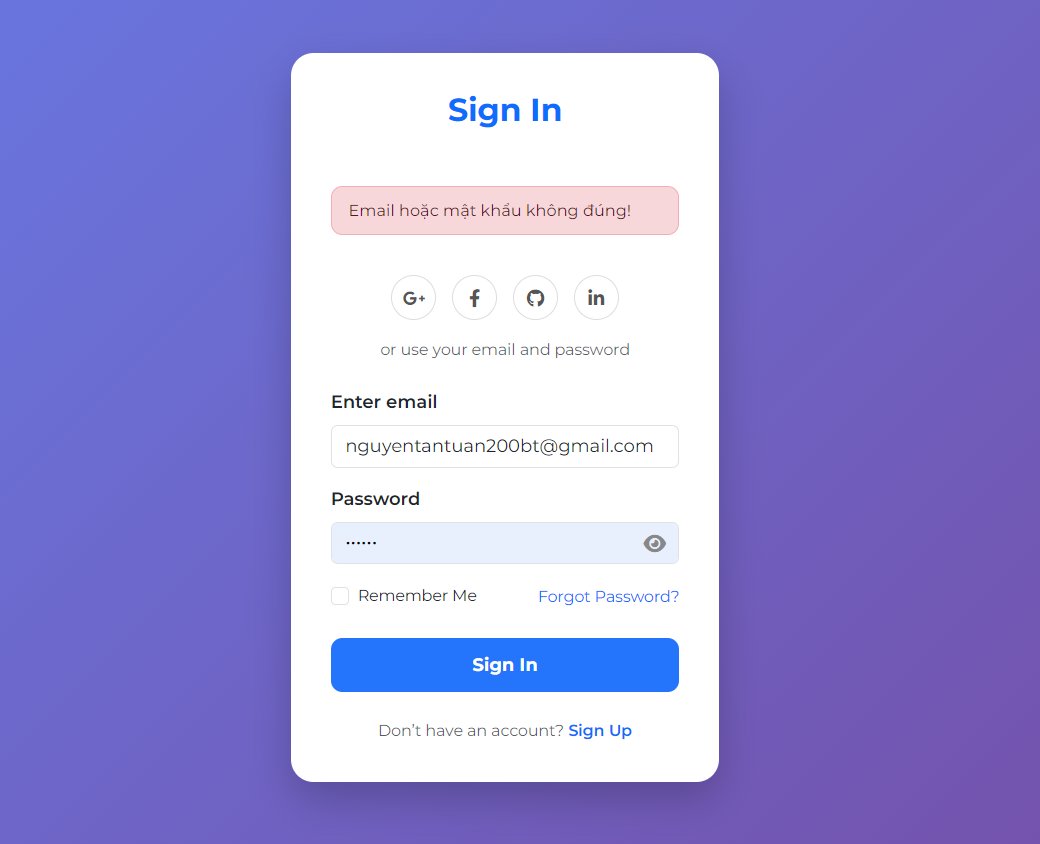
Thông tin từ bảng Users

****

Nhập các thông tin Email, Password vào trong ô và nhấn nút Sign In



Chuyển hướng đến trang chủ của ứng dụng nếu nhập đúng thông tin



Báo lỗi khi nhập sai thông tin

**[dbo].[sp\_RegisterUserWithSeparateName]**

CREATE PROCEDURE sp\_RegisterUserWithSeparateName

@CustomerName NVARCHAR(100),

@UserName NVARCHAR(100),

@Email NVARCHAR(100),

@PasswordHash NVARCHAR(255),

@Role NVARCHAR(50) = 'Customer',

@SoDT NVARCHAR(15) = NULL,

@BirthDayKH DATETIME = NULL,

@GioiTinhKH NVARCHAR(10) = NULL,

@Result INT OUTPUT,

@UserID INT OUTPUT,

@MaKH INT OUTPUT

AS

BEGIN

SET NOCOUNT ON;

IF EXISTS (SELECT 1 FROM Users WHERE Email = @Email OR Name = @UserName)

BEGIN

SET @Result = 1;

SET @UserID = NULL;

SET @MaKH = NULL;

RETURN;

END

BEGIN TRY

BEGIN TRANSACTION;

INSERT INTO Users (Name, Email, Role, Password, CreatedAt)

VALUES (@UserName, @Email, @Role, @PasswordHash, GETDATE());

SET @UserID = SCOPE\_IDENTITY();

INSERT INTO KhachHang (TenKH, Email, SoDT, BirthDayKH, GioiTinhKH, UserId)

VALUES (@CustomerName, @Email, @SoDT, @BirthDayKH, @GioiTinhKH, @UserID);

SET @MaKH = SCOPE\_IDENTITY();

COMMIT TRANSACTION;

SET @Result = 0;

END TRY

BEGIN CATCH

IF @@TRANCOUNT > 0

ROLLBACK TRANSACTION;

SET @Result = -1;

SET @UserID = NULL;

SET @MaKH = NULL;

THROW;

END CATCH

END

* **Chức năng**: sp\_RegisterUserWithSeparateName là Stored Procedure dùng để đăng ký khách hàng mới, đồng thời tạo bản ghi trong cả hai bảng:

Users → chứa thông tin tài khoản đăng nhập

KhachHang → chứa thông tin hồ sơ cá nhân khách hàng

Thủ tục giúp đảm bảo dữ liệu đồng bộ giữa hai bảng và hỗ trợ hệ thống quản lý user + thông tin khách hàng.

* **Cách hoạt động**:

**Kiểm tra trùng lặp**

Nếu Email hoặc UserName đã tồn tại → dừng lại, trả về @Result = 1.

**Bắt đầu Transaction**

INSERT vào bảng **Users**, lấy @UserID bằng SCOPE\_IDENTITY()

INSERT vào bảng **KhachHang**, lấy @MaKH.

**Commit Transaction** nếu cả hai bảng đều thêm thành công.

**Catch lỗi**

Nếu xảy ra lỗi (như ràng buộc, kiểu dữ liệu, độ dài…)  
→ rollback toàn bộ và trả về @Result = -1.

* **Kiểm thử:**

**Test 1 – Đăng ký hợp lệ**

DECLARE @Out\_Result INT;

DECLARE @Out\_UserID INT;

DECLARE @Out\_MaKH INT;

EXEC sp\_RegisterUserWithSeparateName

@CustomerName = N'Nguyễn Văn A',

@UserName = N'user\_test\_02',

@Email = N'test001@email.com',

@PasswordHash = N'matkhau123',

@SoDT = N'0901234567',

@GioiTinhKH = N'Nam',

@Result = @Out\_Result OUTPUT,

@UserID = @Out\_UserID OUTPUT,

@MaKH = @Out\_MaKH OUTPUT;

SELECT

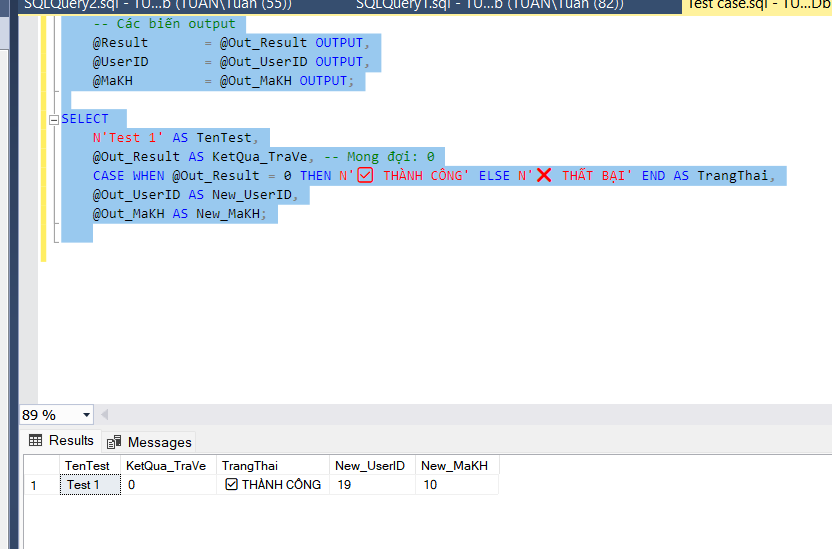
N'Test 1' AS TenTest,

@Out\_Result AS KetQua\_TraVe,

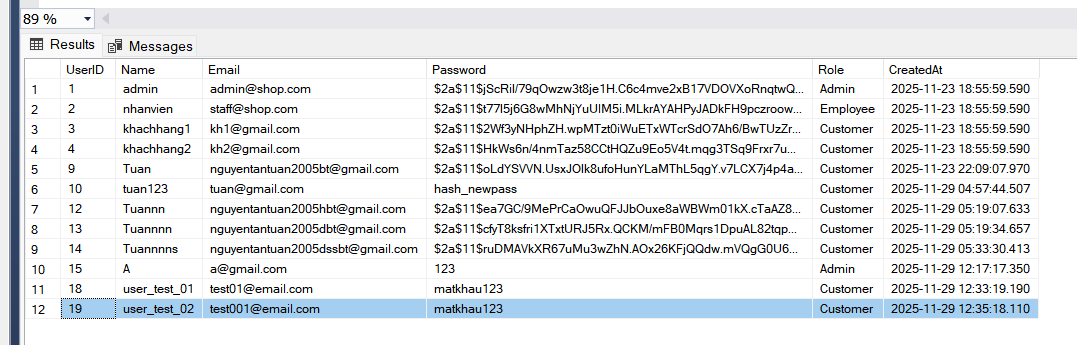
CASE WHEN @Out\_Result = 0 THEN N'✅ THÀNH CÔNG' ELSE N'❌ THẤT BẠI' END AS TrangThai,

@Out\_UserID AS New\_UserID,

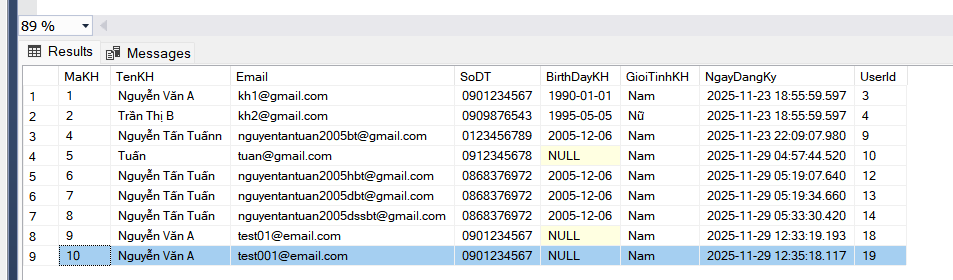
@Out\_MaKH AS New\_MaKH;



Kết quả trả về 0 thể hiện đăng ký thành công



Bảng Users thêm 1 bản ghi mới



Bảng KhachHang lưu thông tin

**Test 2 – Email trùng trong bảng Users**

DECLARE @Out\_Result INT;

DECLARE @Out\_UserID INT;

DECLARE @Out\_MaKH INT;

SET @Out\_Result = NULL; SET @Out\_UserID = NULL; SET @Out\_MaKH = NULL;

EXEC sp\_RegisterUserWithSeparateName

@CustomerName = N'Nguyễn Văn B (Clone)',

@UserName = N'user\_test\_02\_khac',

@Email = N'test01@email.com',

@PasswordHash = N'matkhau123',

@Result = @Out\_Result OUTPUT,

@UserID = @Out\_UserID OUTPUT,

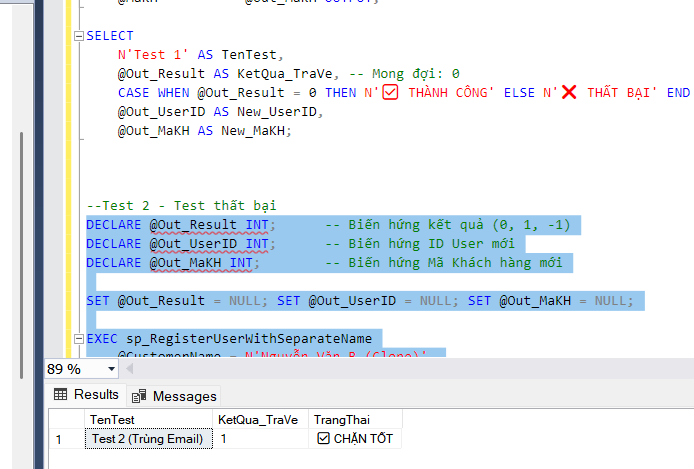
@MaKH = @Out\_MaKH OUTPUT;

SELECT

N'Test 2 (Trùng Email)' AS TenTest,

@Out\_Result AS KetQua\_TraVe,

CASE WHEN @Out\_Result = 1 THEN N'✅ CHẶN TỐT' ELSE N'❌ KHÔNG CHẶN ĐƯỢC' END AS TrangThai;

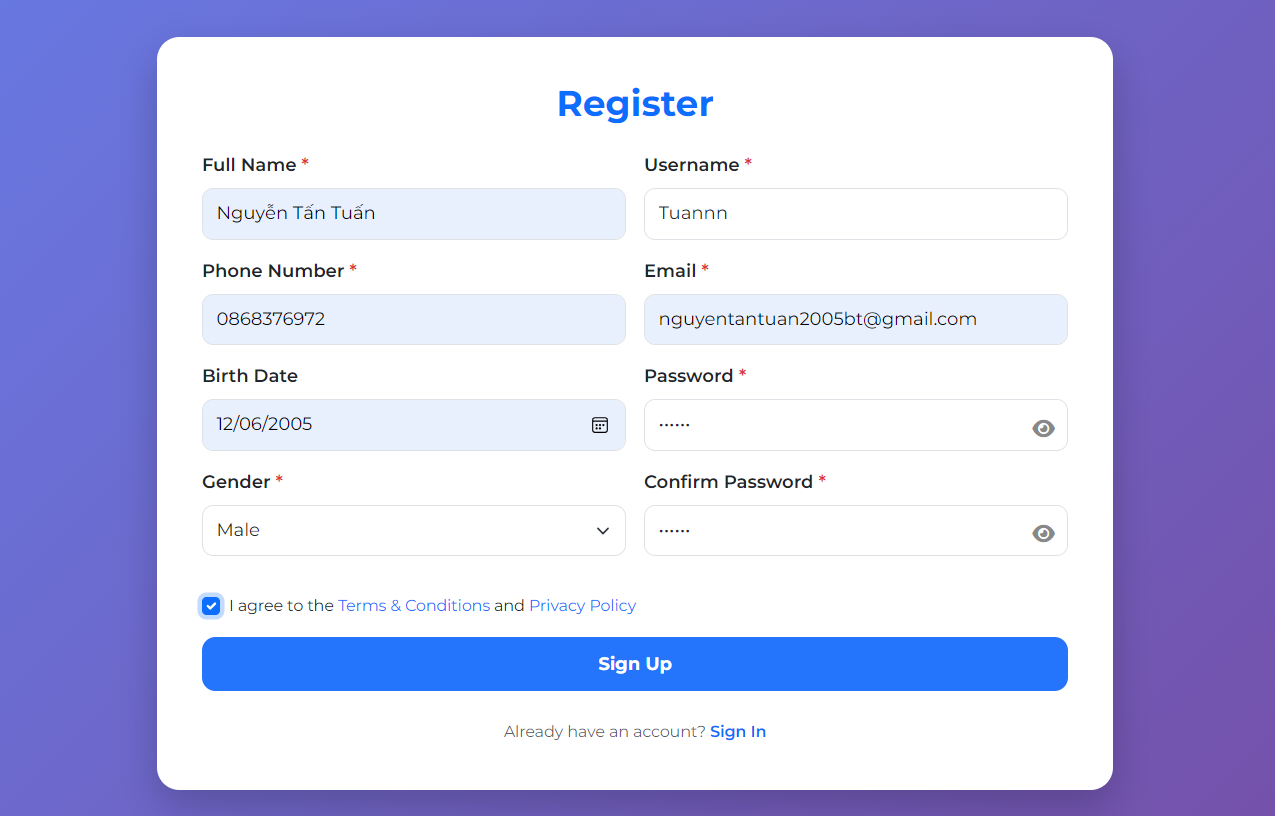


Kết quả trả về 1 Với lỗi trùng với email đã tồn tại

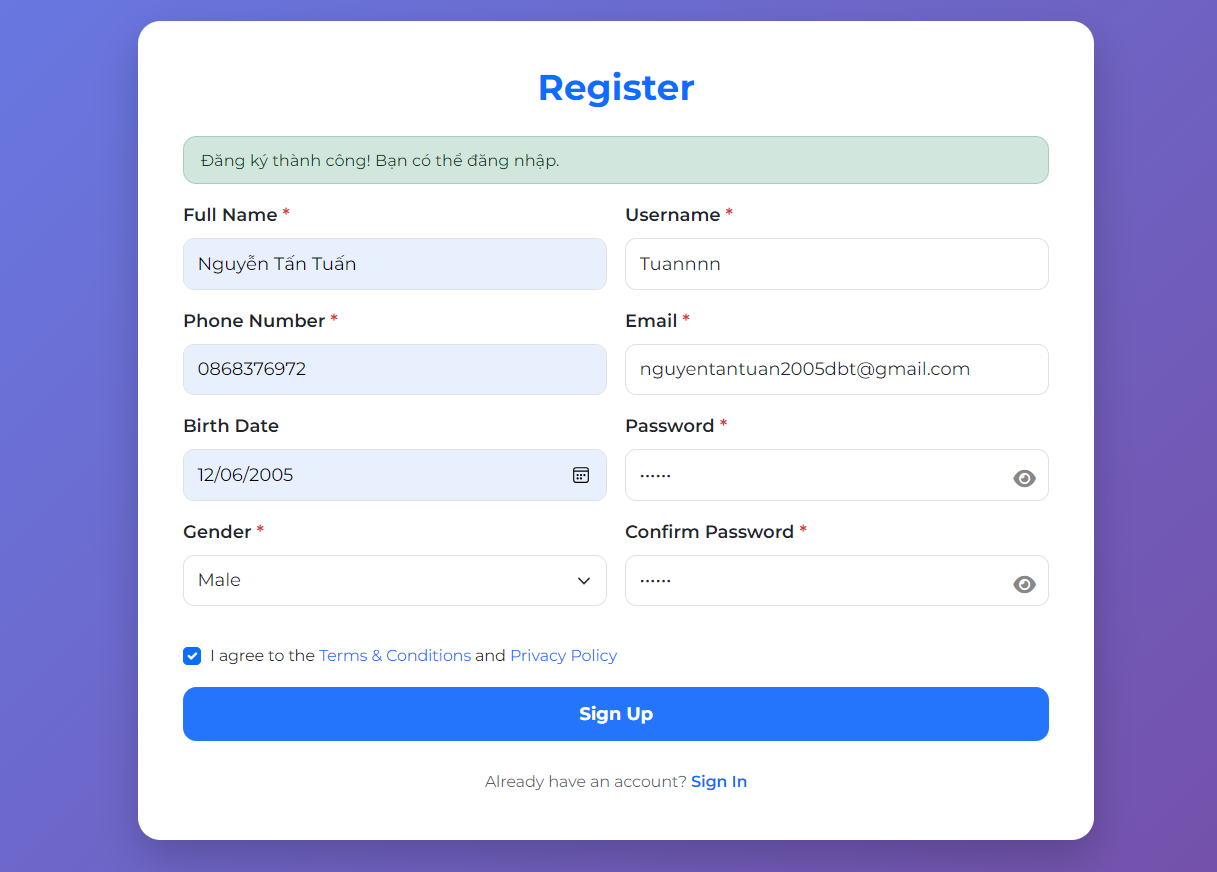
Lệnh ROLLBACK TRANSACTION; sẽ hủy toàn bộ các thao tác đã làm trong transaction

* User không được tạo
* KhachHang không được tạo

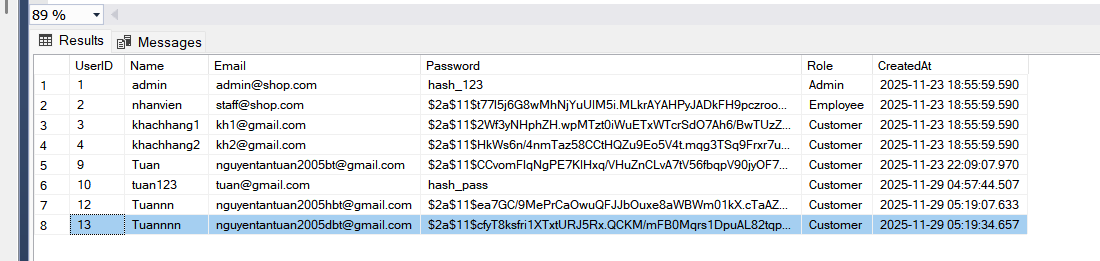
**Test 3 – Đăng ký mới từ ứng dụng**

****

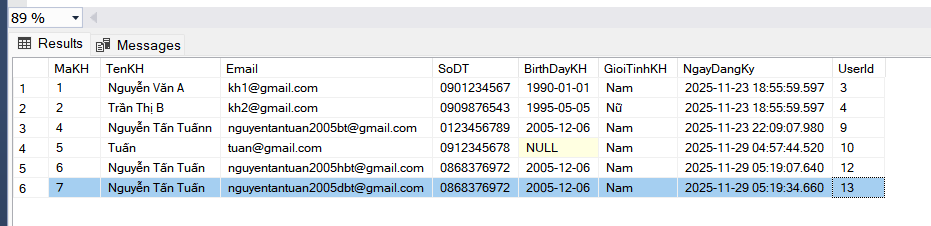
Điền các thông tin vào ô nhập liệu và ấn Sign Up



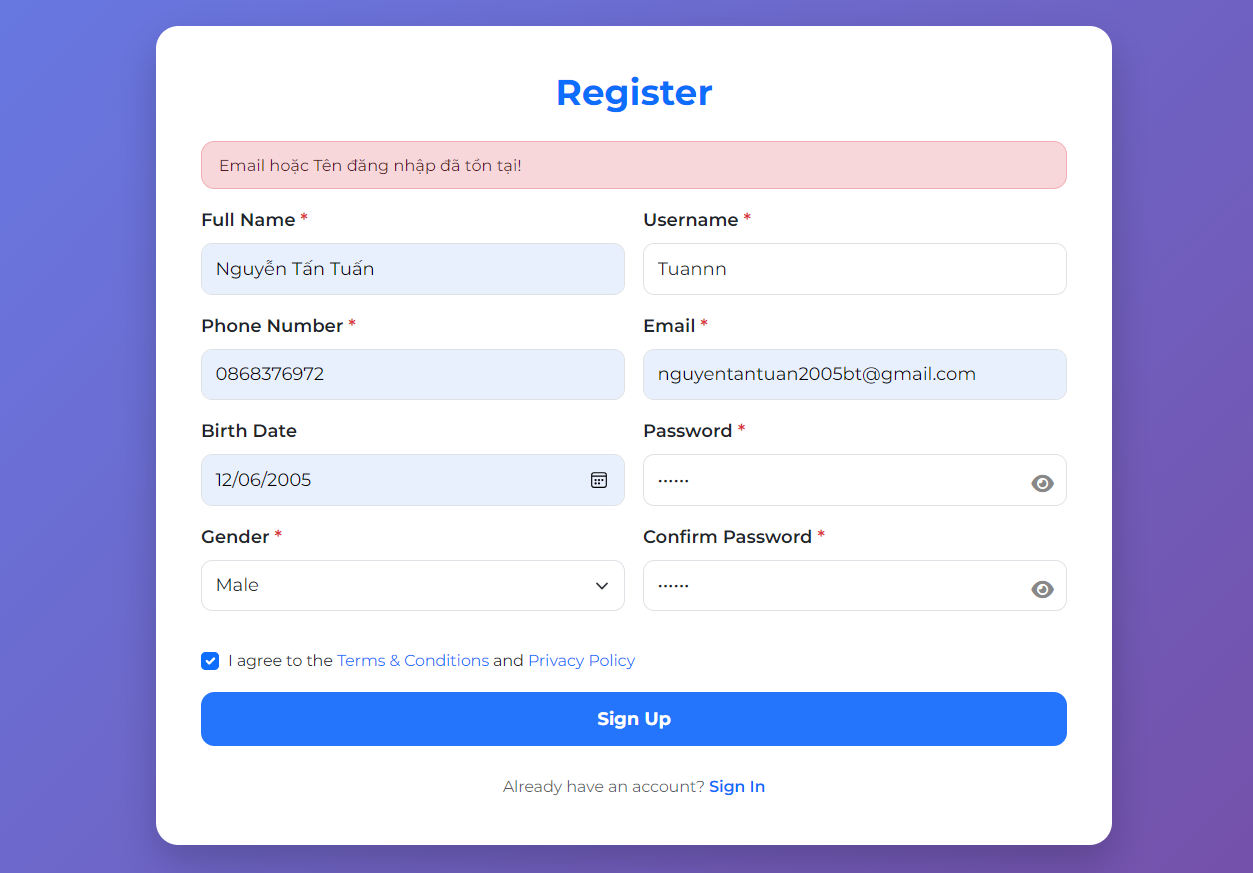
Nếu nhập đúng hệ thống chuyển hướng sang trang đăng nhập đồng thời lưu thông tin vào bảng Users và bảng KhachHang



Lưu thông tin UserName, Email, Password, Role vào bảng Users



Lưu thông tin Full Name, Email, Phone Number, Birth Date, Gender vào bảng KhachHang



Nếu nhập không đúng thông tin hệ thống sẽ báo lỗi

**[dbo].[sp\_ResetUserPassword]**

CREATE PROCEDURE sp\_ResetUserPassword

@TargetEmail NVARCHAR(100),

@NewPassword NVARCHAR(255),

@AdminName NVARCHAR(100),

@Result INT OUTPUT -- 0: Thành công, 1: Không tìm thấy Email, -1: Lỗi

AS

BEGIN

SET NOCOUNT ON;

IF NOT EXISTS (SELECT 1 FROM Users WHERE Email = @TargetEmail)

BEGIN

SET @Result = 1; -- Lỗi: Email không tồn tại

RETURN;

END

BEGIN TRANSACTION;

BEGIN TRY

UPDATE Users

SET Password = @NewPassword

WHERE Email = @TargetEmail;

INSERT INTO AuditLog (ActionName, Description, PerformedBy, TargetUserEmail)

VALUES (

'RESET\_PASSWORD',

N'Đã cấp lại mật khẩu mới cho tài khoản ' + @TargetEmail,

@AdminName,

@TargetEmail

);

COMMIT TRANSACTION;

SET @Result = 0; -- Thành công

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

SET @Result = -1; -- Lỗi hệ thống

END CATCH

END

**Chức năng:** sp\_ResetUserPassword là Stored Procedure dùng để cấp lại mật khẩu cho người dùng.  
Các nhiệm vụ chính:

Kiểm tra email tồn tại trong bảng Users.

Cập nhật mật khẩu mới (đã băm từ phía C#).

Ghi nhật ký hành động (AuditLog).

Email người dùng

Trả kết quả qua biến OUTPUT:

* 0 → Thành công
* 1 → Không tìm thấy email
* -1 → Lỗi hệ thống
* **Cách hoạt động:**

Bước 1 – Kiểm tra Email

IF NOT EXISTS (SELECT 1 FROM Users WHERE Email = @TargetEmail)

* Nếu email không tồn tại → @Result = 1, kết thúc SP.

Bước 2 – Bắt đầu giao dịch

BEGIN TRANSACTION;

* Toàn bộ thao tác UPDATE + INSERT log nằm trong cùng một transaction.

Bước 3 – Cập nhật mật khẩu

UPDATE Users

SET Password = @NewPassword

WHERE Email = @TargetEmail;

Bước 4 – Ghi log

INSERT INTO AuditLog (ActionName, Description, PerformedBy, TargetUserEmail)

VALUES (

'RESET\_PASSWORD',

N'Đã cấp lại mật khẩu mới cho tài khoản ' + @TargetEmail,

@AdminName,

@TargetEmail

);

Bước 5 – Hoàn tất giao dịch

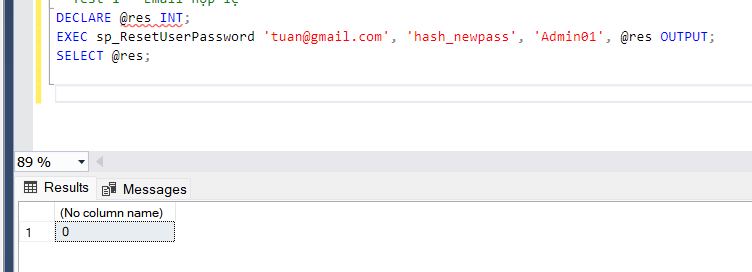
* Nếu không lỗi → COMMIT TRANSACTION, @Result = 0
* Nếu có lỗi → ROLLBACK TRANSACTION, @Result = -1
* **Kiểm thử:**

**Test 1 – Email hợp lệ**

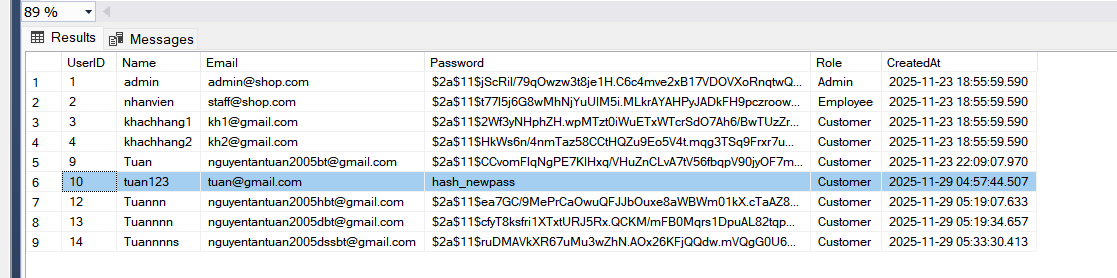
DECLARE @res INT;

EXEC sp\_ResetUserPassword 'tuan@gmail.com', 'hash\_newpass', 'Admin01', @res OUTPUT;

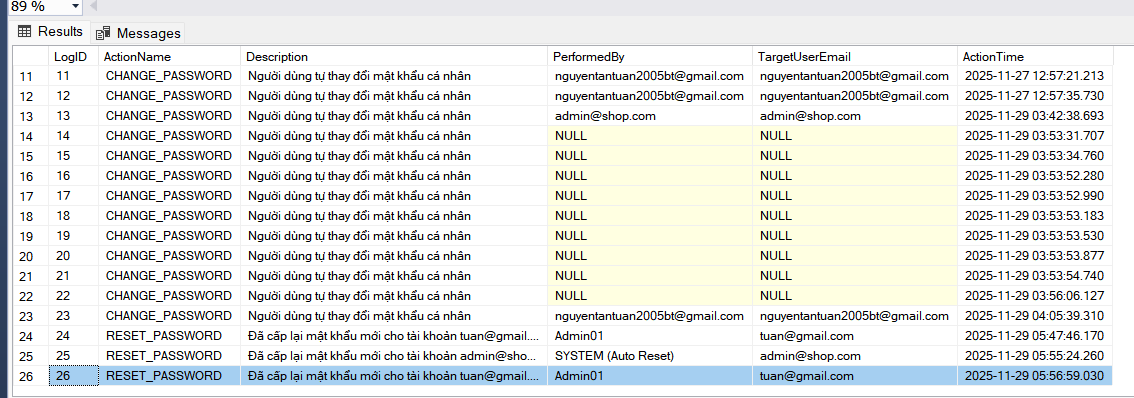
SELECT @res;

****

Kết quả trả về là 0 thể hiện đã thay đổi Password ở tài khoản 'tuan@gmail.com' về hasd\_newpass



Password đã được thay đổi ở bảng Users



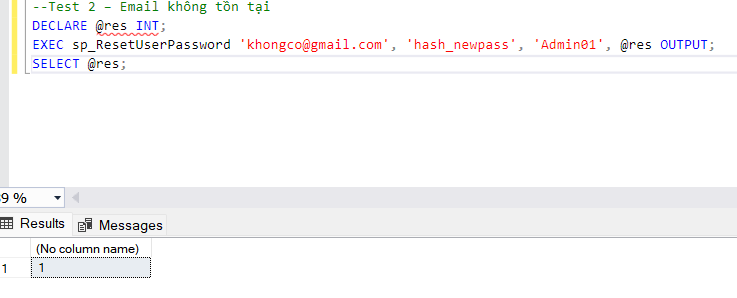
Lưu 1 bản ghi ở AuditLog

**Test 2 – Email không tồn tại**

DECLARE @res INT;

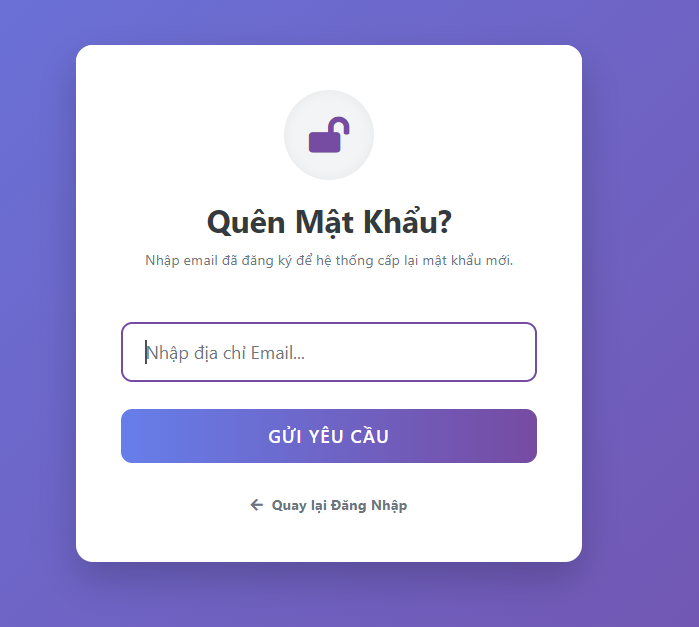
EXEC sp\_ResetUserPassword 'khongco@gmail.com', 'hash\_newpass', 'Admin01', @res OUTPUT;

SELECT @res;

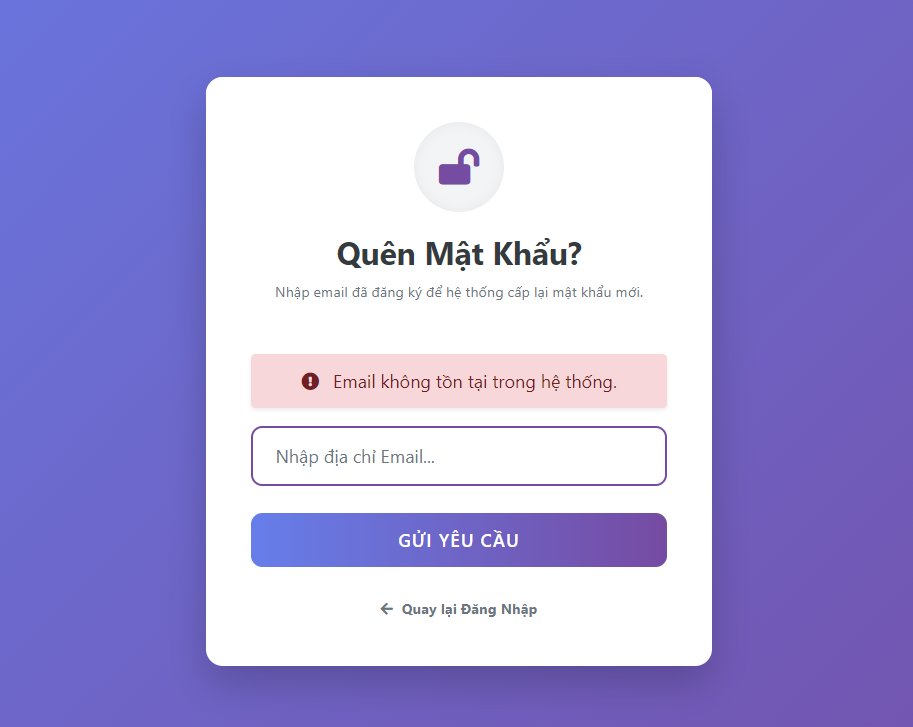
****

Kết quả trả về là 1 thể hiện không tìm thấy Email => Không thay đổi Users, Khôngthêm log

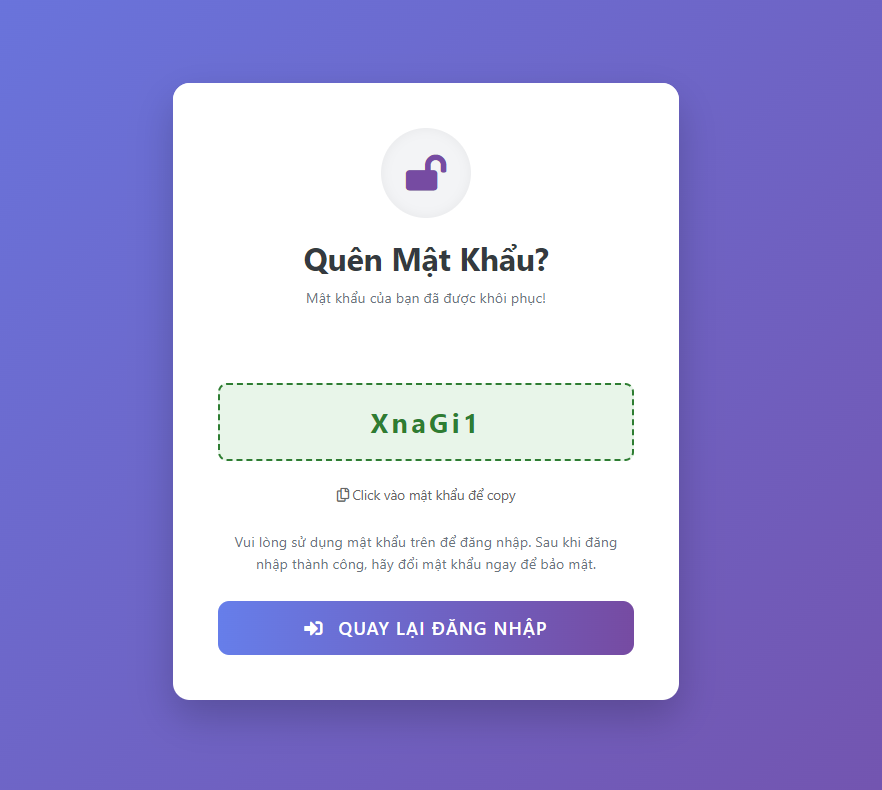
* **Test 3 – Thực hiện từ ứng dụng**

****

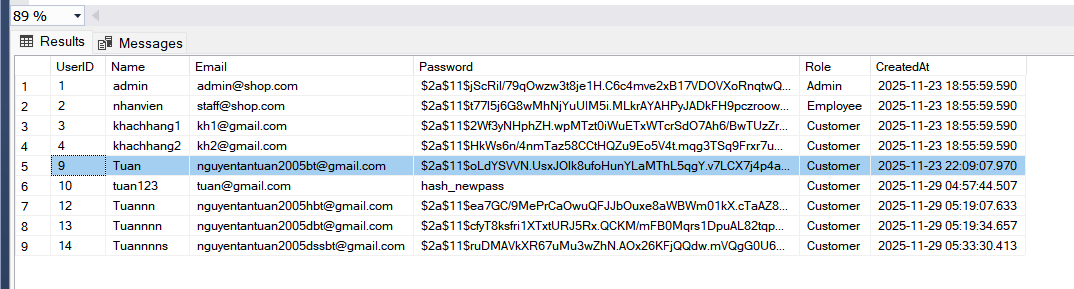
**Nhập Email vào ô nhập liệu để thay đổi mật khẩu**

****

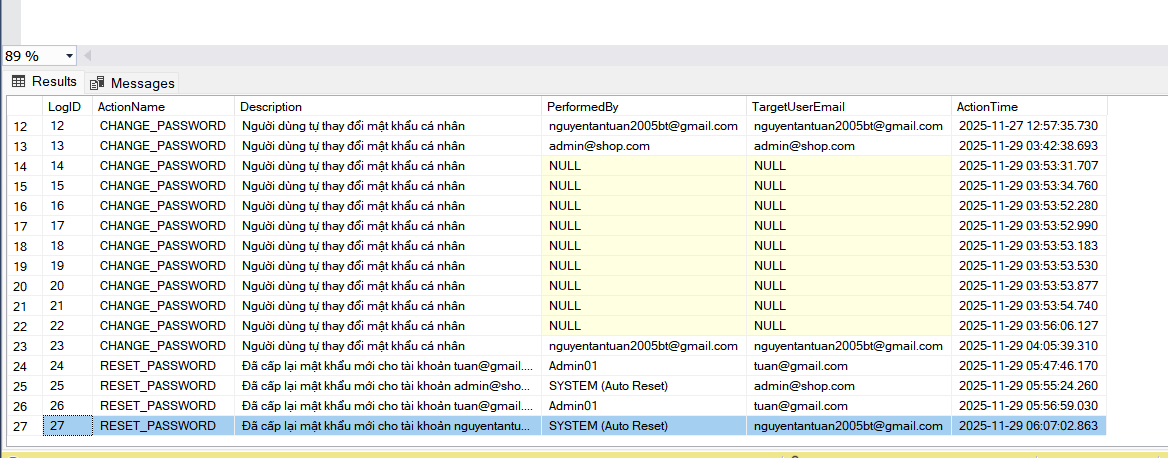
Báo lỗi nếu nhập sai



Nếu nhập đúng thông tin, kết hợp với hàm sinh mật khẩu ngẫu nhiên ở ứng dụng tạo mật khẩu mới cho tài khoản



Cập nhật Password mới ở bảng Users



Thêm bản ghi mới ở AuditLog

**[dbo].[fn\_GetTotalCustomerCount]**

CREATE FUNCTION fn\_GetTotalCustomerCount()

RETURNS INT

AS

BEGIN

DECLARE @CustomerCount INT;

SELECT @CustomerCount = COUNT(MaKH)

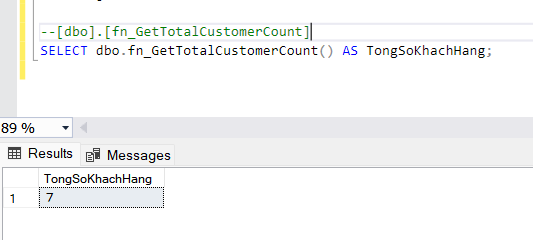
FROM KhachHang;

RETURN @CustomerCount;

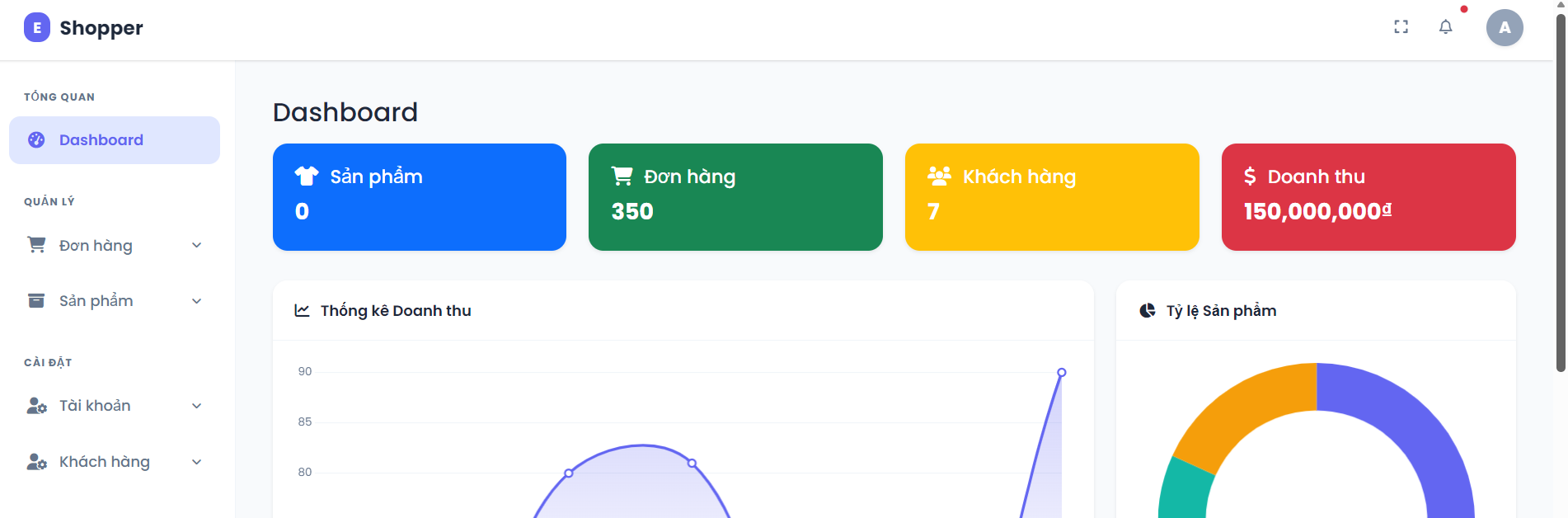
END;

* **Chức năng**: fn\_GetTotalCustomerCount là Function **trả về tổng số khách hàng** hiện có trong bảng KhachHang
* **Cách hoạt động**:
* Khai báo biến @CustomerCount để lưu số lượng khách hàng.
* Dùng COUNT(MaKH) trên bảng KhachHang để đếm tổng số dòng (Mã khách hàng).
* Gán kết quả vào @CustomerCount.
* RETURN @CustomerCount → trả về tổng số
* **Kiểm thử:**

SELECT dbo.fn\_GetTotalCustomerCount() AS TongSoKhachHang;



Trả về kết quả của Function



Hiển thị số lượng khách hàng ở Dashboard

[**tr\_LimitAdminCount]**

CREATE TRIGGER tr\_LimitAdminCount

ON Users

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

IF EXISTS (SELECT 1 FROM inserted WHERE Role = 'Admin')

BEGIN

DECLARE @CurrentAdminCount INT;

SELECT @CurrentAdminCount = COUNT(UserID) FROM Users WHERE Role = 'Admin';

IF @CurrentAdminCount > 2

BEGIN

ROLLBACK TRANSACTION;

THROW 51000, N'Không thể tạo Admin mới. Hệ thống chỉ cho phép tối đa 2 quản trị viên (Admin).', 1;

RETURN;

END

END

END

* **Chức năng:** Trigger tr\_LimitAdminCount dùng để giới hạn số lượng tài khoản Admin tối đa trong hệ thống.

Mỗi khi có bản ghi INSERT vào bảng Users, trigger sẽ kiểm tra xem tài khoản mới có phải Admin không.

Nếu tổng số Admin vượt quá 2, trigger sẽ:

* Hủy bỏ thao tác INSERT
* THROW lỗi để phía ứng dụng (C#, Entity Framework) nhận biết

Mục đích: Bảo vệ hệ thống, tránh việc tạo quá nhiều tài khoản quản **trị.**

* **Cách hoạt động:**

Trigger chạy **AFTER INSERT**, tức là chỉ kích hoạt sau khi có lệnh INSERT vào bảng Users.

Kiểm tra dữ liệu trong bảng **inserted** (dữ liệu vừa được thêm).

IF EXISTS (SELECT 1 FROM inserted WHERE Role = 'Admin')

Đếm toàn bộ số Admin hiện có:

SELECT @CurrentAdminCount = COUNT(UserID) FROM Users WHERE Role = 'Admin';

Nếu số Admin > 2:

ROLLBACK TRANSACTION → hủy việc thêm Admin

Gửi lỗi về ứng dụng:

THROW 51000, N'Không thể tạo Admin mới. Hệ thống chỉ cho phép tối đa 2 quản trị viên (Admin).', 1;

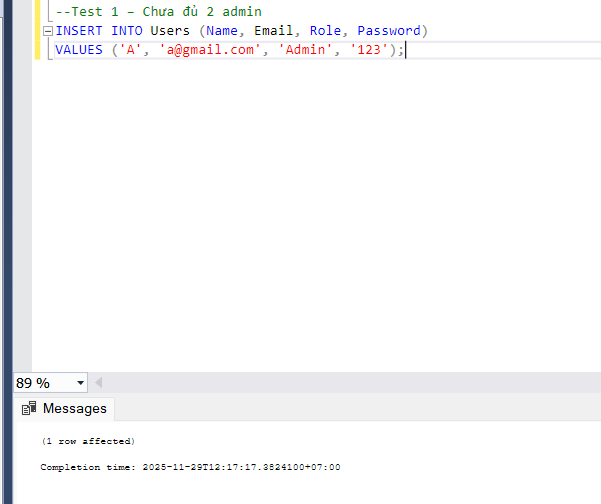
Nhờ đó, không thể thêm quá 2 admin vào hệ thống.

* **Kiểm thử:**

**Test 1 – Chưa đủ 2 admin**

INSERT INTO Users (Name, Email, Role, Password)

VALUES ('A', 'a@gmail.com', 'Admin', '123');

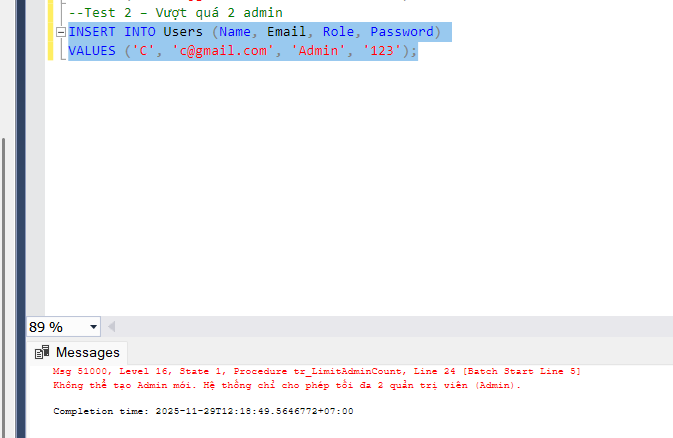


Thêm thành công Admin mới

**Test 2 – Vượt quá 2 admin**

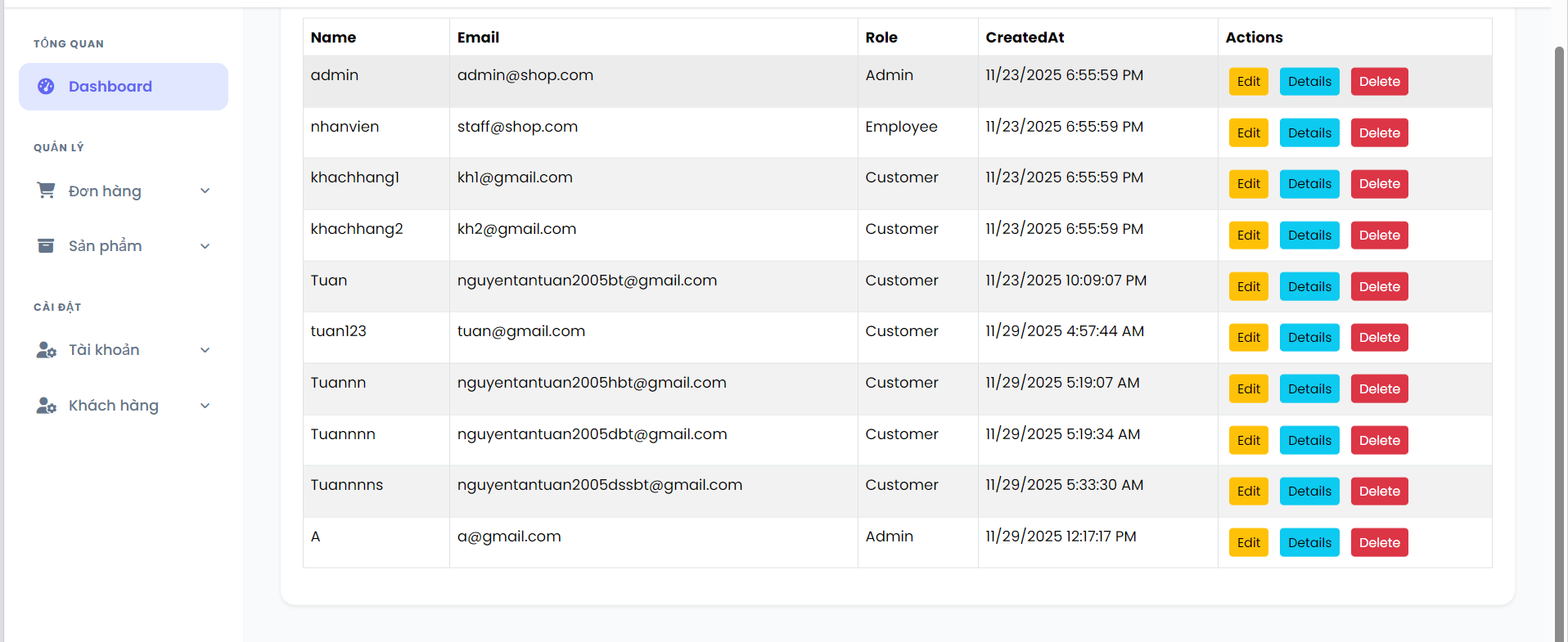
INSERT INTO Users (Name, Email, Role, Password)

VALUES ('C', 'c@gmail.com', 'Admin', '123');

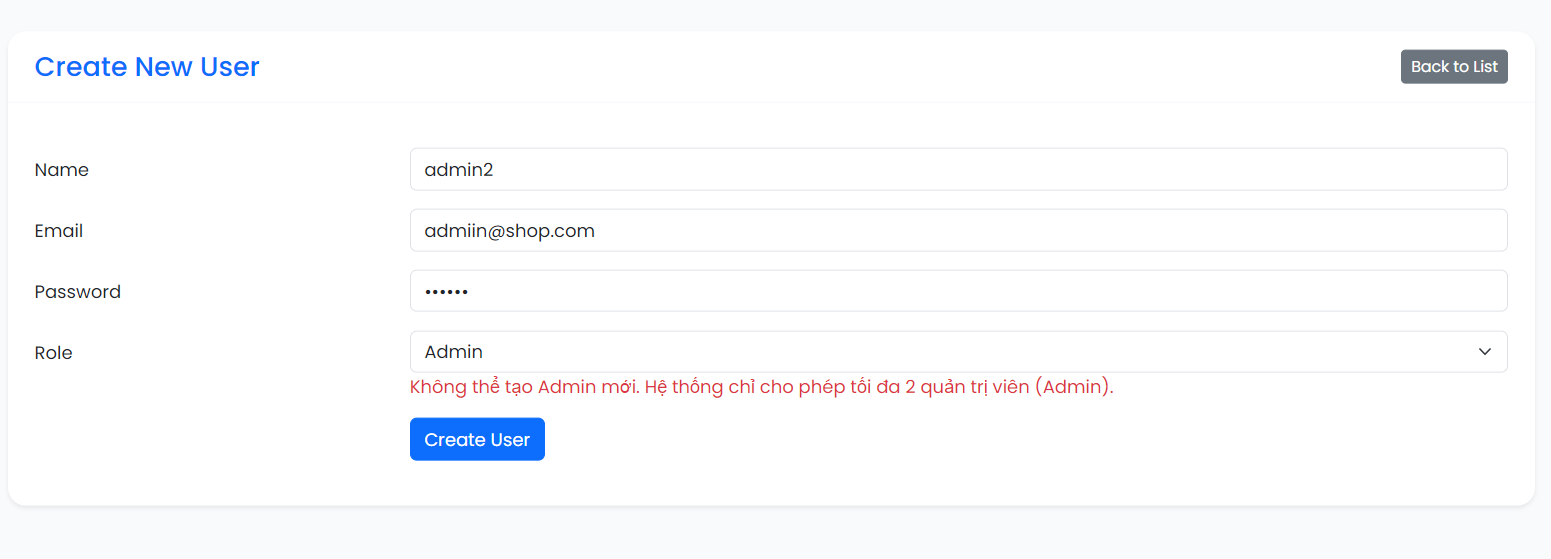
****

Báo lỗi khi vượt quá 2 tài khoản Admin trong bảng Users

**Test 3 – Thêm mới từ ứng dụng**

****

Danh sách tài khoản hiện có, đã tồn tại 2 tài khoản với role là Admin



Hệ thống báo lỗi ngay khi tạo mới

## 3.4. Nhom\_XuLyDonHang

## 3.5. Nhom\_QuanLyKH

### 3.5.1. Mô tả ứng dụng:

Ứng dụng được phát triển là hệ thống quản lý tài khoản người dùng được xây dựng trên nền tảng ASP.NET(MVC ) và SQL Server. Hệ thống quản lý danh sách thông tin khách hàng, giỏ hảng và đánh giá sản phẩm và đơn hàng của khách hàng. Ở quản lý khách hàng có đầy đủ chức năng CRUD của khách hàng, giỏ hàng, đánh giá sản phẩm. Ở đơn hàng và chi tiết đơn hàng chỉ có chức năng tạo cho khách hàng. Vói giao diện tối giản, dễ dùng.

### 3.3.2. Các đối tượng xử lý và điều khiển:

**Những hàm store procedure của khách hàng:**

**- Thêm khách hàng:**

Create Procedure sp\_AddKhachHang

@TenKH nvarchar(100),

@Email nvarchar(100),

@SoDT nvarchar(15),

@BirthDayKH Date,

@GioiTinhKH nvarchar(10),

@UserId int

As

Begin

Set NoCount On;

Begin Try

-- Kiểm tra email có tồn tại hay không --

If Exists(Select 1 From KhachHang Where Email = @Email)

Begin

Select 'Lỗi' As Status, 'Email này đã tồn tại' As Message

Return

End

End Try

End;

Go

* **Xóa khách hàng:**

CREATE PROCEDURE [dbo].[sp\_DeleteCustomer]

@MaKH INT

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

-- Kiểm tra khách hàng tồn tại

IF NOT EXISTS (SELECT 1 FROM KhachHang WHERE MaKH = @MaKH)

BEGIN

SELECT 'ERROR' AS Status, N'Khách hàng không tồn tại!' AS Message;

RETURN;

END

-- Kiểm tra có đơn hàng còn hoạt động không

IF EXISTS (SELECT 1 FROM Orders WHERE MaKH = @MaKH AND IsCancel = 0)

BEGIN

SELECT 'ERROR' AS Status, N' Không thể xóa! Khách hàng còn đơn hàng đang hoạt động.' AS Message;

RETURN;

END

-- Xóa khách hàng (CASCADE sẽ xóa các dữ liệu liên quan)

DELETE FROM KhachHang WHERE MaKH = @MaKH;

SELECT 'SUCCESS' AS Status, N'Xóa khách hàng thành công!' AS Message;

END TRY

BEGIN CATCH

SELECT 'ERROR' AS Status, ERROR\_MESSAGE() AS Message;

END CATCH

END;

GO

* **Cập nhật khách hàng**

Create Procedure sp\_UpdateKhachHang

@MaKH int,

@TenKH nvarchar(100),

@Email nvarchar(100),

@SoDT nvarchar(100),

@BirthDayKH Date,

@GioiTinhKH nvarchar(10)

As

Begin

Set NoCount On;

Begin Try

If Not Exists (Select 1 From KhachHang Where MaKH = @MaKH)

Begin

Select 'Lỗi!' as Status, N'Khách hàng đã tồn tại!' as Message

Return

End

If @Email is not Null and @Email <> (Select Email From KhachHang Where MaKH = @MaKH)

Begin

Select 'Lỗi!' as Status, N'Email đã tồn tại!' as Message

Return

End

Update KhachHang

Set

TenKH = isNull(@TenKH, TenKH),

Email = isNull(@Email, Email),

SoDT = isNull(@SoDT, SoDT),

BirthDayKH = isNUll(@BirthDayKH, BirthDayKH),

GioiTinhKH = isNull(@GioiTinhKH, GioiTinhKH)

Where MaKH = MaKH

Select 'Đã thêm thộng tin khách hàng thành công ' as Status,N'Cập nhật khách hàng thành công' as Message

End Try

Begin Catch

Select 'Lỗi' as Status, ERROR\_MESSAGE() as Message

End Catch

End

Go

* **Lâý mã khách hàng để tìm:**

Create Procedure [dbo].[ sp\_GetKhachHangBangID]

@MaKH int

As Begin

Set NoCount On

Begin Try

Select\*From KhachHang

Where MaKH = @MaKH

If @@Rowcount =0

Begin

Select 'Lỗi!' as Status, N'Khách hàng tồn tại!' as Message

End

End Try

Begin Catch

Select 'Lỗi' as Status , ERROR\_MESSAGE() as Message

End Catch

End

Go

* **Lấy danh sách tất cả các khách hàng:**

CREATE PROCEDURE[dbo].[ sp\_GetAllCustomers]

@PageNumber INT = 1,

@PageSize INT = 10

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

DECLARE @Offset INT = (@PageNumber - 1) \* @PageSize;

-- Lấy danh sách khách hàng

SELECT

MaKH,

TenKH,

Email,

SoDT,

BirthDayKH,

GioiTinhKH,

NgayDangKy,

dbo.fn\_GetCustomerOrderCount(MaKH) AS SoDonHang

FROM KhachHang

ORDER BY NgayDangKy DESC

OFFSET @Offset ROWS

FETCH NEXT @PageSize ROWS ONLY;

-- Lấy tổng số lượng

SELECT COUNT(\*) AS TotalCount FROM KhachHang;

END TRY

BEGIN CATCH

SELECT 'ERROR' AS Status, ERROR\_MESSAGE() AS Message;

END CATCH

END;

GO

* **Lấy tất cả danh sách đơn hàng của khách hàng:**

CREATE PROCEDURE [dbo].[sp\_GetCustomerOrders]

@MaKH INT,

@PageNumber INT = 1,

@PageSize INT = 10

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

-- Kiểm tra khách hàng tồn tại

IF NOT EXISTS (SELECT 1 FROM KhachHang WHERE MaKH = @MaKH)

BEGIN

SELECT 'ERROR' AS Status, N'Khách hàng không tồn tại!' AS Message;

RETURN;

END

DECLARE @Offset INT = (@PageNumber - 1) \* @PageSize;

-- Lấy danh sách đơn hàng

SELECT

Order\_Id,

OrderNo,

Status,

SubTotal,

TotalAmount,

OrderDate,

IsCancel,

(SELECT COUNT(\*) FROM OrderDetail WHERE Order\_Id = o.Order\_Id) AS ItemCount

FROM Orders o

WHERE MaKH = @MaKH

ORDER BY OrderDate DESC

OFFSET @Offset ROWS

FETCH NEXT @PageSize ROWS ONLY;

-- Tổng số đơn hàng

SELECT COUNT(\*) AS TotalOrders FROM Orders WHERE MaKH = @MaKH;

END TRY

BEGIN CATCH

SELECT 'ERROR' AS Status, ERROR\_MESSAGE() AS Message;

END CATCH

END;

GO

* **Thống kê thông tin khách hàng:**

CREATE PROCEDURE[dbo].[ sp\_GetCustomerStatistics]

@MaKH INT

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

-- Kiểm tra khách hàng tồn tại

IF NOT EXISTS (SELECT 1 FROM KhachHang WHERE MaKH = @MaKH)

BEGIN

SELECT 'ERROR' AS Status, N'Khách hàng không tồn tại!' AS Message;

RETURN;

END

SELECT

MaKH,

TenKH,

Email,

SoDT,

NgayDangKy,

dbo.fn\_GetCustomerOrderCount(MaKH) AS TotalOrders,

ISNULL(SUM(o.TotalAmount), 0) AS TotalSpent,

COUNT(DISTINCT o.Order\_Id) AS CompletedOrders,

CASE

WHEN ISNULL(SUM(o.TotalAmount), 0) >= 10000000 THEN N'VIP'

WHEN ISNULL(SUM(o.TotalAmount), 0) >= 5000000 THEN N'Thường xuyên'

WHEN ISNULL(SUM(o.TotalAmount), 0) >= 1000000 THEN N'Bình thường'

ELSE N'Mới'

END AS CustomerLevel

FROM KhachHang k

LEFT JOIN Orders o ON k.MaKH = o.MaKH AND o.IsCancel = 0 AND o.Status = N'Hoàn thành'

WHERE k.MaKH = @MaKH

GROUP BY k.MaKH, k.TenKH, k.Email, k.SoDT, k.NgayDangKy;

END TRY

BEGIN CATCH

SELECT 'ERROR' AS Status, ERROR\_MESSAGE() AS Message;

END CATCH

END;

GO

* **Thống kê tổng quát**

CREATE PROCEDURE[dbo].[ sp\_GetCustomerDashboardStatus]

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

SELECT

dbo.fn\_GetTotalCustomerCount() AS TotalCustomers,

COUNT(DISTINCT CASE WHEN DATEDIFF(DAY, NgayDangKy, GETDATE()) <= 7 THEN MaKH END) AS NewCustomersLastWeek,

COUNT(DISTINCT CASE WHEN DATEDIFF(DAY, NgayDangKy, GETDATE()) <= 30 THEN MaKH END) AS NewCustomersLastMonth

FROM KhachHang;

END TRY

BEGIN CATCH

SELECT 'ERROR' AS Status, ERROR\_MESSAGE() AS Message;

END CATCH

END;

GO

Nhưng hàm function của khách hành:

Tổng tiền khách hàng đã chi tiêu:

CREATE FUNCTION fn\_GetCustomerTotalSpent (@MaKH INT)

RETURNS DECIMAL(18,2)

AS

BEGIN

DECLARE @TotalSpent DECIMAL(18,2);

SELECT @TotalSpent = ISNULL(SUM(TotalAmount), 0)

FROM Orders

WHERE MaKH = @MaKH AND IsCancel = 0 AND Status = N'Hoàn thành';

RETURN @TotalSpent;

END;

GO

* **Kiểm tra email hợp lệ của khách hàng:**

CREATE FUNCTION [dbo].[fn\_ValidateEmail](@Email NVARCHAR(100))

RETURNS BIT

AS

BEGIN

-- Kiểm tra định dạng email cơ bản

IF @Email LIKE '%@%.%' AND @Email NOT LIKE '% %'

RETURN 1;

ELSE

RETURN 0;

END;

GO

* **Định dạng số điện thoại khách hàng:**

CREATE FUNCTION fn\_FormatPhoneNumber (@Phone NVARCHAR(15))

RETURNS NVARCHAR(20)

AS

BEGIN

-- Loại bỏ các ký tự không phải số

DECLARE @CleanPhone NVARCHAR(15) = '';

DECLARE @i INT = 1;

WHILE @i <= LEN(@Phone)

BEGIN

IF SUBSTRING(@Phone, @i, 1) LIKE '[0-9]'

SET @CleanPhone = @CleanPhone + SUBSTRING(@Phone, @i, 1);

SET @i = @i + 1;

END

-- Định dạng: 0XXX-XXX-XXXX

IF LEN(@CleanPhone) = 10

RETURN SUBSTRING(@CleanPhone, 1, 4) + '-' +

SUBSTRING(@CleanPhone, 5, 3) + '-' +

SUBSTRING(@CleanPhone, 8, 3);

RETURN @CleanPhone;

END;

GO

Xếp hạng khách hàng:

CREATE FUNCTION [dbo].[fn\_XepHangKhachHang (@MaKH INT)

RETURNS NVARCHAR(50)

AS

BEGIN

DECLARE @TotalSpent DECIMAL(18,2) = dbo.fn\_GetCustomerTotalSpent(@MaKH);

DECLARE @OrderCount INT = dbo.fn\_GetCustomerOrderCount(@MaKH);

IF @TotalSpent >= 10000000 AND @OrderCount >= 10

RETURN N'VIP Platinum';

ELSE IF @TotalSpent >= 5000000 AND @OrderCount >= 5

RETURN N'VIP Gold';

ELSE IF @TotalSpent >= 1000000 AND @OrderCount >= 3

RETURN N'Khách hàng thân thiết';

ELSE IF @OrderCount >= 1

RETURN N'Khách hàng thường xuyên';

ELSE

RETURN N'Khách hàng mới';

END;

GO

**Những hàm Trigger của quản lý khách hàng**

* **Ngăn khách hàng có đơn hàng đang hoạt động:**

DROP TRIGGER IF EXISTS [dbo].[tr\_NganXoaKhachHangCoDonHangHoatDong];

GO

CREATE TRIGGER [dbo].[tr\_NganXoaKhachHangCoDonHangHoatDong]

ON KhachHang

INSTEAD OF DELETE

AS

BEGIN

SET NOCOUNT ON;

IF EXISTS (

SELECT 1

FROM deleted d

Inner Join KhachHang kh On d.MaKH = kh.MaKH

INNER JOIN Orders o ON d.MaKH = o.MaKH

WHERE o.IsCancel = 0

)

BEGIN

RAISERROR(N'Không thể xóa khách hàng có đơn hàng đang hoạt động!', 16, 1);

RETURN;

END;

DELETE FROM KhachHang WHERE MaKH IN (SELECT MaKH FROM deleted);

END;

GO

**Những hàm procedure của Hóa Đơn và Chi Tiết Hóa Đơn:**

* **Tạo hóa đơn:**

CREATE PROCEDURE sp\_CreateOrder

@MaKH INT,

@ShippingAddress NVARCHAR(500),

@PaymentMethod NVARCHAR(50),

@Coupon\_Id INT = NULL,

@InternalNote NVARCHAR(MAX) = NULL,

@Result INT OUTPUT,

@Order\_Id INT OUTPUT

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRANSACTION;

BEGIN TRY

-- 1. Kiểm tra khách hàng tồn tại

IF NOT EXISTS (SELECT 1 FROM KhachHang WHERE MaKH = @MaKH)

BEGIN

SET @Result = 1; -- Khách hàng không tồn tại

ROLLBACK TRANSACTION;

RETURN;

END

-- 2. Kiểm tra giỏ hàng có sản phẩm không

IF NOT EXISTS (SELECT 1 FROM Cart WHERE MaKH = @MaKH)

BEGIN

SET @Result = 2; -- Giỏ hàng trống

ROLLBACK TRANSACTION;

RETURN;

END

-- 3. Tính tổng tiền

DECLARE @SubTotal DECIMAL(18,2) = 0;

DECLARE @DiscountAmount DECIMAL(18,2) = 0;

DECLARE @TotalAmount DECIMAL(18,2) = 0;

SELECT @SubTotal = ISNULL(SUM(c.Quantity \* dbo.fn\_GetDisplayPrice(p.Product\_Id)), 0)

FROM Cart c

INNER JOIN Product p ON c.Product\_Id = p.Product\_Id

WHERE c.MaKH = @MaKH;

-- 4. Kiểm tra và áp dụng coupon

IF @Coupon\_Id IS NOT NULL

BEGIN

DECLARE @CouponExists INT, @DiscountType INT, @DiscountValue DECIMAL(18,2);

SELECT @CouponExists = 1, @DiscountType = DiscountType, @DiscountValue = DiscountValue

FROM Coupon

WHERE Coupon\_Id = @Coupon\_Id AND IsActive = 1

AND GETDATE() BETWEEN StartDate AND ExpiryDate;

IF @CouponExists = 1

BEGIN

IF @DiscountType = 1 -- Phần trăm

SET @DiscountAmount = @SubTotal \* @DiscountValue / 100;

ELSE -- Tiền cố định

SET @DiscountAmount = @DiscountValue;

END

ELSE

BEGIN

SET @Coupon\_Id = NULL;

END

END

SET @TotalAmount = @SubTotal - @DiscountAmount;

-- 5. Tạo Payment

INSERT INTO Payment (PaymentMethod, PaymentStatus, Amount, MaKH, CreatedDate)

VALUES (@PaymentMethod, N'Chờ thanh toán', @TotalAmount, @MaKH, GETDATE());

DECLARE @Payment\_Id INT = SCOPE\_IDENTITY();

-- 6. Tạo Order

INSERT INTO Orders (OrderNo, Status, SubTotal, TotalAmount, ShippingAddress, MaKH, Payment\_Id, Coupon\_Id, OrderDate, IsCancel, InternalNote)

VALUES (

'ORD' + FORMAT(GETDATE(), 'yyyyMMddHHmmss'),

N'Chờ xác nhận',

@SubTotal,

@TotalAmount,

@ShippingAddress,

@MaKH,

@Payment\_Id,

@Coupon\_Id,

GETDATE(),

0,

@InternalNote

);

SET @Order\_Id = SCOPE\_IDENTITY();

-- 7. Tạo OrderDetail từ Cart

INSERT INTO OrderDetail (Order\_Id, Product\_Id, Quantity, UnitPrice, TotalPrice)

SELECT @Order\_Id, c.Product\_Id, c.Quantity, dbo.fn\_GetDisplayPrice(c.Product\_Id),

c.Quantity \* dbo.fn\_GetDisplayPrice(c.Product\_Id)

FROM Cart c

WHERE c.MaKH = @MaKH;

-- 8. Cập nhật tồn kho

UPDATE WarehouseStock

SET Quantity = Quantity - (

SELECT SUM(Quantity) FROM Cart c WHERE c.Product\_Id = WarehouseStock.Product\_Id AND c.MaKH = @MaKH

)

WHERE Product\_Id IN (SELECT Product\_Id FROM Cart WHERE MaKH = @MaKH);

Ghi lại StockTransaction

INSERT INTO StockTransaction (TransactionType, Quantity, PreviousQuantity, NewQuantity, Reason, ReferenceNo, TransactionDate, CreatedBy, Stock\_Id, Order\_Id)

SELECT N'Xuất', c.Quantity, ws.Quantity + c.Quantity, ws.Quantity, N'Bán hàng', 'ORD' + FORMAT(GETDATE(), 'yyyyMMddHHmmss'), GETDATE(), (SELECT Email FROM KhachHang WHERE MaKH = @MaKH), ws.Stock\_Id, @Order\_Id

FROM Cart c

INNER JOIN WarehouseStock ws ON c.Product\_Id = ws.Product\_Id;

-- 10. Xóa giỏ hàng

DELETE FROM Cart WHERE MaKH = @MaKH;

-- 11. Ghi log nếu dùng coupon

IF @Coupon\_Id IS NOT NULL

BEGIN

INSERT INTO Coupon\_Usage (Coupon\_Id, MaKH, UsedDate)

VALUES (@Coupon\_Id, @MaKH, GETDATE());

END

COMMIT TRANSACTION;

SET @Result = 0; -- Thành công

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

SET @Result = -1; -- Lỗi hệ thống

END CATCH

END;

GO

* **Lấy danh sách đơn hàng:**

CREATE PROCEDURE sp\_GetOrdersList

@PageNumber INT = 1,

@PageSize INT = 10,

@Status NVARCHAR(50) = NULL

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

DECLARE @Offset INT = (@PageNumber - 1) \* @PageSize;

SELECT

o.Order\_Id,

o.OrderNo,

o.Status,

o.SubTotal,

o.TotalAmount,

o.OrderDate,

o.IsCancel,

k.MaKH,

k.TenKH,

k.SoDT,

p.PaymentMethod,

p.PaymentStatus,

(SELECT COUNT(\*) FROM OrderDetail WHERE Order\_Id = o.Order\_Id) AS ItemCount

FROM Orders o

INNER JOIN KhachHang k ON o.MaKH = k.MaKH

LEFT JOIN Payment p ON o.Payment\_Id = p.Payment\_Id

WHERE (@Status IS NULL OR o.Status = @Status)

ORDER BY o.OrderDate DESC

OFFSET @Offset ROWS

FETCH NEXT @PageSize ROWS ONLY;

SELECT COUNT(\*) AS TotalOrders

FROM Orders

WHERE (@Status IS NULL OR Status = @Status);

END TRY

BEGIN CATCH

SELECT 'ERROR' AS Status, ERROR\_MESSAGE() AS Message;

END CATCH

END;

GO

* **Lấy thông tin trong chi tiết hóa đơn:**

CREATE PROCEDURE sp\_GetOrderDetail

@Order\_Id INT

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

IF NOT EXISTS (SELECT 1 FROM Orders WHERE Order\_Id = @Order\_Id)

BEGIN

SELECT 'ERROR' AS Status, N'Đơn hàng không tồn tại!' AS Message;

RETURN;

END

-- Lấy thông tin Order

SELECT

o.Order\_Id,

o.OrderNo,

o.Status,

o.SubTotal,

o.TotalAmount,

o.ShippingAddress,

o.OrderDate,

o.IsCancel,

o.ConfirmedDate,

o.ShippedDate,

o.DeliveredDate,

o.CancelledDate,

o.CancelledBy,

o.CancelReason,

o.InternalNote,

k.MaKH,

k.TenKH,

k.Email,

k.SoDT,

p.PaymentMethod,

p.PaymentStatus,

c.CouponCode,

c.DiscountValue

FROM Orders o

INNER JOIN KhachHang k ON o.MaKH = k.MaKH

LEFT JOIN Payment p ON o.Payment\_Id = p.Payment\_Id

LEFT JOIN Coupon c ON o.Coupon\_Id = c.Coupon\_Id

WHERE o.Order\_Id = @Order\_Id;

-- Lấy danh sách sản phẩm trong đơn hàng

SELECT

od.OrderDetail\_Id,

od.Product\_Id,

pr.ProductName,

od.Quantity,

od.UnitPrice,

od.TotalPrice

FROM OrderDetail od

INNER JOIN Product pr ON od.Product\_Id = pr.Product\_Id

WHERE od.Order\_Id = @Order\_Id;

END TRY

BEGIN CATCH

SELECT 'ERROR' AS Status, ERROR\_MESSAGE() AS Message;

END CATCH

END;

GO

* **Cập nhật trạng thái hóa đơn**

CREATE PROCEDURE sp\_UpdateOrderStatus

@Order\_Id INT,

@NewStatus NVARCHAR(50),

@UpdatedBy NVARCHAR(100)

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

IF NOT EXISTS (SELECT 1 FROM Orders WHERE Order\_Id = @Order\_Id)

BEGIN

SELECT 'ERROR' AS Status, N'Đơn hàng không tồn tại!' AS Message;

RETURN;

END

DECLARE @OldStatus NVARCHAR(50);

SELECT @OldStatus = Status FROM Orders WHERE Order\_Id = @Order\_Id;

-- Cập nhật trạng thái

UPDATE Orders

SET Status = @NewStatus,

ConfirmedDate = CASE WHEN @NewStatus = N'Đã xác nhận' THEN GETDATE() ELSE ConfirmedDate END,

ShippedDate = CASE WHEN @NewStatus = N'Đã gửi' THEN GETDATE() ELSE ShippedDate END,

DeliveredDate = CASE WHEN @NewStatus = N'Hoàn thành' THEN GETDATE() ELSE DeliveredDate END

WHERE Order\_Id = @Order\_Id;

-- Ghi log

INSERT INTO AuditLog (ActionName, Description, PerformedBy, ActionTime)

VALUES ('UPDATE\_ORDER\_STATUS', N'Thay đổi trạng thái Order\_Id: ' + CAST(@Order\_Id AS NVARCHAR) + N' từ ' + @OldStatus + N' sang ' + @NewStatus, @UpdatedBy, GETDATE());

SELECT 'SUCCESS' AS Status, N'Cập nhật trạng thái thành công!' AS Message;

END TRY

BEGIN CATCH

SELECT 'ERROR' AS Status, ERROR\_MESSAGE() AS Message;

END CATCH

END;

GO

Hủy hóa đơn:

CREATE PROCEDURE [dbo].[sp\_CancelOrder]

@Order\_Id INT,

@CancelledBy NVARCHAR(100),

@CancelReason NVARCHAR(MAX)

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRANSACTION;

BEGIN TRY

IF NOT EXISTS (SELECT 1 FROM Orders WHERE Order\_Id = @Order\_Id)

BEGIN

SELECT 'ERROR' AS Status, N'Đơn hàng không tồn tại!' AS Message;

ROLLBACK TRANSACTION;

RETURN;

END

DECLARE @CurrentStatus NVARCHAR(50);

SELECT @CurrentStatus = Status, @Order\_Id = Order\_Id FROM Orders WHERE Order\_Id = @Order\_Id;

-- Kiểm tra chỉ có thể hủy đơn chưa gửi

IF @CurrentStatus = N'Đã gửi' OR @CurrentStatus = N'Hoàn thành'

BEGIN

SELECT 'ERROR' AS Status, N'Không thể hủy đơn hàng đã gửi hoặc hoàn thành!' AS Message;

ROLLBACK TRANSACTION;

RETURN;

END

-- Hoàn lại tồn kho

UPDATE WarehouseStock

SET Quantity = Quantity + (

SELECT SUM(od.Quantity) FROM OrderDetail od WHERE od.Order\_Id = @Order\_Id

)

WHERE Product\_Id IN (SELECT Product\_Id FROM OrderDetail WHERE Order\_Id = @Order\_Id);

-- Ghi StockTransaction

INSERT INTO StockTransaction (TransactionType, Quantity, PreviousQuantity, NewQuantity, Reason, ReferenceNo, TransactionDate, CreatedBy, Stock\_Id, Order\_Id)

SELECT N'Nhập', od.Quantity, ws.Quantity - od.Quantity, ws.Quantity, N'Hủy đơn hàng', 'ORD' + CAST(@Order\_Id AS NVARCHAR), GETDATE(), @CancelledBy, ws.Stock\_Id, @Order\_Id

FROM OrderDetail od

INNER JOIN WarehouseStock ws ON od.Product\_Id = ws.Product\_Id;

-- Cập nhật đơn hàng

UPDATE Orders

SET IsCancel = 1,

Status = N'Đã hủy',

CancelledDate = GETDATE(),

CancelledBy = @CancelledBy,

CancelReason = @CancelReason

WHERE Order\_Id = @Order\_Id;

-- Ghi log

INSERT INTO AuditLog (ActionName, Description, PerformedBy, ActionTime)

VALUES ('CANCEL\_ORDER', N'Hủy đơn hàng Order\_Id: ' + CAST(@Order\_Id AS NVARCHAR) + N' - Lý do: ' + @CancelReason, @CancelledBy, GETDATE());

COMMIT TRANSACTION;

SELECT 'SUCCESS' AS Status, N'Hủy đơn hàng thành công!' AS Message;

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

SELECT 'ERROR' AS Status, ERROR\_MESSAGE() AS Message;

END CATCH

END;

GO

* **Tìm kiếm hóa đơn:**

CREATE PROCEDURE [dbo].sp\_SearchOrders

@Keyword NVARCHAR(100),

@PageNumber INT = 1,

@PageSize INT = 10

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

DECLARE @Offset INT = (@PageNumber - 1) \* @PageSize;

SELECT

o.Order\_Id,

o.OrderNo,

o.Status,

o.SubTotal,

o.TotalAmount,

o.OrderDate,

k.TenKH,

k.SoDT,

(SELECT COUNT(\*) FROM OrderDetail WHERE Order\_Id = o.Order\_Id) AS ItemCount

FROM Orders o

INNER JOIN KhachHang k ON o.MaKH = k.MaKH

WHERE o.OrderNo LIKE '%' + @Keyword + '%'

OR k.TenKH LIKE '%' + @Keyword + '%'

OR k.SoDT LIKE '%' + @Keyword + '%'

ORDER BY o.OrderDate DESC

OFFSET @Offset ROWS

FETCH NEXT @PageSize ROWS ONLY;

SELECT COUNT(\*) AS TotalOrders

FROM Orders o

INNER JOIN KhachHang k ON o.MaKH = k.MaKH

WHERE o.OrderNo LIKE '%' + @Keyword + '%'

OR k.TenKH LIKE '%' + @Keyword + '%'

OR k.SoDT LIKE '%' + @Keyword + '%';

END TRY

BEGIN CATCH

SELECT 'ERROR' AS Status, ERROR\_MESSAGE() AS Message;

END CATCH

END;

GO

Thống kê doanh thu theo ngày:

CREATE PROCEDURE sp\_GetRevenueByDateRange

@StartDate DATETIME,

@EndDate DATETIME

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

SELECT

CAST(OrderDate AS DATE) AS OrderDate,

COUNT(\*) AS OrderCount,

SUM(TotalAmount) AS DailyRevenue,

COUNT(DISTINCT MaKH) AS UniqueCustomers

FROM Orders

WHERE OrderDate BETWEEN @StartDate AND @EndDate

AND IsCancel = 0

AND Status = N'Hoàn thành'

GROUP BY CAST(OrderDate AS DATE)

ORDER BY OrderDate DESC;

END TRY

BEGIN CATCH

SELECT 'ERROR' AS Status, ERROR\_MESSAGE() AS Message;

END CATCH

END;

GO

* **Lấy chi tiết một sản phẩm của đơn hàng:**

CREATE PROCEDURE sp\_GetOrderDetailItem

@OrderDetail\_Id INT

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

IF NOT EXISTS (SELECT 1 FROM OrderDetail WHERE OrderDetail\_Id = @OrderDetail\_Id)

BEGIN

SELECT 'ERROR' AS Status, N'Chi tiết đơn hàng không tồn tại!' AS Message;

RETURN;

END

SELECT

od.OrderDetail\_Id,

od.Order\_Id,

od.Product\_Id,

p.ProductName,

p.ProductSlug,

od.Quantity,

od.UnitPrice,

od.TotalPrice,

c.CategoryName

FROM OrderDetail od

INNER JOIN Product p ON od.Product\_Id = p.Product\_Id

INNER JOIN Category c ON p.Category\_Id = c.Category\_Id

WHERE od.OrderDetail\_Id = @OrderDetail\_Id;

END TRY

BEGIN CATCH

SELECT 'ERROR' AS Status, ERROR\_MESSAGE() AS Message;

END CATCH

END;

GO

Những hàm function của hóa đơn và chi tiết hóa đơn:

* **Tính tổng tiền hóa đơn:**

CREATE FUNCTION fn\_GetOrderTotal (@Order\_Id INT)

RETURNS DECIMAL(18,2)

AS

BEGIN

DECLARE @TotalAmount DECIMAL(18,2);

SELECT @TotalAmount = ISNULL(SUM(TotalPrice), 0)

FROM OrderDetail

WHERE Order\_Id = @Order\_Id;

RETURN @TotalAmount;

END;

GO

Đếm số lượng sản phẩm trong hóa đơn:

CREATE FUNCTION fn\_GetOrderItemCount (@Order\_Id INT)

RETURNS INT

AS

BEGIN

DECLARE @ItemCount INT;

SELECT @ItemCount = COUNT(\*)

FROM OrderDetail

WHERE Order\_Id = @Order\_Id;

RETURN ISNULL(@ItemCount, 0);

END;

GO

Đếm số đơn hàng chưa xử lý:

CREATE FUNCTION fn\_GetPendingOrderCount ()

RETURNS INT

AS

BEGIN

DECLARE @PendingCount INT;

SELECT @PendingCount = COUNT(\*)

FROM Orders

WHERE Status = N'Chờ xác nhận' AND IsCancel = 0;

RETURN ISNULL(@PendingCount, 0);

END;

GO

Kiểm tra đơn hàng có hoàn thành không:

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

If Object\_Id('fn\_IsOrderCompleted') is Null

Drop Function [dbo].[fn\_IsOrderCompleted]

Go

CREATE FUNCTION [dbo].fn\_IsOrderCompleted (@Order\_Id INT)

RETURNS BIT

AS

BEGIN

IF EXISTS (SELECT 1 FROM Orders WHERE Order\_Id = @Order\_Id AND Status = N'Hoàn thành')

RETURN 1;

ELSE

Print N'Không hoàn thành'

RETURN 0; -- Không hoàn thành

END;

GO

Cập nhật thống kê sản phẩm khi tạo chi tiết hóa đơn:

CREATE TRIGGER tr\_UpdateProductSoldCount

ON OrderDetail

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

UPDATE Product

SET SoldCount = SoldCount + inserted.Quantity

FROM Product p

INNER JOIN inserted ON p.Product\_Id = inserted.Product\_Id;

END;

GO

Những hàm trigger của hóa đơn và chi tiết hóa đơn:

* **Ghi log đơn hàng khi được tạo:**

If Object\_Id('trg\_LogOrderCreation')is Null

Drop trigger [dbo].[trg\_LogOrderCreation]

Go

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TRIGGER [dbo].[trg\_LogOrderCreation]

ON Orders

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO AuditLog (ActionName, Description, TargetUserEmail, ActionTime)

SELECT

'CREATE\_ORDER',

N'Tạo đơn hàng mới: ' + i.OrderNo + N' - Tổng tiền: ' + CAST(i.TotalAmount AS NVARCHAR),

(SELECT Email FROM KhachHang kh WHERE kh.MaKH = MaKH),

GETDATE()

FROM inserted i

END;

GO

* **Ngăn thêm chi tiết hóa đơn khi bị sai:**

CREATE TRIGGER [dbo].[trg\_ValidateOrderDetailPrice]

ON OrderDetail

INSTEAD OF INSERT

AS

BEGIN

SET NOCOUNT ON;

IF EXISTS (SELECT 1 FROM inserted WHERE Quantity <= 0 OR UnitPrice < 0 OR TotalPrice < 0)

BEGIN

RAISERROR(N'Dữ liệu OrderDetail không hợp lệ!', 16, 1);

RETURN;

END

INSERT INTO OrderDetail (Order\_Id, Product\_Id, Quantity, UnitPrice, TotalPrice)

SELECT Order\_Id, Product\_Id, Quantity, UnitPrice, TotalPrice FROM inserted;

END;

GO

Cập nhật trạng thái của phương thức thanh toán khi hóa đơn hoàn thành:

CREATE TRIGGER trg\_UpdatePaymentStatusOnOrderComplete

ON Orders

AFTER UPDATE

AS

BEGIN

SET NOCOUNT ON;

UPDATE Payment

SET PaymentStatus = N'Đã thanh toán',

PaymentDate = GETDATE()

FROM Payment p

INNER JOIN inserted i ON p.Payment\_Id IN (SELECT Payment\_Id FROM inserted WHERE Status = N'Hoàn thành');

END;

GO

**Nhưng hàm store procedure của giỏ hàng:**

**Các CRUD của cart:**

* **Thêm khách hàng:**

CREATE PROCEDURE sp\_AddToCart

@MaKH INT,

@Product\_Id INT,

@Quantity INT,

@Cart\_Id INT OUTPUT

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION

-- Kiểm tra khách hàng

IF NOT EXISTS (SELECT 1 FROM KhachHang WHERE MaKH = @MaKH)

BEGIN

RAISERROR('Khách hàng không tồn tại', 16, 1)

RETURN

END

-- Kiểm tra sản phẩm

IF NOT EXISTS (SELECT 1 FROM Product WHERE Product\_Id = @Product\_Id)

BEGIN

RAISERROR('Sản phẩm không tồn tại', 16, 1)

RETURN

END

-- Kiểm tra số lượng

IF @Quantity <= 0

BEGIN

RAISERROR('Số lượng phải lớn hơn 0', 16, 1)

RETURN

END

-- Kiểm tra tồn kho

IF dbo.fn\_CheckStock(@Product\_Id, @Quantity) = 0

BEGIN

RAISERROR('Tồn kho không đủ', 16, 1)

RETURN

END

-- Kiểm tra sản phẩm đã có trong giỏ

IF EXISTS (SELECT 1 FROM Cart WHERE MaKH = @MaKH AND Product\_Id = @Product\_Id)

BEGIN

UPDATE Cart

SET Quantity = Quantity + @Quantity

WHERE MaKH = @MaKH AND Product\_Id = @Product\_Id

SELECT @Cart\_Id = Cart\_Id FROM Cart

WHERE MaKH = @MaKH AND Product\_Id = @Product\_Id

END

ELSE

BEGIN

INSERT INTO Cart (Quantity, CreatedDate, Product\_Id, MaKH)

VALUES (@Quantity, GETDATE(), @Product\_Id, @MaKH)

SET @Cart\_Id = SCOPE\_IDENTITY()

END

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

RAISERROR('Lỗi thêm sản phẩm vào giỏ', 16, 1)

END CATCH

END

GO

Cập nhật số lượng trong giỏ:

CREATE PROCEDURE sp\_UpdateCartQuantity

@Cart\_Id INT,

@NewQuantity INT

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION

IF NOT EXISTS (SELECT 1 FROM Cart WHERE Cart\_Id = @Cart\_Id)

BEGIN

RAISERROR('Sản phẩm trong giỏ không tồn tại', 16, 1)

RETURN

END

IF @NewQuantity <= 0

BEGIN

RAISERROR('Số lượng phải lớn hơn 0', 16, 1)

RETURN

END

DECLARE @Product\_Id INT

SELECT @Product\_Id = Product\_Id FROM Cart WHERE Cart\_Id = @Cart\_Id

IF dbo.fn\_CheckStock(@Product\_Id, @NewQuantity) = 0

BEGIN

RAISERROR('Tồn kho không đủ', 16, 1)

RETURN

END

UPDATE Cart

SET Quantity = @NewQuantity

WHERE Cart\_Id = @Cart\_Id

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

RAISERROR('Lỗi cập nhật số lượng', 16, 1)

END CATCH

END

GO

* **Xóa sản phẩm khỏi giỏ hàng:**

CREATE PROCEDURE sp\_DeleteCartItem

@Cart\_Id INT

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION

IF NOT EXISTS (SELECT 1 FROM Cart WHERE Cart\_Id = @Cart\_Id)

BEGIN

RAISERROR('Sản phẩm trong giỏ không tồn tại', 16, 1)

RETURN

END

DELETE FROM Cart WHERE Cart\_Id = @Cart\_Id

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

RAISERROR('Lỗi xóa sản phẩm', 16, 1)

END CATCH

END

GO

Xóa toàn bộ đơn hàng:

CREATE PROCEDURE sp\_DeleteAllCart

@MaKH INT

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION

IF NOT EXISTS (SELECT 1 FROM Cart WHERE MaKH = @MaKH)

BEGIN

RAISERROR('Khách hàng không có sản phẩm trong giỏ', 16, 1)

RETURN

END

DELETE FROM Cart WHERE MaKH = @MaKH

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

RAISERROR('Lỗi xóa giỏ hàng', 16, 1)

END CATCH

END

GO

Nhưng hàm trigger của giỏ hàng:

Ghi thêm log và giỏ:

CREATE TRIGGER trg\_LogCartInsert

ON Cart

AFTER INSERT

AS

BEGIN

INSERT INTO AuditLog (ActionName, Description, PerformedBy, ActionTime)

SELECT 'ADD\_TO\_CART',

'Thêm sản phẩm ID: ' + CAST(Product\_Id AS NVARCHAR(10)) +

' | Số lượng: ' + CAST(Quantity AS NVARCHAR(10)) +

' | Khách hàng: ' + CAST(MaKH AS NVARCHAR(10)),

'System',

GETDATE()

FROM inserted

END

GO

* **Ghi log khi cập nhật giỏ:**

CREATE TRIGGER trg\_LogCartUpdate

ON Cart

AFTER UPDATE

AS

BEGIN

INSERT INTO AuditLog (ActionName, Description, PerformedBy, ActionTime)

SELECT 'UPDATE\_CART',

'Cập nhật số lượng sản phẩm ID: ' + CAST(i.Product\_Id AS NVARCHAR(10)) +

' | Từ: ' + CAST(d.Quantity AS NVARCHAR(10)) +

' | Sang: ' + CAST(i.Quantity AS NVARCHAR(10)),

'System',

GETDATE()

FROM inserted i

INNER JOIN deleted d ON i.Cart\_Id = d.Cart\_Id

WHERE i.Quantity <> d.Quantity

END

GO

**Ghi xóa log khỏi giỏ:**

CREATE TRIGGER trg\_LogCartDelete

ON Cart

AFTER DELETE

AS

BEGIN

INSERT INTO AuditLog (ActionName, Description, PerformedBy, ActionTime)

SELECT 'DELETE\_CART',

'Xóa sản phẩm ID: ' + CAST(Product\_Id AS NVARCHAR(10)) +

' | Số lượng: ' + CAST(Quantity AS NVARCHAR(10)),

'System',

GETDATE()

FROM deleted

END

GO

* **Kiểm tra tồn kho trước khi thêm giỏ:**

CREATE TRIGGER trg\_CheckStockBeforeInsertCart

ON Cart

INSTEAD OF INSERT

AS

BEGIN

DECLARE @Product\_Id INT, @Quantity INT

SELECT @Product\_Id = Product\_Id, @Quantity = Quantity FROM inserted

IF dbo.fn\_CheckStock(@Product\_Id, @Quantity) = 0

BEGIN

RAISERROR('Tồn kho không đủ', 16, 1)

ROLLBACK TRANSACTION

RETURN

END

INSERT INTO Cart (Quantity, CreatedDate, Product\_Id, MaKH)

SELECT Quantity, GETDATE(), Product\_Id, MaKH FROM inserted

END

GO

* **Cập nhật số lượng tổng của hàng đã hết:**

CREATE TRIGGER trg\_UpdateSoldCount

ON OrderDetail

AFTER INSERT

AS

BEGIN

UPDATE Product

SET SoldCount = ISNULL(SoldCount, 0) + i.Quantity

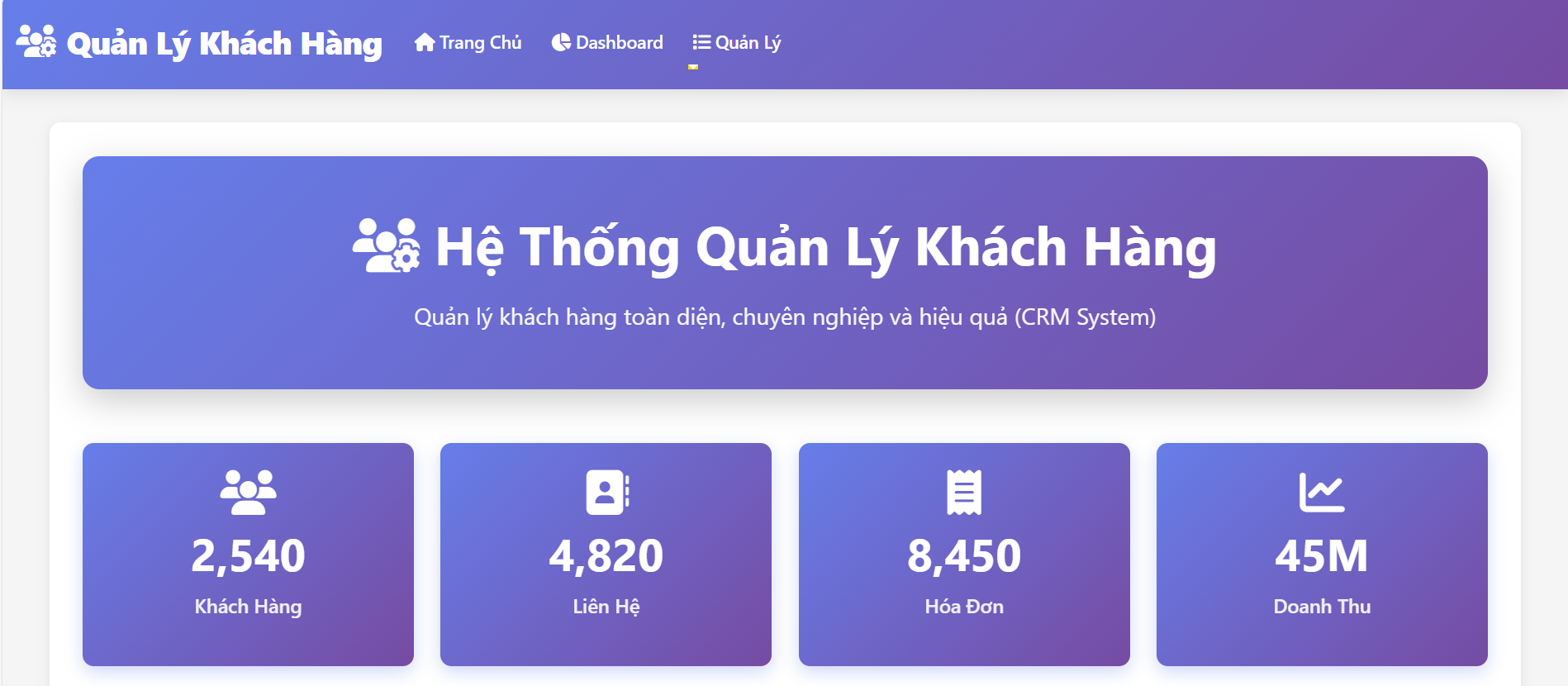
FROM inserted i WHERE Product.Product\_Id = i.Product\_Id

END

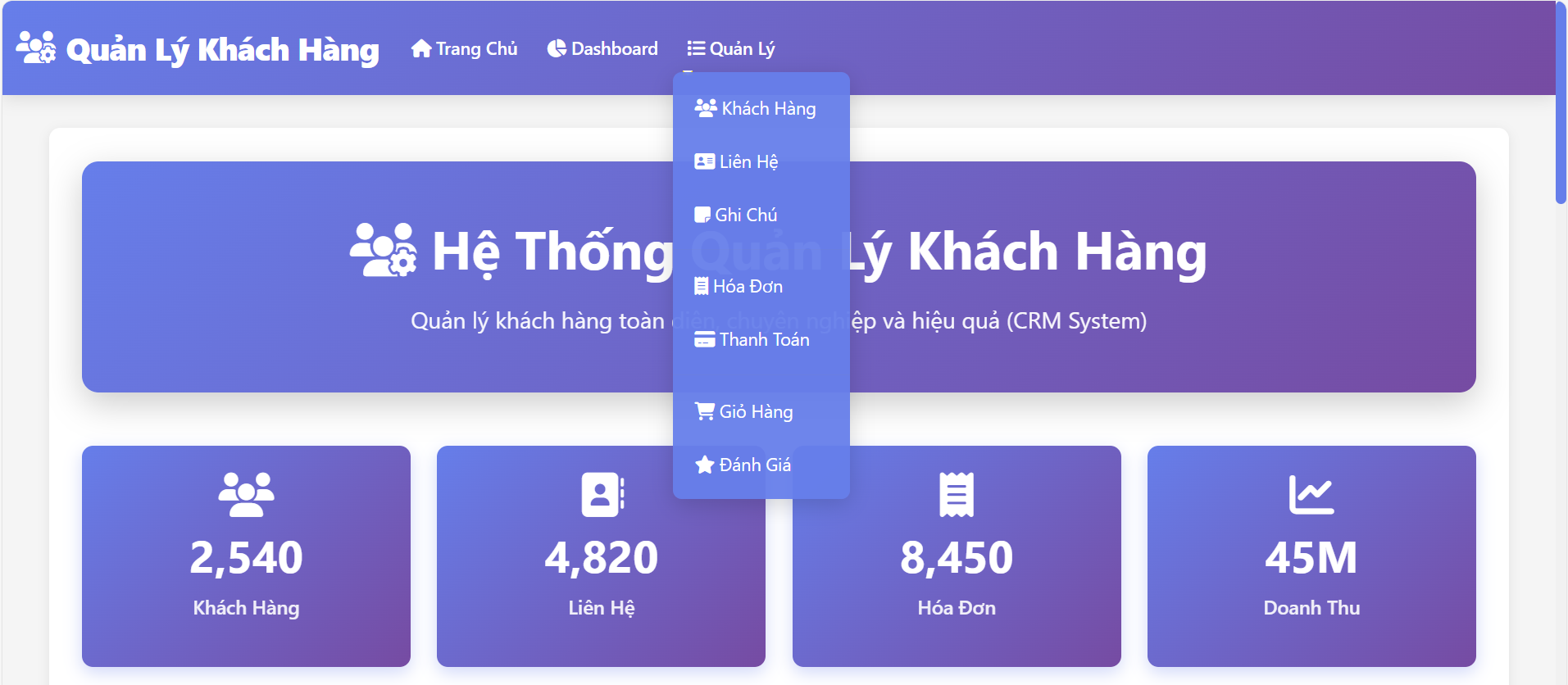
GO

Giao diện của ứng dụng:

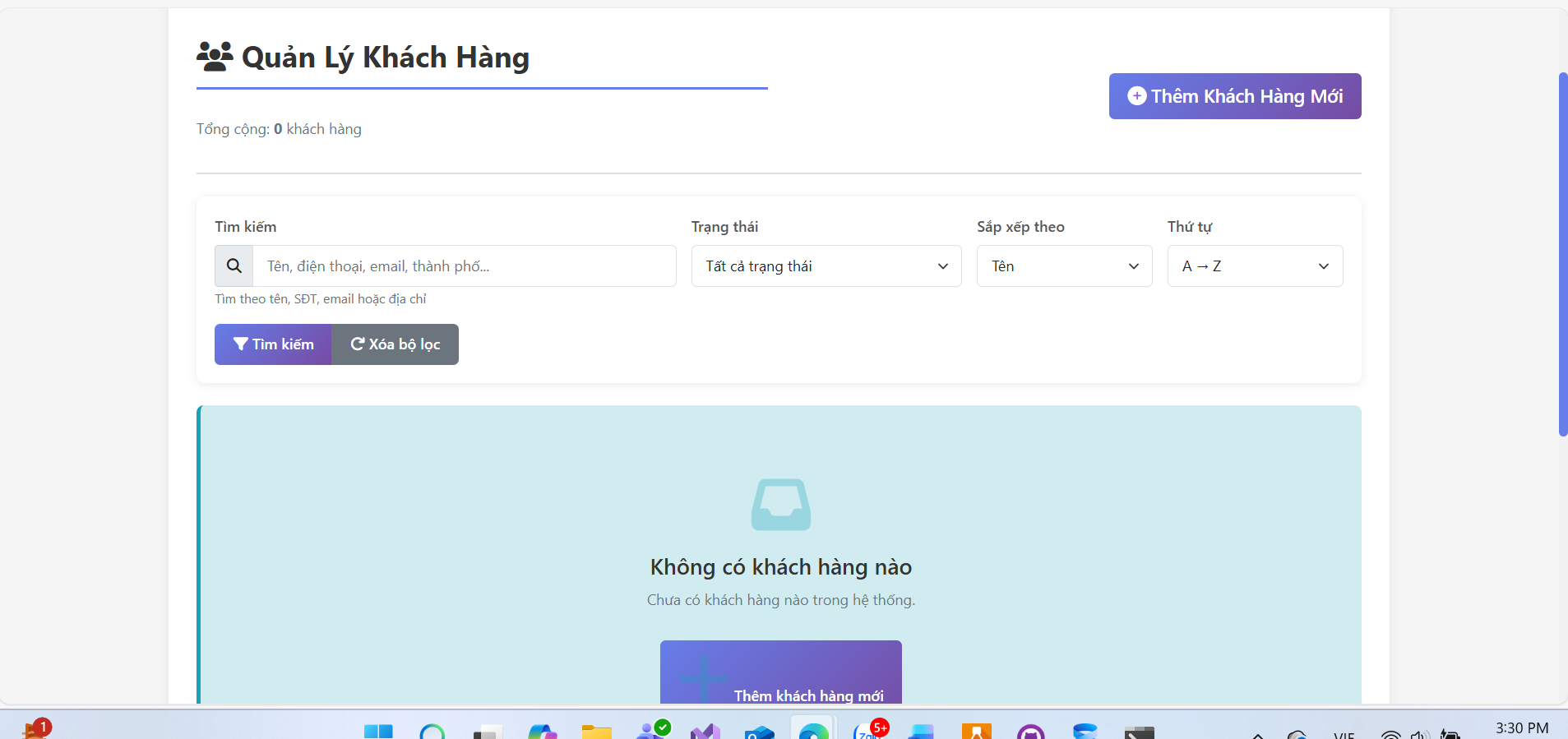
Giao diện chính:



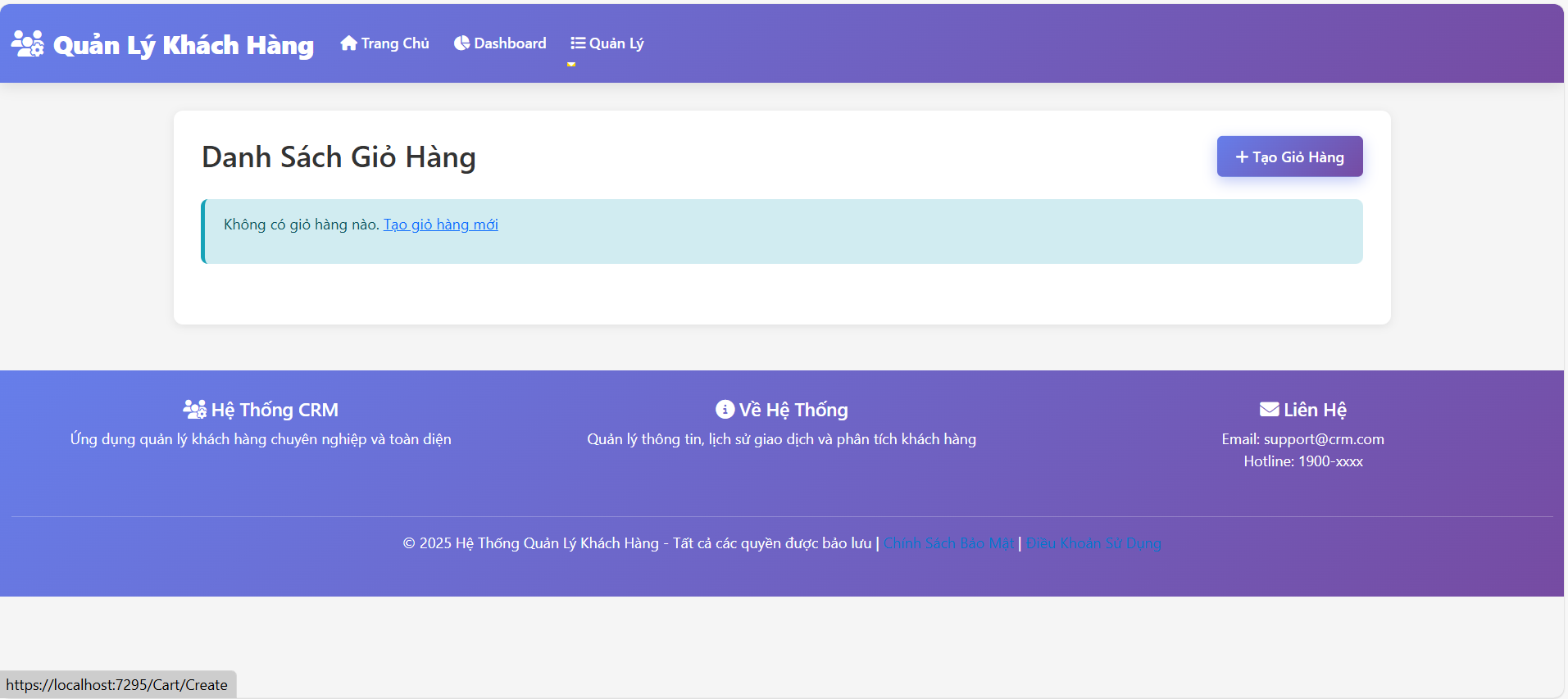
Danh mục chức năng:



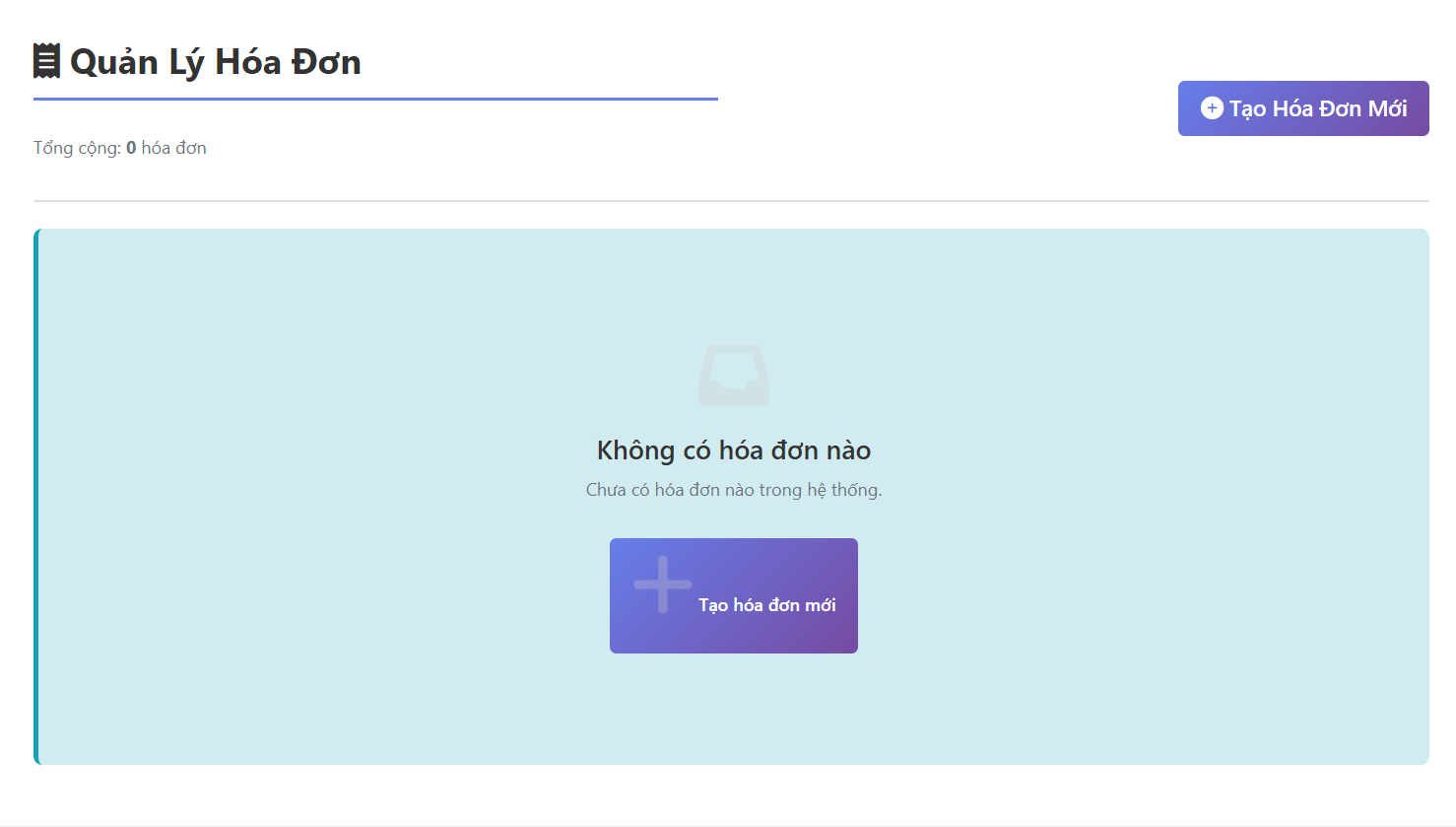
Giao diện quản lý danh sách khách hàng:



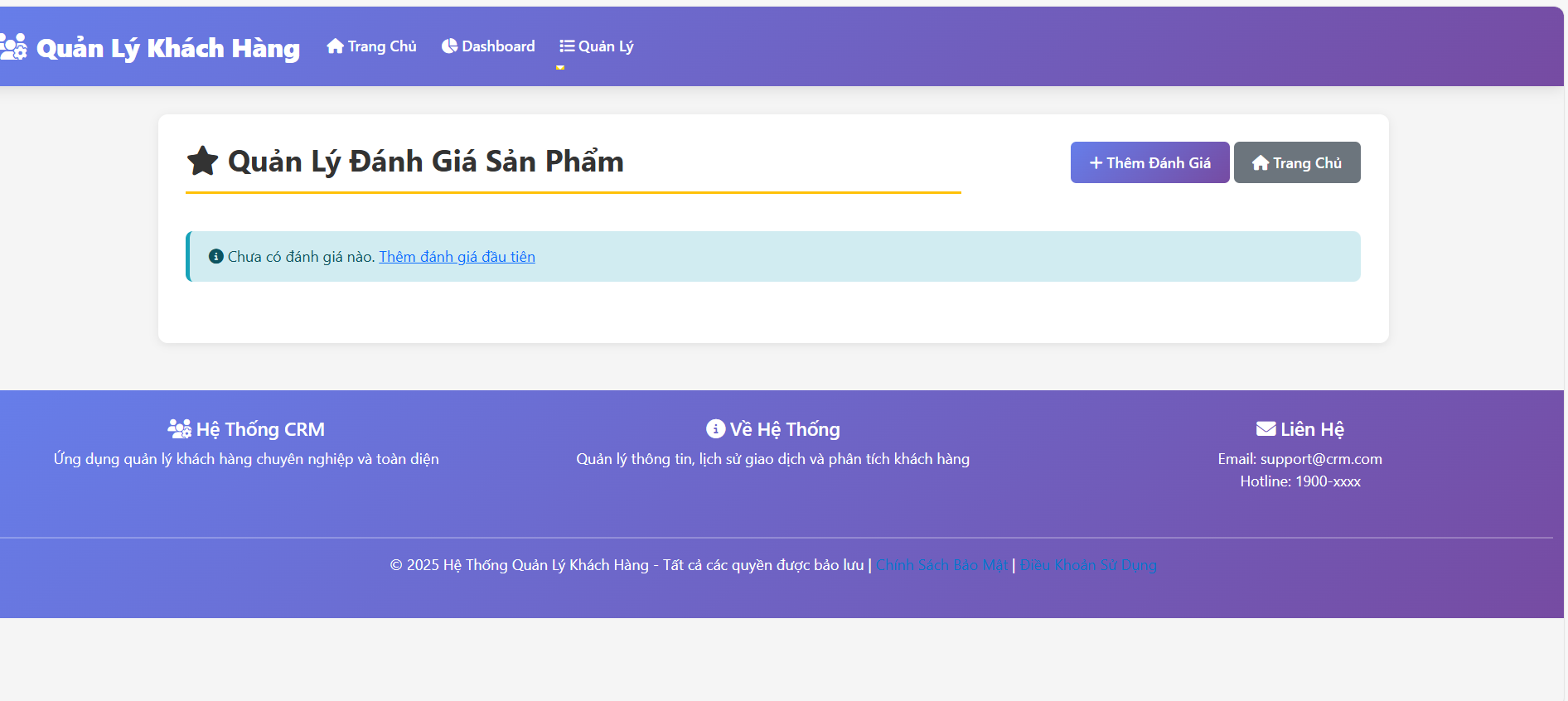
Giao diện quản lý giỏ hàng:



Giao diện quản lý đơn hàng:



Giao diện đánh giá sản phẩm:



# TÀI LIỆU THAM KHẢO

**[1]. Database Star.** (n.d.). *Data model for an ecommerce website*. [https://www.databasestar.com/data-model-ecommerce/](https://www.databasestar.com/data-warehouse/)

**[2]. Microsoft.** (n.d.). *Authentication in SQL Server*. Microsoft Learn. Retrieved December 2, 2025, [from https://learn.microsoft.com/en-us/sql/relational-databases/security/authentication-access/authentication-in-sql-server](https://learn.microsoft.com/en-us/sql/connect/ado-net/sql/authentication-sql-server?view=sql-server-ver17)

**[3]. Microsoft.** (n.d.). *Database-level roles*. Microsoft Learn. Retrieved December 2, 2025, from <https://learn.microsoft.com/en-us/sql/relational-databases/security/authentication-access/database-level-roles>

**[4]. Nguyễn, Đ.** (2018). *Giáo trình hệ quản trị cơ sở dữ liệu SQL Server*. Nhà xuất bản Thông tin và Truyền thông.