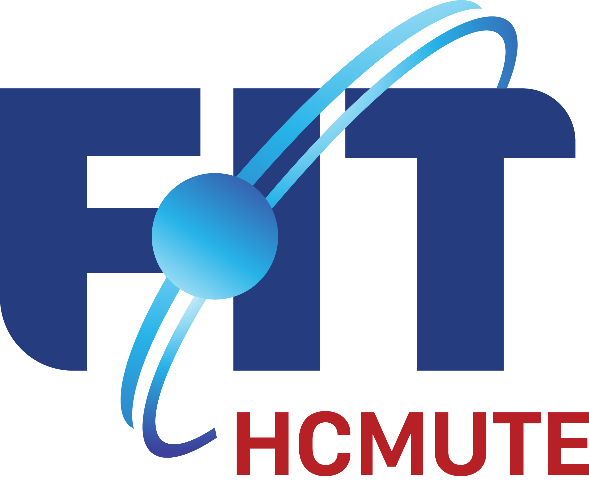
**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BỘ MÔN HỆ THỐNG THÔNG TIN**

**-----….-----**

****

**ĐỀ TÀI CUỐI KÌ MÔN HỆ QUẢN TRỊ CSDL**

**TÊN ĐỀ TÀI:**

**QUẢN LÝ QUÁN CAFÉ “WIBU COFFEE”**

**GVHD:** *TS. Nguyễn Thành Sơn*

**Lớp HP:** *DBMS330284\_23\_2\_03CLC*

**Nhóm thực hiện:** *Nhóm 02*

**Học kỳ:** *2*

**Năm học:** *2023 – 2024*

Thành phố Hồ Chí Minh, tháng 5, năm 2024

**DANH SÁCH SINH VIÊN NHÓM THỰC HIỆN**

**Học kì 2 NĂM HỌC 2023 – 2024**

**Nhóm 02**

***Đề tài: Quản lý quán café “Wibu Coffee”***

|  |  |
| --- | --- |
| 22110205 | Lê Duy Phương |
| 22110196 | Lê Văn Phát |
| 22110118 | Huỳnh Thanh Duy |
| 22110138 | Nguyễn Trung Hiếu |

**NHẬN XÉT CỦA GV**

Tp. Hồ Chí Minh, ngày 15, tháng 03, năm 2024

Giảng viên chấm điểm

**MỤC LỤC**

[**CHƯƠNG 1: TỔNG QUAN VỀ HỆ THỐNG** 1](#_Toc165362067)

[**1.** **Đặc tả đề tài** 1](#_Toc165362068)

[**1.1.** **Mô tả bài toán** 1](#_Toc165362069)

[**1.2.** **Nghiệp vụ bài toán** 4](#_Toc165362070)

[**1.3.** **Mô tả chức năng của bài toán** 9](#_Toc165362071)

[**1.4.** **Tính lương nhân viên** 11](#_Toc165362072)

[**CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ HỆ THỐNG** 13](#_Toc165362073)

[**1.** **Thiết kế cơ sở dữ liệu mức quan niệm** 13](#_Toc165362074)

[**2.** **Thiết kế cơ sở dữ liệu mức logic** 15](#_Toc165362075)

[**3.** **Các ràng buộc cần có** 16](#_Toc165362076)

[**4.** **Cài đặt các CSDL và các ràng buộc** 19](#_Toc165362077)

[**CHƯƠNG 3: THIẾT KẾ CÁC CHỨC NĂNG** 25](#_Toc165362078)

[**1.** **Kết nối Cơ sở dữ liệu:** 25](#_Toc165362079)

[**2.** **Các chức năng của chương trình:** 30](#_Toc165362080)

[**2.1.** **Dashboard** 30](#_Toc165362083)

[**2.2.** **Chức năng Order:** 41](#_Toc165362084)

[**2.3.** **Chức năng Employee:** 79](#_Toc165362085)

[**2.4.** **Chức năng Product:** 105](#_Toc165362086)

[**2.5.** **Chức năng Material:** 122](#_Toc165362087)

[**2.6.** **Chức năng Supplier:** 124](#_Toc165362088)

[**2.7.** **Chức năng Expense Bill** 128](#_Toc165362089)

[**2.8.** **Chức năng ReceiptNote** 133](#_Toc165362090)

[**3.** **Tạo user và phân quyền truy cập** 149](#_Toc165362091)

[**CHƯƠNG 4: THIẾT KẾ GIAO DIỆN HỆ THỐNG** 155](#_Toc165362092)

**LỜI CẢM ƠN**

Kính thưa thầy Nguyễn Thành Sơn,

Nhóm em xin gửi lời cảm ơn chân thành đến thầy vì đã hướng dẫn và giúp đỡ

chúng em trong quá trình hoàn thiện đồ án môn Hệ quản trị cơ sở dữ liệu.

Nhờ vào sự chỉ dẫn và giúp đỡ của thầy, nhóm em đã có thể hoàn thành đồ án một cách hiệu quả và đạt được kết quả như mong đợi. Chúng em rất biết ơn thầy đã dành thời gian và tâm huyết để truyền đạt kiến thức và kinh nghiệm cho nhóm.

Thầy đã giúp chúng em có được những kiến thức và kỹ năng cần thiết để làm

việc với cơ sở dữ liệu một cách hiệu quả, và nhóm em sẽ sử dụng những kiến thức này để phát triển bản thân và đóng góp vào công việc sau này.

Một lần nữa, nhóm em xin chân thành cảm ơn thầy Nguyễn Thành Sơn vì sự giúp đỡ và hướng dẫn của thầy trong quá trình học tập của nhóm.

Trân trọng,

Huỳnh Thanh Duy

Nguyễn Trung Hiếu

Lê Văn Phát

Lê Duy Phương

**LỜI NÓI ĐẦU**

Xin chào thầy và các bạn,

Đồ án quản lý quán café “Wibu Coffee” là kết quả của nhóm em sau một thời gian học tập và thực hành môn học Hệ quản trị cơ sở dữ liệu. Đây là một đồ án đầy thử thách và hứa hẹn đem lại cho chúng ta những kiến thức và kinh nghiệm quý giá.

Trong quá trình hoàn thành đồ án, nhóm em đã đối mặt với rất nhiều khó khăn và thách thức. Tuy nhiên, với sự hướng dẫn và hỗ trợ tận tình của giáo viên hướng dẫn cùng với sự nỗ lực không ngừng của chúng em, đồ án đã được hoàn thành một cách thật sự tốt đẹp.

Đồ án quản lý quán cà phê là một ứng dụng thực tiễn, nhằm mục đích giúp chủ quán cà phê có thể quản lý hoạt động của quán một cách dễ dàng và hiệu quả hơn. Bằng việc áp dụng những kiến thức về hệ quản trị cơ sở dữ liệu, chúng em đã xây dựng được một hệ thống quản lý hoàn chỉnh, bao gồm các chức năng quản lý khách hàng, quản lý nhân viên, quản lý thực đơn và các hoạt động kinh doanh khác.

Nhóm em hy vọng rằng đồ án của chúng em sẽ đem lại cho quý thầy cô và các bạn sinh viên những kiến thức và trải nghiệm thực tế hữu ích. Chúng em cũng mong muốn rằng đồ án này sẽ có thể truyền cảm hứng và khơi gợi những ý tưởng sáng tạo cho các bạn sinh viên khác trong việc ứng dụng hệ quản trị cơ sở dữ liệu vào thực tế.

Cảm ơn quý thầy cô và các bạn sinh viên đã theo dõi và đóng góp ý kiến cho đồ án của chúng em.

# **CHƯƠNG 1: TỔNG QUAN VỀ HỆ THỐNG**

1. **Đặc tả đề tài**
   1. **Mô tả bài toán**

Một quán cà phê cần có một hệ thống để quản lý. Chương trình quản lý được xây dựng cho hai đối tượng người dùng là người quản lý và nhân viên với các quyền khác nhau. Đối với người dùng là nhân viên, họ sẽ có quyền truy cập vào các chức năng như bán hàng, xem doanh thu, xem ca làm việc, … Còn với người dùng là người quản lý, họ sẽ có quyền truy cập vào các chức năng như quản lý kho, quản lý doanh thu, quản lý nhân viên, quản lý ca làm việc, … Hệ thống quản lý quán cà phê cần lưu trữ các thông tin sau:

Quán có nhiều nhân viên, một nhân viên chỉ làm một công việc duy nhất, một công việc có thể có nhiều nhân viên cùng làm. Các công việc phân biệt với nhau bằng mã công việc, mỗi công việc gồm các thông tin như mã công việc, tên công việc, lương (theo giờ). Các nhân viên phân biệt với nhau bằng mã nhân viên, mỗi nhân viên gồm các thông tin như mã nhân viên, họ tên, ngày sinh, địa chỉ, sđt, mã công việc, số ca, thưởng, phạt, ngày tuyển dụng.

Các nhân viên sẽ được chia ca để làm việc. Một nhân viên có thể làm nhiều ca, một ca có thể có nhiều nhân viên. Các ca làm việc được phân biệt với nhau qua mã ca, mỗi ca có các thông tin như mã ca, ngày, giờ bắt đầu, giờ kết thúc.

Khách hàng đến quán sẽ được phân biệt với nhau bằng mã khách hàng, thông tin khách hàng gồm mã khách hàng, họ tên khách hàng, sđt và điểm thưởng (được tích lũy sau mỗi lần order và thanh toán hóa đơn, có giá trị khuyến mãi khi sử dụng dịch vụ tại quán trong những lần tiếp theo).

Trong quán cà phê có nhiều bàn khác nhau để khách hàng sử dụng. Các bàn được phân biệt với nhau qua mã bàn, thông tin bàn gồm có mã bàn và trạng thái bàn (đang có người sử dụng hay còn trống).

Sản phẩm được chia thành nhiều loại, các loại sản phẩm phân biệt với nhau bằng mã loại sản phẩm, thông tin loại sản phẩm gồm mã loại sản phẩm, tên loại. Một loại sản phẩm có thể có nhiều sản phẩm, một sản phẩm chỉ thuộc một loại sản phẩm. Các sản phẩm phân biệt với nhau bằng mã sản phẩm. Mỗi sản phẩm có các thông tin như mã sản phẩm, tên sản phẩm, đơn giá, tình trạng, mã loại sản phẩm.

Sản phẩm được làm ra từ các nguyên liệu. Một sản phẩm được làm ra từ nhiều nguyên liệu, một nguyên liệu có thể dùng trong nhiều sản phẩm. Các nguyên liệu phân biệt nhau bằng mã nguyên liệu. Mỗi nguyên liệu có các thông tin như mã nguyên liệu, tên nguyên liệu, số lượng.

Để quản lý nguyên liệu cần có thông tin của nhà sản xuất nguyên liệu và đơn nhập hàng để đảm bảo luôn có đủ nguyên liệu để sản xuất sản phẩm. Một nhà sản xuất có thể sản xuất ra nhiều nguyên liệu, một nguyên liệu cũng có thể được sản xuất từ nhiều nhà sản xuất. Các nhà sản xuất phân biệt với nhau bằng mã nhà sản xuất. Mỗi nhà sản xuất có các thông tin như mã nhà sản xuất, tên nhà sản xuất, địa chỉ, số điện thoại. Đơn nhập hàng được nhân viên nhập hàng tạo và phân biệt bởi mã đơn nhập hàng, mỗi đơn nhập hàng có các thông tin như mã đơn nhập hàng, ngày nhập hàng, trị giá đơn nhập hàng, nhà cung cấp, nhân viên nhập hàng. Chi tiết đơn nhập hàng gồm có nguyên liệu nhập, số lượng từng nguyên liệu, đơn giá.

Có nhiều loại hóa đơn để phân biệt hóa đơn của khách dùng tại quán hay mang đi hay hóa đơn được đặt qua các ứng dụng online như shopee food, grab food, befood … Một loại hóa đơn có thể bao gồm nhiều hóa đơn, một hóa đơn chỉ thuộc một loại hóa đơn duy nhất. Các loại hóa đơn được phân biệt với nhau qua mã loại hóa đơn, thông tin loại hóa đơn gồm mã loại hóa đơn, tên loại, chiết khấu hóa đơn (lưu thông tin phần trăm chiết khấu khi sản phẩm của quán được đặt qua ứng dụng của bên thứ ba). Các hóa đơn ghi lại đơn mua của khách hàng được phân biệt với nhau qua mã hóa đơn, thông tin mỗi hóa đơn gồm các thông tin như mã hóa đơn, thời gian xuất, mã loại, mã bàn, mã khách hàng, trị giá hóa đơn, hình thức thanh toán, tiền khách . Chi tiết hóa đơn gồm có các sản phẩm và số lượng từng sản phẩm, số tiền nhận và số tiền thối lại khách.

Phiếu chi để tính tổng tiền chi phí trong quá trình kinh doanh. Các hóa đơn chi sẽ được quản lý tạo và phân biệt qua mã phiếu chi, thông tin phiếu chi sẽ gồm tên chi phí, giá trị phiếu, ngày thanh toán.

* 1. **Nghiệp vụ bài toán**

Bài toán quản lý quán cà phê là một yêu cầu cần thiết đối với các chủ doanh nghiệp trong lĩnh vực dịch vụ ăn uống. Để giải quyết hiệu quả những khía cạnh phức tạp của hoạt động kinh doanh này, nhiều quán cafe đang tìm hiểu và đưa đến quyết định sử dụng các phần mềm quản lý chuyên nghiệp. Điều này không chỉ giúp tối ưu hóa quy trình làm việc mà còn nâng cao trải nghiệm của khách hàng. Vì vậy cần xây dựng tài liệu đào tạo nghiệp vụ cho nhân viên về quy trình tác nghiệp, văn hóa phục vụ, cách ứng xử, xử lý tình huống, …

**Quản lý kho và nguyên vật liệu:**

Trong một quán cà phê, việc quản lý kho và nguyên vật liệu không chỉ là việc sắp xếp hàng hóa một cách đơn thuần, mà còn là yếu tố quyết định cho chất lượng và sự đồng nhất của mỗi sản phẩm được phục vụ. Bằng cách tổ chức và quản lý kho hàng một cách chặt chẽ, quán không chỉ đảm bảo sự tiện lợi trong công việc mà còn tạo ra niềm tin và sự hài lòng từ phía khách hàng. Đầu tiên phải bắt đầu với việc lập kế hoạch đặt hàng, người quản lý cần phải xác định đâu là những nguyên vật liệu mà quán đang cần như cà phê, sữa, đường, siro, … đảm bảo đủ nguyên liệu để tạo nên các sản phẩm hoàn chỉnh và phải đảm bảo tìm ra nơi cung cấp các nguyên liệu uy tín, chất lượng và giá cả hợp lý. Sau khi lập được kế hoạch đặt hàng và nhập các nguyên vật liệu về, người quản lý cần dựa trên dữ liệu về lượng tiêu thụ và lượng hàng tồn kho qua các ngày, các tuần để cập nhật và đưa ra những quyết định về lượng nhập loại nguyên liệu phù hợp hoặc có thể tinh chỉnh lại kế hoạch nhập hàng hợp lý để tránh tình trạng thiếu hụt hay lãng phí nguyên liệu. Ví dụ như khi một quán cà phê nhận thấy lượng cà phê hạt bắt đầu cạn kiệt, người quản lý sẽ xem xét lịch sử tiêu thụ và dự đoán nhu cầu tiêu thụ trong thời gian tới để đặt hàng kịp thời từ nhà cung cấp, tránh tình trạng thiếu hụt cà phê khiến quán không thể phục vụ khách hàng. Tiếp đến, để đảm bảo chất lượng của mỗi ly cà phê và các sản phẩm sau những lần nhập đều như nhau thì người nhân viên khi nhận hàng cần kiểm tra kỹ lưỡng chất lượng của cà phê hạt bằng cách kiểm tra mẫu và thử nghiệm pha chế một số ly cà phê để đảm bảo vị và mùi hương đúng chuẩn như ban đầu hay không. Nếu không đúng thì báo lại cho bên cung cấp và quản lý, nếu tình trạng kéo dài có thể đổi nhà cung cấp. Đồng thời nhân viên cần phải ghi chép thông tin về đơn hàng như ngày nhận hàng, số lượng, giá cả và nhà cung cấp trong hệ thống quản lý kho của quán để người quản lý dễ dàng quản lý. Sau khi nhập và đảm bảo các nguyên liệu đạt tiêu chuẩn, nhân viên cần phải tổ chức và sắp xếp các nguyên vật liệu vào kho hàng một cách cẩn thận để dễ dàng tìm kiếm và truy cập (có thể sử dụng hệ thống mã vạch hoặc nhãn dán để đánh dấu và phân loại hàng hóa). Ví dụ như khi nhân viên lưu trữ cà phê vào kho hàng thì cà phê có thể được lưu trữ trong các hộp đậu chứa khô, được gắn nhãn theo nguồn gốc và hạn sử dụng để dễ dàng xác định và sử dụng trước hạn. Cuối mỗi chu kỳ nhập hàng, nhân viên quản lý kho cần thực hiện kiểm kê lượng hàng tồn để xác định lượng nguyên vật liệu còn lại trong kho và so sánh với dự định tiêu thụ để đảm bảo đủ cung cấp cho các đơn hàng và khách hàng trong chu kỳ tới. Song với việc quản lý và kiểm kê nguyên liệu thì việc xử lý hàng tồn cũng là một vấn đề quan trọng trong quá trình giải quyết bài toán quản lý quán cà phê ở mảng quản lý kho. Bên đây là ví dụ về việc giải quyết hàng tồn gần đến hạn, quán có thể sử dụng chúng trong các đồ uống đặc biệt hoặc chương trình khuyến mãi để tiêu thụ nhanh chóng trước khi hết hạn sử dụng, tránh lãng phí và tối ưu hóa lợi ích.

**Quản lý nhân sự:**

Mỗi quán/ cafe cần có một đội ngũ nhân viên chuyên nghiệp để đảm bảo hiệu suất làm việc và chất lượng phục vụ cao. Vì vậy khi tuyển dụng, quán cần đưa ra những yêu cầu và các điều kiện để ứng tuyển vào vị trí (kỹ năng nghiệp vụ, kỹ năng giao tiếp, tính kỷ luật, sự cam kết, …) đồng thời quán cần chi trả một mức lương hợp lý cho nhân viên. Nhân viên sau khi đáp ứng được những tiêu chí và được tuyển dụng sẽ trải qua quá trình đào tạo về nghiệp vụ và một số kỹ năng cần thiết. Người quản lý cần theo dõi, quan sát và đánh giá hiệu suất làm việc của từng nhân sự thông qua thái độ làm việc, phản hồi của khách hàng về chất lượng phục vụ, tính kỷ luật trong công việc, … Từ đó chỉ ra những thiếu sót và đưa ra góp ý để nhân sự đó khắc phục những thiếu sót, nếu nhân viên vi phạm quy tắc mà quán đã đề ra trong thỏa thuận thì sẽ bị thêm điểm phạt và sẽ trừ vào lương của nhân viên. Bên cạnh đó có thể đưa ra những đánh giá tích cực về nhân sự có những phấn đấu nổi bật trong quá trình làm việc hay thêm tiền thưởng để tạo động lực cho nhân sự. Ngoài ra quản lý còn phải lên lịch trình làm việc phù hợp với tình hình quán và vào các ngày lễ, Tết. Cụ thể là quán có thể gia tăng số lượng nhân viên làm việc (có thể tuyển thêm nhân viên part time) trong các ca cao điểm như cuối tuần và các ngày lễ, Tết. Điều này giúp đảm bảo rằng quán có đủ nhân lực để phục vụ cho nhu cầu tăng cao của khách hàng.

**Quy trình thực hiện khi khách đến quán:**

Đầu tiên, nhân viên sẽ chào khách khi khách bước vào quán. Sau đó nhân viên đứng quầy sẽ bấm tạo đơn hàng trên chương trình. Tiếp theo nhân viên sẽ hỏi khách hàng cần dùng món gì trong menu và có thể tư vấn các món nếu khách hàng có nhu cầu. Sau khi khách chọn được món thì nhân viên tiến hành thêm các món vào đơn hàng vừa tạo. Tiếp theo, nhân viên hỏi khách có đăng ký thành viên của quán chưa, nếu chưa thì xin số điện thoại của khách để đăng ký thành viên. Sau đó, nhân viên nhập số điện thoại của khách hàng vào để tích điểm. Cuối cùng, nhân viên xuất hóa đơn để đưa cho khách thanh toán và đưa cho khách một tấm thẻ số (tượng trưng cho số bàn đã tạo đơn trên chương trình). Khi này, nhân viên pha chế bắt đầu làm những món có trong đơn hàng của khách. Sau khi hoàn thành các món trong đơn hàng, nhân viên phục vụ đưa món ra cho khách hàng. Nhân viên cần phải kiểm tra đầy đủ các món trong đơn, vật dụng đi kèm với món rồi mới đem đến cho khách hàng.

**Quy trình thực hiện khi có đơn hàng online qua ứng dụng:**

Nhân viên quán sẽ có một thiết bị (điện thoại, Ipad) đăng nhập vào các ứng dụng đặt hàng online của bên thứ ba. Khi có đơn hàng được đặt qua các ứng dụng, thiết bị sẽ nhận được thông báo của các đơn hàng. Nhân viên sẽ bấm tạo đơn ở mục đơn hàng trên chương trình, thêm các món có trong đơn hàng qua ứng dụng. Sau đó, người pha chế sẽ chế biến các món có trong đơn hàng. Cuối cùng, nhân viên xuất hóa đơn và đưa cả món lẫn hóa đơn cho người giao hàng của ứng dụng đó.

**Quản lý tài chính:**

Việc quản lý tài chính là một phần rất quan trọng vì nó quyết định sự duy trì ổn định của quán. Và để giải quyết được bài toán này, người quản lý cần lập kế hoạch ngân sách, cần phải xác định và ghi chép các chi phí cố định và biến đổi như thuê mặt bằng, lương nhân viên, nguyên liệu, tiện ích, và chi phí quảng cáo. Tạo ra một kế hoạch ngân sách hợp lý để quản lý tài chính hiệu quả. Theo dõi doanh thu và chi phí cần sử dụng phần mềm quản lý tài chính hoặc bảng tính để ghi chép và theo dõi doanh thu hàng ngày, hàng tuần và hàng tháng cũng như các chi phí liên quan. Điều này giúp bạn hiểu rõ về hiệu suất kinh doanh và tìm ra cách cải thiện lợi nhuận. Tối ưu hóa quản lý tồn kho cần theo dõi và kiểm soát tồn kho nguyên vật liệu và hàng hóa để tránh lãng phí và thiếu hụt. Điều chỉnh lịch trình đặt hàng và tồn kho để đảm bảo rằng bạn chỉ mua hàng khi cần thiết và không để hàng tồn quá nhiều. Quản lý nợ phải trả và công nợ phải thu bằng cách theo dõi và quản lý các khoản nợ và công nợ để đảm bảo rằng bạn thanh toán các hóa đơn đúng hạn và thu được tiền từ khách hàng kịp thời. Điều này giúp bạn duy trì mối quan hệ tốt với nhà cung cấp và khách hàng. Tối ưu hóa chi phí vận hành cần xem xét các phương pháp tiết kiệm chi phí như tối ưu hóa sử dụng năng lượng, tái sử dụng và tái chế nguyên liệu, và tối ưu hóa lịch trình làm việc của nhân viên để tránh phí trả cao hơn. Phân tích tài chính định kỳ là việc cần thực hiện phân tích tài chính định kỳ để đánh giá hiệu suất kinh doanh, xác định vấn đề và cơ hội, và đưa ra các chiến lược cải thiện.

**Quản lý không gian:**

Trong việc quản lý không gian của quán cà phê, sự sắp xếp thông minh và linh hoạt là chìa khóa để tạo ra một môi trường thoải mái và thu hút khách hàng. Bằng cách tận dụng mọi khoảng trống, tạo ra các khu vực linh hoạt và sử dụng ánh sáng tự nhiên một cách thông minh, quán cà phê không chỉ là một điểm đến lý tưởng để thưởng thức cà phê mà còn là nơi để thư giãn và tận hưởng không gian xanh giữa lòng thành phố ồn ào. Nhân viên được giao nhiệm vụ theo dõi tình trạng của các bàn trong quán. Nếu một bàn trở nên trống sau khi khách rời đi, nhân viên phục vụ sẽ kiểm tra và dọn sạch bàn đó, chuẩn bị sẵn sàng cho khách hàng tiếp theo sử dụng. Nhân viên cần kiểm tra và dọn dẹp khu vực nhà vệ sinh khoảng 1 - 3 tiếng 1 lần. Điều này giúp duy trì sự sạch sẽ và tạo trải nghiệm tích cực cho khách hàng, đặc biệt là những khách hàng mới. Ngoài ra việc tận dụng không gian ngoài trời nếu có thể, sử dụng không gian ngoài trời để tạo ra một không gian yên tĩnh và thoải mái, đặc biệt là vào những ngày thời tiết đẹp. Ngoài ra việc sử dụng các thiết kế mở trong việc trang trí quán như việc sử dụng các kính cửa sổ lớn hoặc tường ngăn cách bằng kính để tạo cảm giác mở rộng không gian và kết nối với bên ngoài, tạo ra một không gian sáng sủa và thân thiện. Các điều trên tạo nên một không gian quán dễ quản lý và dễ chịu cho khách hàng đến quán lẫn nhân viên trong quá trình phục vụ.

* 1. **Mô tả chức năng của bài toán**

Phần mềm quản lý quán cafe cung cấp một loạt các tính năng linh hoạt như quản lý hóa đơn, theo dõi tồn kho, và tính toán tự động giá trị hóa đơn. Khả năng ghi chép hóa đơn một cách nhanh chóng và theo dõi tình trạng hóa đơn từ lúc gọi nước cho đến khi thanh toán giúp giảm thời gian chờ đợi của khách hàng và dễ dàng quản lý không gian quán và tăng cường sự linh hoạt trong phục vụ.

* Đối với nhân viên:

**Chức năng bán hàng:** cho phép nhân viên chọn bàn, tạo hóa đơn, thêm món, xóa món, xuất hóa đơn thanh toán.

**Quản lý khách hàng:** cho phép nhân viên tạo thẻ thành viên cho khách hàng (khi chưa có), nhập số điện thoại ở bước thanh toán để tích điểm cho khách.

**Theo dõi doanh thu theo ca:** cho phép theo dõi doanh thu bán hàng trong ca làm việc của mình.

**Theo dõi kho hàng:** cho phép theo dõi số lượng nguyên liệu tồn trong kho, ngày nhập hàng và hạn sử dụng của nguyên liệu, có nhiệm vụ báo lại cho người quản lý nếu nguyên liệu trong kho sắp hết hoặc bị quá hạn.

**Theo dõi tình trạng không gian quán:** cho phép theo dõi tình trạng (có khách/trống) của tất cả các bàn trong quán.

* Đối với người quản lý: Bao gồm tất cả các chức năng mà nhân viên có thể làm

**Quản lý nhân viên:** cho phép người quản lý nhập, thêm, xóa, sửa thông tin của nhân viên (bao gồm cả quyền truy cập, lương thưởng và tiền phạt). Người quản lý có thể xem các thông tin để liên lạc được với nhân viên như số điện thoại.

**Quản lý ca làm việc của nhân viên:** các nhân viên cần đăng ký các ca làm trong tuần với người quản lý và người quản lý sẽ thêm ca làm việc vào bảng phân ca cho nhân viên. Quản lý có quyền thêm, sửa, xóa ca làm việc của nhân viên trong bảng phân ca làm việc.

**Quản lý thống kê tài chính:** theo dõi, tính toán lượng hàng/nguyên liệu, hóa đơn, doanh thu theo ngày, tuần, tháng hoặc năm, dựa trên thông tin về hóa đơn và giao dịch tài chính.

**Quản lý hóa đơn:** cho phép xem, xuất, xóa hóa đơn đã xuất.

**Quản lý kho hàng:** cho phép thêm, xóa, sửa đơn nhập hàng.

* 1. **Tính lương nhân viên**

**Lương cơ bản:** Đây là số tiền nhân viên được trả cho công việc cũng như phụ thuộc vào việc là nhân viên bán thời gian hoặc toàn thời gian. Đây là số tiền mà nhân viên sẽ chắc chắn được trả hàng tháng (tính theo giờ).

**Loại nhân viên:** Mỗi loại nhân viên sẽ có lương cơ bản cũng như thời gian làm khác nhau.

**Thời gian làm việc:** Là thời gian mà nhân viên làm việc trong tháng (được tính bằng giờ). Đối với mỗi loại nhân viên sẽ có yêu cầu về thời gian hàng tháng khác nhau.

**Lương theo giờ:** Được tính bằng thời gian làm \* lương cơ bản.

**Lương thưởng:** Lương này là lương thưởng thêm, không cố định. Mỗi nhân viên khi làm đủ số giờ quy định trong một tháng sẽ có lương thưởng. Lương thưởng được tính bằng 5% \* lương cố định đối với nhân viên toàn thời gian, mức thưởng cố định cho nhân viên bán thời gian (cụ thể là 200000 đồng).

**Điểm phạt:** Đây cũng là điểm không cố định. Mỗi khi nhân viên vi phạm các quy tắc mà quán đề ra sẽ bị tích một điểm. Số tiền phạt sẽ phụ thuộc vào điểm phạt của từng nhân viên. Với 1 điểm phạt thì trừ số tiền tương ứng với 1 giờ làm của nhân viên.

**Tiền lương 1 tháng của nhân viên** = (Số giờ làm – điểm phạt) \* lương + lương thưởng

**Các mức lương cơ bản:**

Bồi bàn: 23000 đồng/giờ.

Pha chế: 24000 đồng/giờ.

Bấm bill: 25000 đồng/giờ.

**Ví dụ:**

Có 3 nhân viên lần lượt như sau:

- Anh P bồi bàn là nhân viên part time.(Mức lương 1 giờ là 23000 đồng). Anh D là nhân viên pha chế full time (Mức lương 1 giờ là 24000 đồng). Anh H là nhân viên thanh toán (Mức lương 1 giờ là 25000 đồng).

- Chỉ tiêu nhân viên part-time phải trung bình làm 4 tiếng một ngày, quán làm việc cả tháng(trung bình 30 ngày 1 tháng). 4\*30 = 120 tiếng một tháng.

- Chỉ tiêu nhân viên full-time phải trung bình làm 8 tiếng một ngày, quán làm việc cả tháng (trung bình 30 ngày 1 tháng). 8 \* 30 = 240 tiếng một tháng.

- Anh P làm được 121 giờ trong tháng vừa qua, có 1 điểm phạt. Như vậy mức lương của anh ấy sẽ là: (121-1)\*23000 + 150000 = 2910000 đồng.

- Anh D làm được 235 giờ trong tháng vừa qua, có 0 điểm phạt. Như vậy mức lương của anh D sẽ là: 235\*24000 = 5640000 đồng.

- Anh H làm được 240 giờ trong tháng vừa qua, có 0 điểm phạt. Như vậy mức lương của anh H sẽ là: (240\*25000) + (240\*25000\*5%) = 6300000 đồng.

# **CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ HỆ THỐNG**

1. **Thiết kế cơ sở dữ liệu mức quan niệm**
   1. **Các tập thực thể**

A diagram of a company

Description automatically generated with medium confidence

* 1. **Mối quan hệ các tập thực thể**

Từ các tập thực thể ở 1.1, ta có sơ đồ thực thể kết hợp (ERD):

**A diagram of a flowchart

Description automatically generated**

1. **Thiết kế cơ sở dữ liệu mức logic**

Từ sơ đồ thực thể kết hợp (ERD), ta có các lược đồ quan hệ:

* Employee(ID, name, birthDate, address, phone, recruitmentDate, jobID, penaltyPoint, bonusPoint, numberOfShift)
* Job(ID, jobDetail, salary)
* Shift(ID, startTime, endTime)
* On\_Duty(empID, shiftID, date)
* Table(ID, name, status)
* Customer(ID, name, phone, bonusPoint)
* BillCategory(ID, name, discount)
* Bill(ID, dateTime, tableID, customerID, categoryID, empID, receiptMoney)
* ProductCategory(ID, name)
* Product(ID, name, categoryID, price, status)
* BillInfo(billID, productID, quantity)
* Material(ID, name, status)
* Product\_Material(productID, materialID, quantity)
* ReceiptNote(ID, date, price, supplierID, empID)
* ReceiptNoteDetail(rNoteID, materialID, quantity, unitPrice)
* Supplier(ID, name, address, phone)
* Delivery(materialID, supplierID)
* ExpenseBill(ID, detail, price, date)
* Account(userName, password, role)

1. **Các ràng buộc cần có**

|  |  |  |
| --- | --- | --- |
| **STT** | **Quan hệ** | **Ràng buộc** |
| 1 | Employee | Khóa chính ID,  khoá ngoại jobID đến Job.ID,  name NOT NULL,  check (name <> ‘’),  birthDate NOT NULL,  address NOT NULL,  phone NOT NULL,  check LENGTH(phone) = 10,  jobID NOT NULL,  check (jobID <> ‘’),  numberOfShift NOT NULL,  recruitmentDate NOT NULL |
| 2 | Job | Khóa chính ID,  jobDetail NOT NULL,  salary NOT NULL,  check salary > 0 |
| 3 | Shift | Khóa chính ID,  startTime NOT NULL,  endTime NOT NULL |
| 4 | On\_Duty | Khóa ngoại empID đến Employee.ID,  Khóa ngoại shiftID đến Shift.ID,  Khoá chính (empID, shiftID, date) |
| 5 | Table | Khóa chính ID,  name NOT NULL check (name <> N‘’),  status NOT NULL check (status <> N‘’) |
| 6 | Customer | Khóa chính ID,  name NOT NULL check (name <> N‘’),  phone NOT NULL check (LEN(phone) = 10) |
| 7 | BillCategory | Khóa chính ID,  name NOT NULL check (name <> N‘’),  discount NOT NULL |
| 8 | Bill | Khóa chính ID,  khóa ngoại tableID đến Table.ID,  khoá ngoại categoryID đến BillCategory.ID,  khóa ngoại customerID đến Customer.ID,  khoá ngoại empID đến Employee.ID,  dateTime NOT NULL,  tableID NOT NULL,  customerID NOT NULL,  categoryID NOT NULL,  empID NOT NULL |
| 9 | ProductCategory | Khoá chính ID,  name NOT NULL check (name <> N‘’) |
| 10 | Product | Khoá chính ID,  khoá ngoại categoryID đến ProductCategory.ID,  name NOT NULL check (name <> N‘’),  categoryID NOT NULL,  price NOT NULL,  status NOT NULL |
| 11 | BillInfo | khoá ngoại billID đến Bill.ID,  khoá ngoại productID đến Product.ID,  khoá chính (billID, productID),  quantity NOT NULL |
| 12 | Material | Khoá chính ID,  name NOT NULL check (name <> N‘’),  status NOT NULL |
| 13 | Product\_Material | khoá ngoại productID đến product.ID,  khoá ngoại materialID đến Material.ID,  khoá chính (productID, materialID),  quantity NOT NULL |
| 14 | ReceiptNote | khoá chính ID,  khoá ngoại supplierID đến Supplier.ID,  khoá ngoại empID đến Employee.ID,  date NOT NULL,  supplierID NOT NULL,  empID NOT NULL |
| 15 | ReceiptNoteDetail | khoá ngoại rNoteID đến ReceiptNote.ID,  khoá ngoại materialID đến Material.ID,  khoá chính (rNoteID, materialID),  quantity NOT NULL,  unitPrice NOT NULL |
| 16 | Supplier | khoá chính ID,  name NOT NULL check (name <> N‘’),  address NOT NULL check (address <> N‘’),  phone NOT NULL check LEN(phone) = 10 |
| 17 | Delivery | Khóa ngoại materialID đến Material.ID,  Khóa ngoại supplierID đến Supplier.ID,  Khóa chính (materialID, supplierID) |
| 18 | ExpenseBill | khoá chính ID,  detail NOT NULL check (detail <> N‘’),  price NOT NULL,  date NOT NULL |
| 19 | Account | khóa chính userName,  pass NOT NULL check (pass <> ‘’),  role NOT NULL check (role >=0 AND role <=1) |

1. **Cài đặt các CSDL và các ràng buộc**

***Bảng công việc***

|  |
| --- |
| CREATE TABLE Job (  ID VARCHAR(3) PRIMARY KEY, -- J01, J02, ...  jobDetail NVARCHAR(max) NOT NULL,  salary DECIMAL(10, 2) NOT NULL CHECK (salary > 0)  ); |

***Bảng nhân viên***

|  |
| --- |
| CREATE TABLE Employee (  ID VARCHAR(4) PRIMARY KEY, -- E01, E02, ...  name NVARCHAR(max) NOT NULL CHECK (name != ''),  birthDate DATE NOT NULL,  address NVARCHAR(max) NOT NULL,  phone VARCHAR(10) NOT NULL CHECK (LEN(phone) = 10),  recruitmentDate DATE NOT NULL,  jobID VARCHAR(3) NOT NULL CHECK (jobID != ''),  penaltyPoint INT,  bonusPoint INT,  numberOfShift INT NOT NULL,  FOREIGN KEY (jobID) REFERENCES Job(ID)  ON UPDATE CASCADE  ); |

***Bảng ca làm việc***

|  |
| --- |
| CREATE TABLE [Shift] (  ID VARCHAR(5) PRIMARY KEY, -- S0001, S0002, ...  startTime TIME NOT NULL ,  endTime TIME NOT NULL  ); |

***Bảng nhà cung cấp***

|  |
| --- |
| CREATE TABLE Supplier (  ID VARCHAR(4) PRIMARY KEY, -- SP01, SP02, ...  name NVARCHAR(max) NOT NULL CHECK (name != N''),  address NVARCHAR(max) NOT NULL CHECK (address != N''),  phone VARCHAR(10) CHECK (LEN(phone) = 10) NOT NULL  ); |

***Bảng phân công ca làm việc***

|  |
| --- |
| CREATE TABLE On\_Duty (  empID VARCHAR(4),  shiftID VARCHAR(5),  date DATE,  FOREIGN KEY (empID) REFERENCES Employee(ID)  ON UPDATE CASCADE,  FOREIGN KEY (shiftID) REFERENCES [Shift](ID)  ON UPDATE CASCADE,  PRIMARY KEY (empID, shiftID, date)  ); |

***Bảng thông tin bàn***

|  |
| --- |
| CREATE TABLE [Table] (  ID VARCHAR(3) PRIMARY KEY, -- T01, T02, ...  name NVARCHAR(max) NOT NULL CHECK (name != N''),  status NVARCHAR(max) NOT NULL CHECK (status != N'')  ); |

***Bảng khách hàng***

|  |
| --- |
| CREATE TABLE Customer (  ID VARCHAR(5) PRIMARY KEY, -- C0001, C0002, ...  name NVARCHAR(max) NOT NULL CHECK (name != N''),  phone VARCHAR(10) NOT NULL CHECK (LEN(phone) = 10),  bonusPoint INT  ); |

***Bảng loại hóa đơn***

|  |
| --- |
| CREATE TABLE BillCategory (  ID VARCHAR(4) PRIMARY KEY, -- BC01, ...  name NVARCHAR(max) NOT NULL CHECK (name != N''),  discount DECIMAL(5, 2) NOT NULL  ); |

***Bảng hóa đơn***

|  |
| --- |
| CREATE TABLE Bill (  ID VARCHAR(7) PRIMARY KEY, -- B000001  dateTime DATETIME NOT NULL,  tableID VARCHAR(3) NOT NULL,  customerID VARCHAR(5) NOT NULL CHECK (customerID != ''),  categoryID VARCHAR(4) NOT NULL CHECK (categoryID != ''),  empID VARCHAR(4) NOT NULL CHECK (empID != ''),  receiptMoney DECIMAL(10, 2) NOT NULL,  FOREIGN KEY (tableID) REFERENCES [Table](ID)  ON UPDATE CASCADE,  FOREIGN KEY (customerID) REFERENCES Customer(ID)  ON UPDATE CASCADE,  FOREIGN KEY (categoryID) REFERENCES BillCategory(ID)  ON UPDATE CASCADE,  FOREIGN KEY (empID) REFERENCES Employee(ID)  ON UPDATE CASCADE  ); |

***Bảng loại món***

|  |
| --- |
| CREATE TABLE ProductCategory (  ID VARCHAR(4) PRIMARY KEY, -- PC01, PC02, ...  name VARCHAR(max) NOT NULL CHECK (name != '')  ); |

***Bảng thông tin món***

|  |
| --- |
| CREATE TABLE Product (  ID VARCHAR(3) PRIMARY KEY, -- P01, P02, ...  name NVARCHAR(max) NOT NULL CHECK (name != N''),  categoryID VARCHAR(4) NOT NULL,  price DECIMAL(10, 2) NOT NULL,  status INT NOT NULL,  FOREIGN KEY (categoryID) REFERENCES ProductCategory(ID)  ON UPDATE CASCADE  ); |

***Bảng chi tiết hóa đơn***

|  |
| --- |
| CREATE TABLE BillInfo (  billID VARCHAR(7) ,  productID VARCHAR(3),  quantity INT NOT NULL,  FOREIGN KEY (billID) REFERENCES Bill(ID)  ON UPDATE CASCADE,  FOREIGN KEY (productID) REFERENCES Product(ID)  ON UPDATE CASCADE,  PRIMARY KEY (billID, productID)  ); |

***Bảng nguyên vật liệu***

|  |
| --- |
| CREATE TABLE Material (  ID VARCHAR(3) PRIMARY KEY, -- M01, M02, ...  name NVARCHAR(max) NOT NULL CHECK (name != ''),  status DECIMAL(10,2) NOT NULL  ); |

***Bảng Sản phẩm – Nguyên vật liệu***

|  |
| --- |
| CREATE TABLE Product\_Material (  productID VARCHAR(3) NOT NULL CHECK (productID != ''),  materialID VARCHAR(3) NOT NULL CHECK (materialID != ''),  quantity INT NOT NULL,  FOREIGN KEY (productID) REFERENCES Product(ID)  ON UPDATE CASCADE,  FOREIGN KEY (materialID) REFERENCES Material(ID)  ON UPDATE CASCADE,  PRIMARY KEY (productID, materialID)  ); |

***Bảng phiếu nhập***

|  |
| --- |
| CREATE TABLE ReceiptNote (  ID VARCHAR(7) PRIMARY KEY,  date DATE NOT NULL,  price DECIMAL(10, 2),  supplierID VARCHAR(4) NOT NULL,  empID VARCHAR(4) NOT NULL,  FOREIGN KEY (supplierID) REFERENCES Supplier(ID)  ON UPDATE CASCADE,  FOREIGN KEY (empID) REFERENCES Employee(ID)  ON UPDATE CASCADE,  ); |

***Bảng chi tiết phiếu nhập***

|  |
| --- |
| CREATE TABLE ReceiptNoteDetail (  rNoteID VARCHAR(7),  materialID VARCHAR(3),  quantity INT NOT NULL CHECK (quantity > 0),  unitPrice DECIMAL(10, 2) NOT NULL CHECK (unitPrice > 0),  FOREIGN KEY (rNoteID) REFERENCES ReceiptNote(ID)  ON UPDATE CASCADE,  FOREIGN KEY (materialID) REFERENCES Material(ID)  ON UPDATE CASCADE,  PRIMARY KEY (rNoteID, materialID)  ); |

***Bảng chi tiết vận chuyển***

|  |
| --- |
| CREATE TABLE Delivery (  materialID VARCHAR(3),  supplierID VARCHAR(4),  FOREIGN KEY (materialID) REFERENCES Material(ID)  ON UPDATE CASCADE,  FOREIGN KEY (supplierID) REFERENCES Supplier(ID)  ON UPDATE CASCADE,  PRIMARY KEY (materialID, supplierID)  ); |

***Bảng phiếu chi***

|  |
| --- |
| CREATE TABLE ExpenseBill (  ID VARCHAR(7) PRIMARY KEY, -- EX00001, EX00002  detail NVARCHAR(max) NOT NULL CHECK (detail != N''),  price DECIMAL(18, 2) NOT NULL CHECK (price > 0),  date DATE NOT NULL  ); |

***Bảng tài khoản***

|  |
| --- |
| CREATE TABLE Account (  userName VARCHAR(10) PRIMARY KEY,  pass VARCHAR(max) NOT NULL CHECK (pass != ''),  userRole INT NOT NULL CHECK (userRole >= 0 AND userRole <= 1)); |

***Diagram:***

A screenshot of a computer

Description automatically generated

# **CHƯƠNG 3: THIẾT KẾ CÁC CHỨC NĂNG**

1. **Kết nối Cơ sở dữ liệu:**

SQL:

/\* CHECK LOGIN \*/

CREATE FUNCTION checkLogin

(@userName VARCHAR(10),

@pass VARCHAR(MAX))

RETURNS VARCHAR(MAX)

AS

BEGIN

DECLARE @userRole VARCHAR(MAX)

IF EXISTS (SELECT \* FROM Account WHERE userName = @userName AND pass = @pass)

SELECT @userRole = userRole FROM Account WHERE userName = @userName

RETURN ISNULL(@userRole, 'Invalid')

END

GO

C# code:

public Login()

{

InitializeComponent();

// find the servername.txt file in the same directory as the executable file, if not found, enable the txbServerName to input the server name

if (!System.IO.File.Exists("servername.txt"))

{

pnlServerName.Enabled = true;

}

else

{

pnlServerName.Enabled = false;

}

if(System.IO.File.Exists("servername.txt") && System.IO.File.ReadAllText("servername.txt").Length > 0)

{

txbServerName.Text = System.IO.File.ReadAllText("servername.txt");

}

else if(System.IO.File.Exists("servername.txt") && System.IO.File.ReadAllText("servername.txt").Length == 0)

{

pnlServerName.Enabled = true;

}

}

// Event handler for login

private void btnLogin\_Click(object sender, EventArgs e)

{

object result;

try

{

if (pnlServerName.Enabled)

{

string serverName = txbServerName.Text;

// write the server name to the servername.txt file

System.IO.File.WriteAllText("servername.txt", serverName);

}

if (System.IO.File.Exists("servername.txt") && System.IO.File.ReadAllText("servername.txt").Length > 0)

{

try

{

string username = txbUserName.Text;

string password = txbPassword.Text;

// write the user name to the username.txt file

System.IO.File.WriteAllText("username.txt", username);

// write the password to the password.txt file

System.IO.File.WriteAllText("password.txt", password);

bool isAdmin = false;

result = DataProvider.Instance.ExecuteScalar("SELECT dbo.checkLogin( @username , @password )", new object[] { username, password });

if (result.ToString() != "Invalid")

{

if (result.Equals("0"))

{

isAdmin = true;

}

else if (result.Equals("1"))

{

isAdmin = false;

}

this.Hide();

MainWindow mainWindow = new MainWindow(isAdmin);

mainWindow.Show();

}

else

{

MessageBox.Show("Sai tên đăng nhập hoặc mật khẩu!");

}

}

catch (SqlException err)

{

MessageBox.Show(Text = "Error from SQL!\n" + err.Message);

}

}

else

{

MessageBox.Show("Không tìm thấy file servername.txt");

return;

}

}

catch (SqlException err)

{

MessageBox.Show("Error from SQL!\n" + err.Message);

// delete the servername.txt file

System.IO.File.Delete("servername.txt");

pnlServerName.Enabled = true;

return;

}

}

**DataProvider:**

// get the server name from the servername.txt file

private static string dataSource = System.IO.File.ReadAllText("servername.txt");

private static string user = System.IO.File.ReadAllText("username.txt");

private static string pass = System.IO.File.ReadAllText("password.txt");

public static string connectionString = "Data Source=" + dataSource + ";Initial Catalog=WibuCoffee;User ID=" + user + ";Password=" + pass;

private static DataProvider instance;

public static DataProvider Instance

{

get

{

if (instance == null)

instance = new DataProvider();

return DataProvider.instance;

}

private set

{

DataProvider.instance = value;

}

}

private DataProvider() { }

public DataTable ExecuteQuery(string query, object[] para = null)

{

DataTable data = new DataTable();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlCommand command = new SqlCommand(query, connection);

if(para != null)

{

string[] listPara = query.Split(' ');

int i = 0;

foreach(string item in listPara)

{

if(item.Contains('@'))

{

command.Parameters.AddWithValue(item, para[i]);

i++;

}

}

}

SqlDataAdapter adapter = new SqlDataAdapter(command);

adapter.Fill(data);

connection.Close();

}

return data;

}

public int ExecuteNonQuery(string query, object[] para = null)

{

int data = 0;

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlCommand command = new SqlCommand(query, connection);

if (para != null)

{

string[] listPara = query.Split(' ');

int i = 0;

foreach (string item in listPara)

{

if (item.Contains('@'))

{

command.Parameters.AddWithValue(item, para[i]);

i++;

}

}

}

data = command.ExecuteNonQuery();

connection.Close();

}

return data;

}

public object ExecuteScalar(string query, object[] para = null)

{

object data = 0;

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlCommand command = new SqlCommand(query, connection);

if (para != null)

{

string[] listPara = query.Split(' ');

int i = 0;

foreach (string item in listPara)

{

if (item.Contains('@'))

{

command.Parameters.AddWithValue(item, para[i]);

i++;

}

}

}

data = command.ExecuteScalar();

connection.Close();

}

return data;

}

1. **Các chức năng của chương trình:**
3. 1. **Dashboard**
      1. **Views:**
4. **View hiển thị danh sách tổng hóa đơn: Mã hóa đơn, mã loại hóa đơn, ngày xuất hóa đơn, tổng giá trị hóa đơn, chiết khấu**

SQL:

CREATE VIEW BillDetailView

AS

SELECT BillInfo.billID,

Product.name,

BillInfo.quantity,

Product.price \* BillInfo.quantity AS totalPrice

FROM BillInfo, Product

WHERE BillInfo.productID = Product.ID

GO

/\* VIEW TÍNH VÀ HIỂN THỊ TỔNG TIỀN VÀ CHIẾC KHẤU CỦA TỪNG HÓA ĐƠN.

NẾU HÓA ĐƠN LÀ BC01 VÀ BC02 THÌ CĂN CỨ THEO ID KHÁCH HÀNG TÌM RA BONUSPOINT CỦA HỌ

KHI BONUSPOINT BẰNG 200đ = chiết khấu 5% tổng bill

400đ = chiết khấu 10% tổng bill

500đ = chiết khấu 15% tổng bill

NẾU HÓA ĐƠN LÀ BC03 VÀ BC04 THÌ CĂN CỨ VÀO DISCOUNT TRONG BILLCATEGORY

CUỐI CÙNG LẤY DISCOUNT NHÂN VỚI TỔNG GIÁ TIỀN ĐỂ RA CHIẾC KHẤU\*/

CREATE VIEW BillTotalPriceView

AS

SELECT Bill.ID,

Bill.categoryID,

Bill.dateTime,

SUM(totalPrice) AS totalPrice,

CASE

WHEN Bill.categoryID = 'BC01' OR Bill.categoryID = 'BC02' THEN

CASE

WHEN Customer.bonusPoint < 200 THEN 0

WHEN Customer.bonusPoint >= 200 AND Customer.bonusPoint < 400 THEN SUM(totalPrice) \* 0.05

WHEN Customer.bonusPoint >= 400 AND Customer.bonusPoint < 500 THEN SUM(totalPrice) \* 0.1

WHEN Customer.bonusPoint >= 500 THEN SUM(totalPrice) \* 0.15

END

WHEN Bill.categoryID = 'BC03' OR Bill.categoryID = 'BC04' THEN SUM(totalPrice) \* BillCategory.discount

END AS discount

FROM Bill, BillDetailView, Customer, BillCategory

WHERE Bill.ID = BillDetailView.billID

AND Bill.customerID = Customer.ID

AND Bill.categoryID = BillCategory.ID

GROUP BY Bill.ID, Bill.categoryID, Bill.dateTime, Customer.bonusPoint, BillCategory.discount

GO

1. **View hiển thị top 5 món bán chạy nhất**

SQL:

CREATE VIEW MostPopularProducts

AS

SELECT TOP 5 p.ID, p.name, p.price, SUM(bi.quantity) as totalQuantity

FROM Product p

JOIN BillInfo bi

ON p.ID = bi.productID

GROUP BY p.ID, p.name, p.price

ORDER BY totalQuantity DESC;

GO

C# code:

try

{

// get view MostPopularProducts from database and set to topProducts

top5Products = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.MostPopularProducts");

// Check if there are enough rows returned

if (top5Products.Rows.Count >= 5)

{

// Set the text for lbTop1, lbTop2, lbTop3, lbTop4, lbTop5 from top5Products with format: "Quantity - Price - Product Name"

lbTop1.Text = top5Products.Rows[0]["ID"] + " - " + top5Products.Rows[0]["name"].ToString() + " - "

+ top5Products.Rows[0]["price"] + "VNĐ - Đã bán: " + top5Products.Rows[0]["totalQuantity"].ToString();

lbTop2.Text = top5Products.Rows[1]["ID"] + " - " + top5Products.Rows[1]["name"].ToString() + " - "

+ top5Products.Rows[1]["price"] + "VNĐ - Đã bán: " + top5Products.Rows[1]["totalQuantity"].ToString();

lbTop3.Text = top5Products.Rows[2]["ID"] + " - " + top5Products.Rows[2]["name"].ToString() + " - "

+ top5Products.Rows[2]["price"] + "VNĐ - Đã bán: " + top5Products.Rows[2]["totalQuantity"].ToString();

lbTop4.Text = top5Products.Rows[3]["ID"] + " - " + top5Products.Rows[3]["name"].ToString() + " - "

+ top5Products.Rows[3]["price"] + "VNĐ - Đã bán: " + top5Products.Rows[3]["totalQuantity"].ToString();

lbTop5.Text = top5Products.Rows[4]["ID"] + " - " + top5Products.Rows[4]["name"].ToString() + " - "

+ top5Products.Rows[4]["price"] + "VNĐ - Đã bán: " + top5Products.Rows[4]["totalQuantity"].ToString();

}

else

{

// Handle the case where less than 5 products are returned

// You may want to clear the labels or set them to a default text like "N/A"

lbTop1.Text = lbTop2.Text = lbTop3.Text = lbTop4.Text = lbTop5.Text = "N/A";

}

}

catch (SqlException err)

{

MessageBox.Show(err.Message);

}

1. **View hiển thị tổng thu chi trong 7 ngày gần nhất. Dùng để vẽ biểu đồ**

SQL:

CREATE VIEW BillExpenseCount7DaysAgo

AS

-- Định nghĩa CTE (Common Table Expression, dùng để tạo một bảng tạm thời) đầu tiên để tạo một dãy số từ 0 đến 6

WITH NumberSeries AS (

SELECT 0 AS Number -- Bắt đầu dãy số với 0

UNION ALL SELECT 1 -- Thêm số 1 vào dãy

UNION ALL SELECT 2 -- Thêm số 2 vào dãy

UNION ALL SELECT 3 -- Thêm số 3 vào dãy

UNION ALL SELECT 4 -- Thêm số 4 vào dãy

UNION ALL SELECT 5 -- Thêm số 5 vào dãy

UNION ALL SELECT 6 -- Thêm số 6 vào dãy, hoàn thành dãy số từ 0 đến 6

),

-- Định nghĩa CTE thứ hai để tạo chuỗi ngày từ ngày hiện tại trở về trước 6 ngày

DateSeries AS (

SELECT CAST(DATEADD(DAY, -Number, GETDATE()) AS DATE) AS DateValue -- Tính toán ngày bằng cách trừ số ngày (Number) từ ngày hiện tại

FROM NumberSeries -- Sử dụng dãy số đã tạo ở CTE trên để tính ngày

)

-- Chọn ngày và tính tổng giá trị hóa đơn và chi phí cho mỗi ngày trong khoảng 7 ngày

SELECT

ds.DateValue AS [Date], -- Lấy giá trị ngày từ CTE DateSeries

-- Tính tổng giá trị hóa đơn cho mỗi ngày, nếu không có hóa đơn thì trả về 0

(SELECT ISNULL(SUM(totalPrice), 0) FROM BillTotalPriceView WHERE CAST(dateTime AS DATE) = ds.DateValue) AS TotalBill,

-- Tính tổng chi phí cho mỗi ngày, nếu không có chi phí thì trả về 0

(SELECT ISNULL(SUM(price),0) FROM ExpenseBill WHERE CAST(date AS DATE) = ds.DateValue) AS TotalExpense

FROM

DateSeries ds; -- Sử dụng CTE DateSeries để lặp qua từng ngày

GO

C# code:

try

{

moneyRecents = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.BillExpenseCount7DaysAgo");

if (moneyRecents.Rows.Count >= 7)

{

// get the Date of the first row in moneyRecents and set to lbDate1, lbDate2, lbDate3, lbDate4, lbDate5, lbDate6, lbDate7

lbDate1.Text = ((DateTime)moneyRecents.Rows[0]["Date"]).ToString("dd/MM/yyyy");

lbDate2.Text = ((DateTime)moneyRecents.Rows[1]["Date"]).ToString("dd/MM/yyyy");

lbDate3.Text = ((DateTime)moneyRecents.Rows[2]["Date"]).ToString("dd/MM/yyyy");

lbDate4.Text = ((DateTime)moneyRecents.Rows[3]["Date"]).ToString("dd/MM/yyyy");

lbDate5.Text = ((DateTime)moneyRecents.Rows[4]["Date"]).ToString("dd/MM/yyyy");

lbDate6.Text = ((DateTime)moneyRecents.Rows[5]["Date"]).ToString("dd/MM/yyyy");

lbDate7.Text = ((DateTime)moneyRecents.Rows[6]["Date"]).ToString("dd/MM/yyyy");

int max = 550;

for (int i = 0; i < 7; i++)

{

decimal totalBill = Convert.ToDecimal(moneyRecents.Rows[i]["TotalBill"]);

decimal totalExpense = Convert.ToDecimal(moneyRecents.Rows[i]["TotalExpense"]);

decimal total = totalBill + totalExpense;

int incomeWidth = (total > 0) ? (int)((totalBill / total) \* max) : 0;

int expenseWidth = (total > 0) ? (int)((totalExpense / total) \* max) : 0;

if (incomeWidth + expenseWidth >= max)

{

double scaleFactor = (double)max / (incomeWidth + expenseWidth);

incomeWidth = (int)(incomeWidth \* scaleFactor);

expenseWidth = (int)(expenseWidth \* scaleFactor);

}

string tooltipText = $"Total Bill: {totalBill} VNĐ\nTotal Expense: {totalExpense} VNĐ";

btnToolTip.SetToolTip(btnMoneyChart\_Income[i], tooltipText);

btnToolTip.SetToolTip(btnMoneyChart\_Expense[i], tooltipText);

// Set width and position of income button

btnMoneyChart\_Income[i].Width = incomeWidth;

// Set width and position of expense button

btnMoneyChart\_Expense[i].Width = expenseWidth;

btnMoneyChart\_Expense[i].Left = btnMoneyChart\_Income[i].Left + btnMoneyChart\_Income[i].Width;

}

}

}

catch (SqlException err)

{

MessageBox.Show(err.Message);

}

1. **View hiển thị tổng thu nhập theo 12 tháng trong năm hiện tại**

SQL:

CREATE VIEW MonthlyTotalPriceBillView

AS

-- Định nghĩa một CTE để tạo ra một dãy số từ 0 đến 11

WITH NumberSeries AS (

SELECT 0 AS Number -- Đại diện cho tháng đầu tiên của năm hiện tại

UNION ALL SELECT 1

UNION ALL SELECT 2

UNION ALL SELECT 3

UNION ALL SELECT 4

UNION ALL SELECT 5

UNION ALL SELECT 6

UNION ALL SELECT 7

UNION ALL SELECT 8

UNION ALL SELECT 9

UNION ALL SELECT 10

UNION ALL SELECT 11 -- Đại diện cho tháng cuối cùng của năm hiện tại

),

-- Định nghĩa CTE thứ hai để tạo ra một chuỗi các giá trị tháng từ số đã cho

MonthSeries AS (

SELECT DATEADD(MONTH, Number, DATEADD(YEAR, DATEDIFF(YEAR, 0, GETDATE()), 0)) AS MonthValue

FROM NumberSeries -- Thêm mỗi giá trị Number vào đầu năm hiện tại để tạo ra một ngày trong từng tháng

)

-- Thực hiện truy vấn để lấy tổng giá trị hóa đơn cho mỗi tháng

SELECT

MONTH(ms.MonthValue) AS [Month], -- Lấy số tháng từ giá trị ngày

-- Subquery để tính tổng giá trị hóa đơn cho mỗi tháng tương ứng

(SELECT ISNULL(SUM(totalPrice), 0)

FROM BillTotalPriceView -- Sử dụng VIEW lưu trữ tổng giá trị của các hóa đơn

WHERE MONTH(dateTime) = MONTH(ms.MonthValue) -- So sánh tháng của hóa đơn với tháng hiện tại của CTE

AND YEAR(dateTime) = YEAR(ms.MonthValue) -- So sánh năm của hóa đơn với năm của ngày trong CTE

) AS TotalPrice

FROM MonthSeries ms; -- Lặp qua mỗi giá trị tháng trong MonthSeries

GO

C# code:

try

{

incomeMonthly = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.MonthlyTotalPriceBillView");

//find max value of incomeMonthly

decimal maxIncome = 0;

for (int i = 0; i < 12; i++)

{

decimal total = Convert.ToDecimal(incomeMonthly.Rows[i]["TotalPrice"]);

if (total > maxIncome)

{

maxIncome = total;

}

}

if (incomeMonthly.Rows.Count >= 12)

{

int max = 200;

for (int i = 0; i < 12; i++)

{

decimal total = Convert.ToDecimal(incomeMonthly.Rows[i]["TotalPrice"]);

int height = (maxIncome > 0) ? (int)(total / maxIncome \* max) : 0;

btnIncomeChart[i].Location = new Point(btnIncomeChart\_point[i].X, 726 - height);

btnIncomeChart[i].Height = height;

if (height == 0)

{

lbIncomeChart[i].Location = btnIncomeChart\_point[i];

lbIncomeChart[i].Text = "0 VNĐ";

}

else

{

// Set the location of the lbIncomeChart[i] to the top of the btnIncomeChart[i]

lbIncomeChart[i].Top = btnIncomeChart[i].Top - lbIncomeChart[i].Height;

lbIncomeChart[i].Text = ((int)(total / 100)).ToString() + "k";

}

}

}

}

catch (SqlException err)

{

MessageBox.Show(err.Message);

}

* + 1. **Functions**

**Function tính tổng thu nhập theo ngày, tổng chi tiêu theo ngày, tổng lợi nhuận tháng hiện tại và số bàn trống hiện tại**

SQL:

CREATE FUNCTION getTotalPriceBillbyDate

(@date date)

RETURNS DECIMAL(10,0)

BEGIN

DECLARE @total DECIMAL(10,0);

SELECT @total = SUM(totalPrice)

FROM BillTotalPriceView

WHERE CAST(dateTime AS date) = @date;

RETURN ISNULL(@total, 0); -- Trả về 0 nếu không có dữ liệu

END;

GO

CREATE FUNCTION getTotalExpensebyDate

(@date date)

RETURNS DECIMAL(10,0)

AS

BEGIN

DECLARE @total DECIMAL(10,0);

SELECT @total = SUM(price)

FROM ExpenseBill

WHERE date = @date;

RETURN ISNULL(@total, 0);

END;

GO

CREATE FUNCTION getTotalRevenue1MonthAgo()

RETURNS DECIMAL(18,2) -- Increased precision for financial calculations

AS

BEGIN

DECLARE @totalRevenue DECIMAL(18,2);

DECLARE @totalBills DECIMAL(18,2);

DECLARE @totalExpenses DECIMAL(18,2);

DECLARE @totalReceiptNotes DECIMAL(18,2);

-- Calculate total price from bills in the current month

SELECT @totalBills = ISNULL(SUM(totalPrice), 0)

FROM BillTotalPriceView

WHERE MONTH(dateTime) = MONTH(GETDATE())

AND YEAR(dateTime) = YEAR(GETDATE());

-- Calculate total price from expense bills in the current month

SELECT @totalExpenses = ISNULL(SUM(price), 0)

FROM ExpenseBill

WHERE MONTH(date) = MONTH(GETDATE())

AND YEAR(date) = YEAR(GETDATE());

-- Calculate total price from receipt notes in the current month

SELECT @totalReceiptNotes = ISNULL(SUM(unitPrice \* quantity), 0)

FROM ReceiptNoteDetail

JOIN ReceiptNote ON ReceiptNoteDetail.rNoteID = ReceiptNote.ID

WHERE MONTH(ReceiptNote.date) = MONTH(GETDATE())

AND YEAR(ReceiptNote.date) = YEAR(GETDATE());

-- Calculate total revenue

SET @totalRevenue = @totalBills - @totalExpenses - @totalReceiptNotes;

RETURN @totalRevenue;

END;

GO

CREATE FUNCTION getEmptyTableCount()

RETURNS INT

AS

BEGIN

DECLARE @avail INT;

SELECT @avail = COUNT(\*)

FROM dbo.[Table]

WHERE [status] = 'Available';

RETURN ISNULL(@avail, 0);

END;

GO

C# code:

// get today date

DateTime today = DateTime.Now;

decimal todayIncome = 0;

decimal todayExpense = 0;

decimal monthRevenue = 0;

int emptyTable = 0;

try

{

// get today income

todayIncome = (decimal)DataProvider.Instance.ExecuteScalar("SELECT dbo.getTotalPriceBillbyDate ( @date )", new object[] { today.Date.ToString("yyyy-MM-dd") });

todayExpense = (decimal)DataProvider.Instance.ExecuteScalar("SELECT dbo.getTotalExpensebyDate ( @date )", new object[] { today.Date.ToString("yyyy-MM-dd") });

monthRevenue = (decimal)DataProvider.Instance.ExecuteScalar("SELECT dbo.getTotalRevenue1MonthAgo ()");

emptyTable = (int)DataProvider.Instance.ExecuteScalar("SELECT dbo.getEmptyTableCount ()");

}

catch (SqlException err)

{

MessageBox.Show(err.Message);

}

* 1. **Chức năng Order:**
     1. **View:**

1. **View hiển thị danh sách hóa đơn:**

SQL:

CREATE VIEW BillView

AS

SELECT Bill.ID, Bill.dateTime, BillCategory.name, Bill.receiptMoney

FROM Bill, BillCategory

WHERE Bill.categoryID = BillCategory.ID

GO

1. **View hiển thị chi tiết hóa đơn khi biết mã hóa đơn:**

View hiển thị chi tiết hóa đơn gồm: Tên sản phẩm, số lượng, thành tiền khi biết mã hóa đơn.

SQL:

DROP VIEW IF EXISTS BillDetailView

GO

CREATE VIEW BillDetailView

AS

SELECT BillInfo.billID,

Product.name,

BillInfo.quantity,

Product.price \* BillInfo.quantity AS totalPrice

FROM BillInfo, Product

WHERE BillInfo.productID = Product.ID

GO

1. **View hiển thị danh sách trạng thái bàn:**

SQL:

DROP VIEW IF EXISTS TableStatusView

GO

CREATE VIEW TableStatusView

AS

SELECT ID, status

FROM [Table]

GO

C# code:

try

{

data.Clear();

dataProduct.Clear();

dataCategories.Clear();

data = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.TableStatusView");

dataProduct = DataProvider.Instance.ExecuteQuery("EXEC dbo.selectAllProduct");

dataCategories = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.BillCategory");

}

catch (SqlException ev)

{

MessageBox.Show("Load dữ liệu thất bại! \n Do: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

};

1. **View hiển thị danh sách bàn đang trống:**

SQL:

DROP VIEW IF EXISTS TableAvailableView

GO

CREATE VIEW TableAvailableView

AS

SELECT ID, status

FROM [Table]

WHERE status = 'Available'

GO

1. **View hiển thị danh sách tên các nhân viên đang trong ca trực của ngày hiện tại:**

View hiển thị danh sách gồm: Tên nhân viên, chi tiết công việc, số điện thoại. ID ca trực, ngày trực. Và sắp xếp theo ngày trực.

SQL:

DROP VIEW IF EXISTS EmployeeShiftView

GO

CREATE VIEW EmployeeShiftView

AS

SELECT Employee.name,

Job.jobDetail,

Employee.phone,

Shift.ID,

On\_Duty.date

FROM Employee, Job, Shift, On\_Duty

WHERE Employee.name <> 'NA' AND Employee.jobID = Job.ID AND Employee.ID = On\_Duty.empID AND On\_Duty.shiftID = Shift.ID

GO

1. **View hiển thị tổng tiền và chiếc khấu của từng hóa đơn:**

View tính và hiển thị danh sách các hóa đơn gồm: Tổng tiền và chiếc khấu.

SQL:

DROP VIEW IF EXISTS BillTotalPriceView

GO

CREATE VIEW BillTotalPriceView

AS

SELECT Bill.ID,

Bill.categoryID,

Bill.dateTime,

SUM(totalPrice) AS totalPrice,

CASE

WHEN Bill.categoryID = 'BC01' OR Bill.categoryID = 'BC02' THEN

CASE

WHEN Customer.bonusPoint < 200 THEN 0

WHEN Customer.bonusPoint >= 200 AND Customer.bonusPoint < 400 THEN SUM(totalPrice) \* 0.05

WHEN Customer.bonusPoint >= 400 AND Customer.bonusPoint < 500 THEN SUM(totalPrice) \* 0.1

WHEN Customer.bonusPoint >= 500 THEN SUM(totalPrice) \* 0.15

END

WHEN Bill.categoryID = 'BC03' OR Bill.categoryID = 'BC04' THEN SUM(totalPrice) \* BillCategory.discount

END AS discount

FROM Bill, BillDetailView, Customer, BillCategory

WHERE Bill.ID = BillDetailView.billID

AND Bill.customerID = Customer.ID

AND Bill.categoryID = BillCategory.ID

GROUP BY Bill.ID, Bill.categoryID, Bill.dateTime, Customer.bonusPoint, BillCategory.discount

GO

1. **View lấy ra thông tin sản phẩm:**

View hiển thị danh sách thông tin sản phẩm gồm: Mã sản phẩm, tên sản phẩm, trạng thái, giá sản phẩm.

SQL:

DROP VIEW IF EXISTS ProductView

GO

CREATE VIEW ProductView

AS

SELECT ID, name, status, price

FROM Product

GO

* + 1. **Procedure:**

1. **Procedure thêm khách hàng mới:**

SQL:

DROP PROCEDURE IF EXISTS insertCustomer

GO

CREATE PROCEDURE insertCustomer

(@name NVARCHAR(max),

@phone VARCHAR(10))

AS

BEGIN

IF @name = '' OR @name = 'CUSTOMER DOES NOT EXIST'

THROW 50000, 'Vui lòng điền tên khách hàng', 1

ELSE IF @phone = ''

THROW 50000, 'Vui lòng điền số điện thoại khách hàng', 1

ELSE IF LEN(@phone) != 10

THROW 50000, 'Số điện thoại phải có 10 chữ số', 1

ELSE IF EXISTS (SELECT \* FROM Customer WHERE phone = @phone)

THROW 50000, 'Số điện thoại đã tồn tại', 1

ELSE

BEGIN TRANSACTION

DECLARE @cusID VARCHAR(7)

--LẤY CUSTOMERID CUỐI CÙNG

DECLARE @lastCusID VARCHAR(7)

SELECT @lastCusID = MAX(ID) FROM Customer

--NẾU CHƯA CÓ CUSTOMER NÀO THÌ CUSTOMERID MỚI LÀ C00001

IF @lastCusID IS NULL

SET @cusID = 'C00001'

ELSE

--NẾU ĐÃ CÓ CUSTOMER THÌ CUSTOMERID MỚI LÀ CUSTOMERID CUỐI CÙNG + 1

BEGIN

DECLARE @number INT

SET @number = CAST(RIGHT(@lastCusID, 4) AS INT) + 1

SET @cusID = 'C' + RIGHT('0000' + CAST(@number AS VARCHAR(4)), 4)

END

--THÊM CUSTOMER MỚI

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

INSERT INTO Customer(ID, name, phone)

VALUES (@cusID, @name, @phone)

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

SET @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

END

GO

C# code:

try

{

if (DataProvider.Instance.ExecuteNonQuery("EXEC insertCustomer @name , @phone ", new object[] { name, phone }) > 0)

{

MessageBox.Show("Thêm khách hàng thành công");

btnAddCus.Enabled = false;

btnSearch.Enabled = false;

tbxCustomerName.Enabled = false;

tbxPhone.Enabled = false;

}

}

catch (SqlException ev)

{

MessageBox.Show("Thêm khách hàng thất bại! \n Do: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

1. **Procedure cập nhật trạng thái bàn từ Occupied sang Available:**

SQL:

DROP PROCEDURE IF EXISTS updateTableStatus

GO

CREATE PROCEDURE updateTableStatus

(@tableID VARCHAR(3))

AS

BEGIN

IF (SELECT Status FROM TableStatusView WHERE ID = @tableID) = 'Occupied'

BEGIN

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

UPDATE [Table] SET status = 'Available' WHERE ID = @tableID

END TRY

BEGIN CATCH

SET @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

END

ELSE

THROW 50000, 'Bàn đang trống', 1

RETURN

END

GO

C# code:

try

{

if (check)

{

if (DataProvider.Instance.ExecuteNonQuery("EXEC updateTableStatus @ID ", new object[] { listButton[i].Text }) > 0)

{

paintTable();

addBtnClick();

break;

}

}

}

catch (SqlException ev)

{

MessageBox.Show("Cập nhật trạng thái bàn thất bại! \n Do: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

1. **Procedure thêm bill mới:**

SQL:

DROP PROCEDURE IF EXISTS insertBill

GO

CREATE PROCEDURE insertBill

@ID varchar(7),

@dateTime varchar(max),

@tableID varchar(3),

@customerName nvarchar(max),

@categoryName nvarchar(max),

@empName nvarchar(max),

@receiptMoney DECIMAL(10,2)

AS

BEGIN

IF @categoryName = ''

THROW 50000, 'Vui lòng chọn loại hóa đơn', 1

ELSE IF @customerName = ''

THROW 50000, 'Vui lòng nhập số điện thoại và khách hàng', 1

ELSE IF @empName = ''

THROW 50000, 'Vui lòng chọn tên nhân viên', 1

ELSE IF @tableID = ''

THROW 50000, 'Vui lòng chọn bàn', 1

ELSE IF NOT EXISTS (SELECT \* FROM TableAvailableView WHERE ID = @tableID) AND @tableID <> 'NA'

THROW 50000, 'Bàn không tồn tại hoặc đang được sử dụng', 1

ELSE

BEGIN TRANSACTION

DECLARE @customerID VARCHAR(5)

DECLARE @categoryID VARCHAR(4)

DECLARE @empID VARCHAR(4)

SELECT @customerID = ID FROM Customer WHERE name = @customerName

SELECT @categoryID = ID FROM BillCategory WHERE name = @categoryName

SELECT @empID = ID FROM Employee WHERE name = @empName

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

INSERT INTO Bill(ID, dateTime, tableID, customerID, categoryID, empID, receiptMoney)

VALUES (@ID, CAST(@dateTime AS DATETIME), @tableID, @customerID, @categoryID, @empID, @receiptMoney)

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

SET @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

END

GO

Trigger cập nhật trạng thái bàn sang đang được sử dụng sau khi thêm bill:

DROP TRIGGER IF EXISTS updateTableStatusTrigger

GO

CREATE TRIGGER updateTableStatusTrigger

ON Bill

AFTER INSERT

AS

BEGIN

DECLARE @tableID VARCHAR(3)

DECLARE @categoryID VARCHAR(4)

SELECT @tableID = tableID FROM inserted

SELECT @categoryID = categoryID FROM inserted

IF @categoryID = 'BC01'

BEGIN

IF (SELECT status FROM [Table] WHERE ID = @tableID) = 'Available'

UPDATE [Table] SET status = 'Occupied' WHERE ID = @ta bleID

ELSE

THROW 50000, 'TABLE IS OCCUPIED', 1

END

EXEC sp\_refreshview TableAvailableView

END

GO

C# code:

try

{

if (DataProvider.Instance.ExecuteNonQuery("EXEC insertBill @ID , @date , @tableID , @name , @categories , @emp , @receiptMoney ", new object[] { billID, date, tableID, name, categories, emp, receipMoney }) > 0)

{

MessageBox.Show("Thêm hóa đơn thành công");

lbShowDis.Text = "0 VND";

lbTotalPrice.Text = "0 VND";

cbxCategories.Enabled = false;

tbxIDBill.Enabled = false;

tbxDate.Enabled = false;

tbxPhone.Enabled = false;

tbxCustomerName.Enabled = false;

cbxEmp.Enabled = false;

btnAdd.Enabled = false;

tbxReceiptMoney.Enabled = true;

paintTable();

addBtnClick();

break;

}

}

catch (SqlException ev)

{

MessageBox.Show("Thêm hóa đơn thất bại! \n Do: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

break;

}

1. **Procedure thêm chi tiết hóa đơn mới vào hóa đơn khi biết mã hóa đơn:**

SQL:

CREATE PROCEDURE addBillInfo

(

@billID VARCHAR(7),

@productName NVARCHAR(max),

@quantity INT

)

AS

BEGIN

IF @quantity <= 0

THROW 50000, 'Số lượng không hợp lệ', 1

ELSE IF NOT EXISTS (SELECT \* FROM Product WHERE name = @productName)

THROW 50000, 'Sản phẩm không tồn tại', 1

ELSE IF NOT EXISTS (SELECT \* FROM Bill WHERE ID = @billID)

THROW 50000, 'Hóa đơn không tồn tại', 1

ELSE

BEGIN TRANSACTION

DECLARE @productID VARCHAR(3)

DECLARE @status INT

DECLARE @quantityProductMaterial INT

DECLARE @materialID VARCHAR(3)

-- Lấy ID và trạng thái của sản phẩm

SELECT @productID = ID , @status = status FROM Product WHERE name = @productName

-- Lấy số lượng nguyên liệu cần cho sản phẩm

SELECT @quantityProductMaterial = quantity, @materialID = materialID FROM Product\_Material WHERE productID = @productID

-- Kiểm tra số lượng tồn kho

IF @status >= @quantity

BEGIN

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

-- Cập nhật số lượng nguyên liệu

UPDATE Material SET status = status - @quantityProductMaterial \* @quantity WHERE ID = @materialID

-- Thêm thông tin hóa đơn

INSERT INTO BillInfo VALUES (@billID, @productID, @quantity)

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

SET @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

END

ELSE

THROW 50000, 'Sản phẩm không đủ số lượng mà bạn yêu cầu', 1

END

GO

C# code:

try

{

if (DataProvider.Instance.ExecuteNonQuery("EXEC addBillInfo @billID , @productName , @quantity ", new object[] { billID, productName, quantity }) > 0)

{

MessageBox.Show("Thêm sản phẩm thành công");

tbxAvai.Text = DataProvider.Instance.ExecuteScalar("SELECT dbo.getProductStatus ( @productName )", new object[] { productName }).ToString();

tbxQuantity.Text = "0";

}

}

catch (SqlException ev)

{

MessageBox.Show("Thêm sản phẩm thất bại! \n Do: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

1. **Procedure cập nhật chi tiết hóa đơn khi biết mã hóa đơn:**

SQL:

DROP PROCEDURE IF EXISTS updateBillInfo

GO

CREATE PROCEDURE updateBillInfo

(@billID VARCHAR(7),

@productName NVARCHAR(max),

@quantity INT)

AS

BEGIN

IF @quantity <= 0

THROW 50000, 'Số lượng không hợp lệ', 1

ELSE IF NOT EXISTS (SELECT \* FROM Product WHERE name = @productName)

THROW 50000, 'Sản phẩm không tồn tại', 1

ELSE IF NOT EXISTS (SELECT \* FROM Bill WHERE ID = @billID)

THROW 50000, 'Hóa đơn không tồn tại', 1

ELSE

BEGIN TRANSACTION

DECLARE @productID VARCHAR(3)

DECLARE @status INT

DECLARE @quantityProductMaterial INT

DECLARE @materialID VARCHAR(3)

-- Lấy ID và trạng thái của sản phẩm

SELECT @productID = ID , @status = status FROM Product WHERE name = @productName

-- Lấy số lượng nguyên liệu cần cho sản phẩm

SELECT @quantityProductMaterial = quantity, @materialID = materialID FROM Product\_Material WHERE productID = @productID

-- Kiểm tra số lượng tồn kho

IF @status >= @quantity

BEGIN

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

-- Cập nhật số lượng nguyên liệu

UPDATE Material SET status = status - @quantityProductMaterial \* @quantity WHERE ID = @materialID

-- Cập nhật thông tin hóa đơn

UPDATE BillInfo SET quantity = @quantity WHERE billID = @billID AND productID = @productID

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

SET @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

END

ELSE

THROW 50000, 'Sản phẩm không đủ số lượng mà bạn yêu cầu', 1

END

GO

C# code:

try

{

if (DataProvider.Instance.ExecuteNonQuery("EXEC updateBillInfo @billID , @productName , @quantity ", new object[] { billID, productName, quantity }) > 0)

{

MessageBox.Show("Cập nhật sản phẩm thành công");

tbxAvai.Text = DataProvider.Instance.ExecuteScalar("SELECT dbo.getProductStatus ( @productName )", new object[] { productName }).ToString();

tbxQuantity.Text = "0";

}

}

catch (SqlException ev)

{

MessageBox.Show("Cập nhật sản phẩm thất bại! \n Do: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

1. **Procedure xóa chi tiết hóa đơn khi biết tên sản phẩm:**

SQL:

DROP PROCEDURE IF EXISTS deleteBillInfoByName

GO

CREATE PROCEDURE deleteBillInfoByName

@billID varchar(7),

@productName NVARCHAR(max),

@quantity INT

AS

BEGIN

IF @quantity <= 0

THROW 50000, 'Số lượng không hợp lệ', 1

ELSE IF NOT EXISTS (SELECT \* FROM Product WHERE name = @productName)

THROW 50000, 'Sản phẩm không tồn tại', 1

ELSE IF NOT EXISTS (SELECT \* FROM Bill WHERE ID = @billID)

THROW 50000, 'Hóa đơn không tồn tại', 1

ELSE

BEGIN TRANSACTION

DECLARE @productID VARCHAR(3)

DECLARE @status INT

DECLARE @quantityProductMaterial INT

DECLARE @materialID VARCHAR(3)

-- Lấy ID và trạng thái của sản phẩm

SELECT @productID = ID , @status = status FROM Product WHERE name = @productName

-- Lấy số lượng nguyên liệu cần cho sản phẩm

SELECT @quantityProductMaterial = quantity, @materialID = materialID FROM Product\_Material WHERE productID = @productID

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

-- Cập nhật số lượng nguyên liệu

UPDATE Material SET status = status + @quantityProductMaterial \* @quantity WHERE ID = @materialID

-- Xóa thông tin hóa đơn

DELETE FROM BillInfo WHERE billID = @billID AND productID = @productID

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

SET @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

END

GO

C# code:

try

{

if (DataProvider.Instance.ExecuteNonQuery("EXEC deleteBillInfoByName @billID , @productName , @quantity ", new object[] { billID, productName, quantity }) > 0)

{

MessageBox.Show("Xóa sản phẩm thành công");

tbxAvai.Text = DataProvider.Instance.ExecuteScalar("SELECT dbo.getProductStatus ( @productName )", new object[] { productName }).ToString();

tbxQuantity.Text = "0";

}

}

catch (SqlException ev)

{

MessageBox.Show("Xóa sản phẩm thất bại! \n Do: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

1. **Procedure cập nhật điểm thưởng khách hàng:**

SQL:

DROP PROCEDURE IF EXISTS updateCustomerPoint

GO

CREATE PROCEDURE updateCustomerPoint

(@billID VARCHAR(7),

@phone VARCHAR(10))

AS

BEGIN

BEGIN TRANSACTION

DECLARE @totalPrice DECIMAL(10, 2)

DECLARE @point INT

IF NOT EXISTS (SELECT \* FROM Bill WHERE ID = @billID)

THROW 50000, 'BILL DOES NOT EXIST', 1

SELECT @totalPrice = SUM(totalPrice) FROM BillDetailView WHERE billID = @billID

SELECT @point = @totalPrice / 10000

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

UPDATE Customer SET bonusPoint = bonusPoint + @point WHERE phone = @phone

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

SET @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

END

GO

C# code:

try

{

if (DataProvider.Instance.ExecuteNonQuery("EXEC updateCustomerPoint @billID , @phone ", new object[] { billID, phone }) > 0)

{

MessageBox.Show("Cập nhật số tiền nhận thành công");

tbxReceiptMoney.Enabled = false;

}

}

catch (SqlException ev)

{

MessageBox.Show("Cập nhật số tiền nhận thất bại! \n Do: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

1. **Procedure lấy ra danh sách ca nhân viên có ca trực trong ngày hiện tại:**

SQL:

DROP PROCEDURE IF EXISTS findEmployeeShift

GO

CREATE PROCEDURE findEmployeeShift

AS

BEGIN

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

DECLARE @today DATE

SET @today = GETDATE()

SELECT \* FROM EmployeeShiftView WHERE date = @today

END TRY

BEGIN CATCH

SET @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

END

GO

C# code:

try

{

dataShift.Clear();

dataShiftInfo.Clear();

dataShift = DataProvider.Instance.ExecuteQuery("EXEC dbo.findEmployeeShift");

dataShiftInfo = DataProvider.Instance.ExecuteQuery("EXEC dbo.selectAllShift");

}

catch (SqlException ev)

{

MessageBox.Show("Load dữ liệu thất bại! \n Do: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

1. **Procedure cập nhật số tiền nhận từ khách:**

SQL:

DROP PROCEDURE IF EXISTS updateReceiptMoney

GO

CREATE PROCEDURE updateReceiptMoney

@billID varchar(7),

@receiptMoney DECIMAL(10, 2)

AS

BEGIN

IF @receiptMoney >= (SELECT totalPrice FROM BillTotalPriceView WHERE ID = @billID)

BEGIN

BEGIN TRANSACTION

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

UPDATE Bill

SET receiptMoney = @receiptMoney

WHERE ID = @billID

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

SET @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

END

ELSE IF @receiptMoney = 0

THROW 50000, 'Vui lòng nhập số tiền', 1

ELSE IF @receiptMoney < (SELECT totalPrice FROM BillTotalPriceView WHERE ID = @billID) AND @receiptMoney > 0

THROW 50000, 'Số tiền không đủ', 1

ELSE

THROW 50000, 'Số tiền không hợp lệ', 1

END

GO

C# code:

try

{

if (DataProvider.Instance.ExecuteNonQuery("EXEC updateReceiptMoney @ID , @receiptMoney ", new object[] { tbxIDBill.Text, tbxReceiptMoney.Text }) > 0)

{

string billID = tbxIDBill.Text;

string phone = tbxPhone.Text;

try

{

if (DataProvider.Instance.ExecuteNonQuery("EXEC updateCustomerPoint @billID , @phone ", new object[] { billID, phone }) > 0)

{

MessageBox.Show("Cập nhật số tiền nhận thành công");

tbxReceiptMoney.Enabled = false;

}

}

catch (SqlException ev)

{

MessageBox.Show("Cập nhật số tiền nhận thất bại! \n Do: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

MessageBox.Show("Cập nhật số tiền nhận thành công");

tbxReceiptMoney.Enabled = false;

}

}

catch (SqlException ev)

{

MessageBox.Show("Cập nhật số tiền nhận thất bại! \n Do: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

1. **Procedure xóa bàn:**

SQL:

DROP PROCEDURE IF EXISTS deleteTableByID

GO

CREATE PROCEDURE deleteTableByID

@ID VARCHAR(3)

AS

BEGIN

--NẾU BÀN ĐANG ĐƯỢC DÙNG THÌ KHÔNG XÓA

IF (SELECT status FROM [Table] WHERE ID = @ID) = 'Occupied'

THROW 50000, 'TABLE IS OCCUPIED', 1

ELSE

BEGIN

BEGIN TRANSACTION

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

DELETE FROM [Table] WHERE ID = @ID

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

SET @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

END

END

GO

Trigger chuyển mã bàn sang mã NA trước khi xóa bàn trong tất cả các bill liên quan:

DROP TRIGGER IF EXISTS deleteTableTrigger

GO

CREATE TRIGGER deleteTableTrigger

ON [Table]

INSTEAD OF DELETE

AS

BEGIN

BEGIN TRANSACTION

DECLARE @errorMessage NVARCHAR(max)

DECLARE @tableID VARCHAR(3)

SELECT @tableID = ID FROM deleted

BEGIN TRY

UPDATE Bill SET tableID = 'NA' WHERE tableID = @tableID

DELETE FROM [Table] WHERE ID = @tableID

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

DECLARE @errorMessage NVARCHAR(max)

SET @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

END

GO

Trigger trước khi xóa bill thì xóa hết billInfo liên quan:

DROP TRIGGER IF EXISTS deleteBillTrigger

GO

CREATE TRIGGER deleteBillTrigger

ON Bill

INSTEAD OF DELETE

AS

BEGIN

BEGIN TRANSACTION

DECLARE @billID VARCHAR(7)

DECLARE @tableID VARCHAR(3)

DECLARE @errorMessage NVARCHAR(max)

SELECT @billID = ID FROM deleted

SELECT @tableID = tableID FROM deleted

BEGIN TRY

DELETE FROM BillInfo WHERE billID = @billID

IF (SELECT status FROM [Table] WHERE ID = @tableID) = 'Occupied'

UPDATE [Table] SET status = 'Available' WHERE ID = @tableID

DELETE FROM Bill WHERE ID = @billID

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

SET @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

END

GO

C# code:

try

{

if (DataProvider.Instance.ExecuteNonQuery("EXEC deleteTableByID @ID", new object[] { data.Rows[count - 1]["ID"].ToString() }) > 0)

{

paintTable();

}

}

catch (SqlException ev)

{

MessageBox.Show("Xóa bàn thất bại! \n Do: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

1. **Procedure xóa hóa đơn:**

SQL:

DROP PROCEDURE IF EXISTS deleteBillByID

GO

CREATE PROCEDURE deleteBillByID

@ID VARCHAR(7)

AS

BEGIN

BEGIN TRANSACTION

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

DELETE FROM Bill WHERE ID = @ID

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

SET @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

END

GO

C# code:

try

{

DataProvider.Instance.ExecuteNonQuery("EXEC deleteBillByID @billID ", new object[] { tbxIDBill.Text });

}

catch (SqlException ev)

{

MessageBox.Show("Xóa hóa đơn thất bại! \n Do: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

1. **Procedure lấy ra sanh sách tất cả các ca trực:**

SQL:

DROP PROCEDURE IF EXISTS selectAllShift

GO

CREATE PROCEDURE selectAllShift

AS

BEGIN

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

SELECT \* FROM [Shift]

END TRY

BEGIN CATCH

SET @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

END

GO

C# code:

try

{

dataShift.Clear();

dataShiftInfo.Clear();

dataShift = DataProvider.Instance.ExecuteQuery("EXEC dbo.findEmployeeShift");

dataShiftInfo = DataProvider.Instance.ExecuteQuery("EXEC dbo.selectAllShift");

}

catch (SqlException ev)

{

MessageBox.Show("Load dữ liệu thất bại! \n Do: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

1. **Procedure lấy product từ view ‘ProductView’:**

SQL:

DROP PROCEDURE IF EXISTS selectAllProduct

GO

CREATE PROCEDURE selectAllProduct

AS

BEGIN

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

SELECT \* FROM ProductView

END TRY

BEGIN CATCH

SET @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

END

GO

C# code:

try

{

data.Clear();

dataProduct.Clear();

dataCategories.Clear();

data = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.TableStatusView");

dataProduct = DataProvider.Instance.ExecuteQuery("EXEC dbo.selectAllProduct");

dataCategories = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.BillCategory");

}

catch (SqlException ev)

{

MessageBox.Show("Load dữ liệu thất bại! \n Do: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

};

1. **Procedure thêm bàn mới:**

SQL:

DROP PROCEDURE IF EXISTS insertTable

GO

CREATE PROCEDURE insertTable

AS

BEGIN

BEGIN TRANSACTION

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

INSERT INTO [Table](ID, name, status)

VALUES ('', '', '')

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

SET @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

END

GO

Trigger thêm thông tin bàn cho bàn trước khi thêm:

DROP TRIGGER IF EXISTS insertTableTrigger

GO

CREATE TRIGGER insertTableTrigger

ON [Table]

INSTEAD OF INSERT

AS

BEGIN

BEGIN TRANSACTION

DECLARE @ID VARCHAR(3)

DECLARE @status NVARCHAR(max)

DECLARE @name NVARCHAR(max)

DECLARE @errorMessage NVARCHAR(max)

SET @status = 'Available'

--LẤY TABLEID CUỐI CÙNG

DECLARE @lastTableID VARCHAR(3)

SELECT @lastTableID = MAX(ID) FROM [Table]

--NẾU CHƯA CÓ TABLE NÀO THÌ TABLEID MỚI LÀ T01

IF @lastTableID IS NULL

BEGIN

SET @ID = 'T01'

SET @name = 'Table 1'

END

ELSE

--NẾU ĐÃ CÓ TABLE THÌ TABLEID MỚI LÀ TABLEID CUỐI CÙNG + 1

BEGIN

DECLARE @number INT

SET @number = CAST(RIGHT(@lastTableID, 2) AS INT) + 1

SET @ID = 'T' + RIGHT('00' + CAST(@number AS VARCHAR(2)), 2)

SET @name = 'Table ' + CAST(@number AS VARCHAR(2))

END

BEGIN TRY

INSERT INTO [Table](ID, name, status)

VALUES (@ID, @name, @status)

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

SET @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

END

GO

C# code:

try

{

if (DataProvider.Instance.ExecuteNonQuery("EXEC insertTable") > 0)

{

paintTable();

}

}

catch (SqlException ev)

{

MessageBox.Show("Thêm bàn thất bại! \n Do: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

paintTable();

}

1. **Procedure lấy danh sách hóa đơn:**

SQL:

DROP PROCEDURE IF EXISTS selectAllBillID

GO

CREATE PROCEDURE selectAllBillID

AS

BEGIN

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

SELECT ID FROM Bill

END TRY

BEGIN CATCH

SET @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

END

GO

C# code:

dataBill = DataProvider.Instance.ExecuteQuery("EXEC dbo.selectAllBillView");

1. **Procedure lấy ra danh sách mã hóa đơn:**

SQL:

DROP PROCEDURE IF EXISTS selectAllBillCategory

GO

CREATE PROCEDURE selectAllBillCategory

AS

BEGIN

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

SELECT name FROM BillCategory

END TRY

BEGIN CATCH

SET @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

END

GO

C# code:

DataTable data = DataProvider.Instance.ExecuteQuery("EXEC dbo.selectAllBillID");

* + 1. **Function:**

1. **Function kiểm tra xem khách hàng có tồn tại trong CSDL không:**

SQL:

DROP FUNCTION IF EXISTS checkCustomer

GO

CREATE FUNCTION checkCustomer

(@phone VARCHAR(10))

RETURNS NVARCHAR(max)

AS

BEGIN

DECLARE @result NVARCHAR(max)

IF NOT EXISTS (SELECT \* FROM Customer WHERE phone = @phone)

SET @result = 'CUSTOMER DOES NOT EXIST'

ELSE

SET @result = (SELECT name FROM Customer WHERE phone = @phone)

RETURN @result

END

GO

C# code:

try

{

tbxCustomerName.Text = DataProvider.Instance.ExecuteScalar("SELECT dbo.checkCustomer ( @phone ) ", new object[] { phone }).ToString();

if (tbxCustomerName.Text == "CUSTOMER DOES NOT EXIST")

{

btnAddCus.Enabled = true;

btnSearch.Enabled = true;

tbxCustomerName.Enabled = true;

tbxPhone.Enabled = true;

}

else

{

btnAddCus.Enabled = false;

btnSearch.Enabled = false;

tbxCustomerName.Enabled = false;

tbxPhone.Enabled = false;

}

}

catch (SqlException ev)

{

MessageBox.Show("Tìm kiếm thất bại! \n Do: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

1. **Function trả về bảng hiển thị chi tiết hóa đơn:**

SQL:

DROP FUNCTION IF EXISTS getBillInfo

GO

CREATE FUNCTION getBillInfo

(

@billID VARCHAR(7)

)

RETURNS TABLE

AS

RETURN

(

SELECT Product.name, BillInfo.quantity, Product.price \* BillInfo.quantity AS totalPrice

FROM BillInfo, Product

WHERE BillInfo.productID = Product.ID AND BillInfo.billID = @billID

)

GO

C# code:

try

{

dgvBillInfo.DataSource = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.getBillInfo ( @billID )", new object[] { tbxIDBill.Text });

}

catch (SqlException ev)

{

MessageBox.Show("Load dữ liệu thất bại! \n Do: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

1. **Function lấy ra tổng tiền của một hóa đơn khi biết mã hóa đơn:**

SQL:

DROP FUNCTION IF EXISTS getBillTotalPrice

GO

CREATE FUNCTION getBillTotalPrice

(

@billID VARCHAR(7)

)

RETURNS DECIMAL(10, 2)

AS

BEGIN

DECLARE @totalPrice DECIMAL(10, 2)

SELECT @totalPrice = totalPrice FROM BillTotalPriceView WHERE ID = @billID

RETURN @totalPrice

END

GO

C# code:

try

{

lbShowDis.Text = DataProvider.Instance.ExecuteScalar("SELECT dbo.getBillDiscount ( @billID )", new object[] { tbxIDBill.Text }).ToString();

lbTotalPrice.Text = DataProvider.Instance.ExecuteScalar("SELECT dbo.getBillTotalPrice ( @billID )", new object[] { tbxIDBill.Text }).ToString();

}

catch (SqlException ev)

{

MessageBox.Show("Cập nhật giá trị thất bại! \n Do: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);\

}

1. **Function lấy ra chiếc khấu của một hóa đơn khi biết mã hóa đơn:**

SQL:

DROP FUNCTION IF EXISTS getBillDiscount

GO

CREATE FUNCTION getBillDiscount

(

@billID VARCHAR(7)

)

RETURNS DECIMAL(10, 2)

AS

BEGIN

DECLARE @discount DECIMAL(10, 2)

SELECT @discount = discount FROM BillTotalPriceView WHERE ID = @billID

RETURN @discount

END

GO

C# code:

try

{

lbShowDis.Text = DataProvider.Instance.ExecuteScalar("SELECT dbo.getBillDiscount ( @billID )", new object[] { tbxIDBill.Text }).ToString();

lbTotalPrice.Text = DataProvider.Instance.ExecuteScalar("SELECT dbo.getBillTotalPrice ( @billID )", new object[] { tbxIDBill.Text }).ToString();

}

catch (SqlException ev)

{

MessageBox.Show("Cập nhật giá trị thất bại! \n Do: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

1. **Function kiểm tra xem hóa đơn có tồn tại không:**

SQL:

DROP FUNCTION IF EXISTS checkBillID

GO

CREATE FUNCTION checkBillID

(

@billID VARCHAR(7)

)

RETURNS INT

AS

BEGIN

DECLARE @result INT

IF EXISTS (SELECT \* FROM Bill WHERE ID = @billID)

SET @result = 1

ELSE

SET @result = 0

RETURN @result

END

GO

C# code:

try

{

check = Convert.ToInt32(DataProvider.Instance.ExecuteScalar("SELECT dbo.checkBillID ( @billID )", new object[] { billID }));

}

catch (SqlException ev)

{

MessageBox.Show("Kiểm tra hóa đơn thất bại! \n Do: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

1. **Function lấy ra thông tin hóa đơn:**

SQL:

DROP FUNCTION IF EXISTS getBillInfoViewByID

GO

CREATE FUNCTION getBillInfoViewByID

(

@billID VARCHAR(7)

)

RETURNS TABLE

AS

RETURN

(

SELECT BillCategory.name AS categoryName,

Bill.ID, Employee.name AS empName, Bill.dateTime, Customer.phone, Customer.name AS customerName, [Table].ID AS tableID

FROM Bill, BillCategory, Employee, Customer, [Table]

WHERE Bill.categoryID = BillCategory.ID AND Bill.empID = Employee.ID AND Bill.customerID = Customer.ID AND Bill.tableID = [Table].ID AND Bill.ID = @billID

)

GO

C# code;

try

{

dataBillView = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.getBillInfoViewByID ( @id )", new object[] { id });

}

catch (SqlException ev)

{

MessageBox.Show(" Load dữ liệu thất bại! \n Do: \n " + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

1. **Function tạo Bill ID tự động:**

SQL:

DROP FUNCTION IF EXISTS createBillID

GO

CREATE FUNCTION createBillID

()

RETURNS VARCHAR(7)

AS

BEGIN

DECLARE @billID VARCHAR(7)

--LẤY BILLID CUỐI CÙNG

DECLARE @lastBillID VARCHAR(7)

SELECT @lastBillID = MAX(ID) FROM Bill

--NẾU CHƯA CÓ BILL NÀO THÌ BILLID MỚI LÀ B00001

IF @lastBillID IS NULL

SET @billID = 'B00001'

ELSE

--NẾU ĐÃ CÓ BILL THÌ BILLID MỚI LÀ BILLID CUỐI CÙNG + 1

BEGIN

DECLARE @number INT

SET @number = CAST(RIGHT(@lastBillID, 6) AS INT) + 1

SET @billID = 'B' + RIGHT('000000' + CAST(@number AS VARCHAR(6)), 6)

END

RETURN @billID

END

GO

C# code:

try

{

tbxIDBill.Clear();

tbxIDBill.Text = DataProvider.Instance.ExecuteScalar("SELECT dbo.createBillID ()").ToString();

}

catch (SqlException ev)

{

MessageBox.Show("Tạo ID hóa đơn thất bại! \n Do: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

1. **Function lấy ra trạng thái của sản phẩm:**

SQL:

DROP FUNCTION IF EXISTS getProductStatus

GO

CREATE FUNCTION getProductStatus

(

@productName NVARCHAR(max)

)

RETURNS INT

AS

BEGIN

DECLARE @status NVARCHAR(max)

SELECT @status = status FROM ProductView WHERE name = @productName

RETURN @status

END

GO

C# code:

try

{

if (DataProvider.Instance.ExecuteNonQuery("EXEC addBillInfo @billID , @productName , @quantity ", new object[] { billID, productName, quantity }) > 0)

{

MessageBox.Show("Thêm sản phẩm thành công");

tbxAvai.Text = DataProvider.Instance.ExecuteScalar("SELECT dbo.getProductStatus ( @productName )", new object[] { productName }).ToString();

tbxQuantity.Text = "0";

}

}

catch (SqlException ev)

{

MessageBox.Show("Thêm sản phẩm thất bại! \n Do: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

1. **Function lọc ra danh sách hóa đơn theo mã hóa đơn:**

SQL:

DROP FUNCTION IF EXISTS filterBillByID

GO

CREATE FUNCTION filterBillByID

(

@ID VARCHAR(7)

)

RETURNS TABLE

AS

RETURN

(

SELECT \* FROM BillView

WHERE ID = @ID

)

GO

C# code:

try

{

dataBill = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.filterBillByID ( @id )", new object[] { id });

}

catch (SqlException ev)

{

MessageBox.Show(" Load dữ liệu thất bại! \n Do: \n " + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

1. **Function lọc ra danh sách hóa đơn theo ngày lập hóa đơn:**

SQL:

DROP FUNCTION IF EXISTS filterBillByDate

GO

CREATE FUNCTION filterBillByDate

(

@date DATE

)

RETURNS TABLE

AS

RETURN

(

SELECT \* FROM BillView

WHERE CAST(dateTime AS DATE) = @date

)

GO

C# code:

try

{

dataBill = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.filterBillByDate ( @date )", new object[] { date });

}

catch (SqlException ev)

{

MessageBox.Show(" Load dữ liệu thất bại! \n Do: \n " + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

1. **Function lọc ra hóa đơn theo tên loại hóa đơn:**

SQL:

DROP FUNCTION IF EXISTS filterBillByCategory

GO

CREATE FUNCTION filterBillByCategory

(

@category NVARCHAR(max)

)

RETURNS TABLE

AS

RETURN

(

SELECT \* FROM BillView

WHERE name = @category

)

GO

C# code:

try

{

dataBill = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.filterBillByCategory ( @category )", new object[] { category });

}

catch (SqlException ev)

{

MessageBox.Show(" Load dữ liệu thất bại! \n Do: \n " + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

1. **Function lấy ra tất cả chi tiết hóa đơn của một hóa đơn khi biết mã hóa đơn:**

SQL:

DROP FUNCTION IF EXISTS getBillDetail

GO

CREATE FUNCTION getBillDetail

(

@billID VARCHAR(7)

)

RETURNS TABLE

AS

RETURN

(

SELECT \* FROM BillDetailView

WHERE billID = @billID

)

GO

C# code:

try

{

data = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.getBillDetail ( @id )", new object[] { id });

}

catch (SqlException ev)

{

MessageBox.Show(" Load dữ liệu thất bại! \n Do: \n " + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

* 1. **Chức năng Employee:**

Quản lý nhân viên như thêm, xóa nhân viên, chỉnh sửa thông tin nhân viên và sắp xếp và lên lịch trực cho quán.

* + 1. **Views:**

1. **View hiển thị chi tiết nhân viên gồm tên, công việc, số ca, lương:**

Trong đó:

Số ca được tính bằng tổng số ca trong view employeeshift.

Lương được tính bằng số ca nhân với số giờ làm việc trong một ca (với nhân viên có mã là EF thì 1 ca dài 8 tiếng, còn mã EP thì một ca dài 4 tiếng) nhân với Job.salary.

SQL:

DROP VIEW IF EXISTS EmployeeDetailView

GO

CREATE VIEW EmployeeDetailView

AS

SELECT Employee.name, Job.jobDetail AS job, COUNT(EmployeeShiftView.phone) AS shift,

CASE

WHEN Employee.ID LIKE 'EF%' THEN COUNT(EmployeeShiftView.phone) \* 8 \* Job.salary

WHEN Employee.ID LIKE 'EP%' THEN COUNT(EmployeeShiftView.phone) \* 4 \* Job.salary

END AS salary

FROM Employee , Job , EmployeeShiftView

WHERE Employee.name <> 'NA' AND Employee.jobID = Job.ID AND Employee.phone = EmployeeShiftView.phone

GROUP BY Employee.name, Job.jobDetail, Employee.ID, Job.salary

GO

1. **View hiển thị chi tiết ca trực gồm tên nhân viên, số điện thoại, ca trực, ngày trực:**

Trong đó view được sắp xếp dựa vào ngày trực theo thứ tự tăng dần và nếu có cùng ngày trực thì sắp xếp theo mã ca trực theo thứ tự tăng dần.

SQL:

DROP VIEW IF EXISTS ShiftDetailView

GO

CREATE VIEW ShiftDetailView

AS

SELECT Employee.name, Employee.phone, EmployeeShiftView.ID, EmployeeShiftView.date

FROM Employee, EmployeeShiftView

WHERE Employee.phone = EmployeeShiftView.phone

GO

Procedure sắp xếp:

DROP PROCEDURE IF EXISTS getShiftDetail

GO

CREATE PROCEDURE getShiftDetail

AS

BEGIN

SELECT \* FROM ShiftDetailView

ORDER BY date ASC, ID ASC

END

GO

C# code:

try

{

dataShift.Clear();

dataShift = DataProvider.Instance.ExecuteQuery("EXEC dbo.getShiftDetailByWeek");

}

catch (SqlException ev)

{

MessageBox.Show("Lỗi kết nối.\nDo:\n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

* + 1. **Procedures:**

1. **Procedure lấy ra danh sách nhân viên từ view employeedetailview:**

SQL:

DROP PROCEDURE IF EXISTS getEmployee

GO

CREATE PROCEDURE getEmployee

AS

BEGIN

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

EXEC SP\_REFRESHVIEW 'EmployeeDetailView'

SELECT \* FROM EmployeeDetailView

END TRY

BEGIN CATCH

SELECT @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

END

GO

C# code:

try

{

dataEmpShift = DataProvider.Instance.ExecuteQuery("EXEC dbo.getEmployee");

}

catch (SqlException ev)

{

MessageBox.Show("Load dữ liệu thất bại! \nDo: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

1. **Procedure lấy ra chi tiết công việc từ bảng job:**

SQL:

DROP PROCEDURE IF EXISTS getJob

GO

CREATE PROCEDURE getJob

AS

BEGIN

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

SELECT jobDetail FROM Job WHERE ID <> 'NA'

END TRY

BEGIN CATCH

SELECT @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

END

GO

C# code:

try

{

dataEmp = DataProvider.Instance.ExecuteQuery("EXEC dbo.getEmployeeList");

dataJob = DataProvider.Instance.ExecuteQuery("EXEC dbo.getJob");

}

catch (SqlException ev)

{

MessageBox.Show("Load dữ liệu thất bại! \nDo: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

1. **Procedure lấy ra danh sách ca trực trong bảng shift:**

SQL:

DROP PROCEDURE IF EXISTS getShift

GO

CREATE PROCEDURE getShift

AS

BEGIN

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

SELECT \* FROM Shift

END TRY

BEGIN CATCH

SELECT @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

END

GO

C# code:

try

{

dataShift.Clear();

dataShift = DataProvider.Instance.ExecuteQuery("EXEC dbo.getShiftDetailByWeek");

}

catch (SqlException ev)

{

MessageBox.Show("Lỗi kết nối.\nDo:\n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

1. **Procedure lấy ra danh sách nhân viên trong bảng employee:**

SQL:

DROP PROCEDURE IF EXISTS getEmployeeList

GO

CREATE PROCEDURE getEmployeeList

AS

BEGIN

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

SELECT ID, name, phone,

(SELECT jobDetail FROM Job WHERE ID = jobID ) AS jobDetail

FROM Employee

END TRY

BEGIN CATCH

SELECT @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

END

GO

C# code:

try

{

dataEmp = DataProvider.Instance.ExecuteQuery("EXEC dbo.getEmployeeList");

dataJob = DataProvider.Instance.ExecuteQuery("EXEC dbo.getJob");

}

catch (SqlException ev)

{

MessageBox.Show("Load dữ liệu thất bại! \nDo: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

1. **Procedure cập nhật lương của công việc chỉ định:**

SQL:

DROP PROCEDURE IF EXISTS updateSalary

GO

CREATE PROCEDURE updateSalary

(@job NVARCHAR(max), @salary DECIMAL(10,2))

AS

BEGIN

IF @salary <= 0

THROW 50000, 'Lương không hợp lệ', 1

ELSE IF @job = ''

THROW 50000, 'Công việc không được để trống', 1

ELSE IF NOT EXISTS (SELECT \* FROM Job WHERE jobDetail = @job)

THROW 50000, 'Công việc không tồn tại', 1

ELSE

BEGIN TRANSACTION

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

UPDATE Job

SET salary = @salary

WHERE jobDetail = @job

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

SELECT @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

END

GO

C# code:

try

{

DataProvider.Instance.ExecuteNonQuery("EXEC dbo.updateSalary @job , @salary ", new object[] { job, salary });

MessageBox.Show("Thay đổi lương thành công!", "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);

reload();

}

catch (SqlException ev)

{

MessageBox.Show("Thay đổi lương thất bại! \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Warning);

reload();

return;

}

1. **Procedure thêm nhân viên mới:**

SQL:

DROP PROCEDURE IF EXISTS insertEmployee

GO

CREATE PROCEDURE insertEmployee

@id VARCHAR(4),

@name NVARCHAR(max),

@birth DATETIME,

@address NVARCHAR(max),

@phone VARCHAR(10),

@recruitmentDate DATETIME,

@jobName NVARCHAR(max),

@penatyPoint INT,

@bonusPoint INT,

@numberOfShift INT

AS

BEGIN

IF @name = ''

THROW 50000, 'Tên nhân viên không được để trống', 1

ELSE IF @address = ''

THROW 50000, 'Địa chỉ không được để trống', 1

ELSE IF @phone = ''

THROW 50000, 'Số điện thoại không được để trống', 1

ELSE

BEGIN TRANSACTION

DECLARE @jobID NVARCHAR(4)

DECLARE @empCategory NVARCHAR(max)

--LẤY ID CỦA CÔNG VIỆC

SELECT @jobID = ID

FROM Job

WHERE jobDetail = @jobName

--THÊM NHÂN VIÊN

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

INSERT INTO Employee(ID, name, birthDate, address, phone, recruitmentDate, jobID, penaltyPoint, bonusPoint, numberOfShift)

VALUES(@id, @name, CAST (@birth AS DATE), @address, @phone, CAST (@recruitmentDate AS DATE), @jobID, @penatyPoint, @bonusPoint, @numberOfShift)

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

SELECT @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

--REFESH EmployeeDetailView

EXEC SP\_REFRESHVIEW 'EmployeeDetailView'

END

GO

C# code:

try

{

DataProvider.Instance.ExecuteNonQuery("EXEC dbo.insertEmployee @id , @name , @birth , @address , @phone , @recruitment , @job , @penatyPoint , @bonusPoint , @numberOfShift ", new object[] { id, name, birth, address, phone, recruitment, job, penatyPoint, bonusPoint, numberOfShift });

MessageBox.Show("Thêm nhân viên thành công! \n Vui lòng vào LỊCH TRỰC xếp ca trực cho nhân viên " + name, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);

reload();

}

catch (SqlException ev)

{

MessageBox.Show("Thêm nhân viên thất bại! \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Warning);

reload();

return;

}

1. **Procedure cập nhật thông tin nhân viên:**

SQL:

DROP PROCEDURE IF EXISTS updateEmployee

GO

CREATE PROCEDURE updateEmployee

@id VARCHAR(4),

@name NVARCHAR(max),

@birth DATETIME,

@address NVARCHAR(max),

@phone VARCHAR(10),

@recruitmentDate DATETIME,

@jobName NVARCHAR(max),

@penatyPoint INT,

@bonusPoint INT,

@numberOfShift INT

AS

BEGIN

IF @name = ''

THROW 50000, 'Tên nhân viên không được để trống', 1

ELSE IF @address = ''

THROW 50000, 'Địa chỉ không được để trống', 1

ELSE IF @phone = ''

THROW 50000, 'Số điện thoại không được để trống', 1

ELSE

BEGIN TRANSACTION

DECLARE @jobID NVARCHAR(4)

DECLARE @empCategory NVARCHAR(max)

--LẤY ID CỦA CÔNG VIỆC

SELECT @jobID = ID

FROM Job

WHERE jobDetail = @jobName

--CẬP NHẬT NHÂN VIÊN

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

UPDATE Employee

SET name = @name, birthDate = CAST (@birth AS DATE), address = @address, phone = @phone, recruitmentDate = CAST (@recruitmentDate AS DATE), jobID = @jobID, penaltyPoint = @penatyPoint, bonusPoint = @bonusPoint, numberOfShift = @numberOfShift

WHERE ID = @id

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

SELECT @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

--REFESH EmployeeDetailView

EXEC SP\_REFRESHVIEW 'EmployeeDetailView'

END

GO

C# code:

try

{

DataProvider.Instance.ExecuteNonQuery("EXEC dbo.updateEmployee @id , @name , @birth , @address , @phone , @recruitment , @job , @penatyPoint , @bonusPoint , @numberOfShift ", new object[] { id, name, birth, address, phone, recruitment, job, penatyPoint, bonusPoint, numberOfShift });

MessageBox.Show("Sửa thông tin nhân viên thành công " + name + "!", "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);

reload();

}

catch (SqlException ev)

{

MessageBox.Show("Sửa nhân viên thất bại! \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Warning);

reload();

return;

}

1. **Procedure xóa nhân viên:**

SQL:

DROP PROCEDURE IF EXISTS deleteEmployeeByID

GO

CREATE PROCEDURE deleteEmployeeByID

@id VARCHAR(4)

AS

BEGIN

BEGIN TRANSACTION

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

DELETE FROM Employee

WHERE ID = @id

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

SELECT @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

--REFESH EmployeeDetailView

EXEC SP\_REFRESHVIEW 'EmployeeDetailView'

END

GO

Trigger trước khi xóa nhân viên thì set mã nhân viên trong các bảng bill, receiptnote, on\_duty về nhân viên có mã là NA:

DROP TRIGGER IF EXISTS setEmployeeToNA

GO

CREATE TRIGGER setEmployeeToNA

ON Employee

INSTEAD OF DELETE

AS

BEGIN

BEGIN TRANSACTION

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

UPDATE Bill

SET empID = 'NA'

WHERE empID = (SELECT ID FROM DELETED)

UPDATE ReceiptNote

SET empID = 'NA'

WHERE empID = (SELECT ID FROM DELETED)

UPDATE On\_Duty

SET empID = 'NA'

WHERE empID = (SELECT ID FROM DELETED)

DELETE FROM Employee

WHERE ID = (SELECT ID FROM DELETED)

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

SELECT @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

END

GO

C# code:

string id = tbxIDEmp.Text;

try

{

DataProvider.Instance.ExecuteNonQuery("EXEC dbo.deleteEmployeeByID @id ", new object[] { id });

MessageBox.Show("Xóa nhân viên thành công!", "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);

reload();

}

catch (SqlException ev)

{

MessageBox.Show("Xóa nhân viên thất bại! \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Warning);

reload();

return;

}

1. **Procedure thêm nhân viên vào bảng ca trực:**

SQL:

DROP PROCEDURE IF EXISTS insertEmployeeToShift

GO

CREATE PROCEDURE insertEmployeeToShift

@phone VARCHAR(10),

@date DATE,

@shiftID VARCHAR(5)

AS

BEGIN

BEGIN TRANSACTION

--THÊM NHÂN VIÊN VÀO BẢNG ON DUTY

DECLARE @id VARCHAR(4)

SELECT @id = ID FROM Employee WHERE phone = @phone

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

INSERT INTO On\_Duty(empID, shiftID, date)

VALUES(@id, @shiftID, @date)

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

SET @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

--REFESH EmployeeShiftView

EXEC SP\_REFRESHVIEW 'EmployeeShiftView'

END

GO

C# code:

try

{

string phone = tbxPhone.Text;

string shiftID = cbxShift.Text;

DateTime date = dtpDateOfShift.Value;

DataProvider.Instance.ExecuteNonQuery("EXEC dbo.insertEmployeeToShift @phone , @date , @shiftID", new object[] { phone, date, shiftID });

dataShift = DataProvider.Instance.ExecuteQuery("EXEC dbo.getShiftDetailByWeek");

reloadShiftInfo(dataShift);

}

catch (SqlException ev)

{

MessageBox.Show("Thêm thất bại.\n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

1. **Procedure cập nhật thông tin nhân viên trong trong bảng ca trực:**

SQL:

DROP PROCEDURE IF EXISTS updateEmployeeInShift

GO

CREATE PROCEDURE updateEmployeeInShift

@phone VARCHAR(10),

@date DATE,

@shiftID VARCHAR(5)

AS

BEGIN

BEGIN TRANSACTION

--CẬP NHẬT NHÂN VIÊN TRONG BẢNG ON DUTY

DECLARE @id VARCHAR(4)

SELECT @id = ID FROM Employee WHERE phone = @phone

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

UPDATE On\_Duty

SET empID = @id

WHERE date = @date AND shiftID = @shiftID

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

SELECT @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

--REFESH EmployeeShiftView

EXEC SP\_REFRESHVIEW 'EmployeeShiftView'

END

GO

C# code:

try

{

string phone = tbxPhone.Text;

string shiftID = cbxShift.Text;

DateTime date = dtpDateOfShift.Value;

DataProvider.Instance.ExecuteNonQuery("EXEC dbo.updateEmployeeInShift @phone , @date , @shiftID", new object[] { phone, date, shiftID });

dataShift = DataProvider.Instance.ExecuteQuery("EXEC dbo.getShiftDetailByWeek");

reloadShiftInfo(dataShift);

}

catch (SqlException ev)

{

MessageBox.Show("Cập nhật thất bại.\n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

1. **Procedure xóa nhân viên khỏi ca trực:**

SQL:

DROP PROCEDURE IF EXISTS deleteEmployeeInShift

GO

CREATE PROCEDURE deleteEmployeeInShift

@phone VARCHAR(10)

AS

BEGIN

BEGIN TRANSACTION

--XÓA NHÂN VIÊN TRONG BẢNG ON DUTY

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

DELETE FROM On\_Duty

WHERE empID = (SELECT ID FROM Employee WHERE phone = @phone)

END TRY

BEGIN CATCH

SET @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

--REFESH EmployeeShiftView

EXEC SP\_REFRESHVIEW 'EmployeeShiftView'

END

GO

C# code:

try

{

string phone = tbxPhone.Text;

DataProvider.Instance.ExecuteNonQuery("EXEC dbo.deleteEmployeeInShift @phone ", new object[] { phone });

dataShift = DataProvider.Instance.ExecuteQuery("EXEC dbo.getShiftDetailByWeek");

reloadShiftInfo(dataShift);

reload();

}

catch (SqlException ev)

{

MessageBox.Show("Xóa thất bại.\n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

1. **Procedure lấy ra danh sách từ view shiftdetailview trong tuần của ngày hiện tại:**

SQL:

DROP PROCEDURE IF EXISTS getShiftDetailByWeek

GO

CREATE PROCEDURE getShiftDetailByWeek

AS

BEGIN

DECLARE @firstDayOfWeek DATE --NGÀY ĐẦU TIÊN CỦA TUẦN

DECLARE @lastDayOfWeek DATE --NGÀY CUỐI CÙNG CỦA TUẦN

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

SET @firstDayOfWeek = DATEADD(DAY, -(DATEPART(WEEKDAY, GETDATE()) - 2), CAST(GETDATE() AS DATE))

SET @lastDayOfWeek = DATEADD(DAY, 6 - (DATEPART(WEEKDAY, GETDATE()) - 2), CAST(GETDATE() AS DATE))

SELECT \* FROM ShiftDetailView

WHERE date BETWEEN @firstDayOfWeek AND @lastDayOfWeek

ORDER BY date ASC, ID ASC

END TRY

BEGIN CATCH

SET @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

END

GO

C# code:

try

{

dataShift.Clear();

dataShift = DataProvider.Instance.ExecuteQuery("EXEC dbo.getShiftDetailByWeek");

}

catch (SqlException ev)

{

MessageBox.Show("Lỗi kết nối.\nDo:\n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

1. **Procedure lấy ra danh sách từ view shiftdetailview trong tuần tiếp theo của ngày hiện tại:**

SQL:

CREATE PROCEDURE getNextWeekShiftDetail

AS

BEGIN

DECLARE @firstDayOfWeek DATE --NGÀY ĐẦU TIÊN CỦA TUẦN

DECLARE @lastDayOfWeek DATE --NGÀY CUỐI CÙNG CỦA TUẦN

DECLARE @errorMessage NVARCHAR(max)

BEGIN TRY

SET @firstDayOfWeek = DATEADD(DAY, -(DATEPART(WEEKDAY, GETDATE()) - 2) + 7, CAST(GETDATE() AS DATE))

SET @lastDayOfWeek = DATEADD(DAY, 6 - (DATEPART(WEEKDAY, GETDATE()) - 2) + 7, CAST(GETDATE() AS DATE))

SELECT \* FROM ShiftDetailView

WHERE date BETWEEN @firstDayOfWeek AND @lastDayOfWeek

ORDER BY date ASC, ID ASC

END TRY

BEGIN CATCH

SET @errorMessage = ERROR\_MESSAGE()

RAISERROR(@errorMessage, 16, 1)

END CATCH

END

GO

C# code:

try

{

dataShift.Clear();

dataShift = DataProvider.Instance.ExecuteQuery("EXEC dbo.getNextWeekShiftDetail");

}

catch (SqlException ev)

{

MessageBox.Show("Lỗi kết nối.\nDo:\n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

* + 1. **Functions:**

1. **Function lấy job detail từ bảng job khi biết id công việc:**

SQL:

DROP FUNCTION IF EXISTS getJobDetail

GO

CREATE FUNCTION getJobDetail

(

@jobID NVARCHAR(3)

)

RETURNS NVARCHAR(max)

AS

BEGIN

DECLARE @jobDetail NVARCHAR(max)

SELECT @jobDetail = jobDetail

FROM Job

WHERE ID = @jobID

RETURN @jobDetail

END

GO

C# code:

try

{

cbxJob.Text = DataProvider.Instance.ExecuteScalar("SELECT dbo.getJobDetail ( @job )", new object[] { data.Rows[0][6].ToString() }).ToString();

tbxPenatyPoint.Text = data.Rows[0][7].ToString();

}

catch (SqlException ev)

{

MessageBox.Show("Load dữ liệu thất bại! \nDo: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

1. **Function lấy ra thông tin nhân viên từ bảng employee khi biết id nhân viên:**

SQL:

DROP FUNCTION IF EXISTS getEmployeeInfo

GO

CREATE FUNCTION getEmployeeInfo

(

@name NVARCHAR(max),

@job NVARCHAR(max)

)

RETURNS TABLE

AS

RETURN

(

SELECT \* FROM Employee

WHERE name = @name AND jobID = (SELECT ID FROM Job WHERE jobDetail = @job)

)

GO

C# code:

try

{

data.Clear();

data = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.getEmployeeInfoByName ( @name )", new object[] { name });

}

catch (SqlException ev)

{

MessageBox.Show("Tìm kiếm nhân viên thất bại! \nDo: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

1. **Function lấy ra lương của công việc khi biết tên công viêc:**

SQL:

DROP FUNCTION IF EXISTS getSalary

GO

CREATE FUNCTION getSalary

(@job NVARCHAR(max))

RETURNS INT

AS

BEGIN

DECLARE @salary INT

SELECT @salary = salary

FROM Job

WHERE jobDetail = @job

RETURN @salary

END

GO

C# code:

try

{

tbxCurentSalary.Text = DataProvider.Instance.ExecuteScalar("SELECT dbo.getSalary ( @job )", new object[] { row[0].ToString() }).ToString();

}

catch (SqlException ev)

{

MessageBox.Show("Load dữ liệu thất bại! \nDo: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

1. **Function lấy ra mã nhân viên từ bảng employee khi biết kí tự đầu là EF hoặc EP:**

SQL:

DROP FUNCTION IF EXISTS getEmployeeID

GO

CREATE FUNCTION getEmployeeID

(

@empCategory NVARCHAR(max)

)

RETURNS TABLE

AS

RETURN

(

SELECT name FROM Employee

WHERE (@empCategory = 'Fulltime' AND ID LIKE 'EF%')

OR (@empCategory <> 'Fulltime' AND ID LIKE 'EP%')

)

GO

C# code:

try

{

dataName.Clear();

dataName = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.getEmployeeID ( @cate )", new object[] { cate });

}

catch (SqlException ev)

{

MessageBox.Show("Tìm kiếm nhân viên thất bại! \nDo: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

1. **Function lấy ra thông tin nhân viên từ view employeedetailView khi biết danh sách tên nhân viên:**

SQL:

DROP FUNCTION IF EXISTS getEmployeeInfoByName

GO

CREATE FUNCTION getEmployeeInfoByName

(

@name NVARCHAR(max)

)

RETURNS TABLE

AS

RETURN

(

SELECT \* FROM EmployeeDetailView

WHERE name = @name

)

GO

C# code:

try

{

data.Clear();

data = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.getEmployeeInfoByName ( @name )", new object[] { name });

}

catch (SqlException ev)

{

MessageBox.Show("Tìm kiếm nhân viên thất bại! \nDo: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

1. **Function lấy ra thông tin nhân viên từ view employeedetailview khi biết công việc:**

SQL:

DROP FUNCTION IF EXISTS getEmployeeInfoByJob

GO

CREATE FUNCTION getEmployeeInfoByJob

(

@job NVARCHAR(max)

)

RETURNS TABLE

AS

RETURN

(

SELECT \* FROM EmployeeDetailView

WHERE job = @job

)

GO

C# code:

try

{

dataEmpShift.Clear();

dataEmpShift = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.getEmployeeInfoByJob ( @job )", new object[] { job });

}

catch (SqlException ev)

{

MessageBox.Show("Tìm kiếm nhân viên thất bại! \nDo: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

1. **Function lấy ra tên nhân viên từ bảng employee khi ngày tuyển dụng:**

SQL:

DROP FUNCTION IF EXISTS getEmployeeNameByRecruitmentDate

GO

CREATE FUNCTION getEmployeeNameByRecruitmentDate

(

@recruitmentDate DATE

)

RETURNS TABLE

AS

RETURN

(

SELECT name FROM Employee

WHERE recruitmentDate = @recruitmentDate

)

GO

C# code:

try

{

data.Clear();

data = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.getEmployeeNameByRecruitmentDate ( @date )", new object[] { date });

}

catch (SqlException ev)

{

MessageBox.Show("Tìm kiếm nhân viên thất bại! \nDo: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

1. **Function tạo ID nhân viên tự động:**

SQL:

DROP FUNCTION IF EXISTS createID

GO

CREATE FUNCTION createID

(

@empCategory VARCHAR(max)

)

RETURNS NVARCHAR(4)

AS

BEGIN

DECLARE @id NVARCHAR(4)

--LẤY ID NHÂN VIÊN CUỐI CÙNG

DECLARE @lastID NVARCHAR(4)

IF @empCategory = 'Fulltime'

SELECT @lastID = MAX(ID)

FROM Employee

WHERE ID LIKE 'EF%'

ELSE

SELECT @lastID = MAX(ID)

FROM Employee

WHERE ID LIKE 'EP%'

--NẾU CHƯA CÓ NHÂN VIÊN NÀO THÌ ID SẼ BẮT ĐẦU BẰNG 'EF01' HOẶC EP01

IF @lastID IS NULL

IF @empCategory = 'Fulltime'

SET @id = 'EF01'

ELSE

SET @id = 'EP01'

ELSE

IF @empCategory = 'Fulltime'

SET @id = 'EF' + RIGHT('00' + CAST(CAST(RIGHT(@lastID, 2) AS INT) + 1 AS NVARCHAR(2)), 2)

ELSE

SET @id = 'EP' + RIGHT('00' + CAST(CAST(RIGHT(@lastID, 2) AS INT) + 1 AS NVARCHAR(2)), 2)

RETURN @id

END

GO

C# code:

try

{

tbxIDEmp.Text = DataProvider.Instance.ExecuteScalar("select dbo.createID ( @cate )", new object[] { cate }).ToString();

}

catch (SqlException ev)

{

MessageBox.Show("Load dữ liệu thất bại! \nDo: \n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

1. **Function lọc nhân viên theo tên nhân viên trong shiftdetailview:**

SQL:

DROP FUNCTION IF EXISTS filterEmployeeByName

GO

CREATE FUNCTION filterEmployeeByName

(@name NVARCHAR(max))

RETURNS TABLE

AS

RETURN

(

SELECT \* FROM ShiftDetailView

WHERE name LIKE '%' + @name + '%'

)

GO

C# code:

try

{

dataShift.Clear();

dataShift = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.filterEmployeeByName ( @name )", new object[] { name });

}

catch (SqlException ev)

{

MessageBox.Show("Lỗi kết nối.\nDo:\n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

1. **Function lọc nhân viên theo ngày trực trong shiftdetailview:**

SQL:

DROP FUNCTION IF EXISTS filterEmployeeByDate

GO

CREATE FUNCTION filterEmployeeByDate

(@date DATE)

RETURNS TABLE

AS

RETURN

(

SELECT \* FROM ShiftDetailView

WHERE date = @date

)

GO

C# code:

try

{

dataShift.Clear();

dataShift = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.filterEmployeeByDate ( @date )", new object[] { date });

}

catch (SqlException ev)

{

MessageBox.Show("Lỗi kết nối.\nDo:\n" + ev.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

* 1. **Chức năng Product:**

Quản lý tất cả các món, bao gồm loại món và nguyên liệu cho từng món

* + 1. **Views:**

SQL:

1. **View hiển thị danh sách Product**

CREATE VIEW ProductView AS

SELECT p.id, p.name, p.price, pc.name AS productcategory, p.status

FROM Product p, ProductCategory pc

WHERE p.categoryID = pc.id AND p.status != -1

GO

1. **View hiển thị danh sách loại Product**

CREATE VIEW ProductCategoryView AS

SELECT id, name

FROM ProductCategory

GO

C# code:

try

{

dtProductCategory = DataProvider.Instance.ExecuteQuery("SELECT \* FROM ProductCategoryView");

foreach (DataRow row in dtProductCategory.Rows)

{

cbbProductCategories.Items.Add(row["Name"]);

}

dgvProductCategory.DataSource = dtProductCategory;

dgvProductCategory.Columns["id"].HeaderText = "Mã";

dgvProductCategory.Columns["name"].HeaderText = "Tên";

dtProduct = DataProvider.Instance.ExecuteQuery("SELECT \* FROM ProductView");

dgvProduct.DataSource = dtProduct;

dgvProduct.Columns["id"].HeaderText = "Mã";

dgvProduct.Columns["name"].HeaderText = "Tên";

dgvProduct.Columns["price"].HeaderText = "Giá";

dgvProduct.Columns["productcategory"].HeaderText = "Loại";

dgvProduct.Columns["status"].HeaderText = "Trạng thái";

dtMaterial = DataProvider.Instance.ExecuteQuery("SELECT \* FROM Material");

foreach (DataRow row in dtMaterial.Rows)

{

cbbMaterialName.Items.Add(row["Name"]);

}

}

catch (SqlException err)

{

MessageBox.Show(err.Message, "Lỗi", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

* + 1. **Functions:**

SQL:

**Liệt kê các nguyên liệu theo Product**

CREATE FUNCTION GetMaterials(@productID VARCHAR(3))

RETURNS TABLE

AS

RETURN

(

SELECT m.name AS materialname, pm.quantity

FROM Material m, Product\_Material pm, Product p

WHERE pm.materialID = m.ID AND pm.productID = p.ID AND pm.productID = @productID

)

GO

C# code:

try

{

// Get selected row

int index = dgvProduct.CurrentCell.RowIndex;

DataGridViewRow selectedRow = dgvProduct.Rows[index];

IDProduct = tbxIDProduct.Text = selectedRow.Cells["id"].Value.ToString();

tbxNameProduct.Text = selectedRow.Cells["name"].Value.ToString();

tbxPriceProduct.Text = selectedRow.Cells["price"].Value.ToString();

cbbProductCategories.Text = selectedRow.Cells["productcategory"].Value.ToString();

tbxStatusProduct.Text = selectedRow.Cells["status"].Value.ToString();

dtProductMaterial = DataProvider.Instance.ExecuteQuery("SELECT \* FROM GetMaterials ( @productID )"

, new object[] { selectedRow.Cells["id"].Value.ToString() });

dgvMaterialProduct.DataSource = dtProductMaterial;

dgvMaterialProduct.Columns["materialname"].HeaderText = "Tên";

dgvMaterialProduct.Columns["quantity"].HeaderText = "SL";

}

catch (SqlException err)

{

MessageBox.Show(err.Message, "Lỗi", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

* + 1. **Procedure:**

1. **Thủ tục thêm Product**

SQL:

CREATE PROCEDURE AddProduct

@id VARCHAR(3),

@name NVARCHAR(MAX),

@categoryID VARCHAR(4),

@price DECIMAL(10, 2),

@status INT

AS

BEGIN

SET NOCOUNT ON;

-- Check for null or empty inputs for varchar/nvarchar fields

IF @id IS NULL OR LTRIM(RTRIM(@id)) = ''

BEGIN

RAISERROR('Error: ID must not be null or empty.', 16, 1);

RETURN;

END

IF @name IS NULL OR LTRIM(RTRIM(@name)) = ''

BEGIN

RAISERROR('Error: Name must not be null or empty.', 16, 1);

RETURN;

END

IF @categoryID IS NULL OR LTRIM(RTRIM(@categoryID)) = ''

BEGIN

RAISERROR('Error: Category ID must not be null or empty.', 16, 1);

RETURN;

END

-- Check for valid decimal and status values

IF @price IS NULL OR @price <= 0 OR LTRIM(RTRIM(@categoryID)) = ''

BEGIN

RAISERROR('Error: Price must not be null and cannot be negative.', 16, 1);

RETURN;

END

IF @status IS NULL OR @status < 0 OR LTRIM(RTRIM(@categoryID)) = ''

BEGIN

RAISERROR('Error: Status must be >= 0 and cannot be null.', 16, 1);

RETURN;

END

DECLARE @ErrorMessage NVARCHAR(4000);

-- If all checks pass, proceed with the insert

BEGIN TRY

INSERT INTO Product (id, name, categoryID, price, status)

VALUES (@id, @name, @categoryID, @price, @status);

END TRY

BEGIN CATCH

-- Catch block to handle SQL errors during the insert operation

SET @ErrorMessage = ERROR\_MESSAGE();

RAISERROR(@ErrorMessage, 16, 1);

END CATCH

END;

GO

C# code:

Try { // Add product

DataProvider.Instance.ExecuteNonQuery("EXEC AddProduct @id , @name , @categoryID , @price , @status"

, new object[] { "P00", tbxNameProduct.Text, productCategoryID, tbxPriceProduct.Text, tbxStatusProduct.Text });

MessageBox.Show("Thêm sản phẩm thành công!", "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);

LoadData();

} catch (SqlException err)

{

MessageBox.Show("Thêm sản phẩm thất bại!\n" + err.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

**Trigger: Cập nhật ID tự động cho Product mới**

CREATE TRIGGER UpdateProductID

ON Product

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

BEGIN TRANSACTION; -- Start the transaction

DECLARE @lastID VARCHAR(3);

SELECT TOP 1 @lastID = ID

FROM Product

ORDER BY ID DESC;

DECLARE @newID VARCHAR(3);

SET @newID = 'P' + RIGHT('00' + CAST(CAST(RIGHT(@lastID, 2) AS INT) + 1 AS VARCHAR(2)), 2);

UPDATE Product

SET ID = @newID

WHERE ID IN (SELECT ID FROM inserted);

COMMIT TRANSACTION; -- Commit the transaction if no errors

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION; -- Roll back the transaction on error

-- Capture and raise the error information

DECLARE @ErrorMessage NVARCHAR(4000) = ERROR\_MESSAGE();

DECLARE @ErrorSeverity INT = ERROR\_SEVERITY();

DECLARE @ErrorState INT = ERROR\_STATE();

RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH

END;

GO

1. **Thủ tục sửa Product:**

SQL:

CREATE PROCEDURE UpdateProductByID

@ID VARCHAR(3),

@name NVARCHAR(MAX),

@categoryID VARCHAR(4),

@price DECIMAL(10, 2),

@status INT

AS

BEGIN

SET NOCOUNT ON;

-- Check for null or empty inputs for varchar/nvarchar fields

IF @ID IS NULL OR LTRIM(RTRIM(@ID)) = ''

BEGIN

RAISERROR('Error: ID must not be null or empty.', 16, 1);

RETURN;

END

IF @name IS NULL OR LTRIM(RTRIM(@name)) = ''

BEGIN

RAISERROR('Error: Name must not be null or empty.', 16, 1);

RETURN;

END

IF @categoryID IS NULL OR LTRIM(RTRIM(@categoryID)) = ''

BEGIN

RAISERROR('Error: Category ID must not be null or empty.', 16, 1);

RETURN;

END

-- Check for valid decimal and status values

IF @price IS NULL OR @price <= 0

BEGIN

RAISERROR('Error: Price must not be null and cannot be negative.', 16, 1);

RETURN;

END

IF @status IS NULL OR @status < 0 OR LTRIM(RTRIM(@categoryID)) = ''

BEGIN

RAISERROR('Error: Status must be >= 0 and cannot be null.', 16, 1);

RETURN;

END

DECLARE @ErrorMessage NVARCHAR(4000);

-- If all checks pass, proceed with the update

BEGIN TRY

UPDATE Product

SET name = @name, categoryID = @categoryID, price = @price, status = @status

WHERE ID = @ID;

END TRY

BEGIN CATCH

-- Catch block to handle SQL errors during the update operation

SET @ErrorMessage = ERROR\_MESSAGE();

RAISERROR(@ErrorMessage, 16, 1);

END CATCH

END;

GO

C# code:

try

{

// Edit product

DataProvider.Instance.ExecuteNonQuery("EXEC UpdateProductByID @id , @name , @categoryID , @price , @status"

, new object[] { tbxIDProduct.Text, tbxNameProduct.Text, productCategoryID, tbxPriceProduct.Text, tbxStatusProduct.Text });

MessageBox.Show("Sửa sản phẩm thành công", "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);

LoadData();

}

catch (SqlException err)

{

MessageBox.Show("Sửa sản phẩm thất bại\n" + err.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

1. **Thủ tục xóa Product:** Sửa status = -1 , được coi như đã xóa

SQL:

CREATE PROCEDURE DeleteProductByID

@ID VARCHAR(3)

AS

BEGIN

SET NOCOUNT ON;

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

BEGIN TRY

BEGIN TRANSACTION; -- Start the transaction

-- First, delete associated data from Product\_Material

DELETE FROM Product\_Material

WHERE productID = @ID;

-- Then, update the status in the Product table

UPDATE Product

SET status = -1

WHERE ID = @ID;

COMMIT TRANSACTION; -- Commit the transaction if no errors

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION; -- Roll back the transaction on error

-- If an error occurs, catch it and throw an error message

SET @ErrorMessage = ERROR\_MESSAGE();

SET @ErrorSeverity = ERROR\_SEVERITY();

SET @ErrorState = ERROR\_STATE();

RAISERROR(@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH

END;

GO

1. **Thủ tục thêm loại Product:**

SQL:

CREATE PROCEDURE addProductCategory

@ID varchar(4),

@name nvarchar(max)

AS

BEGIN

-- Check for null or empty inputs for varchar/nvarchar fields

IF @id IS NULL OR LTRIM(RTRIM(@id)) = ''

BEGIN

RAISERROR('Error: ID must not be null or empty.', 16, 1);

RETURN;

END

IF @name IS NULL OR LTRIM(RTRIM(@name)) = ''

BEGIN

RAISERROR('Error: Name must not be null or empty.', 16, 1);

RETURN;

END

DECLARE @ErrorMessage NVARCHAR(4000);

-- If all checks pass, proceed with the insert

BEGIN TRY

INSERT INTO ProductCategory(ID, name)

VALUES (@ID, @name)

END TRY

BEGIN CATCH

-- Catch block to handle SQL errors during the insert operation

SET @ErrorMessage = ERROR\_MESSAGE();

RAISERROR(@ErrorMessage, 16, 1);

END CATCH

END

GO

C# code:

try

{

// Add product category

DataProvider.Instance.ExecuteNonQuery("EXEC AddProductCategory @id , @name"

, new object[] { "C00", tbxNameCate.Text });

MessageBox.Show("Thêm loại món thành công", "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);

LoadData();

}

catch (SqlException err)

{

MessageBox.Show("Thêm loại món thất bại\n" + err.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

**Trigger: Cập nhật lại ID tự động cho loại Product**

CREATE TRIGGER UpdateProductCategoryID

ON ProductCategory

AFTER INSERT

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION;

DECLARE @lastID VARCHAR(4)

SELECT TOP 1 @lastID = ID

FROM ProductCategory

ORDER BY ID DESC

DECLARE @newID VARCHAR(4)

SET @newID = 'PC' + RIGHT(CAST(CAST(RIGHT(@lastID, 2) AS INT) + 1 AS VARCHAR(2)), 2)

UPDATE ProductCategory

SET ID = @newID

WHERE ID IN (SELECT ID FROM inserted)

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

-- Roll back the transaction on error

ROLLBACK TRANSACTION;

-- Printing the error message and the severity

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

SELECT

@ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE();

RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH

END

GO

1. **Thủ tục sửa loại Product:**

SQL:

CREATE PROCEDURE updateProductCategoryByID

@ID varchar(4),

@name nvarchar(max)

AS

BEGIN

-- Check for null or empty inputs for varchar/nvarchar fields

IF @id IS NULL OR LTRIM(RTRIM(@id)) = ''

BEGIN

RAISERROR('Error: ID must not be null or empty.', 16, 1);

RETURN;

END

IF @name IS NULL OR LTRIM(RTRIM(@name)) = ''

BEGIN

RAISERROR('Error: Name must not be null or empty.', 16, 1);

RETURN;

END

DECLARE @ErrorMessage NVARCHAR(4000);

-- If all checks pass, proceed with the insert

BEGIN TRY

UPDATE ProductCategory

SET name = @name

WHERE ID = @ID

END TRY

BEGIN CATCH

-- Catch block to handle SQL errors during the insert operation

SET @ErrorMessage = ERROR\_MESSAGE();

RAISERROR(@ErrorMessage, 16, 1);

END CATCH

END

GO

C# code:

try

{

// Edit product category

DataProvider.Instance.ExecuteNonQuery("EXEC updateProductCategoryByID @id , @name"

, new object[] { tbxIDCate.Text, tbxNameCate.Text });

MessageBox.Show("Sửa loại món thành công", "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);

LoadData();

}

catch (SqlException err)

{

MessageBox.Show("Sửa loại món thất bại\n" + err.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

1. **Thủ tục thêm nguyên liệu cho Product:**

SQL:

CREATE PROCEDURE addProductMaterial

@productID varchar(3),

@materialID varchar(3),

@quantity INT

AS

BEGIN

-- Check for null or empty inputs for varchar/nvarchar fields

IF @productID IS NULL OR LTRIM(RTRIM(@productID)) = ''

BEGIN

RAISERROR('Error: ProductID must not be null or empty.', 16, 1);

RETURN;

END

IF @materialID IS NULL OR LTRIM(RTRIM(@materialID)) = ''

BEGIN

RAISERROR('Error: MaterialID must not be null or empty.', 16, 1);

RETURN;

END

IF @quantity IS NULL OR @quantity <= 0 OR LTRIM(RTRIM(@quantity)) = ''

BEGIN

RAISERROR('Error: Quantity must be > 0 and cannot be null.', 16, 1);

RETURN;

END

DECLARE @ErrorMessage NVARCHAR(4000);

-- If all checks pass, proceed with the insert

BEGIN TRY

INSERT INTO Product\_Material(productID, materialID, quantity)

VALUES (@productID, @materialID, @quantity)

END TRY

BEGIN CATCH

-- Catch block to handle SQL errors during the insert operation

SET @ErrorMessage = ERROR\_MESSAGE();

RAISERROR(@ErrorMessage, 16, 1);

END CATCH

END

GO

C# code:

try

{

string materialID = "";

foreach (DataRow row in dtMaterial.Rows)

{

if (row["Name"].ToString() == cbbMaterialName.SelectedItem.ToString())

{

materialID = row["ID"].ToString();

break;

}

}

// Add material

DataProvider.Instance.ExecuteNonQuery("EXEC addProductMaterial @productID , @materialID , @quantity"

, new object[] { IDProduct, materialID, txbMaterialCount.Text });

MessageBox.Show("Thêm nguyên liệu thành công", "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);

LoadData();

LoadProductMaterial();

}

catch (SqlException err)

{

MessageBox.Show("Thêm nguyên liệu thất bại\n" + err.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

1. **Thủ tục sửa nguyên liệu cho Product:**

SQL:

CREATE PROCEDURE updateProductMaterialByID

@productID varchar(3),

@materialID varchar(3),

@quantity INT

AS

BEGIN

-- Check for null or empty inputs for varchar/nvarchar fields

IF @productID IS NULL OR LTRIM(RTRIM(@productID)) = ''

BEGIN

RAISERROR('Error: ProductID must not be null or empty.', 16, 1);

RETURN;

END

IF @materialID IS NULL OR LTRIM(RTRIM(@materialID)) = ''

BEGIN

RAISERROR('Error: MaterialID must not be null or empty.', 16, 1);

RETURN;

END

IF @quantity IS NULL OR @quantity <= 0 OR LTRIM(RTRIM(@quantity)) = ''

BEGIN

RAISERROR('Error: Quantity must be > 0 and cannot be null.', 16, 1);

RETURN;

END

DECLARE @ErrorMessage NVARCHAR(4000);

-- If all checks pass, proceed with the insert

BEGIN TRY

UPDATE Product\_Material

SET quantity = @quantity

WHERE productID = @productID AND materialID = @materialID

END TRY

BEGIN CATCH

-- Catch block to handle SQL errors during the insert operation

SET @ErrorMessage = ERROR\_MESSAGE();

RAISERROR(@ErrorMessage, 16, 1);

END CATCH

END

GO

C# code:

try

{

string materialID = "";

foreach (DataRow row in dtMaterial.Rows)

{

if (row["Name"].ToString() == cbbMaterialName.SelectedItem.ToString())

{

materialID = row["ID"].ToString();

break;

}

}

// Edit material

DataProvider.Instance.ExecuteNonQuery("EXEC updateProductMaterialByID @productID , @materialID , @quantity"

, new object[] { IDProduct, materialID, txbMaterialCount.Text });

MessageBox.Show("Sửa nguyên liệu thành công", "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);

LoadData();

LoadProductMaterial();

}

catch (SqlException err)

{

MessageBox.Show("Sửa nguyên liệu thất bại\n" + err.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

1. **Thủ tục xóa nguyên liệu cho Product:**

SQL:

CREATE PROCEDURE deleteProductMaterialByID

@productID VARCHAR(3),

@materialID VARCHAR(3)

AS

BEGIN

-- Check for null or empty inputs for varchar/nvarchar fields

IF @productID IS NULL OR LTRIM(RTRIM(@productID)) = ''

BEGIN

RAISERROR('Error: ProductID must not be null or empty.', 16, 1);

RETURN;

END

IF @materialID IS NULL OR LTRIM(RTRIM(@materialID)) = ''

BEGIN

RAISERROR('Error: MaterialID must not be null or empty.', 16, 1);

RETURN;

END

DECLARE @ErrorMessage NVARCHAR(4000);

-- If all checks pass, proceed with the insert

BEGIN TRY

DELETE FROM Product\_Material

WHERE productID = @productID AND materialID = @materialID

END TRY

BEGIN CATCH

-- Catch block to handle SQL errors during the insert operation

SET @ErrorMessage = ERROR\_MESSAGE();

RAISERROR(@ErrorMessage, 16, 1);

END CATCH

END

GO

C# code:

try

{

// show confirm dialog

DialogResult result = MessageBox.Show("Bạn có chắc chắn muốn xóa nguyên liệu này?",

"Xác nhận", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (result == DialogResult.No)

{

return;

}

string materialID = "";

foreach (DataRow row in dtMaterial.Rows)

{

if (row["Name"].ToString() == cbbMaterialName.SelectedItem.ToString())

{

materialID = row["ID"].ToString();

break;

}

}

// Delete material

DataProvider.Instance.ExecuteNonQuery("EXEC deleteProductMaterialByID @productID , @materialID"

, new object[] { IDProduct, materialID });

MessageBox.Show("Xóa nguyên liệu thành công", "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);

LoadData();

LoadProductMaterial();

}

catch (SqlException err)

{

MessageBox.Show("Xóa nguyên liệu thất bại\n" + err.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

**Trigger cho 3 thủ tục thêm, sửa, xóa nguyên liệu của Product:** gọi thủ tục cập nhật trạng thái món mỗi khi trạng thái của Material được cập nhật

SQL:

CREATE TRIGGER UpdateProduct\_ProductMaterialTrigger

ON Product\_Material

AFTER INSERT, UPDATE, DELETE

AS

BEGIN

SET NOCOUNT ON;

-- Prevent trigger from firing recursively

IF TRIGGER\_NESTLEVEL() > 1 RETURN;

-- Call the procedure to update product statuses

EXEC updateProductStatus;

END;

GO

**Procedure cập nhật status cho Product:** cập nhật số lượng món khả dụng theo số lượng nguyên liệu còn lại

SQL:

CREATE PROCEDURE updateProductStatus

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

BEGIN TRANSACTION; -- Start of transaction

-- Temporary table to store minimum product counts based on material availability

CREATE TABLE #ProductStatusUpdate (

ProductID VARCHAR(50),

MaxProductCount INT

);

-- Insert the calculated maximum product counts into the temporary table

INSERT INTO #ProductStatusUpdate (ProductID, MaxProductCount)

SELECT

pm.ProductID,

MIN(m.status / pm.quantity) AS MaxProductCount

FROM

Product\_Material pm

INNER JOIN

Material m ON pm.MaterialID = m.ID

GROUP BY

pm.ProductID;

-- Update the Product table's status with the least number of products that can be made

UPDATE

Product

SET

status = psu.MaxProductCount

FROM

Product p

INNER JOIN

#ProductStatusUpdate psu ON p.ID = psu.ProductID;

DROP TABLE #ProductStatusUpdate;

COMMIT TRANSACTION; -- Commit the transaction if no errors

END TRY

BEGIN CATCH

-- Roll back the transaction on error

ROLLBACK TRANSACTION;

-- Printing the error message and the severity

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

SELECT

@ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE();

RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH

END;

GO

* 1. **Chức năng Material:**
     1. **View:**

Tạo view hiển thị nguyên liệu:

SQL:

|  |
| --- |
| CREATE VIEW MaterialDetailss  AS  SELECT \*  FROM Material; |

C# code:

|  |
| --- |
| try  {  dtMaterial = DataProvider.Instance.ExecuteQuery("SELECT \* FROM MaterialDetailss");  dgvMaterial.DataSource = dtMaterial;  dgvMaterial.Columns["id"].HeaderText = "Mã";  dgvMaterial.Columns["name"].HeaderText = "Tên";  dgvMaterial.Columns["status"].HeaderText = "Trạng thái";  }  catch(SqlException err)  {  MessageBox.Show(err.Message);  } |

* + 1. **Procedure**

**a. Procedure thêm nguyên liệu mới:**

SQL:

|  |
| --- |
| CREATE PROCEDURE AddMaterial  @ID VARCHAR(3),  @name NVARCHAR(max),  @status DECIMAL(10,2)    AS  BEGIN  IF (@name = '' OR @name = NULL OR @status > -1 )  THROW 50000, 'Không được để trống tên hoặc trạng thái ', 1  ELSE  INSERT INTO Material (ID, name, status)  VALUES (@ID, @name, @status) |

C# code:

|  |
| --- |
| try  {  DataProvider.Instance.ExecuteNonQuery("EXEC AddMaterial @id , @name , @status"  , new object[] { "M00", tbxNameMaterial.Text, tbxStatusMaterial.Text});  MessageBox.Show("Thêm nguyên liệu thành công", "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);  LoadData();  }  catch (SqlException err)  {  MessageBox.Show("Thêm nguyên liệu thất bại\n" + err.Message, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Warning);  return;  } |

**Trigger tự động đặt ID**

|  |
| --- |
| CREATE TRIGGER UpdateMaterialID  ON Material  AFTER INSERT  AS  BEGIN  DECLARE @lastID VARCHAR(3)  SELECT TOP 1 @lastID = ID  FROM Material  ORDER BY ID DESC  DECLARE @newID VARCHAR(3)  SET @newID = 'M' + RIGHT(CAST(CAST(RIGHT(@lastID, 2) AS INT) + 1 AS VARCHAR(2)), 2)  UPDATE Material  SET ID = @newID  WHERE ID IN (SELECT ID FROM inserted)  END; |

**Trigger Roll back lại khi người dùng nhập tên trùng**

|  |
| --- |
| CREATE OR ALTER TRIGGER PreventDuplicateMaterialName  ON Material  INSTEAD OF INSERT  AS  BEGIN  DECLARE @ExistingName NVARCHAR(max);  SELECT @ExistingName = name FROM Material WHERE name IN (SELECT name FROM inserted);  IF @ExistingName IS NOT NULL  BEGIN  RAISERROR ('Cannot insert. Material with the same name already exists.', 16, 1);  ROLLBACK TRANSACTION;  END  ELSE  BEGIN  INSERT INTO Material (ID, name, status)  SELECT ID, name, status FROM inserted;  END  END; |

**b. Procedure sửa nguyên liệu đã có**

SQL:

|  |
| --- |
| CREATE PROCEDURE UpdateMaterial  @ID VARCHAR(3),  @name NVARCHAR(max),  @status DECIMAL(10,2)  AS  BEGIN  IF (@name = ‘’ OR @name = NULL OR @status > -1 )  THROW 50000, ‘Không được để trống tên hoặc trạng thái ‘, 1  ELSE  UPDATE Material  SET name = @name, status = @status  WHERE ID = @ID |

C# Code:

|  |
| --- |
| try  {  DataProvider.Instance.ExecuteNonQuery(“EXEC UpdateMaterial @id , @name , @status”  , new object[] { IDMaterial, tbxNameMaterial.Text, tbxStatusMaterial.Text }) ;  MessageBox.Show(“Sửa thông tin nhà cung cấp thành công”, “Thông báo”, MessageBoxButtons.OK, MessageBoxIcon.Information);  LoadData();  }  catch (SqlException err)  {  MessageBox.Show(“Sửa thông tin nhà cung cấp thất bại\n” + err.Message, “Thông báo”, MessageBoxButtons.OK, MessageBoxIcon.Warning);  return;  } |

* 1. **Chức năng Supplier:**
     1. **View:**

Tạo view hiển thị các nhà cung cấp

SQL:

|  |
| --- |
| CREATE VIEW SupplierList AS  SELECT \*  FROM Supplier  WHERE address <> N'Hết hạn hợp đồng'; |

C#:

|  |
| --- |
| try  {  dtSupplier = DataProvider.Instance.ExecuteQuery("SELECT \* FROM SupplierList");  dgvsupplier.DataSource = dtSupplier;  dgvsupplier.Columns["ID"].HeaderText = "Mã";  dgvsupplier.Columns["name"].HeaderText = "Tên";  dgvsupplier.Columns["address"].HeaderText = "Địa chỉ";  dgvsupplier.Columns["phone"].HeaderText = "Số điện thoại";  }  catch (SqlException err)  {  MessageBox.Show(err.Message);  } |

* + 1. **Procedure**

**a. Procedure thêm nhà cung cấp mới:**

SQL:

|  |
| --- |
| CREATE PROCEDURE AddSupplier  @ID VARCHAR(4),  @name NVARCHAR(max),  @phone VARCHAR(10),  @address NVARCHAR(max)  AS  BEGIN  IF (@name = '' OR @name = NULL OR @phone = NULL OR @phone = '' OR @address = '' OR @address = NULL )  THROW 50000, 'Không được để trống tên hoặc số điện thoại hoặc địa chỉ ', 1  ELSE  INSERT INTO Supplier (ID, name, address, phone)  VALUES (@ID, @name, @address, @phone)  END |

C# code:

|  |
| --- |
| try  {  // Add Supplier  DataProvider.Instance.ExecuteNonQuery("EXEC AddSupplier @ID , @name , @phone , @address"  , new object[] { "SP00", tbxNameSupplier.Text, tbxPhoneSupplier.Text, tbxAddressSupplier.Text});  MessageBox.Show("Thêm nhà cung cấp thành công", "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);  LoadData();  }  catch (SqlException err)  {  MessageBox.Show(err.Message);  } |

**Trigger tự động đặt ID**

|  |
| --- |
| DROP TRIGGER IF EXISTS UpdateSupplierID  GO  CREATE TRIGGER UpdateSupplierID  ON Supplier  AFTER INSERT  AS  BEGIN  SET NOCOUNT ON;  DECLARE @lastID VARCHAR(4);  DECLARE @newID VARCHAR(4);  -- Find the maximum ID from the current table  SELECT @lastID = MAX(ID) FROM Supplier;  IF @lastID IS NOT NULL  SET @newID = 'SP' + RIGHT('00' + CAST(CAST(SUBSTRING(@lastID, 3, 2) AS INT) + 1 AS VARCHAR(2)), 2);  ELSE  SET @newID = 'SP01'; -- Default starting ID if no existing data  -- Update the newly inserted record with a new ID  -- This assumes that only one record is inserted at a time  UPDATE Supplier  SET ID = @newID  WHERE ID IN (SELECT ID FROM inserted);  END;  GO |

**Trigger kiểm tra xem có nhà cung cấp nào trùng số điện thoại không**

|  |
| --- |
| DROP TRIGGER IF EXISTS SamePhone;  GO  CREATE TRIGGER SamePhone  ON Supplier  AFTER INSERT, UPDATE  AS  BEGIN  SET NOCOUNT ON;  DECLARE @phone VARCHAR(20);  DECLARE cur CURSOR FOR  SELECT phone FROM inserted;  OPEN cur;  FETCH NEXT FROM cur INTO @phone;  WHILE @@FETCH\_STATUS = 0  BEGIN  IF EXISTS (  SELECT 1  FROM Supplier  WHERE phone = @phone  GROUP BY phone  HAVING COUNT(\*) > 1  )  BEGIN  RAISERROR ('Không thể chèn. Nhà cung cấp với cùng số điện thoại đã tồn tại.', 16, 1);  ROLLBACK;  RETURN;  END  FETCH NEXT FROM cur INTO @phone;  END  CLOSE cur;  DEALLOCATE cur;  END; |

**b. Procedure sửa giá trị**

SQL:

|  |
| --- |
| CREATE PROCEDURE UpdateSupplier  @ID VARCHAR(4),  @name NVARCHAR(max),  @phone VARCHAR(10),  @address NVARCHAR(max)    AS  BEGIN  IF (@name = '' OR @name = NULL OR @phone = NULL OR @phone = '' OR @address = '' OR @address = NULL )  THROW 50000, 'Không được để trống tên hoặc số điện thoại hoặc địa chỉ ', 1  ELSE  UPDATE Supplier  SET name = @name, address = @address, phone = @phone  WHERE ID = @ID  END |

C# code:

|  |
| --- |
| try  {  // Edit product  DataProvider.Instance.ExecuteNonQuery(“EXEC UpdateSupplier @ID , @name , @phone , @address”  , new object[] { IDSupplier, tbxNameSupplier.Text, tbxPhoneSupplier.Text, tbxAddressSupplier.Text });  MessageBox.Show(“Sửa thông tin nhà cung cấp thành công”, “Thông báo”, MessageBoxButtons.OK, MessageBoxIcon.Information);  LoadData();  }  catch (SqlException err)  {  MessageBox.Show(err.Message);  } |

**c. Procedure xóa một nhà cung cấp**

SQL:

|  |
| --- |
| CREATE PROCEDURE DeleteSupplier  @ID VARCHAR(4)  AS  BEGIN  UPDATE Supplier  SET address = N'Hết hạn hợp đồng'  WHERE ID = @ID;  END |

C# code:

|  |
| --- |
| try  {  // show confirm dialog  DialogResult result = MessageBox.Show("Bạn có chắc chắn muốn xóa nhà cung cấp này?", "Xác nhận", MessageBoxButtons.YesNo, MessageBoxIcon.Question);  if (result == DialogResult.No)  {  return;  }  // Delete product  DataProvider.Instance.ExecuteNonQuery("EXEC DeleteSupplier @id"  , new object[] { tbxIDSupplier.Text });  MessageBox.Show("Xóa sản phẩm thành công", "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);  LoadData();  }  catch (SqlException err)  {  MessageBox.Show(err.Message);  } |

* 1. **Chức năng Expense Bill**
     1. **View**

**View ExpenseBillView để hiển thị dữ liệu phiếu chi lên UI**

|  |
| --- |
| CREATE VIEW dbo.ExpenseBillView  AS  SELECT ID, date, price, detail FROM ExpenseBill;  GO |

*C# code:*

|  |
| --- |
| DataTable data = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.ExpenseBillView");  …  dgvExpenseBill.DataSource = data; |

* + 1. **Procedure**

1. **Procedure thêm mới phiếu chi**

|  |
| --- |
| CREATE PROC addNewExpenseBill  @date DATE, @price DECIMAL(18,2), @detail NVARCHAR(MAX)  AS  BEGIN  IF (@date > GETDATE())  THROW 50000, N'Không được nhập vào ngày trong tương lai', 255  ELSE IF (@price <= 0)  THROW 50000, N'Giá trị phiếu chi phải lớn hơn 0', 255  ELSE IF (@detail = N'')  THROW 50000, N'Chi tiết phiếu chi không được để trống', 255  INSERT INTO ExpenseBill VALUES ('', @detail, @price, @date);  END;  GO |

*C# code:*

|  |
| --- |
| try  {  if (tbxID.Text != "")  {  MessageBox.Show("Vui lòng nhập thông tin phiếu chi mới!", "Thông báo");  initComponent();  }  else if (!double.TryParse(tbxPrice.Text, out double result))  MessageBox.Show("Vui lòng nhập vào giá trị là một số.", "Báo lỗi!");  else  {  DateTime date = dtpDate.Value;  string price = tbxPrice.Text;  string detail = tbxDetail.Text;  DataProvider.Instance.ExecuteNonQuery("EXEC addNewExpenseBill @date , @price , @detail",  new object[] { date, price, detail });  MessageBox.Show("Đã thêm phiếu chi mới thành công.");  initComponent();  }  }  catch (SqlException sqlException)  {  MessageBox.Show(sqlException.Message, "Báo lỗi!");  } |

**Trigger bắt sự kiện thêm phiếu chi mới để tự set ID cho phiếu**

|  |
| --- |
| CREATE TRIGGER setExpenseBillIDTrigger  ON ExpenseBill  AFTER INSERT  AS  BEGIN  DECLARE @max\_id INT;  DECLARE @new\_id VARCHAR(7);    SELECT @max\_id = MAX(CAST(SUBSTRING(ID, 3, LEN(ID)) AS INT)) FROM ExpenseBill WHERE ID LIKE 'EX%';    -- Tạo ID mới  IF @max\_id IS NULL  SET @new\_id = 'EX00001';  ELSE  SET @new\_id = 'EX' + RIGHT('00000' + CAST(@max\_id + 1 AS VARCHAR), 5);    -- Gán ID mới cho bản ghi sẽ được chèn vào  UPDATE ExpenseBill SET ID = @new\_id WHERE ID = '';  END;  GO |

1. **Procedure sửa phiếu chi**

|  |
| --- |
| CREATE PROC updateExpenseBill  @id VARCHAR(7), @date DATE, @price DECIMAL(18,2), @detail NVARCHAR(MAX)  AS  BEGIN  IF (@date > GETDATE())  THROW 50000, N'Không được nhập vào ngày trong tương lai', 255  ELSE IF (@price <= 0)  THROW 50000, N'Giá trị phiếu chi phải lớn hơn 0', 255  ELSE IF (@detail = N'')  THROW 50000, N'Chi tiết phiếu chi không được để trống', 255  UPDATE ExpenseBill SET date = @date, price = @price, detail = @detail  WHERE ID = @id  END; |

*C# code:*

|  |
| --- |
| try  {  if (tbxID.Text == "")  MessageBox.Show("Vui lòng chọn phiếu chi cần sửa trong bảng dữ liệu.", "Thông báo");  else if (!double.TryParse(tbxPrice.Text, out double result))  MessageBox.Show("Vui lòng nhập vào giá trị là một số.", "Báo lỗi!");  else  {  string id = tbxID.Text;  DateTime date = dtpDate.Value;  string price = tbxPrice.Text;  string detail = tbxDetail.Text;  DataProvider.Instance.ExecuteNonQuery("EXEC updateExpenseBill @id , @date , @price , @detail",  new object[] { id, date, price, detail });  MessageBox.Show("Đã sửa phiếu chi thành công.", "Thông báo");  initComponent();  }  }  catch (SqlException sqlException)  {  MessageBox.Show(sqlException.Message, "Báo lỗi!");  } |

1. **Procedure xóa Phiếu chi**

|  |
| --- |
| CREATE PROC deleteExpenseBill  @id VARCHAR(7)  AS  BEGIN  IF (@id = '')  THROW 50000, N'Vui lòng chọn phiếu chi cần xóa trong bảng dữ liệu.', 255  DELETE ExpenseBill  WHERE ID = @id  END; |

*C# code:*

|  |
| --- |
| try  {  string id = tbxID.Text;  DataProvider.Instance.ExecuteNonQuery("EXEC deleteExpenseBill @id",  new object[] { id });  MessageBox.Show("Đã xóa phiếu chi thành công.", "Thông báo");  initComponent();  }  catch (SqlException sqlException)  {  MessageBox.Show(sqlException.Message, "Báo lỗi!");  } |

* + 1. **Function**

1. **Function lọc Phiếu chi theo ID**

|  |
| --- |
| CREATE FUNCTION filterExpenseBillByID  (  @id VARCHAR(7)  )  RETURNS TABLE  AS  RETURN  SELECT \* FROM ExpenseBillView WHERE ID = @id;  GO |

*C# code:*

|  |
| --- |
| DataTable data = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.filterExpenseBillByID( @id )", new object[] {id}); |

1. **Function lọc Phiếu chi theo ngày nhập**

|  |
| --- |
| CREATE FUNCTION filterExpenseBillByDate  (  @date DATE  )  RETURNS TABLE  AS  RETURN  SELECT \* FROM ExpenseBillView WHERE date = @date  GO |

*C# code:*

|  |
| --- |
| DataTable data = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.filterExpenseBillByDate( @date )", new object[] { date }); |

* 1. **Chức năng ReceiptNote**
     1. **View**

1. **View ReceiptNoteView để hiển thị thông tin đơn nhập hàng trên UI (UCReceiptNote)**

|  |
| --- |
| CREATE VIEW ReceiptNoteView  AS  SELECT rn.ID, rn.date, s.name [Supplier Name], e.name [Employee Name], rn.price FROM ReceiptNote rn, Supplier s, Employee e  WHERE rn.empID = e.ID AND rn.supplierID = s.ID  GO |

*C# code:*

|  |
| --- |
| DataTable data = DataProvider.Instance.ExecuteQuery("SELECT \* FROM ReceiptNoteView");  …  DataGridView dgvReceiptNote.DataSource = data; |

1. **View ReceiptNoteListView để hiển thị sơ lược đơn nhập hàng trên UI (UCReceiptNoteDetail)**

|  |
| --- |
| CREATE VIEW ReceiptNoteListView  AS  SELECT ID, date, [Employee Name], [Supplier Name] FROM ReceiptNoteView  GO |

*C# code:*

|  |
| --- |
| DataTable dataRN = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.ReceiptNoteListView");  …  DataGridView dgvRN.DataSource = dataRN; |

* + 1. **Procedure**

1. **Procedure Thêm Đơn nhập hàng**

|  |
| --- |
| CREATE PROC addNewReceiptNote  @date DATE,  @empName NVARCHAR(MAX),  @supName NVARCHAR(MAX)  AS  BEGIN  IF (@date > GETDATE())  THROW 50000, N'Vui lòng không nhập ngày trong tương lai.', 255  ELSE IF (@supName = N'')  THROW 50000, N'Vui lòng chọn nhà cung cấp.', 255  ELSE IF (@empName = N'')  THROW 50000, N'Vui lòng chọn nhân viên nhập hàng.', 255  DECLARE @supID VARCHAR(4), @empID VARCHAR(4);  SELECT @supID = ID FROM Supplier WHERE name = @supName;  SELECT @empID = ID FROM Employee WHERE name = @empName;  INSERT INTO ReceiptNote(ID, date, price, supplierID, empID) VALUES('', @date, 0, @supID, @empID);  END;  GO |

*C# code:*

|  |
| --- |
| try  {  if (tbxID.Text != "")  {  MessageBox.Show("Vui lòng nhập thông tin đơn nhập hàng mới!");  initComponent();  }  else  {  DateTime date = dtpDate.Value;  string supName = cbxSupplier.Text;  string empName = cbxEmployee.Text;  DataProvider.Instance.ExecuteNonQuery("EXEC addNewReceiptNote @date , @empName , @supName",  new object[] { date, empName, supName });  MessageBox.Show("Đã thêm đơn nhập hàng mới thành công.");  initComponent();  loadData();  }  }  catch (SqlException sqlException)  {  MessageBox.Show(sqlException.Message, "Báo lỗi!");  } |

**Trigger tự set ID cho Đơn nhập hàng (ReceiptNote) khi thêm mới**

|  |
| --- |
| CREATE TRIGGER setReceiptNoteIDTrigger  ON ReceiptNote  AFTER INSERT  AS  BEGIN  DECLARE @max\_id INT;  DECLARE @new\_id VARCHAR(7);    SELECT @max\_id = MAX(CAST(SUBSTRING(ID, 3, LEN(ID)) AS INT)) FROM ReceiptNote WHERE ID LIKE 'RN%';    -- Tạo ID mới  IF @max\_id IS NULL  SET @new\_id = 'RN00001';  ELSE  SET @new\_id = 'RN' + RIGHT('00000' + CAST(@max\_id + 1 AS VARCHAR), 5);    -- Gán ID mới cho bản ghi sẽ được chèn vào  UPDATE ReceiptNote SET ID = @new\_id WHERE ID = '';  END;  GO |

1. **Procedure Sửa Đơn nhập hàng**

|  |
| --- |
| CREATE PROC updateNewReceiptNote  @id VARCHAR(7),  @date DATE,  @empName NVARCHAR(MAX),  @supName NVARCHAR(MAX)  AS  BEGIN  IF (@date > GETDATE())  THROW 50000, N'Vui lòng không nhập ngày trong tương lai.', 255  DECLARE @supID VARCHAR(4), @empID VARCHAR(4);  SELECT @supID = ID FROM Supplier WHERE name = @supName;  SELECT @empID = ID FROM Employee WHERE name = @empName;  UPDATE ReceiptNote  SET date = @date, supplierID = @supID, empID = @empID  WHERE ID = @id;  END;  GO |

*C# code:*

|  |
| --- |
| try  {  if (tbxID.Text == "")  {  MessageBox.Show("Vui lòng chọn đơn nhập hàng cần sửa trong bảng dữ liệu!");  }  else  {  string id = tbxID.Text;  DateTime date = dtpDate.Value;  string supName = cbxSupplier.Text;  string empName = cbxEmployee.Text;  DataProvider.Instance.ExecuteNonQuery("EXEC updateNewReceiptNote @id , @date , @empName , @supName",  new object[] { id, date, empName, supName });  MessageBox.Show("Đã sửa đơn nhập hàng thành công.");  initComponent();  loadData();  }  }  catch (SqlException sqlException)  {  MessageBox.Show(sqlException.Message, "Báo lỗi!");  } |

1. **Procedure Xóa Đơn nhập hàng**

|  |
| --- |
| CREATE PROC deleteReceiptNote  @id VARCHAR(7)  AS  BEGIN  IF (@id = '')  THROW 50000, N'Vui lòng chọn đơn nhập hàng cần xóa trong bảng dữ liệu.', 255  DELETE ReceiptNote WHERE ID = @id;  END; |

*C# code:*

|  |
| --- |
| try  {  string id = tbxID.Text;  DataProvider.Instance.ExecuteNonQuery("EXEC deleteReceiptNote @id",  new object[] { id });  MessageBox.Show("Đã xóa đơn nhập hàng thành công.");  initComponent();  loadData();  }  catch (SqlException sqlException)  {  MessageBox.Show(sqlException.Message, "Báo lỗi!");  } |

**Trigger xóa hết Chi tiết đơn nhập hàng khi Xóa Đơn nhập hàng**

|  |
| --- |
| CREATE TRIGGER deleteReceiptNoteDetailTrigger  ON ReceiptNote  INSTEAD OF DELETE  AS  BEGIN  SET NOCOUNT ON;  DELETE FROM ReceiptNoteDetail  WHERE rNoteID IN (SELECT deleted.ID FROM deleted);  DELETE FROM ReceiptNote  WHERE ID IN (SELECT ID FROM deleted);  END;  GO |

1. **Procedure Thêm Chi tiết đơn nhập hàng**

|  |
| --- |
| CREATE PROC addReceiptNoteDetail  @rNoteID VARCHAR(7),  @materialID VARCHAR(3),  @quantity INT,  @unitPrice DECIMAL(10,2)  AS  BEGIN  IF @materialID IS NULL  THROW 50000, N'Vui lòng chọn nguyên liệu cần nhập.', 255  ELSE IF (@quantity <= 0 )  THROW 50000, N'Vui lòng nhập số lượng lớn hơn 0.', 255  ELSE IF (@unitPrice <= 0)  THROW 50000, N'Vui lòng nhập đơn giá lớn hơn 0.', 255  ELSE IF (SELECT COUNT(\*) FROM ReceiptNoteDetail WHERE rNoteID = @rNoteID AND materialID = @materialID) > 0  THROW 50000, N'Nguyên liệu được thêm đã có trong phiếu nhập. Nếu có sự thay đổi, vui lòng chọn SỬA!', 255  INSERT INTO ReceiptNoteDetail VALUES (@rNoteID, @materialID, @quantity, @unitPrice);  END;  GO |

*C# code:*

|  |
| --- |
| if (cbxMaterial.Text == "")  MessageBox.Show("Vui lòng chọn nguyên liệu cần nhập!");  else if (!int.TryParse(tbxQuantity.Text, out int result1))  MessageBox.Show("Vui lòng nhập số lượng là một số nguyên hợp lệ!");  else if (!double.TryParse(tbxUnitPrice.Text, out double result2))  MessageBox.Show("Vui lòng nhập đơn giá là một số hợp lệ!");  else  {  try  {  string rnID = dgvRN.Rows[dgvRN.SelectedCells[0].RowIndex].Cells[0].Value  .ToString(); //Lấy ID trong dgvRN  string mID =  (string)DataProvider.Instance.ExecuteQuery(  "SELECT ID FROM Material WHERE name = '" + cbxMaterial.Text + "'").Rows[0]["ID"];  decimal quantity = decimal.Parse(tbxQuantity.Text);  decimal unitPrice = decimal.Parse(tbxUnitPrice.Text);  DataProvider.Instance.ExecuteNonQuery(  "EXEC addReceiptNoteDetail @rnNoteID , @materialID , @quantity , @unitPrice",  new object[] { rnID, mID, quantity, unitPrice });  MessageBox.Show("Đã thêm chi tiết đơn nhập hàng mới thành công.");  dataRNDetail =  DataProvider.Instance.ExecuteQuery("SELECT \* from dbo.seeReceiptNoteDetail( @id )",  new object[] { rnID });  loadDataRNDetail();  decimal price =  (decimal)DataProvider.Instance.ExecuteScalar("SELECT dbo.calReceiptNotePrice( @id )",  new object[] { rnID });  lblPrice.Text = price.ToString() + " VND";  }  catch (SqlException sqlException)  {  MessageBox.Show(sqlException.Message, "Báo lỗi!");  }  } |

1. **Procedure Sửa Chi tiết đơn nhập hàng**

|  |
| --- |
| CREATE PROC updateReceiptNoteDetail  @rNoteID VARCHAR(7),  @materialID VARCHAR(3),  @quantity INT,  @unitPrice DECIMAL(10,2)  AS  BEGIN  IF @materialID IS NULL  THROW 50000, N'Vui lòng chọn thông tin chi tiết cần sửa.', 255  ELSE IF (@quantity <= 0 )  THROW 50000, N'Vui lòng nhập số lượng lớn hơn 0.', 255  ELSE IF (@unitPrice <= 0)  THROW 50000, N'Vui lòng nhập đơn giá lớn hơn 0.', 255  UPDATE ReceiptNoteDetail  SET quantity = @quantity, unitPrice = @unitPrice  WHERE rNoteID = @rNoteID AND materialID = @materialID;  END;  GO |

*C# code:*

|  |
| --- |
| if (cbxMaterial.Text == "")  {  MessageBox.Show("Vui lòng chọn thông tin chi tiết đơn nhập hàng cần sửa!");  }  else if (!int.TryParse(tbxQuantity.Text, out int result1))  MessageBox.Show("Vui lòng nhập số lượng lớn hơn 0!");  else if (!double.TryParse(tbxUnitPrice.Text, out double result2))  MessageBox.Show("Vui lòng nhập đơn giá lớn hơn 0!");  else  {  try  {  string rnID = dgvRN.Rows[dgvRN.SelectedCells[0].RowIndex].Cells[0].Value  .ToString(); //Lấy ID trong dgvRN  string mID =  (string)DataProvider.Instance.ExecuteQuery(  "SELECT ID FROM Material WHERE name = '" + cbxMaterial.Text + "'").Rows[0]["ID"];  decimal quantity = decimal.Parse(tbxQuantity.Text);  decimal unitPrice = decimal.Parse(tbxUnitPrice.Text);  DataProvider.Instance.ExecuteNonQuery(  "EXEC updateReceiptNoteDetail @rnNoteID , @materialID , @quantity , @unitPrice",  new object[] { rnID, mID, quantity, unitPrice });  MessageBox.Show("Đã sửa chi tiết đơn nhập hàng thành công.");  dataRNDetail =  DataProvider.Instance.ExecuteQuery("SELECT \* from dbo.seeReceiptNoteDetail( @id )",  new object[] { rnID });  loadDataRNDetail();  decimal price =  (decimal)DataProvider.Instance.ExecuteScalar("SELECT dbo.calReceiptNotePrice( @id )",  new object[] { rnID });  lblPrice.Text = price.ToString() + " VND";  }  catch (SqlException sqlException)  {  MessageBox.Show(sqlException.Message, "Báo lỗi!");  }  } |

1. **Procedure Xóa Chi tiết đơn nhập hàng**

|  |
| --- |
| CREATE PROC deleteReceiptNoteDetail  @rNoteID VARCHAR(7),  @materialID VARCHAR(3)  AS  BEGIN  IF @materialID IS NULL  THROW 50000, N'Vui lòng chọn thông tin chi tiết cần xóa.', 255  DELETE ReceiptNoteDetail WHERE rNoteID = @rNoteID AND materialID = @materialID;  END;  GO |

*C# code:*

|  |
| --- |
| if (cbxMaterial.Text == "")  {  MessageBox.Show("Vui lòng chọn thông tin chi tiết đơn nhập hàng cần xóa!");  }  else  {  try  {  string rnID = dgvRN.Rows[dgvRN.SelectedCells[0].RowIndex].Cells[0].Value  .ToString(); //Lấy ID trong dgvRN  string mID =  (string)DataProvider.Instance.ExecuteQuery(  "SELECT ID FROM Material WHERE name = '" + cbxMaterial.Text + "'").Rows[0]["ID"];  DataProvider.Instance.ExecuteNonQuery("EXEC deleteReceiptNoteDetail @rnNoteID , @materialID",  new object[] { rnID, mID });  MessageBox.Show("Đã xóa chi tiết đơn nhập hàng thành công.");  dataRNDetail =  DataProvider.Instance.ExecuteQuery("SELECT \* from dbo.seeReceiptNoteDetail( @id )",  new object[] { rnID });  loadDataRNDetail();  decimal price =  (decimal)DataProvider.Instance.ExecuteScalar("SELECT dbo.calReceiptNotePrice( @id )",  new object[] { rnID });  lblPrice.Text = price.ToString() + " VND";  }  catch (SqlException sqlException)  {  MessageBox.Show(sqlException.Message, "Báo lỗi!");  }  } |

**TRIGGER sau khi thêm, sửa, xóa Chi tiết đơn nhập hàng thì cập nhật lại giá trị cho đơn**

|  |
| --- |
| CREATE TRIGGER updateRNPriceTrigger  ON ReceiptNoteDetail  AFTER INSERT, UPDATE, DELETE  AS  BEGIN  DECLARE @rnID VARCHAR(7)  IF EXISTS (SELECT \* FROM inserted)  BEGIN  SELECT @rnID = rNoteID FROM inserted;  END  ELSE  BEGIN  SELECT @rnID = rNoteID FROM deleted;  END;  UPDATE ReceiptNote  SET price = (SELECT SUM(unitPrice \* quantity) FROM ReceiptNoteDetail WHERE rNoteID = @rnID)  WHERE ID = @rnID;  END;  GO |

* + 1. **Function**

1. **Function lọc ReceiptNoteView theo ID**

|  |
| --- |
| CREATE FUNCTION filterReceiptNoteViewByID  (  @id VARCHAR(7)  )  RETURNS TABLE  AS  RETURN  SELECT \* FROM ReceiptNoteView WHERE ID = @id;  GO |

*C# code:*

|  |
| --- |
| DataTable data = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.filterReceiptNoteViewByID( @id )", new object[] { id }); |

1. **Function lọc ReceiptNoteView theo nhà cung cấp**

|  |
| --- |
| CREATE FUNCTION filterReceiptNoteViewBySupplier  (  @supName NVARCHAR(MAX)  )  RETURNS TABLE  AS  RETURN  SELECT \* FROM ReceiptNoteView WHERE [Supplier Name] = @supName;  GO |

*C# code:*

|  |
| --- |
| DataTable data = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.filterReceiptNoteViewBySupplier( @supName )", new object[] { supName }); |

1. **Function lọc ReceiptNoteView theo Nhân viên**

|  |
| --- |
| CREATE FUNCTION filterReceiptNoteViewByEmployee  (  @empName NVARCHAR(MAX)  )  RETURNS TABLE  AS  RETURN  SELECT \* FROM ReceiptNoteView WHERE [Employee Name] = @empName;  GO |

*C# code:*

|  |
| --- |
| DataTable data = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.filterReceiptNoteViewByEmployee( @empName )", new object[] { empName }); |

1. **Function lọc ReceiptNoteView theo Ngày nhập**

|  |
| --- |
| CREATE FUNCTION filterReceiptNoteViewByDate  (  @date DATE  )  RETURNS TABLE  AS  RETURN  SELECT \* FROM ReceiptNoteView WHERE date = @date;  GO |

*C# code:*

|  |
| --- |
| DateTime date = dtpSearch.Value;  DataTable data = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.filterReceiptNoteViewByDate( @date )", new object[] { date }); |

1. **Function lọc ReceiptNoteListView theo ID**

|  |
| --- |
| CREATE FUNCTION filterReceiptNoteListViewByID  (  @id VARCHAR(7)  )  RETURNS TABLE  AS  RETURN  SELECT \* FROM ReceiptNoteListView WHERE ID = @id;  GO |

*C# code:*

|  |
| --- |
| DataTable dataRN = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.filterReceiptNoteListViewByID( @id )", new object[] { id }); |

1. **Function lọc ReceiptNoteListView theo Nhà cung cấp**

|  |
| --- |
| CREATE FUNCTION filterReceiptNoteListViewBySupplier  (  @supName VARCHAR(MAX)  )  RETURNS TABLE  AS  RETURN  SELECT \* FROM ReceiptNoteListView WHERE [Supplier Name] = @supName;  GO |

*C# code:*

|  |
| --- |
| DataTable dataRN = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.filterReceiptNoteListViewByDate( @supName )", new object[] { supName }); |

1. **Function lọc ReceiptNoteListView theo Nhân viên**

|  |
| --- |
| CREATE FUNCTION filterReceiptNoteListViewByEmployee  (  @empName VARCHAR(MAX)  )  RETURNS TABLE  AS  RETURN  SELECT \* FROM ReceiptNoteListView WHERE [Employee Name] = @empName;  GO |

*C# code:*

|  |
| --- |
| DataTable dataRN = DataProvider.Instance.ExecuteQuery("SELECT \* FROM dbo.filterReceiptNoteListViewByDate( @empName )", new object[] { empName }); |

1. **Function lọc ReceiptNoteListView theo Ngày nhập**

|  |
| --- |
| CREATE FUNCTION filterReceiptNoteListViewByDate  (  @date DATE  )  RETURNS TABLE  AS  RETURN  SELECT \* FROM ReceiptNoteListView WHERE date = @date;  GO |

1. **Function trả về bảng Chi tiết đơn nhập hàng khi click vào Đơn nhập hàng**

|  |
| --- |
| CREATE FUNCTION seeReceiptNoteDetail  (  @id VARCHAR(7)  )  RETURNS TABLE  AS  RETURN  (  SELECT m.name, rnd.quantity, rnd.unitPrice, rnd.quantity\*rnd.unitPrice as price FROM ReceiptNote rn, ReceiptNoteDetail rnd, Material m  WHERE rn.ID = rnd.rNoteID AND rnd.materialID = m.ID AND rn.ID = @id  );  GO |

*C# code:*

|  |
| --- |
| DataTable dataRNDetail = DataProvider.Instance.ExecuteQuery("SELECT \* from dbo.seeReceiptNoteDetail( @id )", new object[] { id }); |

1. **Function trả về tổng giá trị Đơn nhập hàng dựa trên Chi tiết đơn**

|  |
| --- |
| CREATE FUNCTION calReceiptNotePrice  (  @id VARCHAR(7)  )  RETURNS DECIMAL(18,2)  AS  BEGIN  IF ((SELECT COUNT(materialID) FROM ReceiptNoteDetail WHERE rNoteID = @id) = 0)  RETURN 0;  DECLARE @result DECIMAL(18,2);  SELECT @result = SUM(quantity \* unitPrice)  FROM dbo.seeReceiptNoteDetail(@id);  RETURN @result;  END;  GO |

*C# code:*

|  |
| --- |
| decimal price = (decimal)DataProvider.Instance.ExecuteScalar("SELECT dbo.calReceiptNotePrice( @id )", new object[] { rnID }); |

1. **Tạo user và phân quyền truy cập**

SQL:

------------- CREATE USER ADMIN WITH ROLE DB\_OWNER ----------------

USE master

GO

CREATE LOGIN admin

WITH PASSWORD = 'admin',

CHECK\_POLICY = OFF, CHECK\_EXPIRATION = OFF,

DEFAULT\_DATABASE = WibuCoffee;

-- Create login for the user

CREATE LOGIN emp

WITH PASSWORD = 'emp',

CHECK\_POLICY = OFF, CHECK\_EXPIRATION = OFF,

DEFAULT\_DATABASE = WibuCoffee;

GO

USE WibuCoffee

GO

CREATE USER admin

FOR LOGIN admin

GO

-- Granting db\_owner role to admin within WibuCoffee database

ALTER ROLE db\_owner ADD MEMBER admin;

GO

---------- CREATE USER EMP WITH LIMITED ROLE ---------------

USE WibuCoffee

GO

-- Create user from the login

CREATE USER emp

FOR LOGIN emp;

GO

CREATE ROLE role\_emp

GO

ALTER ROLE role\_emp ADD MEMBER emp

GO

GRANT EXECUTE TO role\_emp;

GO

GRANT SELECT TO role\_emp;

GO

-- Granting SELECT, INSERT, UPDATE on Product\_Material

GRANT SELECT, INSERT, UPDATE ON Product\_Material TO role\_emp;

GO

GRANT SELECT, INSERT, UPDATE ON Bill TO role\_emp;

GO

-- Granting SELECT, INSERT, UPDATE, DELETE on BillInfo

GRANT SELECT, INSERT, UPDATE, DELETE ON BillInfo TO role\_emp;

GO

GRANT EXECUTE ON dbo.checkLogin TO role\_emp;

GO

-- Explicitly deny DELETE on all other tables

DENY DELETE ON DATABASE::WibuCoffee TO role\_emp;

GO

-- Explicitly deny any access to specific tables

DENY SELECT, INSERT, UPDATE, DELETE ON Account TO role\_emp;

DENY SELECT, INSERT, UPDATE, DELETE ON Employee TO role\_emp;

DENY SELECT, INSERT, UPDATE, DELETE ON Shift TO role\_emp;

DENY SELECT, INSERT, UPDATE, DELETE ON On\_Duty TO role\_emp;

DENY SELECT, INSERT, UPDATE, DELETE ON Product TO role\_emp;

DENY SELECT, INSERT, UPDATE, DELETE ON Supplier TO role\_emp;

DENY SELECT, INSERT, UPDATE, DELETE ON ReceiptNote TO role\_emp;

DENY SELECT, INSERT, UPDATE, DELETE ON ReceiptNoteDetail TO role\_emp;

GO

--------- FOR ORDER --------------

REVOKE CONTROL ON dbo.BillCategory FROM role\_emp;

GO

GRANT SELECT ON dbo.BillCategory TO role\_emp;

GO

REVOKE CONTROL ON dbo.insertBill FROM role\_emp;

GO

GRANT EXECUTE ON dbo.insertBill TO role\_emp;

GO

GRANT SELECT ON dbo.TableStatusView TO role\_emp;

GO

DENY EXECUTE ON dbo.insertTable TO role\_emp;

DENY EXECUTE ON dbo.deleteTableByID TO role\_emp;

DENY EXECUTE ON dbo.deleteBillByID TO role\_emp;

GO

------- FOR EMPLOYEE -----------

DENY SELECT ON dbo.EmployeeDetailView TO role\_emp;

DENY SELECT ON dbo.ShiftDetailView to role\_emp;

DENY EXECUTE ON dbo.getJobDetail TO role\_emp;

DENY SELECT ON dbo.getEmployeeInfo TO role\_emp;

DENY SELECT ON dbo.filterEmployeeByName TO role\_emp;

DENY SELECT ON dbo.filterEmployeeByDate TO role\_emp;

DENY EXECUTE ON dbo.getSalary TO role\_emp;

DENY SELECT ON dbo.getEmployeeID TO role\_emp;

DENY SELECT ON dbo.getEmployeeInfoByName TO role\_emp;

DENY SELECT ON dbo.getEmployeeInfoByJob TO role\_emp;

DENY SELECT ON dbo.getEmployeeNameByRecruitmentDate TO role\_emp;

DENY EXECUTE ON dbo.createID TO role\_emp;

GO

DENY EXECUTE ON dbo.getEmployee TO role\_emp;

DENY EXECUTE ON dbo.getJob TO role\_emp;

DENY EXECUTE ON dbo.getShift TO role\_emp;

DENY EXECUTE ON dbo.getEmployeeList TO role\_emp;

DENY EXECUTE ON dbo.updateSalary TO role\_emp;

DENY EXECUTE ON dbo.insertEmployee TO role\_emp;

DENY EXECUTE ON dbo.updateEmployee TO role\_emp;

DENY EXECUTE ON dbo.deleteEmployeeByID TO role\_emp;

DENY EXECUTE ON dbo.insertEmployeeToShift TO role\_emp;

DENY EXECUTE ON dbo.updateEmployeeInShift TO role\_emp;

DENY EXECUTE ON dbo.deleteEmployeeInShift TO role\_emp;

DENY EXECUTE ON dbo.getShiftDetail TO role\_emp;

DENY EXECUTE ON dbo.getShiftDetailByWeek TO role\_emp;

DENY EXECUTE ON dbo.getNextWeekShiftDetail TO role\_emp;

GO

------- FOR PRODUCT -----------

DENY SELECT ON dbo.ProductView TO role\_emp;

DENY SELECT ON dbo.ProductCategoryView TO role\_emp;

GO

DENY SELECT ON dbo.GetMaterials TO role\_emp;

GO

DENY EXECUTE ON dbo.AddProduct TO role\_emp;

DENY EXECUTE ON dbo.UpdateProductByID TO role\_emp;

DENY EXECUTE ON dbo.DeleteProductByID TO role\_emp;

DENY EXECUTE ON dbo.addProductCategory TO role\_emp;

DENY EXECUTE ON dbo.updateProductCategoryByID TO role\_emp;

DENY EXECUTE ON dbo.addProductMaterial TO role\_emp;

DENY EXECUTE ON dbo.updateProductMaterialByID TO role\_emp;

DENY EXECUTE ON dbo.deleteProductMaterialByID TO role\_emp;

GO

------- FOR SUPPLIER -----------

DENY SELECT ON dbo.SupplierList TO role\_emp

GO

DENY EXECUTE ON dbo.AddSupplier TO role\_emp;

DENY EXECUTE ON dbo.UpdateSupplier TO role\_emp;

DENY EXECUTE ON dbo.DeleteSupplier TO role\_emp;

GO

------- FOR RECEIPT NOTE -----------

DENY SELECT ON dbo.ReceiptNoteView TO role\_emp;

DENY SELECT ON dbo.ReceiptNoteListView TO role\_emp;

GO

DENY SELECT ON dbo.filterReceiptNoteViewByID TO role\_emp;

DENY SELECT ON dbo.filterReceiptNoteViewBySupplier TO role\_emp;

DENY SELECT ON dbo.filterReceiptNoteViewByEmployee TO role\_emp;

DENY SELECT ON dbo.filterReceiptNoteViewByDate TO role\_emp;

DENY SELECT ON dbo.filterReceiptNoteListViewByID TO role\_emp;

DENY SELECT ON dbo.filterReceiptNoteListViewBySupplier TO role\_emp;

DENY SELECT ON dbo.filterReceiptNoteListViewByEmployee TO role\_emp;

DENY SELECT ON dbo.filterReceiptNoteListViewByDate TO role\_emp;

DENY SELECT ON dbo.seeReceiptNoteDetail TO role\_emp;

DENY EXECUTE ON dbo.calReceiptNotePrice TO role\_emp;

GO

DENY EXECUTE ON dbo.addNewReceiptNote TO role\_emp;

DENY EXECUTE ON dbo.updateNewReceiptNote TO role\_emp;

DENY EXECUTE ON dbo.deleteReceiptNote TO role\_emp;

DENY EXECUTE ON dbo.deleteReceiptNoteDetail TO role\_emp;

DENY EXECUTE ON dbo.addReceiptNoteDetail TO role\_emp;

DENY EXECUTE ON dbo.updateReceiptNoteDetail TO role\_emp;

GO

**Procedure tạo user:**

CREATE PROCEDURE AddAccount

@userName VARCHAR(10),

@pass VARCHAR(MAX),

@userRole INT

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

-- Validate user role

IF @userRole < 0 OR @userRole > 1

RAISERROR('Invalid user role. Must be 0 or 1.', 16, 1);

-- Insert the new account

INSERT INTO Account (userName , pass , userRole)

VALUES (@userName , @pass , @userRole);

END TRY

BEGIN CATCH

-- Error handling

DECLARE @ErrorMessage NVARCHAR(4000) = ERROR\_MESSAGE();

DECLARE @ErrorSeverity INT = ERROR\_SEVERITY();

DECLARE @ErrorState INT = ERROR\_STATE();

RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH

END;

GO

**Trigger tạo tài khoản:**

CREATE TRIGGER [dbo].[Trigger\_CreateSQLAccount]

ON [dbo].[Account]

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

BEGIN TRANSACTION; -- Start transaction

DECLARE @userName NVARCHAR(10), @pass NVARCHAR(MAX), @userRole INT;

SELECT @userName = i.userName, @pass = '' + i.pass, @userRole = i.userRole

FROM inserted i;

DECLARE @sqlString NVARCHAR(2000);

-- Create SQL Server login for the new account

SET @sqlString = 'CREATE LOGIN [' + @userName + '] WITH PASSWORD = ''' + @pass +

''', DEFAULT\_DATABASE=[WibuCoffee], CHECK\_EXPIRATION=OFF, CHECK\_POLICY=OFF';

--print @sqlString

EXEC (@sqlString);

-- Create user for the login

SET @sqlString = 'CREATE USER [' + @userName + '] FOR LOGIN [' + @userName + ']';

EXEC (@sqlString);

-- Assign role based on userRole

IF @userRole = 0

SET @sqlString = 'ALTER ROLE db\_owner ADD MEMBER [' + @userName + ']';

ELSE IF @userRole = 1

SET @sqlString = 'ALTER ROLE role\_emp ADD MEMBER [' + @userName + ']';

EXEC (@sqlString);

COMMIT TRANSACTION; -- Commit the transaction if no errors

END TRY

BEGIN CATCH

IF @@TRANCOUNT > 0

ROLLBACK TRANSACTION; -- Roll back the transaction on error

-- Capture and raise the error information

DECLARE @ErrorMessage NVARCHAR(4000) = ERROR\_MESSAGE();

DECLARE @ErrorSeverity INT = ERROR\_SEVERITY();

DECLARE @ErrorState INT = ERROR\_STATE();

RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH

END;

GO

C# code:

* **Try/ catch 1 lệnh Select từ bảng cần truy cập, nếu SQL trả về mã lỗi 229, có nghĩa là user đã bị chặn (deny) khỏi bảng đó**

try

{

DataProvider.Instance.ExecuteNonQuery("SELECT \* FROM dbo.Account");

}

catch (SqlException err)

{

if (err.Number == 229)

MessageBox.Show("Lỗi phân quyền!\n" + err.Message);

return;

}

# **CHƯƠNG 4: THIẾT KẾ GIAO DIỆN HỆ THỐNG**

**Môi trường cài đặt và công nghệ sử dụng**

- Hệ cơ sở dữ liệu Microsoft SQL Server Management Studio 19.

- Chương trình Winform được viết bằng Microsoft Visual Studio 2022.

**Giao diện và chức năng của chương trình:**

1. **Form đăng nhập**

**A screenshot of a login screen

Description automatically generated**

**-** Đối với việc đăng nhập lần đầu, chương trình sẽ yêu cầu nhập tên server (nếu không nhớ rõ, vui lòng copy trực tiếp từ SSMS) cùng với user name và password.

- Từ lần thứ 2 trở đi, chương trình tự lưu server name và người dùng chỉ cần nhập user name và password.

1. **Form chính**

**A screenshot of a computer

Description automatically generated**

- 4 ô ở trên cùng lần lượt thể hiện doanh thu hôm nay, chi tiêu hôm nay, lợi nhuận của tháng hiện tại cùng với số bàn còn trống.

- Ở dưới chia thành 2 bên, bên trái thể hiện 5 món bán được nhiều nhất tất cả hóa đơn từ trước đến nay, bên phải biểu đồ thống kê tỉ lệ thu – chi trong 7 ngày gần nhất.

- Cuối cùng là biểu đồ thống kê thu nhập trong 12 tháng.

1. **Giao diện order** (cả quyền admin và nhân viên đều thao tác được, riêng nhân viên không thể xóa hóa đơn, thêm/xóa bàn)

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**\*Lưu ý: Phải đảm bảo có nhân viên trực tại thời điểm đó, nếu không có vui lòng qua tab Employee để thêm ca trực cho nhân viên, chi tiết xem tại mục d. Ban đầu quán không bán 24/7, nhưng vì lý do đang trong thời gian test, chúng em đã thêm 2 ca trực từ 22h tới 23h59 ngày hiện tại và từ 0h tới 7h ngày hôm sau.**

**-** Muốn thêm hóa đơn, ta lần lượt thêm loại hóa đơn (nếu là tại chỗ “In place” thì phải chọn bàn ở bên phải giao diện, không thì không chọn bàn), số điện thoại và bấm tìm để xác định tên khách hàng (Nếu khách hàng mới mua lần đầu “CUSTOMER DOES NOT EXISTS” thì tạo khách hàng mới).

- Sau đó ta mới có thể thêm sản phẩm cũng như số lượng cho hóa đơn được.

- Cuối cùng, để hoàn thành hóa đơn, ta thêm số tiền nhận từ khách vào ô “Tiền nhận” và bấm Cập nhật.

- Để tạo hóa đơn mới, vui lòng bấm “Làm mới”.

1. **Bảng Employee** ( chỉ quyền admin mới có quyền truy cập và thao tác)

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

- Ở bảng Employee ta có thể xem tên nhân viên và kiểm soát các ca trực.

- Ta thao tác việc thêm, sửa xóa ca trực ở bên trái và theo dõi kết quả ở bên phải.

1. **Bảng Product** ( chỉ quyền admin mới có quyền truy cập và thao tác)

A screenshot of a computer

Description automatically generated

- Tương tự, ở bảng Product cho phép ta thêm, sửa, xóa, theo dõi các sản phẩm. Xác định loại sản phẩm cũng như nguyên liệu làm ra chúng.

1. **Bảng Material** (cả quyền admin và nhân viên đều được thao tác)

**A screenshot of a computer

Description automatically generated**

- Ở bảng Material có 2 thao tác, thêm hoặc sửa nguyên liệu.

1. **Bảng Receipt Note** ( chỉ quyền admin mới có quyền truy cập và thao tác)

- Bảng này cho biết danh sách các đơn nhập hàng, theo dõi chi tiết đơn nhập hàng cũng như thêm sửa xóa chúng. **A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**

1. **Bảng Supplier** ( chỉ quyền admin mới có quyền truy cập và thao tác)

- Bảng này giúp người dùng theo dõi các nhà cung cấp đồng thời thêm sửa xóa chúng.

**A screenshot of a computer

Description automatically generated**

1. **Bảng expense Bill** (cả quyền quản lí và nhân viên đều được thao tác)

- Bảng này cho biết danh sách các hoạt động như bảo dưỡng thiết bị, lau dọn,… theo dõi chi tiết đơn nhập hàng cũng như thêm sửa xóa chúng.

**A screenshot of a computer

Description automatically generated**

1. **Tạo user:** (chỉ quyền admin mới có thể truy cập và thao tác)

**A screenshot of a computer login screen

Description automatically generated**

* Có 2 quyền :

+ Số 0: admin , tức quyền quản lý, chủ quán

+ Số 1: emp, tức quyền nhân viên