

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**  
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**  
**Khoa Hệ Thống Thông Tin**



**ĐỒ ÁN MÔN**  
**HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU**

**ĐỀ TÀI: ỨNG DỤNG QUẢN LÝ KHO CUNG CẤP VẬT**  
**LIỆU XÂY DỰNG**

**Giảng viên hướng dẫn:**

**ThS. Đỗ Thị Minh Phụng**

**Sinh viên thực hiện - Nhóm 1:**

**Nguyễn Ngọc Công - 15520069**

**Nguyễn Xuân Hội – 15520268**

**Đặng Xuân Phóng - 15520621**

**TP. Hồ Chí Minh, Tháng 07 năm 2020**

# MỤC LỤC

<b>LỜI CẢM ƠN .....</b>	<b>5</b>
<b>NHẬN XÉT CỦA GIẢNG VIÊN .....</b>	<b>6</b>
<b>CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI.....</b>	<b>7</b>
1.1.Đặt vấn đề .....	7
1.2.Mục tiêu .....	8
1.3.Công cụ sử dụng .....	8
<b>CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ CƠ SỞ DỮ LIỆU .....</b>	<b>8</b>
2.1.Phân tích nghiệp vụ:.....	8
1.1.1. Những nghiệp vụ chung .....	8
1.1.2. Nghiệp vụ riêng của người quản lý .....	9
2.2.Phân tích yêu cầu .....	9
1.1.3. Yêu cầu chức năng.....	9
1.1.4. Yêu cầu phi chức năng.....	10
1.1.5. Thiết kế mô hình quan hệ.....	11
<b>CHƯƠNG 3. CÔNG NGHỆ SỬ DỤNG .....</b>	<b>18</b>
3.1.Oracle.....	18
3.1.1. Giới thiệu Oracle.....	18
3.1.2. Các phiên bản.....	18
3.1.3. Oracle 12c .....	19
3.2.Hibernate.....	19
3.2.1. ORM framework .....	19
3.2.2. Hibernate framework .....	20
3.3.JPA.....	20
3.4.Java Spring.....	21
3.5.Github .....	21
3.5.1. Git .....	21
3.5.2. Github .....	22

3.6.IntelliJ IDEA.....	22
<b>CHƯƠNG 4. XÂY DỰNG VÀ QUẢN LÝ GIAO TÁC .....</b>	<b>23</b>
4.1.Trigger .....	23
4.1.1. Trigger trong Oracle.....	23
4.1.2. Danh sách các Trigger.....	24
4.1.3. Mô tả một số Trigger .....	25
4.2.Store Procedure.....	28
4.2.1. Store procedure trong Oracle .....	28
4.2.2. Transaction trong Oracle.....	29
4.2.3. Danh sách Store Procedure .....	30
4.2.4. Mô tả một số Store Procedure .....	31
<b>CHƯƠNG 5. XỬ LÝ TRUY XUẤT ĐỒNG THỜI .....</b>	<b>37</b>
5.1.Các mức cô lập trong Oracle.....	37
5.1.1. Read committed .....	37
5.1.2. Serializable.....	37
5.1.3. Read-only .....	39
5.2.Cơ chế khóa .....	39
5.3.Deadlock .....	42
5.4.Mô tả trong đồ án môn học .....	43
5.4.1. Non-repeatable read .....	43
5.4.2. Phantom read.....	50
5.4.3. Lost update.....	53
<b>CHƯƠNG 6. THIẾT KẾ GIAO DIỆN .....</b>	<b>60</b>
6.1.Danh sách các màn hình .....	60
6.1.1. Đăng nhập .....	60
6.1.2. Menu bảng điều khiển.....	60
6.1.3. Category .....	60
6.1.4. Sản phẩm.....	60
6.1.5. Nhập kho.....	61

6.1.6. Xuất kho.....	61
6.1.7. Sản phẩm trong kho .....	61
6.1.8. Lịch sử kho.....	61
6.1.9. User.....	61
6.1.10. Menu .....	62
6.1.11. Quyền.....	62
6.2.Mô tả các màn hình.....	62
6.2.1. Đăng nhập .....	62
6.2.2. Menu bảng điều khiển.....	63
6.2.3. Category .....	65
6.2.4. Sản phẩm.....	68
6.2.5. Nhập kho .....	72
6.2.6. Xuất kho.....	74
6.2.7. Sản phẩm trong kho .....	76
6.2.8. Lịch sử kho.....	77
6.2.9. User .....	77
6.2.10. Menu .....	80
6.2.11. Quyền.....	81
<b>CHƯƠNG 7. KẾT LUẬN .....</b>	<b>82</b>
7.1.Kết quả đạt được .....	82
7.1.1. Kiến thức.....	82
7.1.2. Làm việc nhóm.....	82
7.1.3. Ứng dụng Quản lý kho.....	82
7.2.Hạn chế .....	82
7.3.Hướng phát triển .....	83
<b>Phụ lục.....</b>	<b>84</b>
Phụ lục 1: Bảng phân chia công việc .....	84
Phụ lục 2: Tài liệu tham khảo .....	84

## **LỜI CẢM ƠN**

Đầu tiên, em xin gửi lời cảm ơn chân thành đến quý Thầy cô giảng viên Trường Đại học Công nghệ thông tin – Đại học Quốc gia TP.HCM và quý thầy cô khoa Hệ thống Thông tin đã giúp cho em có những kiến thức cơ bản làm nền tảng để thực hiện đồ án này.

Đặc biệt, em xin gửi lời cảm ơn và lòng biết ơn sâu sắc nhất tới cô – ThS. Đỗ Thị Minh Phụng, người đã hướng dẫn chúng em trong suốt thời gian làm đồ án. Cô đã trực tiếp hướng dẫn tận tình, sửa chữa và đóng góp nhiều ý kiến quý báu giúp nhóm chúng em hoàn thành tốt báo cáo môn học của mình. Một lần nữa em chân thành cảm ơn cô và chúc cô dồi dào sức khỏe.

Trong thời gian một học kỳ thực hiện đồ án, chúng em đã vận dụng những kiến thức nền tảng đã tích lũy đồng thời kết hợp với việc học hỏi và nghiên cứu những kiến thức mới từ thầy cô, bạn bè cũng như nhiều nguồn tài liệu tham khảo. Từ đó, nhóm chúng em đã vận dụng tối đa những gì đã thu thập được để hoàn thành một báo cáo đồ án tốt nhất. Tuy nhiên, vì kiến thức chuyên môn còn hạn chế và bản thân còn thiếu nhiều kinh nghiệm thực tiễn nên nội dung của báo cáo không tránh khỏi những thiếu sót, em rất mong nhận được sự góp ý, chỉ bảo thêm của quý thầy cô nhằm hoàn thiện những kiến thức của mình để em có thể dùng làm hành trang thực hiện tiếp các đề tài khác trong tương lai cũng như là trong việc học tập và làm việc sau này.

Một lần nữa xin gửi đến thầy cô, bạn bè lời cảm ơn chân thành và tốt đẹp nhất.

Thành phố Hồ Chí Minh, tháng 07 năm 2020

Nhóm sinh viên thực hiện

Nhóm 1

[illegible]

## CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI

### 1.1. Đặt vấn đề

Ngày nay Công nghệ thông tin đã phát triển với tốc độ nhanh chóng. CNTT đã ứng dụng hầu như vào mọi hoạt động của con người từ nguyên cứu khoa học, phát triển kinh tế, quân sự, nghệ thuật, kinh doanh... Cùng với sự phát triển không ngừng về kỹ thuật máy tính và mạng điện tử, công nghệ thông tin cũng được những công nghệ có đẳng cấp cao và lần lượt chinh phục hết đỉnh cao này đến đỉnh cao khác. Mạng Internet là một trong những sản phẩm có giá trị hết sức lớn lao và ngày càng trở nên một công cụ không thể thiếu, là nền tảng chính cho sự truyền tải, trao đổi thông tin trên toàn cầu.

Giờ đây, mọi việc liên quan đến thông tin trở nên thật dễ dàng cho người sử dụng: chỉ cần có một máy tính kết nối Internet và một dòng dữ liệu truy tìm thì gần như lập tức... cả thế giới về vấn đề mà bạn đang quan tâm sẽ hiện ra, có đầy đủ thông tin, hình ảnh và thậm chí đôi lúc có cả những âm thanh nếu bạn cần...

Bằng Internet, chúng ta đã thực hiện được nhiều công việc với tốc độ nhanh hơn và chi phí thấp hơn nhiều so với cách thức truyền thống. Chính điều này, đã thúc đẩy sự khai sinh và phát triển của thương mại điện tử và chính phủ điện tử trên khắp thế giới, làm biến đổi đáng kể bộ mặt văn hóa, nâng cao chất lượng cuộc sống con người.

Trong hoạt động sản xuất, kinh doanh, giờ đây, việc công nghệ thông tin hóa quy trình kinh doanh đã khẳng định được vai trò xúc tiến và thúc đẩy sự phát triển của doanh nghiệp. Quản lý kho là một công việc rất vất vả cho người chủ shop và nhân viên bán hàng. Khi cửa hàng, doanh nghiệp ngày càng phát triển thì số lượng hàng hóa trong kho cũng ngày càng tăng cao. Quản lý kho thế nào để hàng hóa bán ra – nhập vào không bị nhầm lẫn, thất thoát là bài toán nan giải được các doanh nghiệp đặt ra. Nhận thấy việc quản lý kho bãi thủ công bằng giấy tờ hiện tại gặp nhiều bất cập và sai sót, gây ảnh hưởng lớn đến doanh nghiệp. Nhóm em quyết định lên ý tưởng thực hiện đề tài Quản trị cơ sở dữ liệu Quản lý kho cho doanh nghiệp. Với đề tài này, ứng dụng Quản lý kho có thể áp dụng cho những doanh nghiệp có hình thức kinh doanh có kho hàng. Đối với phạm vi đề tài, nhóm chúng em chọn triển khai ví dụ cho doanh nghiệp Kinh doanh Vật liệu Xây Dựng.

## 1.2. Mục tiêu

Dùng những kiến thức đã học ở trường, đặc biệt là kiến thức môn Hệ quản trị cơ sở dữ liệu để xây dựng một ứng dụng Quản lý kho. Ứng dụng được xây dựng đáp ứng đầy đủ các nhu cầu cơ bản cho nghiệp vụ quản lý kho hàng.

## 1.3. Công cụ sử dụng

Trong quá trình thực hiện, nhóm sử dụng một số công cụ sau:

- SQL Developer
- Oracle SQL Developer Data Modeler
- Draw.io
- IntelliJ IDEA
- Github

# CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ CƠ SỞ DỮ LIỆU

## 2.1. Phân tích nghiệp vụ:

Lĩnh vực kinh doanh: Kinh doanh vật liệu xây dựng

### 1.1.1. *Những nghiệp vụ chung*

- Nghiệp vụ quản lý sản phẩm: Mục đích quản lý các sản phẩm, nhà sản cung cấp, giá cả, doanh số bán hàng của từng loại sản phẩm.
- Nghiệp vụ quản lý tồn kho: Mục đích quản lý tồn kho sản phẩm, xem xét số lượng tồn, đề lên phương án giảm giá thúc đẩy bán sản phẩm tồn hoặc liên hệ nhà cung cấp để mua hàng mới.
- Nghiệp vụ nhập hàng: Mục đích quản lý nhập hàng, giá cả, nhà cung cấp, quản lý các chứng từ liên quan.
- Nghiệp vụ xuất hàng: Mục đích quản lý hàng hóa xuất đi, khách hàng, quản lý các chứng từ liên quan.



- Nghiệp vụ báo cáo, thống kê: Mục đích báo cáo/ thống kê tổng quát hiệu suất công việc bán hàng, kho bãi, lượng hàng tồn, để người quản lý có các chiến lược phương án bán hàng, nhập hàng phù hợp.

### **1.1.2. *Nghiệp vụ riêng của người quản lý***

- Nghiệp vụ quản lý nhân viên: Mục đích quản lý danh sách nhân viên đang làm việc, tính lương, tính hiệu xuất cũng như đánh giá nhân viên.

## **2.2. Phân tích yêu cầu**

### **1.1.3. *Yêu cầu chức năng***

Chức năng phải đáp ứng tốt những nhu cầu cơ bản của một doanh nghiệp kinh doanh có kho hàng. Người dùng bao gồm: Người quản lý và nhân viên, trong đó người quản lý có toàn quyền với mọi chức năng của ứng dụng, nhân viên bị hạn chế một số chức năng nhất định, từ đây đến cuối đề án, đề cập đến tác nhân người dùng có nghĩa là bao gồm cả nhân viên và người quản lý.

#### **- Chức năng hệ thống**

- a. Chức năng đăng nhập / đăng xuất: Người dùng sau khi đăng nhập, dựa vào cơ sở dữ liệu hệ thống sẽ xác định quyền của người dùng là quyền nhân viên hay quản lý. Từ đó danh sách menu trong bảng điều khiển sẽ hiển thị những menu thích hợp.

#### **- Những chức năng chung**

- a. Chức năng quản lý sản phẩm: Chức năng này quản lý thông tin về sản phẩm. Người dùng có thể xem danh sách sản phẩm, thực hiện các chức năng thêm, sửa, xóa, tìm kiếm sản phẩm.
- b. Chức năng quản lý loại sản phẩm: Chức năng này quản lý thông tin về loại sản phẩm. Người dùng có thể xem danh sách loại sản phẩm, thực hiện các chức năng thêm, sửa, xóa, tìm kiếm loại sản phẩm.
- c. Chức năng quản lý hàng tồn kho: Chức năng này quản lý số lượng hàng đang tồn trong kho. Người dùng có thể xem chi tiết số lượng còn tồn trong kho với mỗi mặt hàng.

- d. Chức năng quản lý nhập hàng: Chức năng này quản lý quá trình nhập hàng của người dùng. Ở chức năng này, hóa đơn ghi lại thông tin, chi tiết nhập hàng.
- e. Chức năng xuất hàng: Chức năng này quản lý quá trình xuất hàng của người dùng. Ở chức năng này, hóa đơn ghi lại thông tin, chi tiết xuất hàng.
- f. Chức năng báo cáo / thống kê: Chức năng hỗ trợ người dùng có thể xuất báo cáo, thống kê doanh thu về các mặt hàng riêng biệt, cũng như doanh số xuất / nhập theo thời gian.

- **Những chức năng riêng cho người quản lý**

- a. Chức năng quản lý nhân viên: Chức năng này quản lý thông tin về nhân viên. Người quản có thể xem danh sách loại sản phẩm, thực hiện các chức năng thêm, sửa, xóa, tìm kiếm nhân viên.
- b. Chức năng quản lý phân quyền: Chức năng này giúp người quản lý có thể phân quyền nhân viên của họ. Cho phép cấp và thu hồi quyền của mỗi nhân viên, từ đó nâng cao hiệu suất, gia tăng tính bảo mật và hiệu quả của hệ thống.

#### **1.1.4. Yêu cầu phi chức năng**

**Yêu cầu về giao diện:** giao diện ứng dụng gần gũi, thân thiện. Người dùng có thể sử dụng dễ dàng, thuận tiện và có tính tương tác cao. Thao tác trên ứng dụng nhanh gọn và dễ sử dụng.

**Yêu cầu về chất lượng:**

Tính tiến hóa: Dễ dàng trong việc nâng cấp hay thêm các modules tiện ích khác.

**Tính tiện dụng:**

- Ứng dụng có giao diện trực quan, thân thiện và dễ sử dụng.
- Các đặc tả và hướng dẫn sử dụng rõ ràng.
- Thao tác đơn giản và nhanh gọn, đáp ứng nhu cầu của học sinh.

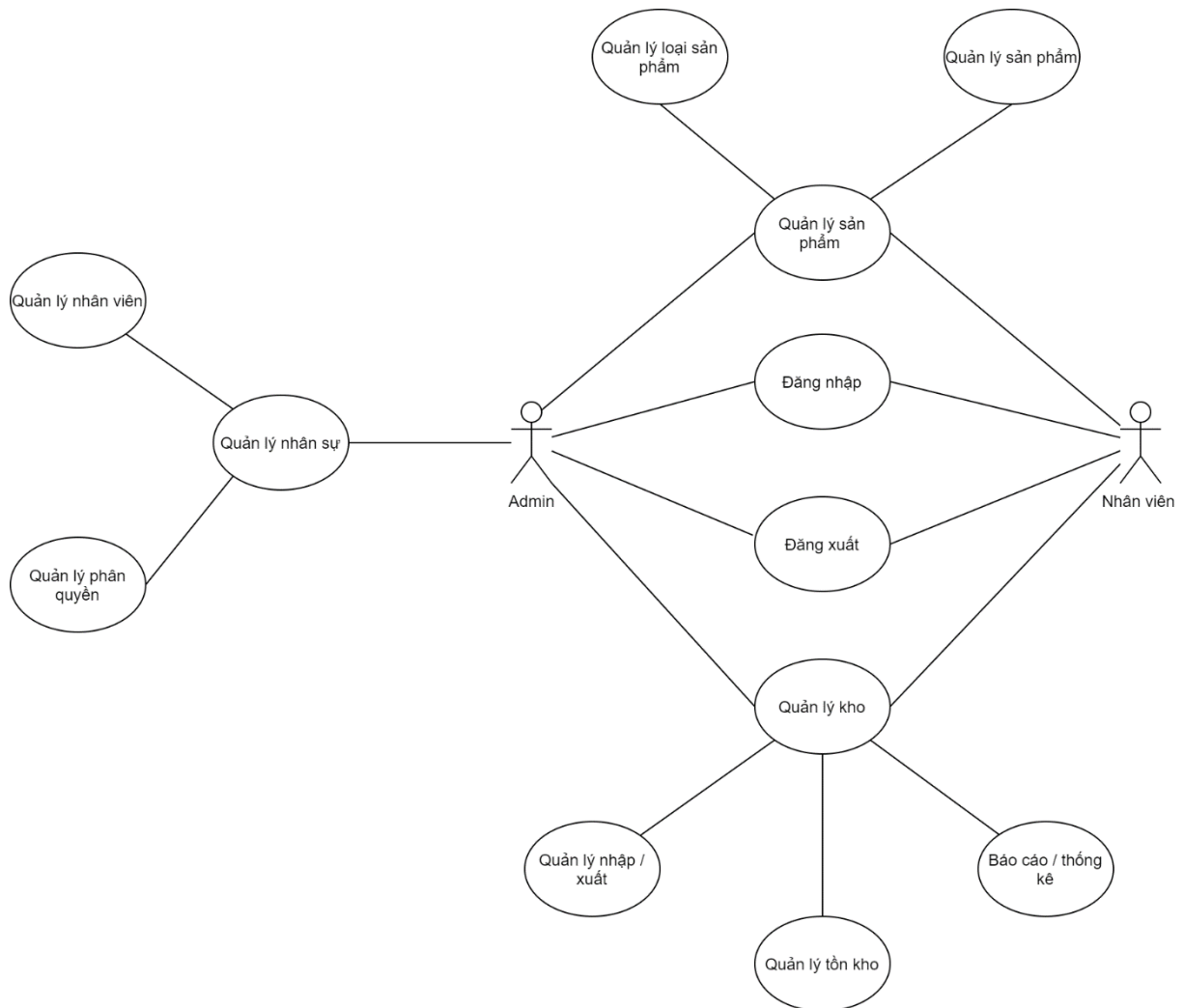
**Tính tương thích:** Hoạt động tốt trên mọi nền tảng, trên ứng dụng, mobile, tablet.

**Tính hiệu quả:**

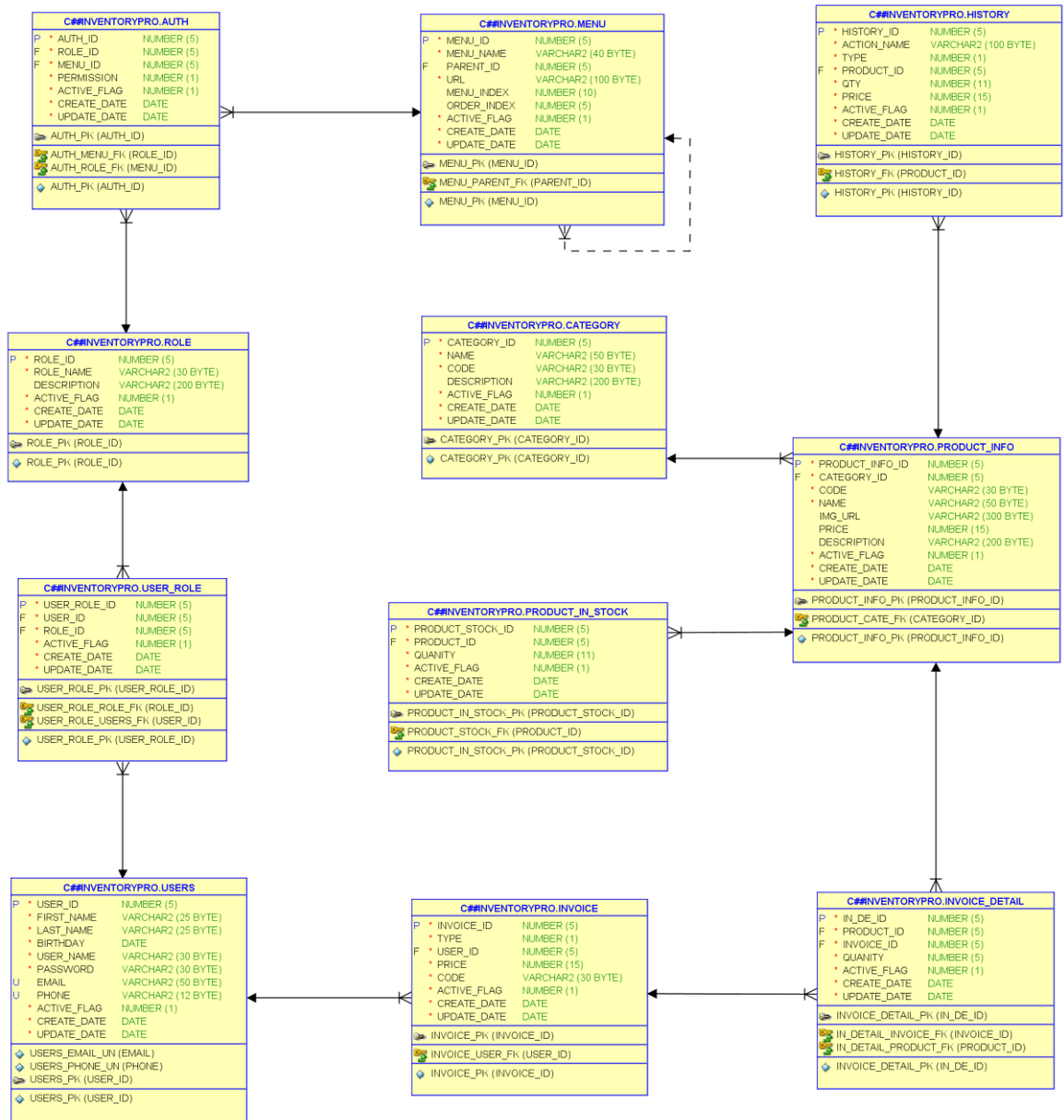
- Ứng dụng hoạt động tốt và đầy đủ những yêu cầu chức năng cơ bản của nghiệp vụ quản lý kho.
- Trang ứng dụng hoạt động ổn định, và đáng tin cậy, có thể truy cập và xử lý đồng thời nhiều hành động

### 1.1.5. Thiết kế mô hình quan hệ

#### a. Sơ đồ Use-case



#### b. Mô hình dữ liệu quan hệ



### c. Mô tả thành phần dữ liệu

Bảng AUTH:

STT	Thuộc tính	Kiểu dữ liệu	Diễn giải
1	AUTH_ID	NUMBER(5,0)	Mã số duy nhất của bảng AUTH
2	ROLE_ID	NUMBER(5,0)	Mã quyền
3	MENU_ID	NUMBER(5,0)	Mã menu
4	PERMISSION	NUMBER(1,0)	Quyền
5	ACTIVE_FLAG	NUMBER(1,0)	Trạng thái hoạt động: 0 – không hoạt động, 1 – hoạt động
6	CREATE_DATE	DATE	Ngày tạo
7	UPDATE_DATE	DATE	Ngày cập nhật

Bảng CATEGORY

STT	Thuộc tính	Kiểu dữ liệu	Diễn giải
1	CATEGORY_ID	NUMBER(5,0)	Mã số duy nhất của bảng CATEGORY
2	NAME	VARCHAR2(50 BYTE)	Tên thể loại
3	CODE	VARCHAR2(30 BYTE)	Mã loại
4	DESCRIPTION	VARCHAR2(200 BYTE)	Mô tả loại
5	ACTIVE_FLAG	NUMBER(1,0)	Trạng thái hoạt động: 0 – không hoạt động, 1 – hoạt động
6	CREATE_DATE	DATE	Ngày tạo
7	UPDATE_DATE	DATE	Ngày cập nhật

Bảng HISTORY

STT	Thuộc tính	Kiểu dữ liệu	Diễn giải
1	HISTORY_ID	NUMBER(5,0)	Mã số duy nhất của bảng HISTORY
2	ACTION_NAME	VARCHAR2(50 BYTE)	Hành động thao tác trên hóa đơn : Thêm/Xóa/Sửa
3	TYPE	NUMBER(1,0)	Loại hóa đơn: 0 – xuất, 1 – nhập
4	PRODUCT_ID	NUMBER(5,0)	Mã số sản phẩm
5	QTY	NUMBER(11,0)	Số lượng
6	PRICE	NUMBER(15,0)	Giá
7	ACTIVE_FLAG	NUMBER(1,0)	Trạng thái hoạt động: 0 – không hoạt động, 1 – hoạt động
8	CREATE_DATE	DATE	Ngày tạo
9	UPDATE_DATE	DATE	Ngày cập nhật

Bảng INVOICE

STT	Thuộc tính	Kiểu dữ liệu	Diễn giải
1	INVOICE_ID	NUMBER(5,0)	Mã số duy nhất của bảng INVOICE

2	TYPE	NUMBER(1,0)	Loại hóa đơn: 0 – xuất, 1 – nhập
3	USER_ID	NUMBER(5,0)	Mã số nhân viên thực hiện hóa đơn
4	PRICE	NUMBER(15,0)	Giá tổng của hóa đơn
5	CODE	VARCHAR2(30 BYTE)	Mã hóa đơn
6	ACTIVE_FLAG	NUMBER(1,0)	Trạng thái hoạt động: 0 – không hoạt động, 1 – hoạt động
7	CREATE_DATE	DATE	Ngày tạo
8	UPDATE_DATE	DATE	Ngày cập nhật

Bảng INVOICE\_DETAIL

STT	Thuộc tính	Kiểu dữ liệu	Diễn giải
1	IN_DE_ID	NUMBER(5,0)	Mã số duy nhất của bảng INVOICE_DETAIL
2	PRODUCT_ID	NUMBER(5,0)	Mã sản phẩm
3	INVOICE_ID	NUMBER(5,0)	Mã hóa đơn
4	QUANTITY	NUMBER(5,0)	Số lượng
5	ACTIVE_FLAG	NUMBER(1,0)	Trạng thái hoạt động: 0 – không hoạt động, 1 – hoạt động
6	CREATE_DATE	DATE	Ngày tạo
7	UPDATE_DATE	DATE	Ngày cập nhật

Bảng MENU

STT	Thuộc tính	Kiểu dữ liệu	Diễn giải
1	MENU_ID	NUMBER(5,0)	Mã số duy nhất của bảng MENU
2	MENU_NAME	VARCHAR2(40 BYTE)	Tên menu
3	PARENT_ID	NUMBER(5,0)	Mã số Menu cha
4	URL	VARCHAR2(100 BYTE)	Link menu
5	ORDER_INDEX	NUMBER(5,0)	Thứ tự sắp xếp menu
6	ACTIVE_FLAG	NUMBER(1,0)	Trạng thái hoạt động: 0 – không hoạt động, 1 – hoạt động

7	CREATE_DATE	DATE	Ngày tạo
8	UPDATE_DATE	DATE	Ngày cập nhật

Bảng PRODUCT\_INFO

STT	Thuộc tính	Kiểu dữ liệu	Diễn giải
1	PRODUCT_INFO_ID	NUMBER(5,0)	Mã số duy nhất của sản phẩm
2	CATEGORY_ID	NUMBER(5,0)	Mã số thể loại
3	CODE	VARCHAR2(30 BYTE)	Mã định danh sản phẩm
4	NAME	VARCHAR2(50 BYTE)	Tên sản phẩm
5	IMG_URL	VARCHAR2(300 BYTE)	Link hình ảnh sản phẩm
6	PRICE	NUMBER(15,0)	Giá sản phẩm
7	DESCRIPTION	VARCHAR2(200 BYTE)	Mô tả
8	ACTIVE_FLAG	NUMBER(1,0)	Trạng thái hoạt động: 0 – không hoạt động, 1 – hoạt động
9	CREATE_DATE	DATE	Ngày tạo
10	UPDATE_DATE	DATE	Ngày cập nhật

Bảng PRODUCT\_IN\_STOCK

STT	Thuộc tính	Kiểu dữ liệu	Diễn giải
1	PRODUCT_STOCK_ID	NUMBER(5,0)	Mã số duy nhất của bảng PRODUCT_IN_STOCK
2	PRODUCT_ID	NUMBER(5,0)	Mã số sản phẩm
3	QUANTITY	NUMBER(11,0)	Số lượng
4	ACTIVE_FLAG	NUMBER(1,0)	Trạng thái hoạt động: 0 – không hoạt động, 1 – hoạt động
5	CREATE_DATE	DATE	Ngày tạo
6	UPDATE_DATE	DATE	Ngày cập nhật

Bảng ROLE

STT	Thuộc tính	Kiểu dữ liệu	Diễn giải
1	ROLE_ID	NUMBER(5,0)	Mã số duy nhất của bảng ROLE
2	ROLE_NAME	VARCHAR2(30 BYTE)	Tên quyền
3	DESCRIPTION	VARCHAR2(200 BYTE)	Mô tả
4	ACTIVE_FLAG	NUMBER(1,0)	Trạng thái hoạt động: 0 – không hoạt động, 1 – hoạt động
5	CREATE_DATE	DATE	Ngày tạo
6	UPDATE_DATE	DATE	Ngày cập nhật

Bảng USERS

STT	Thuộc tính	Kiểu dữ liệu	Diễn giải
1	USER_ID	NUMBER(5,0)	Mã số duy nhất của nhân viên
2	FIRST_NAME	VARCHAR2(25 BYTE)	Tên người dùng
3	LAST_NAME	VARCHAR2(25 BYTE)	Họ người dùng
4	BIRTHDAY	DATE	Ngày sinh
5	USER_NAME	VARCHAR2(30 BYTE)	Tên đăng nhập
6	PASSWORD	VARCHAR2(30 BYTE)	Mật khẩu
7	EMAIL	VARCHAR2(50 BYTE)	Địa chỉ email
8	PHONE	VARCHAR2(12 BYTE)	Điện thoại
9	ACTIVE_FLAG	NUMBER(1,0)	Trạng thái hoạt động: 0 – inactive, 1 – active
10	CREATE_DATE	DATE	Ngày tạo
11	UPDATE_DATE	DATE	Ngày cập nhật

Bảng USER\_ROLE

STT	Thuộc tính	Kiểu dữ liệu	Diễn giải
1	USER_ROLE_ID	NUMBER(5,0)	Mã số duy nhất của bảng USER_ROLE
2	USER_ID	NUMBER(5,0)	Mã người dùng
3	ROLE_ID	NUMBER(5,0)	Mã phân quyền
4	ACTIVE_FLAG	NUMBER(1,0)	Trạng thái hoạt động: 0 – không hoạt động, 1 – hoạt động



5	CREATE_DATE	DATE	Ngày tạo
6	UPDATE_DATE	DATE	Ngày cập nhật

## **CHƯƠNG 3. CÔNG NGHỆ SỬ DỤNG**

### **3.1. Oracle**

#### **3.1.1. Giới thiệu Oracle**

Oracle Database hay còn gọi là Oracle RDBMS hoặc đơn giản là Oracle (do đây có lẽ là sản phẩm nổi tiếng nhất của hãng), là 1 hệ quản trị cơ sở dữ liệu quan hệ, được phát triển và phân phối bởi tập đoàn Oracle.

Là hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) mang tính mềm dẻo, linh động, thích ứng cao với các quy mô xử lý giao dịch, an toàn hệ thống. Cung cấp các công cụ xây dựng và quản lý cơ sở dữ liệu.

Giống như các phần mềm RDBMS khác, Oracle Database được xây dựng dựa trên ngôn ngữ SQL. Phần mềm Oracle được bổ sung PL/SQL, được Oracle phát triển nhằm bổ sung một số extension độc quyền cho SQL chuẩn – khá phổ biến trong các nhà cung cấp RDBMS.

Cơ sở dữ liệu Oracle cũng hỗ trợ lập trình bằng Java, tích hợp web (kết nối ứng dụng với công nghệ Web được tích hợp trong Oracle WebServer)

#### **3.1.2. Các phiên bản**

- Phiên bản 1 (năm 1977), Phiên bản 2 (năm 1979)
- Phiên bản 3 (năm 1983), Phiên bản 4 (1984)
- Phiên bản 5 phát hành năm 1985 (SQLNet: hệ thống khách/chủ (client/server)).
- Phiên bản 6 phát hành năm 1988 (Sequence, thao tác ghi trễ).
- Oracle7 được phát hành năm 1992 (SQL\*DBA).
- Năm 1999 Oracle giới thiệu Oracle8i (i:internet).
- Năm 2001-2002: 2 phiên bản Oracle9i (Release 1&2).
- Năm 2004-2005: 2 phiên bản Oracle10g (g:Grid) (Release 1&2).
- Năm 2008 – 2009: Phiên bản 11g
- Năm 2013: Phiên bản 12c (c: cloud)

- Năm 2018: Phiên bản 18c, Cơ sở Dữ liệu Tự động của Oracle (Oracle Autonomous Databases, cloud database có khả năng tự quản lý, tự bảo vệ và tự sửa chữa nhờ công nghệ học máy (machine learning) giúp cung cấp hiệu suất cao, khả năng bảo mật mà không cần sự can thiệp của con người, với chi phí thấp hơn một nửa so với Amazon Web Services.

### 3.1.3. Oracle 12c

Để theo kịp xu hướng Cloud Computing, PaaS (Platform as a Service), từ phiên bản Oracle 12c thì Oracle đã giới thiệu tính năng mới là Pluggable Database. Tính năng này cho phép 1 Database độc lập có thể được di chuyển dễ dàng qua các hệ thống khác nhau hoặc qua các nền tảng khác nhau hoặc giữa các bản phân phối (release) Database khác nhau.

Multitenant là 1 khái niệm mới được sử dụng trong phiên bản Oracle 12c và là 1 bước thay đổi lớn trong lịch sử kiến trúc của Oracle. Multitenant bao gồm 2 khái niệm nhỏ là Container Database (CDB) and Pluggable Database (PDB).

Khái niệm database trong 11g tương ứng với khái niệm Container Database (CDB) trong 12c. Cụ thể sau khi cài đặt phần mềm Oracle 12c, có thể tạo 1 hoặc nhiều Container Database (CDB). (Thực tế là chỉ cần 1).

Trong Oracle 12C có một khái niệm mới là CDB\$ROOT (Hoặc gọi là CDB Root) là một đối tượng nằm trong CDB. Các SCHEMA có thể gắn vào trên CDB\$ROOT. Hoặc có các Pluggable Database (PDB) thông thường khác có gắn vào CDB\$ROOT. Mỗi Pluggable Database chứa 0 hoặc nhiều SCHEMA.

PDB\$SEED là một Pluggable database mẫu (Template) nó được sử dụng để làm mẫu (mặc định) để tạo ra một Pluggable Database mới. Tất nhiên có thể lấy một Pluggable Database bất kỳ nào đó làm mẫu để tạo ra một Pluggable Database mới.

Mối quan hệ giữa CDB và PDB là mối quan hệ 1-nhiều. Trong phiên bản Oracle 12c R1 thì 1 CDB có thể chứa tối đa là 250 PDB.

## 3.2. Hibernate

### 3.2.1. ORM framework

ORM (Object Relational Mapping) framework là một cơ chế cho phép người lập trình thao tác với database một cách hoàn toàn tự nhiên thông qua các đối tượng. Lập trình viên hoàn toàn không quan tâm đến loại database sử dụng SQL Server, Oracle, MySQL, PostgreSQL, ...

ORM giúp đơn giản hoá việc tạo ra dữ liệu, thao tác dữ liệu và truy cập dữ liệu. Đó là một kỹ thuật lập trình để ánh xạ đối tượng vào dữ liệu được lưu trữ trong cơ sở dữ liệu.

### **3.2.2. Hibernate framework**

Hibernate là một trong những ORM Framework. Hibernate framework là một framework cho persistence layer. Như vậy, nhờ có Hibernate framework mà giờ đây khi bạn phát triển ứng dụng bạn chỉ còn chú tâm vào những layer khác mà không phải bận tâm nhiều về persistence layer nữa.

Hibernate giúp lập trình viên viết ứng dụng Java có thể map các object (POJO) với hệ quản trị cơ sở dữ liệu quan hệ (database), và hỗ trợ thực hiện các khái niệm lập trình hướng đối tượng với cơ sở dữ liệu quan hệ.

Hibernate giúp lưu trữ và truy vấn dữ liệu quan hệ mạnh mẽ và nhanh. Hibernate cho phép bạn truy vấn dữ liệu thông qua Java Persistence API (JPA) hoặc bằng ngôn ngữ SQL mở rộng của Hibernate (HQL) hoặc bằng SQL thuần (Native SQL).

Kiến trúc Hibernate bao gồm nhiều đối tượng như đối tượng persistent D, session factory, transaction factory, connection factory, session, transaction, ...

### **3.3. JPA**

JPA là viết tắt của Java Persistence API, nó là một đặc tả Java cho việc ánh xạ giữa các đối tượng Java với cơ sở dữ liệu quan hệ sử dụng công nghệ phổ biến là ORM (Object Relational Mapping).

JPA cung cấp đầy đủ các công cụ cho phép chúng ta có thể thao tác với cơ sở dữ liệu một cách đơn giản và nhanh chóng. JPA có thể dùng để persist một đối tượng Java (POJO – Plain Old Java Object) vào trong cơ sở dữ liệu hoặc lấy dữ liệu từ cơ sở dữ liệu và ánh xạ (mapping) ra các đối tượng Java một cách đơn giản.

JPA hoạt động như một cầu nối giữa các table/ các mối quan hệ giữa các table trong database và các class/ mối quan hệ giữa các object. Ví dụ: table USER với các column (Id, username, password) sẽ tương ứng với class User.java với các field Id, username, password. Từ đó mỗi khi truy vấn table hay các column ta sẽ gọi trực tiếp các phương thức trên các class, các field của class mà không cần quan tâm tới việc đang dùng loại database nào, kiểu dữ liệu database ra sao, ...

### **3.4. Java Spring**

Spring là framework phát triển ứng dụng phổ biến nhất dành cho Java Enterprise. Ban đầu nó được viết bởi Rod Johnson và lần đầu tiên được phát hành theo giấy phép Apache 2.0 vào tháng 6 năm 2003. Spring có kích thước nhẹ, phiên bản cơ bản của Spring framework có kích thước khoảng 2MB.

Spring framework là một Java Platform mã nguồn mở, một giải pháp gọn nhẹ dành cho Java Enterprise. Với Spring Framework các nhà phát triển có thể tạo ra các mã có hiệu suất cao, dễ kiểm thử và có thể sử dụng lại được.

Các tính năng core của Spring Framework có thể được sử dụng trong việc phát triển bất kỳ ứng dụng Java nào. Bên cạnh đó, phần mở rộng được sử dụng để xây dựng các ứng dụng web trên nền tảng Java EE. Mục tiêu của Spring Framework là làm cho việc phát triển ứng dụng J2EE dễ dàng hơn và thúc đẩy việc lập trình tốt hơn bằng mô hình POJO-based.

### **3.5. Github**

#### **3.5.1. Git**

Git là tên gọi là một hệ thống quản lý phiên bản phân tán phổ biến nhất hiện nay (Distributed Version Control System – DVCS). Git sẽ giúp người dùng lưu lại các phiên bản của những lần thay đổi vào mã nguồn và sẽ dễ dàng cho việc khôi phục lại mà không cần phải thủ công copy rồi paste vào đâu đó, phiên bản đó đã được sao lưu. Khi chúng ta phát hiện ra lỗi ở đâu đó và muốn backup lại phiên làm việc trước khi bị lỗi xảy ra thì sẽ thật đơn giản khi chúng ta sử dụng Git. Một điểm đặc biệt nữa là một thành viên trong cùng một team khi làm việc với nhau hoàn toàn có thể theo dõi online được các thay đổi của các thành viên khác ở từng phiên bản làm việc mà không nhất thiết phải ngồi ngay

cạnh nhau, họ cũng có thể đối chiếu được những thay đổi đó để rồi gộp phiên bản của thành viên khác vào phiên bản của họ. Cuối cùng là tất cả có thể đưa các thay đổi vào mã nguồn của mình lên một kho chứa mã nguồn.

### **3.5.2. Github**

Github được cho ra đời với mục đích cung cấp dịch vụ Git server miễn phí. Ngoài ra trang này cũng cung cấp các giao diện để người dùng có thể dễ dàng theo dõi các sự thay đổi trên trình duyệt mà không cần phải cài phần mềm Git trên máy tính.

## **3.6. IntelliJ IDEA**

IntelliJ IDEA là một IDE Java thông minh cung cấp một sự kết hợp mạnh mẽ của các công cụ phát triển phần mềm. IntelliJ IDEA là công cụ nhằm tạo ra những dự án lập trình cho điện thoại hoặc cho khả năng mã hóa sâu sắc và điều hướng nhanh, phần mềm này còn cung cấp cho người dùng một danh sách các biểu tượng và ký hiệu phục vụ trong công việc lập trình của người dùng. Chức năng của IntelliJ IDEA được tiếp tục mở rộng bởi người dùng và bên thứ ba thông qua các plugin. IntelliJ IDEA cung cấp hỗ trợ cho Java EE, Spring / Hibernate và các ngôn ngữ công nghệ khác.

## CHƯƠNG 4. XÂY DỰNG VÀ QUẢN LÝ GIAO TÁC

### 4.1. Trigger

#### 4.1.1. Trigger trong Oracle

Trigger là một đơn vị chương trình lưu trữ trong database và thực thi (fire) để đáp ứng một sự kiện nào đó.

Sự kiện này được kết hợp với một table, view, schema, hoặc database, và là một trong những sự kiện sau:

- Một câu lệnh DML (DELETE, INSERT, hoặc UPDATE).
- Một câu lệnh DDL (CREATE, ALTER, DROP)
- Một tác vụ trên database (SERVERERROR, LOGON, LOGOFF, STARTUP, hoặc SHUTDOWN).

#### Cú pháp tạo Trigger:

```
CREATE [OR RELACE] TRIGGER trigger_name
    {BEFORE | AFTER}
    {DELETE, INSERT, UPDATE [OF column_name....]}
    ON table_name
    [REFERENCING {OLD AS old, NEW AS new}]
    [FOR EACH ROW [WHEN condition]]
    DECLARE
        Variable declaration;
        Constant declaration;
    BEGIN
        PL/SQL subprogram body;
    [EXCEPTION
        exception PL/SQL block;
    END;
```

### **Giải thích:**

CREATE [OR REPLACE ] TRIGGER trigger\_name: tạo hoặc thay thế một trigger đã tồn tại thành trigger\_name.

{ BEFORE | AFTER }: cái này chỉ định khi nào trigger được thực thi.

{DELETE, INSERT, UPDATE [OF column\_name....]} chỉ định cụ thể lệnh nào thuộc loại lệnh DML được thực thi. [OF col\_name]: chỉ định cụ thể cột nào sẽ được cập nhật.

ON table\_name: chỉ định trigger sẽ được thực thi trên bảng nào.

[REFERENCING {OLD AS old, NEW AS new}]: cho phép dùng giá trị new hay old cho các lệnh DML, như là Insert, Update, Delete.

[FOR EACH ROW [WHEN condition]]: cho biết cụ thể trigger sẽ thực thi trên từng dòng. WHEN condition: cho biết dòng đó có điều kiện gì thì trigger mới thực thi được.

#### **4.1.2. Danh sách các Trigger**

ST T	Tên	Thao tác	Bảng	Nội dung	Ghi chú
1	update_quantity	INSERT	INVOICE_DE TAIL	Cập nhập số hàng tồn kho khi thêm một hóa đơn nhập hàng hoặc hóa đơn xuất hàng bất kì.	
2	check_quantity_in _stock	INSERT, UPDATE	PRODUCT_IN _STOCK	Kiểm tra số hàng tồn kho khi thêm mới hoặc sửa một dòng có vượt quá 150.	
3	validator_email	INSERT, UPDATE	USERS	Kiểm tra Email khi thêm mới hoặc sửa thông tin một User có đúng chuẩn hay chưa.	
4	update_price_on_i nvoice	INSERT, UPDATE, DELTE	INVOICE_DE TAIL	Cập nhập tổng số tiền của hóa đơn khi thêm, sửa hoặc xóa một dòng invoice_detail	



#### 4.1.3. Mô tả một số Trigger

- a. Cập nhập số lượng sản phẩm khi thêm một hóa đơn

Tên: update\_quantity

Thao tác: INSERT

Trên bảng: INVOICE\_DETAIL

**Mã PL/SQL:**

```
CREATE OR REPLACE TRIGGER update_quantity
CREATE OR REPLACE TRIGGER update_quantity
BEFORE INSERT ON INVOICE_DETAIL
FOR EACH ROW
DECLARE
    v_type INVOICE.TYPE%TYPE;
    v_quantity PRODUCT_IN_STOCK.QUANTITY%TYPE;
BEGIN
    SELECT TYPE INTO v_type
    FROM INVOICE
    WHERE INVOICE_ID = :NEW.INVOICE_ID AND ACTIVE_FLAG = 1;

    SELECT QUANTITY INTO v_quantity
    FROM PRODUCT_IN_STOCK
    WHERE PRODUCT_ID = :NEW.PRODUCT_ID;

    IF v_type = 1 THEN
        --NHAP HANG
        UPDATE PRODUCT_IN_STOCK
        SET QUANTITY = QUANTITY + :NEW.QUANTITY
```

```

WHERE PRODUCT_ID = :NEW.PRODUCT_ID AND ACTIVE_FLAG = 1;

ELSE

--XUAT HANG

IF v_quantity - :NEW.QUANTITY > 0 THEN

    UPDATE PRODUCT_IN_STOCK

    SET QUANTITY = QUANTITY - :NEW.QUANTITY

    WHERE PRODUCT_ID = :NEW.PRODUCT_ID AND ACTIVE_FLAG = 1;

ELSE

    raise_application_error(-20001, 'Quantity is not enough to export!');

END IF;

END IF;

EXCEPTION

WHEN NO_DATA_FOUND THEN

    DBMS_OUTPUT.PUT_LINE('INSERT NEW PRODUCT_IN_STOCK!');

    INSERT INTO PRODUCT_IN_STOCK(PRODUCT_ID, QUANTITY) VALUES
(:NEW.PRODUCT_ID, :NEW.QUANTITY);

WHEN OTHERS THEN

    DBMS_OUTPUT.PUT_LINE('OTHERS EXCEPTION!');

END;

```

### **Các bước thực hiện:**

[1]: Lấy ra loại của hóa đơn thêm vào

[2]: Lấy ra QUANTITY tương ứng với product\_id thêm vào. Nếu QUANTITY khác rỗng có nghĩa là product\_id đã có trong kho hàng thì ta sẽ thực hiện cập nhập số lượng, ngược lại ta sẽ thêm mới một dòng dữ liệu vào kho hàng.

[3]:Kiểm tra giá trị type.

[3.1]: Nếu type = 1 thì ta sẽ cập nhập số lượng mới sẽ bằng số lượng cũ cộng cho số lượng đơn nhập hàng.

[3.2]: Ngược lại, nếu type = 0 thì ta sẽ cập nhập số lượng mới sẽ bằng số lượng cũ trừ cho số lượng đơn nhập hàng với kết quả sau khi trừ phải lớn hơn 0.

b. Kiểm tra Email có đúng định dạng

Tên: validator\_email

Thao tác: INSERT, UPDATE

Trên bảng: USERS

**Mã PL/SQL:**

```
CREATE OR REPLACE TRIGGER validator_email
BEFORE INSERT OR UPDATE ON USERS
FOR EACH ROW
BEGIN
    IF (REGEXP_LIKE (:NEW.EMAIL , '^[A-Za-z]+[A-Za-z0-9.]++@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}$')) THEN
        DBMS_OUTPUT.PUT('Email validate success!');
    ELSE
        raise_application_error(-20001,'Email incorrect!');
    END IF;
END;
```

[1]: Kiểm tra email của User khi insert hay update vào bảng USERS có đúng định dạng hay chưa:

[1.1]: Nếu đúng thì không làm gì cả

[1.2]: Nếu sai thì raise ra lỗi và dòng không được thêm hay cập nhập vào bảng.

## 4.2. Store Procedure

### 4.2.1. Store procedure trong Oracle

Một chương trình con (PL/SQL subprogram) chính là một khối lệnh PL/SQL được đặt tên và được gọi với một tập các đối số. Một chương trình con có thể là một thủ tục (procedure) hoặc là một hàm (function).

Thông thường, procedure được sử dụng để thực hiện một tác vụ nào đó còn function được sử dụng để tính toán và trả về kết quả.

Một thủ tục (procedure) và một hàm (function) có cùng cấu trúc, ngoại trừ:

Phần đầu của function phải chứa mệnh đề RETURN (return clause) xác định kiểu dữ liệu trả về. Còn procedure không chứa mệnh đề RETURN này.

Một function phải chứa ít nhất một câu lệnh RETURN (return statement) trong phần thực thi. Trong procedure, câu lệnh RETURN không bắt buộc.

#### Cú pháp:

```
CREATE [OR REPLACE] PROCEDURE procedure_name
    [ (parameter [parameter]) ]
IS
    [declaration_section]
BEGIN
    executable_section
[EXCEPTION
    exception_section]
END [procedure_name];
```

#### Giải thích:

- procedure\_name : Tên của procedure
- [OR REPLACE]: Thay thế những procedure đang tồn tại.
- IS: [declaration\_section] Những biến muốn khai báo thì khai báo ở đây
- executable\_section: chứa những câu lệnh của procedure.
- [EXCEPTION exception\_section]: Xử lý những ngoại lệ khi xảy ra.

#### 4.2.2. Transaction trong Oracle

##### a. Định nghĩa

Một giao tác là một đơn vị công việc nguyên tử chứa một hoặc nhiều câu lệnh SQL.

Nguyên tố: không thể phân chia được nữa. Hoặc là thực hiện thành công (committed) hoặc là bị hủy bỏ (rolled back)

Tính (ACID) của giao tác: giao tác phải đảm bảo tính nguyên tố (Atomic), nhất quán (Consistency), cô lập (Isolation), bền vững (Durability).

Một session bắt đầu một transaction khi gặp câu lệnh DML và kết thúc khi gặp một câu lệnh COMMIT hoặc ROLLBACK.

##### b. Cấu trúc Transaction

Một transaction có bắt đầu và kết thúc.

Bắt đầu của một Transaction:

Một Transaction bắt đầu khi câu lệnh SQL đầu tiên được thực thi, bao gồm câu lệnh DDL, DML và SET TRANSACTION.

```
SET TRANSACTION
```

```
[ ISOLATION LEVEL [ SERIALIZE | READ COMMITTED ]
```

```
[ NAME 'transaction_name' ];
```

- ISOLATION LEVEL: mức cô lập của transaction sẽ có hai lựa chọn là SERIALIZE hoặc READ COMMITTED.

- NAME 'transaction\_name' đặt tên cho transaction.
- Khi 1 transaction mới bắt đầu , hệ quản trị oracle sẽ gán nó vào undo data segment ( ghi nhận lại các thao tác của transaction trước khi commit, để có thể rollback khi có lỗi).

Kết thúc một Transaction:

Một Transaction kết thúc khi có bất kì hành động nào dưới đây:

- Gặp lệnh COMMIT hoặc ROLLBACK mà không có SAVEPOINT.
- Chạy một câu lệnh DDL như CREATE, DROP, RENAME hoặc ALTER.
- User ngắt kết nối đến hệ quản trị đột ngột, transaction sẽ tự động commit.
- Các ứng dụng đang kết nối đến hệ quản trị bị dừng đột ngột, transaction sẽ tự động rollback.

#### c. Transaction control

Gồm các lệnh để quản lý sự thay đổi của DML lên database, gồm 1 số lệnh chính:

- SAVEPOINT: Xác định 1 điểm trong transaction để rollback về khi có sự cố.
- COMMIT: Kết thúc transaction, lưu thay đổi vĩnh viễn, xóa tất cả SAVEPOINT, mở transaction locks.
- ROLLBACK: phục hồi lại dữ liệu trước khi thay đổi.

ROLLBACK:

- Hoàn tác mọi thay đổi.
- Mở tất cả khóa.
- Xóa toàn bộ savepoints.
- Kết thúc transaction.

#### 4.2.3. Danh sách Store Procedure

STT	Tên	Tham số đầu vào	Tham số đầu ra	Ý nghĩa	Ghi chú
-----	-----	-----------------	----------------	---------	---------

1	get_amount	Table of an Obejct with 2 properties (1*)	NUMBER	Lấy tổng số tiền của một danh sách các mặt hàng sắp được thêm vào hóa đơn	
2	add_goods_receipt_invoice	(1*), v_type NUMBER, v_user_id NUMBER, v_code VARCHAR2		Với đầu vào là một danh sách mặt hàng để thêm vào invoice. Khi thêm invoice thì procedure này sẽ thêm danh sách invoice_detail tương ứng.	
3	create_category	(v_cate_id NUMBER, v_name VARCHAR2, v_description VARCHAR2, v_code VARCHAR2		Nếu tham số đầu vào là một category đã có trong CSDL thì ta cập nhập giá trị, ngược lại nếu là category chưa có trong CLDL thì ta thêm vào.	
4	check_invoice_price	v_invoice_id INVOICE.INVOICE_ID%TYPE	NUMBER	Kiểm tra tổng tiền trong hóa đơn có bằng tổng tiền của các chi tiết hóa đơn	
5	get_quantity_product_in_stock			Lấy ra quantity hiện tại của một Product	
6	get_info_user	v_id_user USERS.USER_ID%TYPE		Lấy và in ra thông tin của User với user_id	
7	get_product_price	v_product_id PRODUCT_INFO.PRODUCT_INFO_ID%TYPE	PRODUCT_INFO.PRICE%TYPE	Lấy giá tiền của sản phẩm với product_info_id	

#### 4.2.4. Mô tả một số Store Procedure

##### a. Lấy tổng số tiền của một hóa đơn

Tên: get\_amount

Nội dung: Lấy tổng số tiền của một danh sách các mặt hàng sắp được thêm vào hóa đơn

Tham số đầu vào: v\_l\_info\_gr, list\_info\_goods\_receipt

Tham số đầu ra: NUMBER

## Mã PL/SQL:

```
CREATE OR REPLACE TYPE info_goods_receipt AS OBJECT (v_product_id NUMBER(5,0),
v_quantity NUMBER(5,0));

CREATE OR REPLACE TYPE list_info_goods_receipt IS TABLE OF info_goods_receipt;

CREATE OR REPLACE FUNCTION get_amount(v_l_info_gr list_info_goods_receipt)
RETURN NUMBER
AS
    v_amount NUMBER;
    v_price PRODUCT_INFO.PRICE%TYPE;
BEGIN
    v_amount := 0;

    FOR ind IN v_l_info_gr.FIRST..v_l_info_gr.LAST
    LOOP
        --GET PRICE FROM PRODUCT ID

        SELECT PRICE INTO v_price
        FROM PRODUCT_INFO
        WHERE PRODUCT_INFO_ID = v_l_info_gr(ind).v_product_id;

        v_amount := v_amount + v_l_info_gr(ind).v_quantity * v_price;
    END LOOP;

    RETURN v_amount;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('NO_DATA_FOUND!');
    WHEN OTHERS THEN
```



```
DBMS_OUTPUT.PUT_LINE('OTHERS EXCEPTION!');  
  
END;
```

### Các bước thực hiện:

Bước nằm ngoài function: tạo 2 TYPE là tham số đầu vào của function là info\_goods\_receipt và list\_info\_goods\_receipt.

[1]: Duyệt danh sách v\_l\_info\_gr là tham số đầu vào

[2]: Lấy ra giá trị PRICE của sản phẩm tương ứng với product\_id

[3]: Giá trị tổng tiền(v\_amount) của hóa đơn bằng tổng tiền trước đó cộng cho số lượng sản phẩm nhân cho số lượng mà được thêm vào.

[4]: Trả về giá trị tổng tiền

#### b. Lấy tổng số tiền của một hóa đơn

Tên: add\_goods\_receipt\_invoice

Nội dung: Với đầu vào là một danh sách mặt hàng để thêm vào invoice. Khi thêm invoice thì procedure này sẽ thêm danh sách invoice\_detail tương ứng.

Tham số đầu vào: v\_l\_info\_gr list\_info\_goods\_receipt, v\_invoice\_id, v\_type NUMBER, v\_user\_id NUMBER, v\_code VARCHAR2

Tham số đầu ra:

### Mã PL/SQL:

```
CREATE OR REPLACE PROCEDURE add_goods_receipt_invoice(v_l_info_gr  
list_info_goods_receipt,  
v_invoice_id NUMBER, v_type NUMBER, v_user_id NUMBER, v_code  
VARCHAR2)  
AS  
v_mount NUMBER;  
  
BEGIN
```

```

--INSERT INTO INVOICE

v_mount := get_amount(v_l_info_gr);

INSERT INTO INVOICE(INVOICE_ID, TYPE, USER_ID, PRICE, CODE)
VALUES(v_invoice_id, v_type, v_user_id, v_mount, v_code);

FOR ind IN v_l_info_gr.FIRST..v_l_info_gr.LAST
LOOP

--INSERT INTO INVOICE DETAIL

INSERT INTO INVOICE_DETAIL(INVOICE_ID, PRODUCT_ID, QUANTITY)

VALUES (v_invoice_id, v_l_info_gr(ind).v_product_id, v_l_info_gr(ind).v_quantity);

END LOOP;

END;

```

### **Các bước thực hiện:**

[1]: Tính tổng tiền cho hóa đơn

[2]: Thêm một dòng dữ liệu cho bảng INVOICE với các tham số là v\_invoice\_id, v\_type, v\_user\_id, v\_mount, v\_code.

[3]: Duyệt từng phần tử trong danh sách list\_info\_goods\_receipt và tương ứng với mỗi phần tử thì ta sẽ thêm một dòng cho bảng INVOICE\_DETAIL.

#### **c. Tạo một category**

Tên: create\_category

Nội dung: Nếu tham số đầu vào là một category đã có trong CSDL thì ta cập nhật giá trị, ngược lại nếu là category chưa có trong CLDL thì ta thêm vào.

Tham số đầu vào: v\_cate\_id NUMBER, v\_name VARCHAR2, v\_description VARCHAR2, v\_code VARCHAR2

Tham số đầu ra:

**Mã PL/SQL:**

```
CREATE OR REPLACE PROCEDURE create_category(v_cate_id NUMBER, v_name
VARCHAR2,
        v_description VARCHAR2, v_code VARCHAR2)
AS
    v_id NUMBER;
BEGIN
    SELECT CATEGORY_ID INTO v_id
    FROM CATEGORY
    WHERE CATEGORY_ID = v_cate_id;

    IF SQL%ROWCOUNT = 1 THEN
        DBMS_OUTPUT.PUT_LINE('UPDATE CATEGORY!');
        UPDATE CATEGORY
        SET NAME = v_name, DESCRIPTION = v_description, CODE = v_code
        WHERE CATEGORY_ID = v_cate_id;
    END IF;

    EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('INSERT CATEGORY!');
        INSERT INTO CATEGORY(CATEGORY_ID, NAME, DESCRIPTION, CODE)
        VALUES(v_cate_id, v_name, v_description, v_code);
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('OTHERS EXCEPTION!');
END;
```

**Các bước thực hiện:**

[1]: Lấy ra category\_id tương ứng với giá trị v\_cate\_id tham số đầu vào để kiểm tra giá trị v\_cate\_id đã có trong bảng CATEGORY hay chưa

[1.1]: Nếu đã có giá category\_id trong bảng CATEGORY thì ta sẽ cập nhật giá trị mới cho category đó.

[1.2]: Ngược lại nếu chưa có giá category\_id trong bảng CATEGORY thì ta sẽ thêm một dòng vào bảng CATEGORY.

## CHƯƠNG 5. XỬ LÝ TRUY XUẤT ĐỒNG THỜI

### 5.1. Các mức cô lập trong Oracle

#### 5.1.1. Read committed

a. Trong Read Committed , mọi truy vấn được thực hiện bởi một transaction chỉ nhìn thấy dữ liệu đã được commit. Mức cô lập này là mặc định. Nó phù hợp với môi trường cơ sở dữ liệu trong đó ít transaction có khả năng xung đột.

#### b. Xung đột trong Read committed

Trong Read Committed, một xung đột ghi xảy ra khi transaction cố gắng thay đổi một dòng đã được cập nhật bởi một transaction khác mà chưa được COMMIT.

#### c. Ưu điểm và khuyết điểm

- Ưu điểm:
  - o Giải quyết vấn đề Dirty Reads.
  - o Shared Lock được giải phóng ngay, không cần phải giữ cho đến hết giao tác nên không cản trở nhiều đến thao tác cập nhật của các giao tác khác.
- Khuyết điểm:
  - o Chưa giải quyết được vấn đề Unrepeatable Reads, Phantoms, Lost Updates.
  - o Phải chờ nếu đơn vị dữ liệu cần đọc đang được giữ khoá ghi.

#### 5.1.2. Serializable

a. Ở mức cô lập tuần tự (serializable isolation level), transaction chỉ thấy những thay đổi đã được commit trước thời điểm transaction có mức cô lập tuần tự này bắt đầu và các thay đổi được thực hiện bởi chính transaction này.

#### b. Ưu điểm và khuyết điểm:

- Ưu điểm:

- Giải quyết thêm được vấn đề Phantoms.
- Khuyết điểm:
  - Phải chờ nếu đơn vị dữ liệu cần đọc đang được giữ khoá ghi.
  - Cản trở nhiều đến việc cập nhật dữ liệu của các giao tác khác.

c. Một ví dụ cho serializable

Session 1	Session 2	Explanation
SELECT last_name, salary FROM employees WHERE last_name IN ('Banda', 'Greene', 'Hintz');  LAST_NAME SALARY ----- Banda                      6300 Greene                      9500	No action	Session 1 thực hiện truy vấn lấy ra họ và lương của nhân viên có họ là Banda, Greene, Hintz.
UPDATE employees SET salary = 7000 WHERE last_name='Banda';	No action	Session 1 thực hiện update lương cho nhân viên có họ là Banda
No action	SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;	Session 2 bắt đầu transaction và thiết lập mức cô lập là SERIALIZABLE
No action	SELECT last_name, salary FROM employees WHERE last_name IN ('Banda', 'Greene', 'Hintz');  LAST_NAME SALARY ----- - Banda                      6300 Greene                      9500	Session 2 thực hiện truy vấn lấy ra họ và lương của nhân viên có họ là Banda, Greene, Hintz.

	UPDATE employees SET salary = 9900 WHERE last_name = 'Greene';	Session 2 thực hiện update lương cho nhân viên có họ là Greene.
INSERT INTO employees (employee_id, last_name, email, hire_date, job_id) VALUES (210, 'Hintz', 'JHINTZ', SYSDATE, 'SH_CLERK');		Session 1 thực hiện insert một nhân viên có họ là Hintz
COMMIT;		Session 1 thực hiện COMMIT
SELECT last_name, salary FROM employees WHERE last_name IN ( 'Banda', 'Greene', 'Hintz' );  LAST_NAME SALARY ----- Banda                      7000 Greene                     9500 Hintz	SELECT last_name, salary FROM employees WHERE last_name IN ( 'Banda', 'Greene', 'Hintz' );  LAST_NAME SALARY ----- - Banda                      6300 Greene                     9900	Session 1 và Session 2 thực hiện truy vấn lấy ra họ và lương của nhân viên có họ là Banda, Greene, Hintz. Kết quả hiển thị ở Session 1 cho thấy chỉ có lương của nhân viên có họ là Banda được cập nhập. Tương tự ở Session 2 cũng chỉ thấy lương của nhân viên có họ là Greene được cập nhập mặc dù Session 1 đã COMMIT.
	COMMIT;	Session 2 thực hiện COMMIT

### 5.1.3. Read-only

Mức cô lập Read-only tương tự như mức cô lập Serializable, nhưng các transaction có mức cô lập Read-only chỉ được đọc dữ liệu, không được sửa đổi dữ liệu trong transaction (trừ trường hợp là người dùng là SYS).

## 5.2. Cơ chế khóa

Khóa là một cơ chế để ngăn chặn các tương tác phá hoại.

Tương tác phá hoại là khi nó cập nhập dữ liệu sai hoặc thay đổi không chính xác cấu trúc dữ liệu khiến dữ liệu không nhất quán giữa, thường xảy ra khi nhiều transaction cùng thao tác lên 1 đơn vị dữ liệu.

Vì vậy, cơ chế khóa đóng vai trò cực kì quan trọng để đảm bảo cơ sở dữ liệu đồng thời và nhất quán.

Tóm tắt các hành vi khóa:

- Cơ sở dữ liệu(CSDL) có một số khóa khác nhau, tùy thuộc vào loại hoạt động lên dữ liệu. Nhìn chung, CSDL sử dụng hai loại khóa chính: exclusive locks (khóa độc quyền) và share locks (khóa chia sẻ).
- Tại một thời điểm chỉ có 1 exclusive lock được thực thi trên 1 đơn vị dữ liệu, nhưng nhiều share locks có thể cùng thực thi trên 1 đơn vị dữ liệu.
- Khóa ảnh hưởng tới tương tác giữa việc đọc và ghi dữ liệu:
  - Một dòng chỉ bị lock khi có một hành động ghi dữ liệu.
  - Một hành động ghi lên một dòng dữ liệu sẽ chặn những hành động khác cùng ghi lên dòng dữ liệu đó.
  - Một người đọc dữ liệu sẽ không chặn một người ghi lên dữ liệu. Chỉ có một trường hợp ngoại lệ là câu lệnh SELECT ... FOR UPDATE sẽ lock đơn vị dữ liệu nó đang đọc.
  - Hành động ghi dữ liệu không chặn hành động đọc dữ liệu. Khi một đơn vị dữ liệu đang được thao tác bởi hành động ghi, hệ quản trị sẽ sử dụng phiên bản trước đó của đơn vị dữ liệu để trả về cho hành động đọc.

Sử dụng Locks:

Khi có nhiều người dùng đang truy cập và sửa đổi dữ liệu, CSDL phải cung cấp một cách để ngăn chặn sửa đổi đồng thời của một dữ liệu.

Cơ chế khóa phải thỏa mãn các yêu cầu quan trọng của cơ sở dữ liệu: tính nhất quán và toàn vẹn dữ liệu.

**Mô tả một ví dụ khóa dòng:**



Session 1	Session 2	Explanation
SELECT PRODUCT_INFO_ID AS PRO_ID, CODE, PRICE FROM PRODUCT_INFO WHERE PRODUCT_INFO_ID = 26;  PRO_ID CODE PRICE ----- ----- 26 XMXT11 170000	No action	Session 1 thực hiện truy vấn lấy ra id, code và price của sản phẩm có id là 26.
	SELECT PRODUCT_INFO_ID AS PRO_ID, CODE, PRICE FROM PRODUCT_INFO WHERE PRODUCT_INFO_ID = 26;  PRO_ID CODE PRICE ----- ----- 26 XMXT11 170000	Session 2 thực hiện truy vấn lấy ra id, code và price của sản phẩm có id là 26. Có cùng kết quả với session 1.
UPDATE PRODUCT_INFO SET CODE = 'XMXT00' WHERE PRODUCT_INFO_ID = 26 AND CODE = 'XMXT11';  1 row updated.		Session 1 thực hiện update code = 'XMXT00' cho sản phẩm có id là 26 và code = 'XMXT11'.
No action	UPDATE PRODUCT_INFO SET CODE = 'XMXT22' WHERE PRODUCT_INFO_ID = 26 AND CODE = 'XMXT11';	Session 2 thực hiện update code = 'XMXT22' cho sản phẩm có id là 26 và code = 'XMXT11'. Vì session 2 cập nhập cùng một dòng với session 1 nên session 2 phải chờ cho đến khi session 1 hoàn tất.
COMMIT;  Commit complete.		Session 1 thực hiện COMMIT, hoàn tất Transaction.
	0 rows updated.	Session 2 lấy được khóa nhưng lúc này không còn sản phẩm với id là 26 và code = 'XMXT11' nên session 2 không cập nhập dòng dữ liệu nào.

### 5.3. Deadlock

Deadlock là tình huống trong đó hai hoặc nhiều người đang chờ dữ liệu bị khóa bởi nhau. Deadlock ngăn cản một số transaction tiếp tục hoạt động.

Cơ sở dữ liệu Oracle tự động phát hiện các deadlock và giải quyết chúng bằng cách roll back một transaction gây deadlock và giải phóng một dòng đang bị khóa. Cơ sở dữ liệu sẽ trả về thông báo cho transaction bị rollback.

Mô tả một ví dụ cho deadlock:

Session 1	Session 2	Explanation
UPDATE PRODUCT_INFO SET PRICE = 180000 WHERE PRODUCT_INFO_ID = 25;	UPDATE PRODUCT_INFO SET PRICE = 160000 WHERE PRODUCT_INFO_ID = 26;	Session 1 thực hiện update price cho sản phẩm có id là 25. Session 2 thực hiện update price cho sản phẩm có id là 26.
UPDATE PRODUCT_INFO SET PRICE = 150000 WHERE PRODUCT_INFO_ID = 26;	UPDATE PRODUCT_INFO SET PRICE = 170000 WHERE PRODUCT_INFO_ID = 25;	Session 1 thực hiện update price cho sản phẩm có id là 26. Session 2 thực hiện update price cho sản phẩm có id là 25.
SQL Error: ORA-00060: deadlock detected while waiting for resource		Session 1 báo lỗi là xảy ra deadlock và roll back câu lệnh update sản phẩm với id là 26.
COMMIT;  Commit complete.		Session 1 thực hiện COMMIT, hoàn tất Transaction.

	1 row updated.	Session nhận được khóa và thực hiện update price cho sản phẩm với id là 25.
	COMMIT;	Session 2 thực hiện COMMIT, hoàn tất Transaction.

## 5.4. Mô tả trong đồ án môn học

### 5.4.1. Non-repeatable read

Khái niệm: Một transaction đọc lại dữ liệu mà nó đã đọc trước đó và thấy rằng một transaction khác đã commit dữ liệu đã được thay đổi hoặc xóa. Ví dụ: người dùng truy vấn một hàng và sau đó truy vấn cùng một hàng, chỉ để phát hiện ra rằng dữ liệu đã thay đổi.

Ví dụ, một giao dịch truy vấn số lượng nhân viên. Năm phút sau thực hiện cùng một truy vấn, nhưng bây giờ số lượng đã tăng lên một vì một người dùng khác đã thêm một nhân viên mới.

#### a. Mô tả tình huống

Nhân viên thứ nhất thêm một hóa đơn nhập hàng cho mặt hàng với product\_id là 21. Sau khi thêm xong nhân viên 1 lấy ra danh sách sản phẩm trong kho và kiểm tra số lượng mặt hàng mới thêm vào. Trong lúc đó thì nhân viên thứ 2 đang thêm một hóa đơn xuất hàng cũng cho mặt hàng với product\_id là 21, giao dịch thành công. Cuối cùng nhân viên 1 lấy ra danh sách sản phẩm trong kho và kiểm tra số lượng mặt hàng với product\_id là 21 thì thấy dữ liệu đã khác so với ban đầu.

#### b. Mã PL/SQL:

Function: get\_amount()

```

CREATE OR REPLACE TYPE info_goods_receipt AS OBJECT (v_product_id
NUMBER(5,0), v_quantity NUMBER(5,0));

CREATE OR REPLACE TYPE list_info_goods_receipt IS TABLE OF info_goods_receipt;

CREATE OR REPLACE FUNCTION get_amount(v_l_info_gr list_info_goods_receipt)
RETURN NUMBER
AS
    v_amount NUMBER;
    v_price PRODUCT_INFO.PRICE%TYPE;
BEGIN
    v_amount := 0;
    FOR ind IN v_l_info_gr.FIRST..v_l_info_gr.LAST
    LOOP
        --GET PRICE FROM PRODUCT ID
        SELECT PRICE INTO v_price
        FROM PRODUCT_INFO
        WHERE PRODUCT_INFO_ID = v_l_info_gr(ind).v_product_id;
        v_amount := v_amount + v_l_info_gr(ind).v_quantity * v_price;
    END LOOP;

    RETURN v_amount;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('NO_DATA_FOUND!');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('OTHERS EXCEPTION!');
END;

```

Store procedure: add\_goods\_receipt\_invoice()

```
CREATE OR REPLACE PROCEDURE add_goods_receipt_invoice(v_l_info_gr
list_info_goods_receipt,
                v_invoice_id NUMBER, v_type NUMBER, v_user_id NUMBER,
v_code VARCHAR2)
AS
    v_mount NUMBER;
BEGIN
    --INSERT INTO INVOICE
    v_mount := get_amount(v_l_info_gr);
    INSERT INTO INVOICE(INVOICE_ID, TYPE, USER_ID, PRICE, CODE)
VALUES(v_invoice_id, v_type, v_user_id, v_mount, v_code);

    FOR ind IN v_l_info_gr.FIRST..v_l_info_gr.LAST
    LOOP
        --INSERT INTO INVOICE DETAIL
        INSERT INTO INVOICE_DETAIL(INVOICE_ID, PRODUCT_ID, QUANTITY)
        VALUES (v_invoice_id, v_l_info_gr(ind).v_product_id, v_l_info_gr(ind).v_quantity);

    END LOOP;
END;
```

Trigger: update\_quantity

```
CREATE OR REPLACE TRIGGER update_quantity
BEFORE INSERT ON INVOICE_DETAIL
FOR EACH ROW
```

```

DECLARE

v_type INVOICE.TYPE%TYPE;

v_product_id PRODUCT_IN_STOCK.PRODUCT_ID%TYPE;

BEGIN

SELECT TYPE INTO v_type

FROM INVOICE

WHERE INVOICE_ID = :NEW.INVOICE_ID AND ACTIVE_FLAG = 1;


SELECT PRODUCT_ID INTO v_product_id

FROM PRODUCT_IN_STOCK

WHERE PRODUCT_ID = :NEW.PRODUCT_ID;


IF v_type = 1 THEN

--NHAP HANG

UPDATE PRODUCT_IN_STOCK

SET QUANTITY = QUANTITY + :NEW.QUANTITY

WHERE PRODUCT_ID = :NEW.PRODUCT_ID AND ACTIVE_FLAG = 1;

ELSE

--XUAT HANG

UPDATE PRODUCT_IN_STOCK

SET QUANTITY = QUANTITY - :NEW.QUANTITY

WHERE PRODUCT_ID = :NEW.PRODUCT_ID AND ACTIVE_FLAG = 1;

END IF;


EXCEPTION

WHEN NO_DATA_FOUND THEN

DBMS_OUTPUT.PUT_LINE('INSERT NEW PRODUCT_IN_STOCK!');

```

```

INSERT INTO PRODUCT_IN_STOCK(PRODUCT_ID, QUANTITY) VALUES
(:NEW.PRODUCT_ID, :NEW.QUANTITY);

```

```

WHEN OTHERS THEN

```

```

DBMS_OUTPUT.PUT_LINE('OTHERS EXCEPTION!');

```

```

END;

```

### c. Bảng mô tả Non-repeatable read

Session 1	Session 2	Explanation
<pre> SELECT PRODUCT_STOCK_ID,PRODUCT_ ID, QUANTITY FROM PRODUCT_IN_STOCK WHERE ACTIVE_FLAG = 1;  PRODUCT_STOCK_ID PRODUCT_ID  QUANTITY ----- 61      21      75 81      22      5 </pre>		Giá trị Quantity ban đầu của sản phẩm với PRODUCT_ID = 21 là 75
<pre> DECLARE v_l_info_gr list_info_goods_receipt := list_info_goods_receipt( info_goods_receipt( 21, 15) ); v_type NUMBER; v_user_id NUMBER; v_code VARCHAR2(30); v_invoice_id NUMBER; BEGIN v_type := 1; v_user_id := 1; v_code := 'HD015'; v_invoice_id := 38;  add_goods_receipt_invoice(v_l_info_g r, v_invoice_id, v_type, v_user_id, v_code); END;  PL/SQL procedure successfully completed. </pre>		Session 1 tương ứng với nhân viên thứ nhất thêm mới một hóa đơn nhập hàng với product_id = 21 và quantity = 15 sử dụng add_goods_receipt_invoice()

<pre> SELECT PRODUCT_STOCK_ID,PRODUCT_ ID, QUANTITY FROM PRODUCT_IN_STOCK WHERE ACTIVE_FLAG = 1;  PRODUCT_STOCK_ID PRODUCT_ID  QUANTITY ----- 61      21      90 81      22      5 </pre>		<p>Nhân viên thứ nhất lấy ra danh sách hàng tồn kho và kiểm tra số lượng của sản phẩm với product_id = 21 lúc này có giá trị là 90.</p>
	<pre> SET TRANSACTION ISOLATION LEVEL READ COMMITTED;  Transaction ISOLATION succeeded. </pre>	<p>Session 2 thiết lập mức cô lập là read committed.</p>
	<pre> SELECT PRODUCT_STOCK_ID,PROD UCT_ID, QUANTITY FROM PRODUCT_IN_STOCK WHERE ACTIVE_FLAG = 1;  PRODUCT_STOCK_ID PRODUCT_ID  QUANTITY ----- 61      21      75 81      22      5 </pre>	<p>Session 2 lấy ra danh sách hàng tồn kho, kiểm tra giá trị Quantity của sản phẩm với product_id lúc này vẫn là 75 vì Session 1 chưa COMMIT</p>
	<pre> DECLARE v_l_info_gr list_info_goods_receipt := list_info_goods_receipt( info_goods_receipt( 21, 5) ); v_type NUMBER; v_user_id NUMBER; v_code VARCHAR2(30); v_invoice_id NUMBER; BEGIN v_type := 0; v_user_id := 1; v_code := 'HD014'; v_invoice_id := 37;  add_goods_receipt_invoice(v_l_ info_gr, v_invoice_id, v_type, v_user_id, v_code); END; </pre>	<p>Session 2 Thêm mới một hóa đơn xuất hàng với cùng một product_id với session 1 product_id = 21 và quantity = 5 sử dụng add_goods_receipt_invoice().</p> <p>Lúc này Session vẫn đang chạy vì Session 1 vẫn đang giữ khóa trên dòng với product_id = 21 của TABLE PRODUCT_IN_STOCK</p>



COMMIT;  Commit complete.		Session 1 COMMIT
	PL/SQL procedure successfully completed.	Session 2 lúc này nhận được khóa và hiển thị kết quả là đã thực hiện xong thao tác thêm hóa đơn xuất hàng.
	COMMIT;  Commit complete.	Session 2 COMMIT
SELECT PRODUCT_STOCK_ID,PRODUCT_ID, QUANTITY FROM PRODUCT_IN_STOCK WHERE ACTIVE_FLAG = 1;  PRODUCT_STOCK_ID PRODUCT_ID    QUANTITY ----- 61        21        85 81        22        5		Lúc này Session 1 lấy lại danh sách hàng tồn kho và nhận ra số lượng tồn kho của sản phẩm với product_id = 21 đã thay đổi. Số lượng lúc này đã thay đổi thành 85(giá trị trước đó là 90).

### c. Nguyên nhân và giải pháp

Vấn đề xảy ra: Transaction 1 thêm một hóa đơn nhập hàng và thực hiện truy vấn lấy ra danh sách hàng tồn kho để kiểm tra số lượng mới của sản phẩm được nhập. Ngay lúc đó Transaction 2 thực hiện thêm một hóa đơn xuất hàng cho cùng một sản phẩm ở Transaction 1. Cuối cùng Transaction 1 thực hiện truy vấn lấy lại danh sách thì thấy dữ liệu đã bị thay đổi.

Nguyên nhân: Transaction 1 thiết lập mức cô lập là READ COMMITTED nên mỗi lần truy vấn nó sẽ đọc dữ liệu đã được COMMIT từ CSDL mặc dù những dữ liệu này tác động trên cùng một đơn vị dữ liệu.

Giải pháp: Thiết lập mức cô lập là SERIALIZABLE cho Transaction 1 thay cho mức cô lập là READ COMMITTED sử dụng câu lệnh “set transaction isolation level serializable”.

#### 5.4.2. Phantom read

Khái niệm: Một transaction chạy một truy vấn để lấy danh sách các hàng với điều kiện (Where) và thấy rằng một transaction khác đã thêm một hàng thỏa với điều kiện như trên.

a. Mô tả tình huống: Trong khi nhân viên thứ nhất đang xem có bao nhiêu đơn nhập hàng hôm nay thì nhân viên thứ 2 thêm một đơn nhập hàng và cập nhập vào hệ thống, lúc này nhân viên thứ nhất lập tổng hóa đơn nhập hàng trong ngày thì thấy dữ liệu đã bị thay đổi.

Store procedure được sử dụng: add\_goods\_receipt\_invoice()

Function được sử dụng: get\_amount()

Trigger được sử dụng: update\_quantity

b. Mô tả Phantom read

Session 1	Session 2	Explanation
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;  Transaction ISOLATION succeeded.	No action	Session 1 thiết lập mức cô lập là READ COMMITTED.
SELECT INVOICE_ID, USER_ID, PRICE FROM INVOICE WHERE TYPE = 1 AND ACTIVE_FLAG = 1 AND TRUNC(UPDATE_DATE) = '26-JUN-20';  INVOICE_ID    USER_ID    PRICE ----- 34            1            1350000 38            1            1350000 33            1            1350000		Session 1 thực hiện truy vấn lấy tất cả hóa đơn nhập hàng trong ngày 26-Jun-20

No action	<pre> DECLARE   v_l_info_gr list_info_goods_receipt := list_info_goods_receipt(   info_goods_receipt( 24, 5),   info_goods_receipt( 26, 3 ) ); v_type NUMBER; v_user_id NUMBER; v_code VARCHAR2(30); v_invoice_seq NUMBER(5, 0); BEGIN   v_type := 1;   v_user_id := 1;   v_code := 'HD016';   --GET INVOICE_SEQ NEXT   VALUE   SELECT INVOICE_SEQ.NEXTVAL   INTO v_invoice_seq FROM DUAL;    add_goods_receipt_invoice(v_l_info_gr,   v_invoice_seq, v_type, v_user_id,   v_code); END;  PL/SQL procedure successfully completed.</pre>	Session 2 Thêm mới một hóa đơn nhập hàng cũng vào ngày 26-Jun-20 sử dụng add_goods_receipt_invoice()																		
No action	<pre> COMMIT;  Commit complete.</pre>	Transaction 2 COMMIT																		
<pre> SELECT INVOICE_ID, USER_ID, PRICE FROM INVOICE WHERE TYPE = 1 AND ACTIVE_FLAG = 1 AND TRUNC(UPDATE_DATE) = '26-JUN-20';</pre> <table> <thead> <tr> <th>INVOICE_ID</th><th>USER_ID</th><th>PRICE</th></tr> <tr> <th>-----</th><th>-----</th><th>-----</th></tr> </thead> <tbody> <tr> <td>42</td><td>1</td><td>1245000</td></tr> <tr> <td>34</td><td>1</td><td>1350000</td></tr> <tr> <td>38</td><td>1</td><td>1350000</td></tr> <tr> <td>33</td><td>1</td><td>1350000</td></tr> </tbody> </table>	INVOICE_ID	USER_ID	PRICE	-----	-----	-----	42	1	1245000	34	1	1350000	38	1	1350000	33	1	1350000	No action	Transaction 1 thực hiện lại truy vấn lấy tất cả hóa đơn nhập hàng trong ngày 26-Jun-20 thì thấy dữ liệu đã có thêm một dòng dữ liệu mới.
INVOICE_ID	USER_ID	PRICE																		
-----	-----	-----																		
42	1	1245000																		
34	1	1350000																		
38	1	1350000																		
33	1	1350000																		

#### d. Nguyên nhân và giải pháp

Vấn đề xảy ra: Transaction 1 thực hiện truy vấn lấy ra danh sách đơn hàng trong ngày, ngay sau đó Transaction 2 thực hiện thêm một hóa đơn nhập hàng. Cuối cùng

Transaction 1 thực hiện truy vấn lấy lại danh sách đơn hàng nhập thì thấy dữ liệu đã có thêm một hóa đơn mới.

Nguyên nhân: Transaction 1 thiết lập mức cô lập là READ COMMITED nên mỗi lần truy vấn nó sẽ đọc dữ liệu đã được COMMIT từ CSDL.

Giải pháp: Thiết lập mức cô lập là SERIALIZABLE cho Transaction 1 thay cho mức cô lập là READ COMMITED sử dụng câu lệnh “set transaction isolation level serializable”.

#### Mô tả giải pháp Phantom read

Session 1	Session 2	Explanation															
<b>SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;</b>  Transaction ISOLATION succeeded.	No action	Session 1 thiết lập mức cô lập là SERIALIZABLE.															
<b>SELECT INVOICE_ID, USER_ID, PRICE</b> <b>FROM INVOICE</b> <b>WHERE TYPE = 1 AND</b> <b>ACTIVE_FLAG = 1</b> <b>AND TRUNC(UPDATE_DATE) =</b> <b>'26-JUN-20';</b>  <table> <tr> <th>INVOICE_ID</th><th>USER_ID</th><th>PRICE</th></tr> <tr> <td>-----</td><td>-----</td><td>-----</td></tr> <tr> <td>34</td><td>1</td><td>1350000</td></tr> <tr> <td>38</td><td>1</td><td>1350000</td></tr> <tr> <td>33</td><td>1</td><td>1350000</td></tr> </table>	INVOICE_ID	USER_ID	PRICE	-----	-----	-----	34	1	1350000	38	1	1350000	33	1	1350000		Session 1 thực hiện truy vấn lấy tất cả hóa đơn nhập hàng trong ngày 26-Jun-20
INVOICE_ID	USER_ID	PRICE															
-----	-----	-----															
34	1	1350000															
38	1	1350000															
33	1	1350000															
No action	<b>DECLARE</b> v_l_info_gr list_info_goods_receipt := list_info_goods_receipt( info_goods_receipt( 24, 5), info_goods_receipt( 26, 3 ) ); v_type NUMBER; v_user_id NUMBER; v_code VARCHAR2(30); v_invoice_seq NUMBER(5, 0); <b>BEGIN</b> v_type := 1; v_user_id := 1; v_code := 'HD016'; --GET INVOICE_SEQ NEXT VALUE	Session 2 Thêm mới một hóa đơn nhập hàng cũng vào ngày 26-Jun-20 sử dụng add_goods_receipt_invoice()															

	SELECT INVOICE_SEQ.NEXTVAL INTO v_invoice_seq FROM DUAL;  add_goods_receipt_invoice(v_l_info_gr, v_invoice_seq, v_type, v_user_id, v_code); END;  PL/SQL procedure successfully completed.																
No action	COMMIT;  Commit complete.	Transaction 2 COMMIT															
SELECT INVOICE_ID, USER_ID, PRICE FROM INVOICE WHERE TYPE = 1 AND ACTIVE_FLAG = 1 AND TRUNC(UPDATE_DATE) = '26-JUN-20';  <table> <thead> <tr> <th>INVOICE_ID</th><th>USER_ID</th><th>PRICE</th></tr> <tr> <th>-----</th><th>-----</th><th>-----</th></tr> </thead> <tbody> <tr> <td>34</td><td>1</td><td>1350000</td></tr> <tr> <td>38</td><td>1</td><td>1350000</td></tr> <tr> <td>33</td><td>1</td><td>1350000</td></tr> </tbody> </table>	INVOICE_ID	USER_ID	PRICE	-----	-----	-----	34	1	1350000	38	1	1350000	33	1	1350000	No action	Transaction 1 thực hiện lại truy vấn lấy tất cả hóa đơn nhập hàng trong ngày 26-Jun-20 và thấy dữ liệu không thay đổi. Vấn đề phantom read đã được giải quyết.
INVOICE_ID	USER_ID	PRICE															
-----	-----	-----															
34	1	1350000															
38	1	1350000															
33	1	1350000															

#### 5.4.3. Lost update

- Mô tả tình huống: Nhân viên thứ nhất cập nhập giá tiền cho sản phẩm với product\_id là 25. Cùng lúc đó nhân viên thứ hai cũng cập nhập giá tiền cho sản phẩm với product\_id là 25. Giá trị cập nhập của nhân viên thứ nhất sẽ bị mất vì giá trị của sản phẩm lúc này sẽ là giá trị cập nhập của nhân viên thứ 2.
- Mô tả Lost update

Session 1	Session 2	Explanation
SELECT PRODUCT_INFO_ID, CODE, NAME, PRICE FROM PRODUCT_INFO WHERE ACTIVE_FLAG = 1 AND PRODUCT_INFO_ID IN (25, 26, 41);	No action	Session 1 truy vấn CODE, NAME, PRICE cho những sản phẩm có ID là 25, 26, 41.

PRODUCT_INFO_ID CODE NAME PRICE ----- ----- ----- 25 XMNS11 Xi măng Nghi Sơn 150000 26 XMXT11 Xi măng Xuân Thành 140000		
UPDATE PRODUCT_INFO SET PRICE = 145000 WHERE PRODUCT_INFO_ID = 25;	No action	Session 1 thực hiện update PRICE cho sản phẩm có ID là 25. Isolation level mặc định là READ COMMITTED.
No action	SET TRANSACTION ISOLATION LEVEL READ COMMITTED;	Bắt đầu Session 2 và thiết lập Isolation level là READ COMMITTED.
No action	SELECT PRODUCT_INFO_ID, CODE, NAME, PRICE FROM PRODUCT_INFO WHERE ACTIVE_FLAG = 1 AND PRODUCT_INFO_ID IN (25, 26, 41);  PRODUCT_INFO_ID CODE NAME PRICE ----- ----- ----- 25 XMNS11 Xi măng Nghi Sơn 150000 26 XMXT11 Xi măng Xuân Thành 140000	Session 2 truy vấn CODE, NAME, PRICE cho những sản phẩm có ID là 25, 26, 41. Giá trị PRICE của sản phẩm với ID là 25 vẫn chưa được cập nhập do Transaction 1 chưa được COMMIT.
No action	UPDATE PRODUCT_INFO SET PRICE = 170000 WHERE PRODUCT_INFO_ID = 26;	Session 2 thực hiện update PRICE cho sản phẩm có ID là 26 thành công do Transaction 1 chỉ lock dòng có ID là 25.

INSERT INTO PRODUCT_INFO(CATEGORY_ID, CODE, NAME) VALUES (62, 'FGH11A', 'Xi mang Hoa Tho');	No action	Transaction 1 insert một dòng cho PRODUCT_INFO với ID là 41, nhưng không thực hiện COMMIT.
No action	SELECT PRODUCT_INFO_ID, CODE, NAME, PRICE FROM PRODUCT_INFO WHERE ACTIVE_FLAG = 1 AND PRODUCT_INFO_ID IN (25, 26, 41);  PRODUCT_INFO_ID CODE NAME PRICE ----- -----  25 XMNS11 Xi măng Nghi Sơn 150000 26 XMXT11 Xi măng Xuân Thành 170000	Session 2 truy vấn CODE, NAME, PRICE cho những sản phẩm có ID là 25, 26,41. Giá trị PRICE của sản phẩm với ID là 26 đã được cập nhật. Lúc này vẫn chưa thấy dữ liệu cập nhật của sản phẩm với ID là 25 hay ID là 41.
No action	UPDATE PRODUCT_INFO SET PRICE = 175000 WHERE PRODUCT_INFO_ID = 25;	Transaction 2 thực hiện update PRICE cho sản phẩm có ID là 25, dòng này đã bị khóa trên Transaction 1. Lúc này Transaction 2 phải chờ Transaction 1 thực hiện xong.
COMMIT;	No action	Transaction 1 COMMIT thành công, kết thúc transaction.
No action	1 row updated.	Transaction 2 nhận được khóa và thực hiện update PRICE cho sản phẩm có ID là 25.
No action	SELECT PRODUCT_INFO_ID, CODE, NAME, PRICE FROM PRODUCT_INFO WHERE ACTIVE_FLAG = 1;	Transaction 2 truy vấn CODE, NAME, PRICE cho những sản phẩm có ID là 25, 26,41. Transaction 2 thấy được giá trị đã update của sản phẩm với ID là 25 và sản

	PRODUCT_INFO_ID CODE NAME PRICE ----- ----- ----- 25 XMNS11 Xi măng Nghi Sơn 175000 26 XMXT11 Xi măng Xuân Thành 170000 41 FGH11A Xi măng Hoa Tho	phẩm mới được thêm vào với ID là 41.
No action	COMMIT;	Transaction 2 COMMIT thành công, kết thúc transaction.
SELECT PRODUCT_INFO_ID, CODE, NAME, PRICE FROM PRODUCT_INFO WHERE ACTIVE_FLAG = 1;		Transaction 1 truy vấn CODE, NAME, PRICE cho những sản phẩm có ID là 25, 26, 41. Giá trị PRICE của sản phẩm với ID = 25 bây giờ là 175000. Việc update PRICE cho sản phẩm với ID = 21 của Transaction 1 đã bị mất.

### c. Nguyên nhân và giải pháp

Vấn đề xảy ra: Transaction 1 thực hiện cập nhập giá tiền cho sản phẩm với product\_id, ngay lúc đó Transaction 2 cũng thực hiện cập nhập giá tiền cho sản phẩm với cùng một sản phẩm ở Transaction 1. Giá tiền cập nhập được cho sản phẩm sẽ là giá trị cập nhập ở Transaction 2 vì thế giá trị cập nhập ở Transaction 1 đã bị mất.

Nguyên nhân: Transaction 1 thực hiện câu lệnh UPDATE lên bảng PRODUCT\_INFO nhưng Transaction 2 vẫn có thể tác động là thực câu lệnh UPDATE lên bảng PRODUCT\_INFO.

Giải pháp: Thiết lập khóa trên bảng PRODUCT\_INFO sử dụng câu lệnh “LOCK TABLE PRODUCT\_INFO IN SHARE MODE” để khi một transaction thực hiện ghi trên bảng PRODUCT\_INFO thì những transaction khác chỉ được đọc mà không được ghi.



## Mô tả giải pháp

Session 1	Session 2	Explanation
LOCK TABLE PRODUCT_INFO IN SHARE MODE;		
SELECT PRODUCT_INFO_ID, CODE, NAME, PRICE FROM PRODUCT_INFO WHERE ACTIVE_FLAG = 1 AND PRODUCT_INFO_ID IN (25, 26, 41);  PRODUCT_INFO_ID CODE NAME PRICE ----- ----- ----- 25 XMNS11 Xi măng Nghi Sơn 150000 26 XMXT11 Xi măng Xuân Thành 140000	No action	Session 1 truy vấn CODE, NAME, PRICE cho những sản phẩm có ID là 25, 26, 41.
UPDATE PRODUCT_INFO SET PRICE = 145000 WHERE PRODUCT_INFO_ID = 25;	No action	Session 1 thực hiện update PRICE cho sản phẩm có ID là 25. Isolation level mặc định là READ COMMITTED.
No action	SET TRANSACTION ISOLATION LEVEL READ COMMITTED;	Bắt đầu Session 2 và thiết lập Isolation level là READ COMMITTED.
No action	SELECT PRODUCT_INFO_ID, CODE, NAME, PRICE FROM PRODUCT_INFO WHERE ACTIVE_FLAG = 1 AND PRODUCT_INFO_ID IN (25, 26, 41);  PRODUCT_INFO_ID CODE NAME PRICE ----- ----- -----	Session 2 truy vấn CODE, NAME, PRICE cho những sản phẩm có ID là 25, 26, 41.  Giá trị PRICE của sản phẩm với ID là 25 vẫn chưa được cập nhập do Transaction 1 chưa được COMMIT.

	25 XMNS11 Xi măng Nghi Sơn 150000 26 XMXT11 Xi măng Xuân Thành 140000	
No action	UPDATE PRODUCT_INFO SET PRICE = 170000 WHERE PRODUCT_INFO_ID = 26;	Session 2 thực hiện update PRICE cho sản phẩm có ID là 26 thành công do Transaction 1 chỉ lock dòng có ID là 25.
INSERT INTO PRODUCT_INFO(CATEGORY_ID, CODE, NAME) VALUES (62, 'FGH11A', 'Xi măng Hoa Tho');	No action	Transaction 1 insert một dòng cho PRODUCT_INFO với ID là 41, nhưng không thực hiện COMMIT.
No action	SELECT PRODUCT_INFO_ID, CODE, NAME, PRICE FROM PRODUCT_INFO WHERE ACTIVE_FLAG = 1 AND PRODUCT_INFO_ID IN (25, 26, 41);  PRODUCT_INFO_ID CODE NAME PRICE ----- - ----- ----- 25 XMNS11 Xi măng Nghi Sơn 150000 26 XMXT11 Xi măng Xuân Thành 170000	Session 2 truy vấn CODE, NAME, PRICE cho những sản phẩm có ID là 25, 26,41. Giá trị PRICE của sản phẩm với ID là 26 đã được cập nhập. Lúc này vẫn chưa thấy dữ liệu cập nhập của sản phẩm với ID là 25 hay ID là 41.
No action	UPDATE PRODUCT_INFO SET PRICE = 175000 WHERE PRODUCT_INFO_ID = 25;	Transaction 2 không thực hiện được bởi vì T1 đã giữ khóa.
COMMIT;	No action	Transaction 1 COMMIT thành công, kết thúc transaction.

SELECT PRODUCT_INFO_ID, CODE, NAME, PRICE FROM PRODUCT_INFO WHERE ACTIVE_FLAG = 1;  PRODUCT_INFO_ID CODE NAME PRICE ----- ----- ----- 25 XMNS11 Xi măng Nghi Sơn 145000 26 XMXT11 Xi măng Xuân Thành 140000		Transaction 1 truy vấn CODE, NAME, PRICE cho những sản phẩm có ID là 25, 26, 41. Giá trị PRICE của sản phẩm với ID = 25 bây giờ là 145000. Transaction 1 cập nhật thành công.
---	--	--

## CHƯƠNG 6. THIẾT KẾ GIAO DIỆN

### 6.1. Danh sách các màn hình

#### 6.1.1. Đăng nhập

STT	Màn hình	Chức năng
1	Login	Đăng nhập

#### 6.1.2. Menu bảng điều khiển

STT	Màn hình	Chức năng
1	Sản phẩm	Hiện menu Sản phẩm
2	Kho	Hiện menu Kho
3	Quản lý	Hiện menu Quản lý

#### 6.1.3. Category

STT	Màn hình	Chức năng
1	Add Category	Thêm loại sản phẩm
2	Delete Category	Xóa loại sản phẩm
3	Edit Category	Sửa loại sản phẩm
4	View Category	Xem loại sản phẩm

#### 6.1.4. Sản phẩm

STT	Màn hình	Chức năng
1	Add ProductInfo	Thêm loại sản phẩm
2	Delete ProductInfo	Xóa loại sản phẩm
3	Edit ProductInfo	Sửa loại sản phẩm

4	Product In Stock List	Hiển thị số lượng sản phẩm có trong kho
5	Product Info List	Hiển thị các sản phẩm
6	View ProductInfo	Hiện thị thông tin chi tiết sản phẩm

#### 6.1.5. Nhập kho

STT	Màn hình	Chức năng
1	Goods Receipt	Hiện danh sách nhập kho
2	Add Goods Receipt	Nhập kho một lô sản phẩm
3	View Goods	Xem chi tiết nhập kho

#### 6.1.6. Xuất kho

STT	Màn hình	Chức năng
1	Goods Issue	Hiện danh sách xuất kho
2	Add Goods Issue	Thêm kho một lô sản phẩm
3	View Goods Issue	Xem chi tiết xuất kho

#### 6.1.7. Sản phẩm trong kho

STT	Màn hình	Chức năng
1	Product In Stock List	Hiện danh sách sản phẩm trong kho

#### 6.1.8. Lịch sử kho

STT	Màn hình	Chức năng
1	History List	Hiện lịch sử xuất / nhập kho

#### 6.1.9. User

STT	Màn hình	Chức năng
-----	----------	-----------

1	User List	Hiện danh sách người dùng
2	Edit User	Chỉnh sửa thông tin người dùng
3	Pop-up delete records	Xác nhận xóa hoặc hủy xóa người dùng
4	Add Users	Thêm người dùng mới
5	View User	Xem chi tiết thông tin người dùng

#### 6.1.10. Menu

STT	Màn hình	Chức năng
1	Menu List	Hiển thị danh sách menu

#### 6.1.11. Quyền

STT	Màn hình	Chức năng
1	Role List	Phân quyền user

### 6.2. Mô tả các màn hình

#### 6.2.1. Đăng nhập

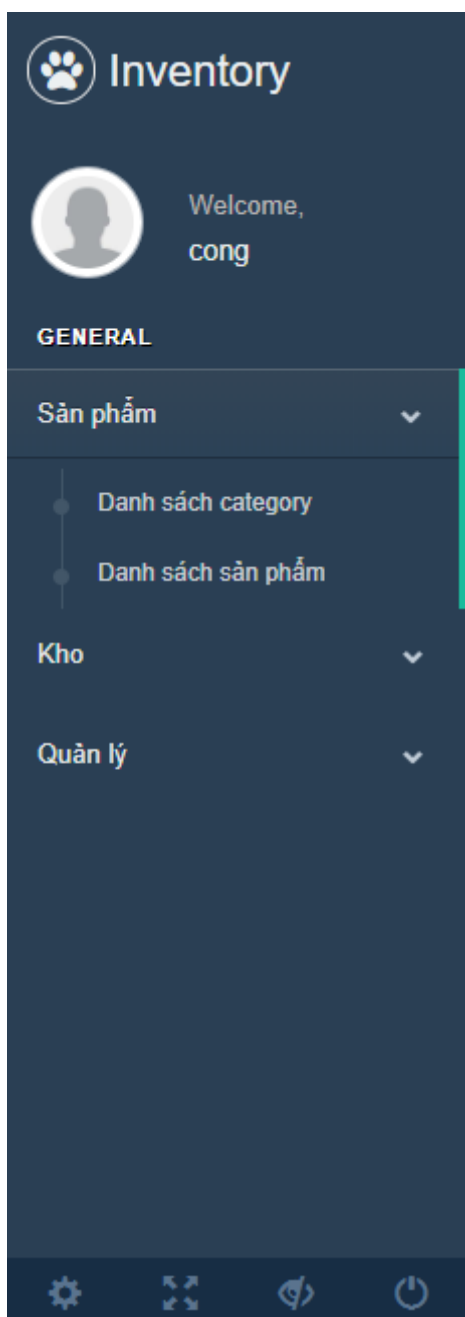
The screenshot displays a web browser window with the URL `localhost:8080/inventory_management_war/login`. The page features a central 'Login Form' with two input fields: the first contains the username 'ngoccong' and the second contains a masked password '\*\*\*\*\*'. A 'Log in' button is positioned below these fields. At the bottom of the page, the text 'Inventory Management System!' is displayed alongside a paw print icon, followed by the copyright notice '©2019 Inventory Management System. Privacy and Terms'.

### Mô tả chi tiết các chức năng:

Người dùng nhập tài khoản và mật khẩu vào hai label username và password, sau đó nhấn button Login để đăng nhập.

#### 6.2.2. Menu bảng điều khiển

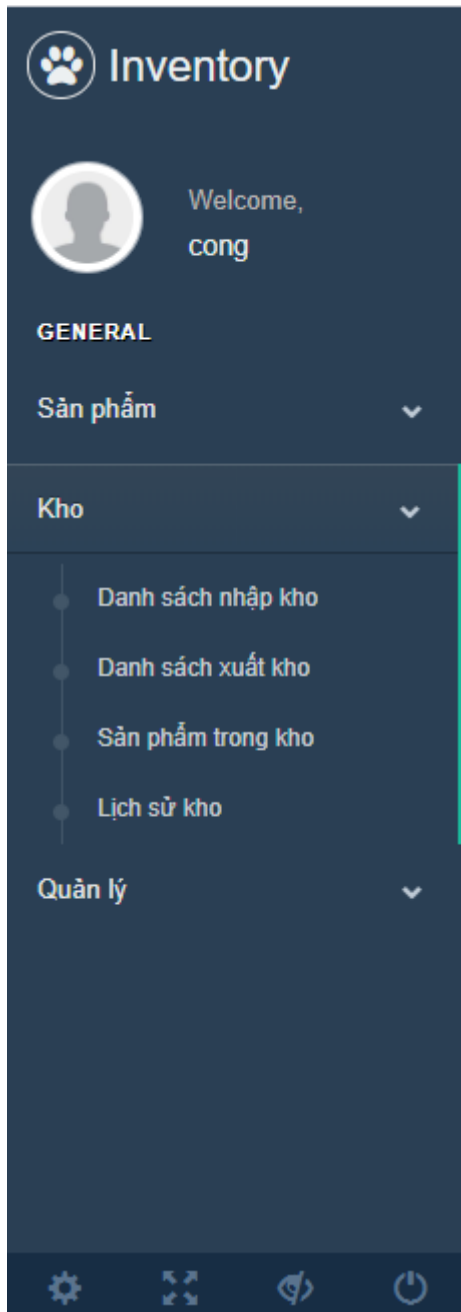
##### a. Màn hình Menu Sản Phẩm



### Mô tả chi tiết các chức năng:

Người dùng khi bấm vào danh mục sản phẩm sẽ hiện ra Danh sách category và Danh sách sản phẩm.

b. Màn hình Menu Kho

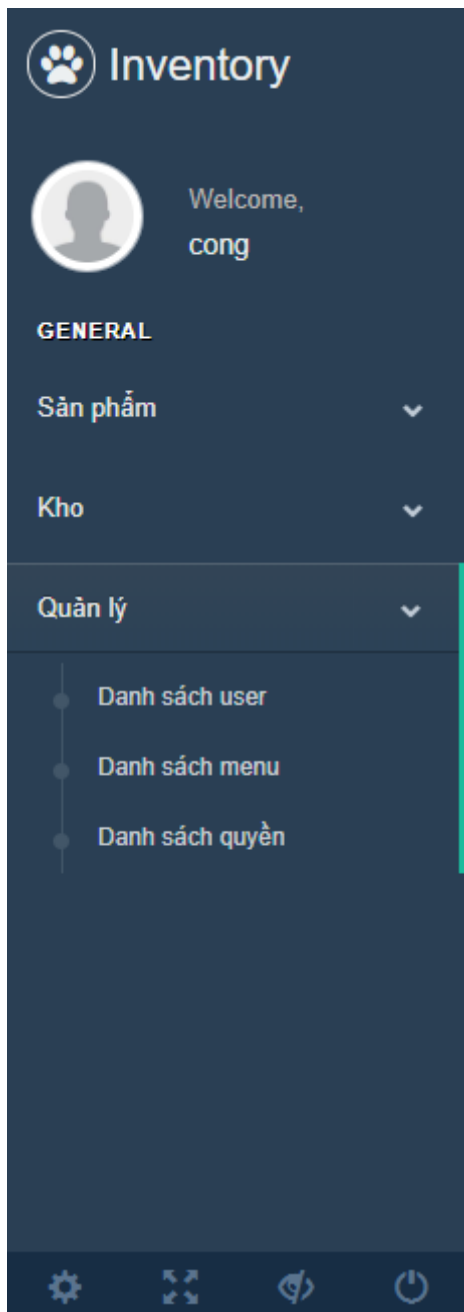


**Mô tả chi tiết các chức năng:**

Người dùng khi bấm vào danh mục Kho sẽ hiện ra Danh sách nhập kho, Danh sách xuất kho, Sản phẩm trong kho, Lịch sử kho.

c. Màn hình Menu quản lý



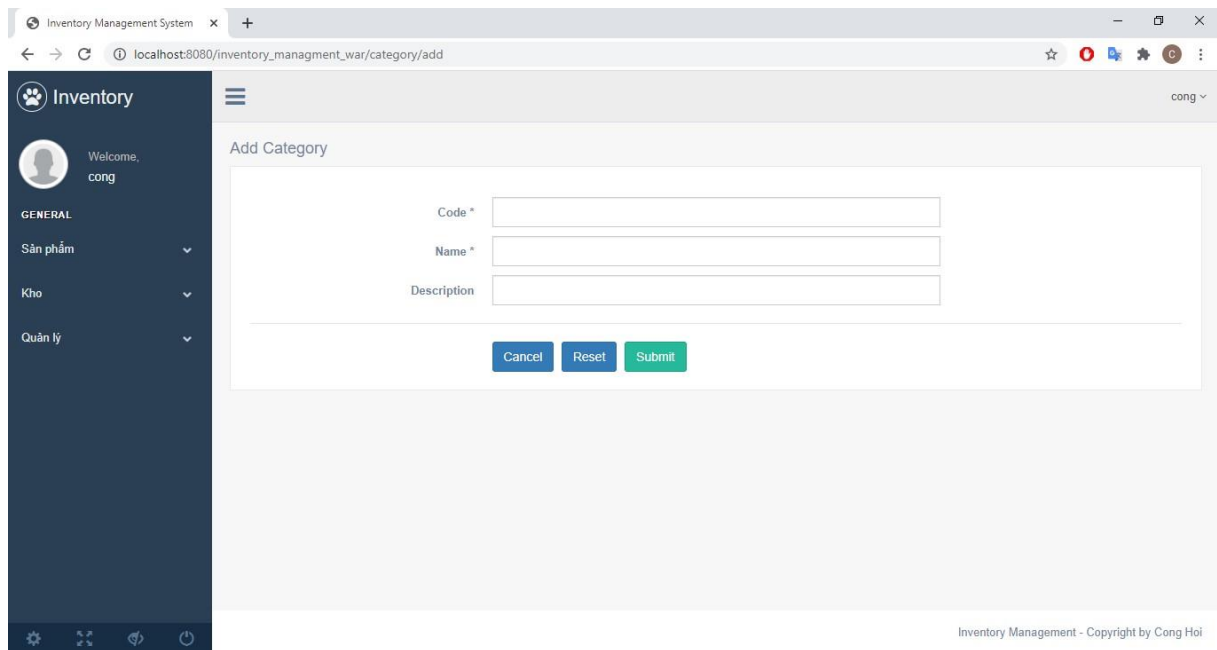


### **Mô tả chi tiết các chức năng:**

Người dùng khi bấm vào danh mục Quản lý sẽ hiển thị Danh sách User, Danh sách Menu, Danh sách quyền.

#### **6.2.3. Category**

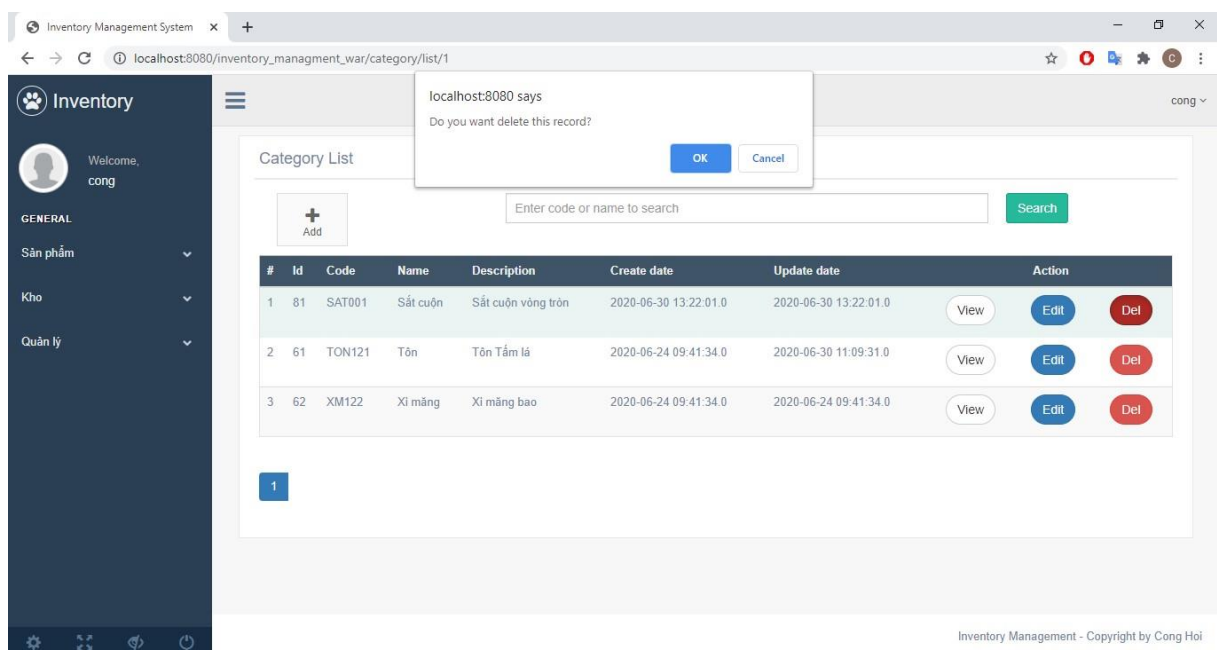
- a. Add Category



### Mô tả chi tiết các chức năng:

Người dùng nhập các thông tin: Code (Mã loại sản phẩm), Name (Tên loại sản phẩm), Description (Mô tả loại sản phẩm) sau đó bấm Submit để thêm sản phẩm, bấm Reset để nhập lại, bấm Cancel để hủy bỏ

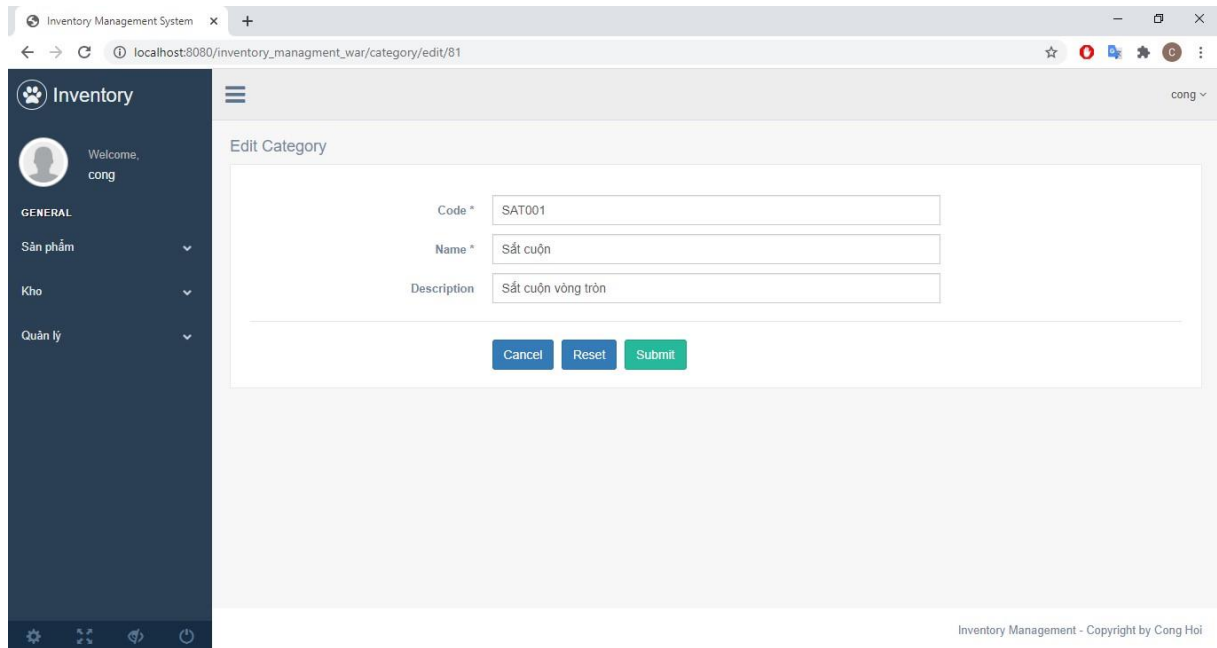
#### b. Delete Category



### Mô tả chi tiết các chức năng:

Màn hình hiển thị thông tin các loại sản phẩm. Người dùng bấm button view để hiển thị thông tin chi tiết loại sản phẩm, bấm Edit để tới màn hình chỉnh sửa thông tin loại sản phẩm, bấm Del để xóa sản phẩm. Ngoài ra người dùng có thể nhập category code hoặc category name trên thanh tìm kiếm sau đó nhấn button Search để tìm loại sản phẩm muốn xóa.

### c. Edit Category

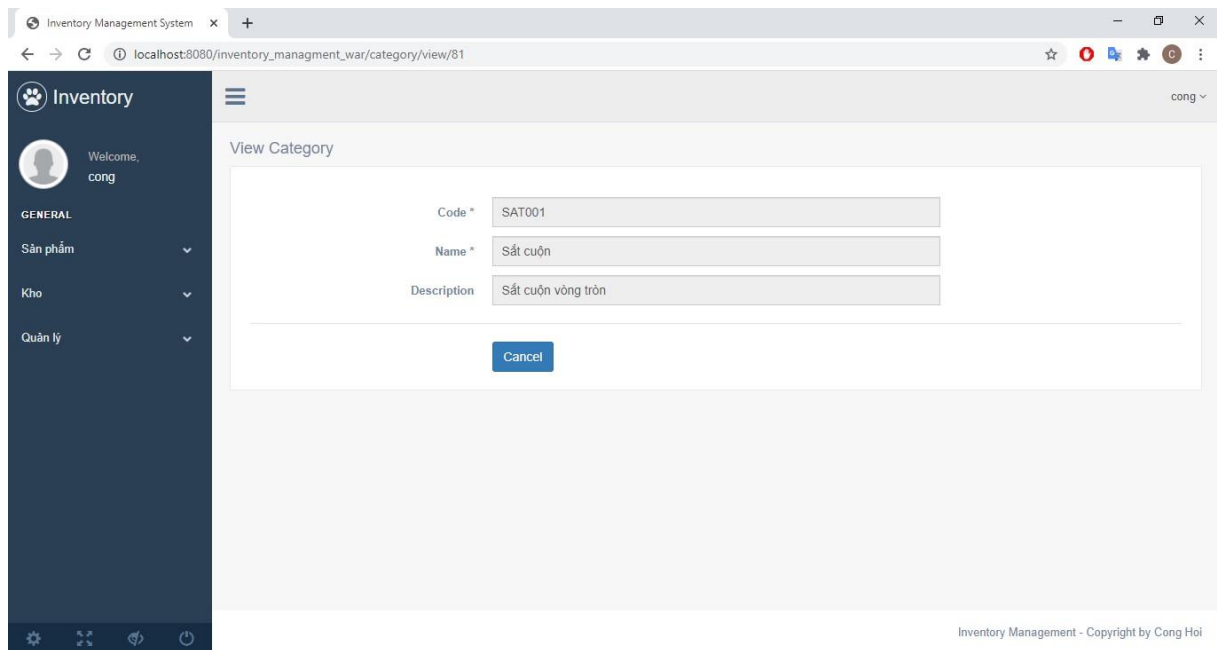


The screenshot shows a web application interface for an Inventory Management System. The main content area is titled 'Edit Category' and contains a form with three input fields: 'Code \*' (containing 'SAT001'), 'Name \*' (containing 'Sắt cuộn'), and 'Description' (containing 'Sắt cuộn vòng tròn'). Below the form are three buttons: 'Cancel', 'Reset', and 'Submit'. The left sidebar shows the 'Inventory' menu with options like 'GENERAL', 'Sản phẩm', 'Kho', and 'Quản lý'. The top bar shows the user 'Welcome, cong' and the system name 'Inventory Management System'.

### Mô tả chi tiết các chức năng:

Người dùng sửa lại thông tin loại sản phẩm bằng cách nhập lại các thông tin Code (Mã loại sản phẩm), Name (Tên loại sản phẩm), Description (Mô tả loại sản phẩm) sau đó nhấn Submit để hoàn tất việc chỉnh sửa. Có thể bấm Reset để làm trắng toàn bộ thông tin sản phẩm.

### d. View Category

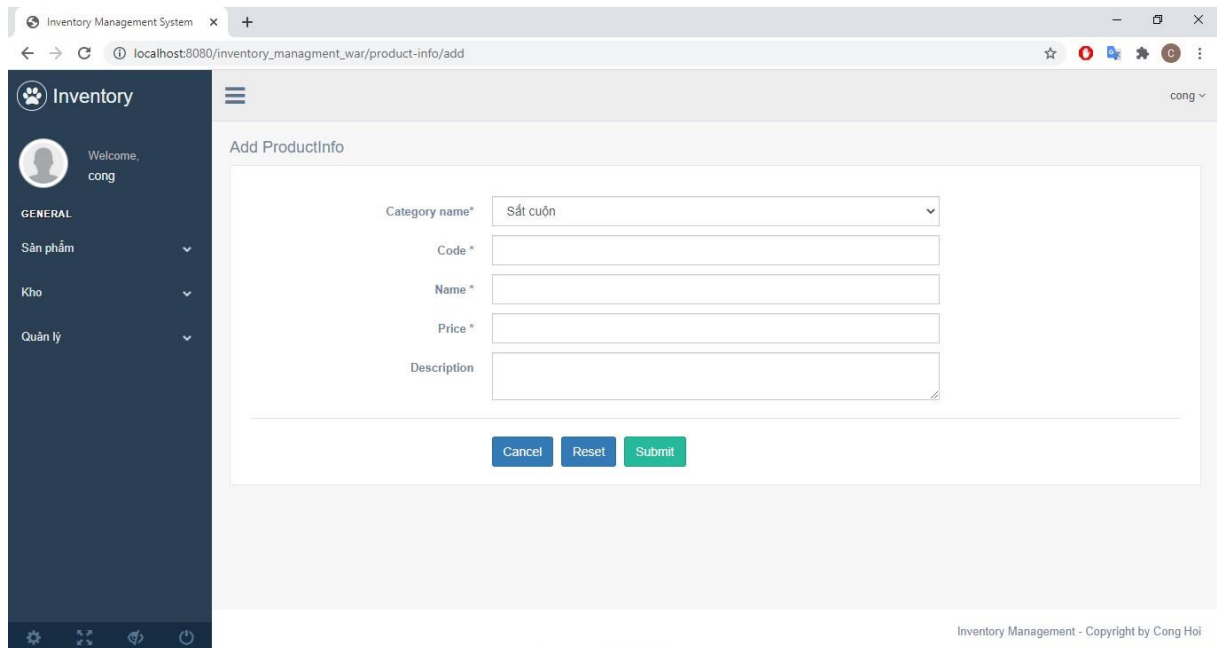


### Mô tả chi tiết các chức năng:

Màn hình hiển thị thông tin chi tiết loại sản phẩm, bao gồm: Code (Mã loại sản phẩm), Name (Tên loại sản phẩm), Description (Mô tả loại sản phẩm).

#### 6.2.4. Sản phẩm

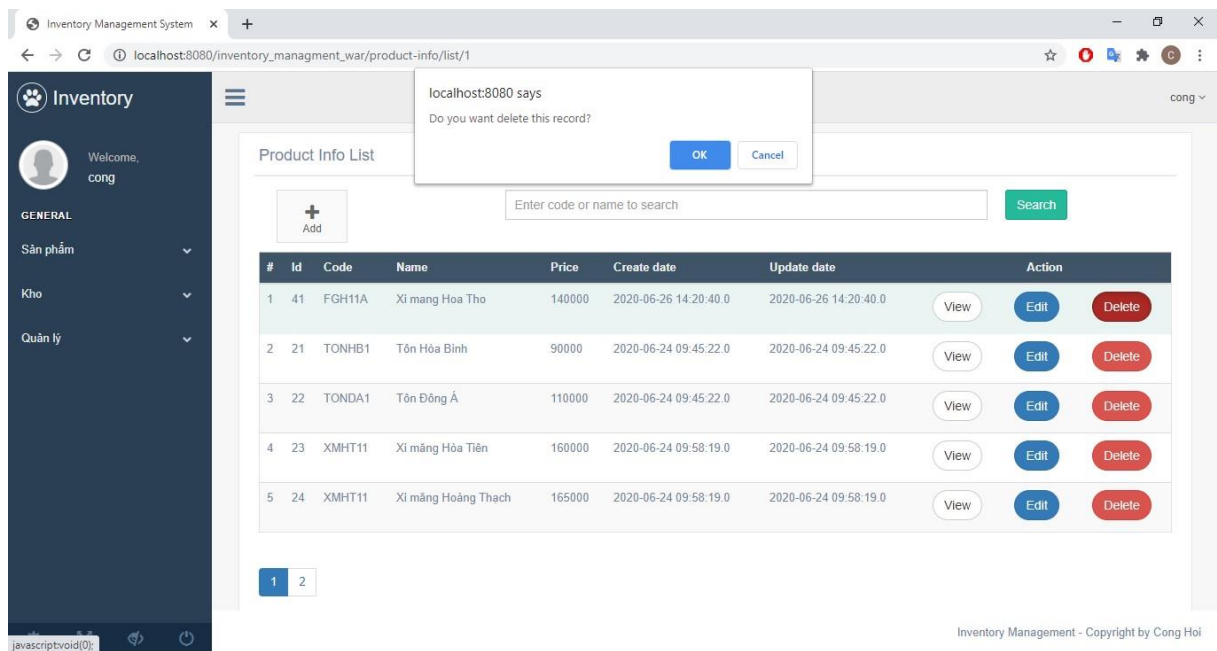
##### a. Add ProductInfo



### Mô tả chi tiết các chức năng:

Người dùng nhập các thông tin sản phẩm: Category name (Loại sản phẩm), Code (Mã sản phẩm), Name (Tên sản phẩm), Price (Giá sản phẩm), Description (Mô tả sản phẩm). Sau đó bấm submit để thêm sản phẩm. Có thể bấm Reset để làm trắng toàn bộ thông tin đã nhập, bấm Cancel để hủy bỏ.

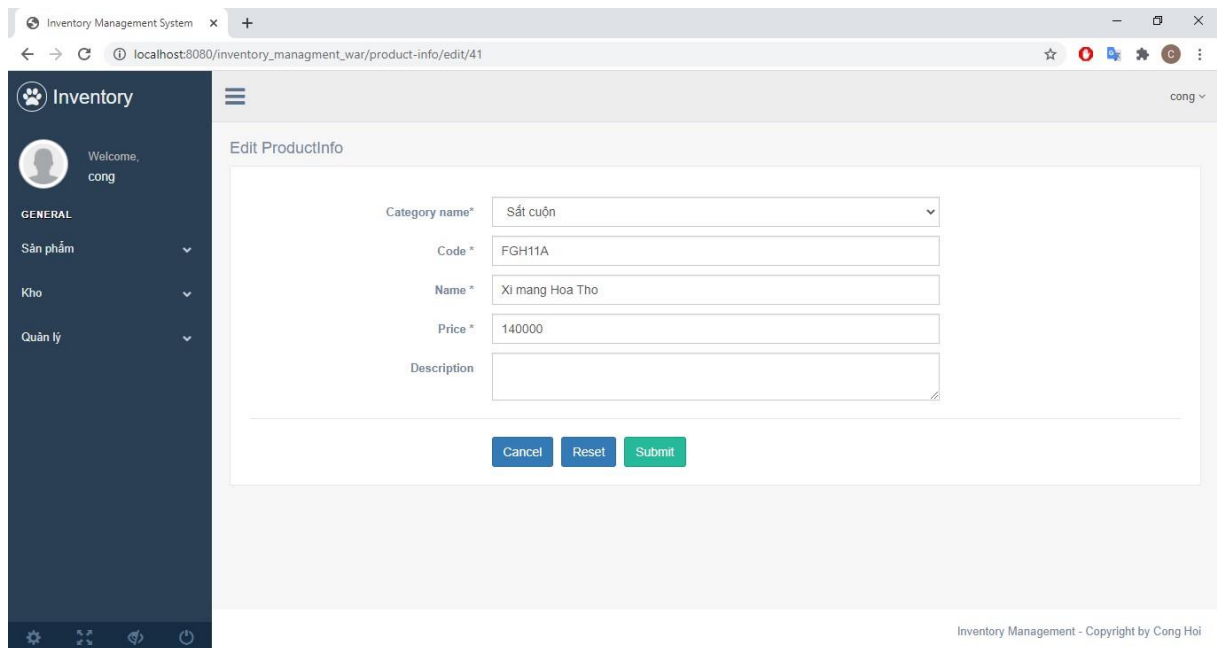
### b. Del ProductInfo



### Mô tả chi tiết các chức năng:

Màn hình hiển thị thông tin các sản phẩm. Người dùng bấm button view để hiển thị thông tin chi sản phẩm, bấm Edit để tới màn hình chỉnh sửa thông tin sản phẩm, bấm Del để xóa sản phẩm. Ngoài ra người dùng có thể nhập code hoặc name trên thanh tìm kiếm sau đó nhấn button Search để tìm sản phẩm muốn xóa.

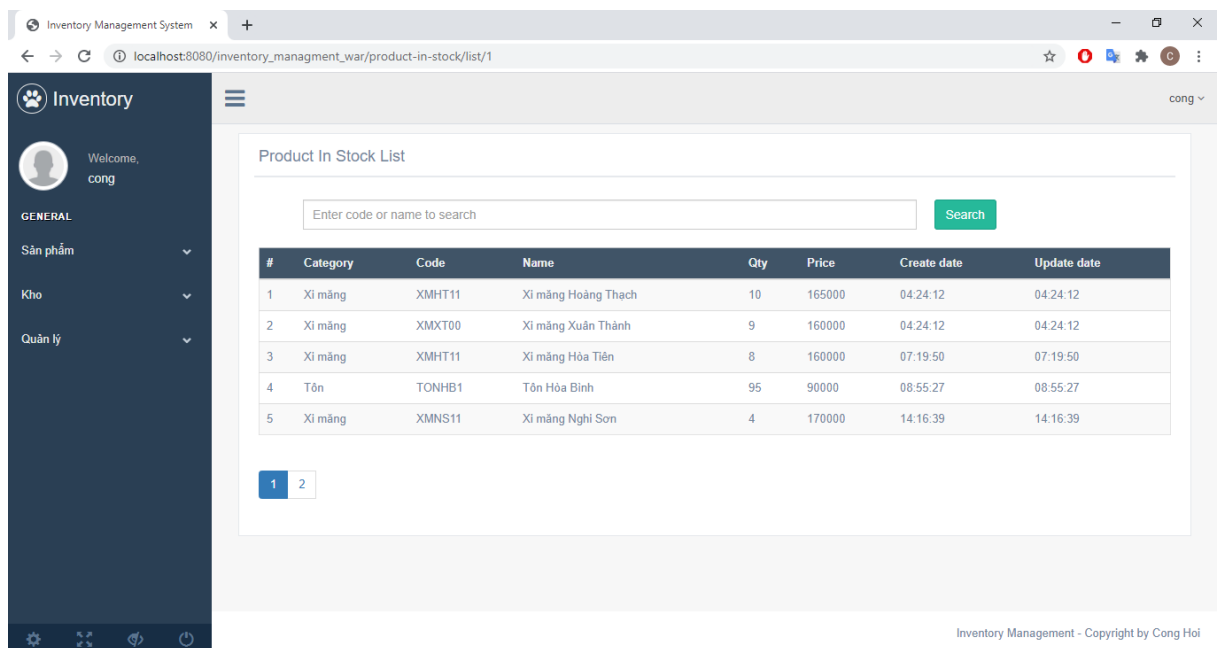
### c. Edit ProductInfo



### Mô tả chi tiết các chức năng:

Người dùng sửa lại thông tin sản phẩm bằng cách nhập lại các thông tin Category name (Tên loại sản phẩm), Code (Mã sản phẩm), Name (Tên sản phẩm), Price (Giá sản phẩm), Description (Mô tả loại sản phẩm) sau đó nhấn Submit để hoàn tất việc chỉnh sửa. Có thể bấm Reset để làm trống toàn bộ thông tin sản phẩm.

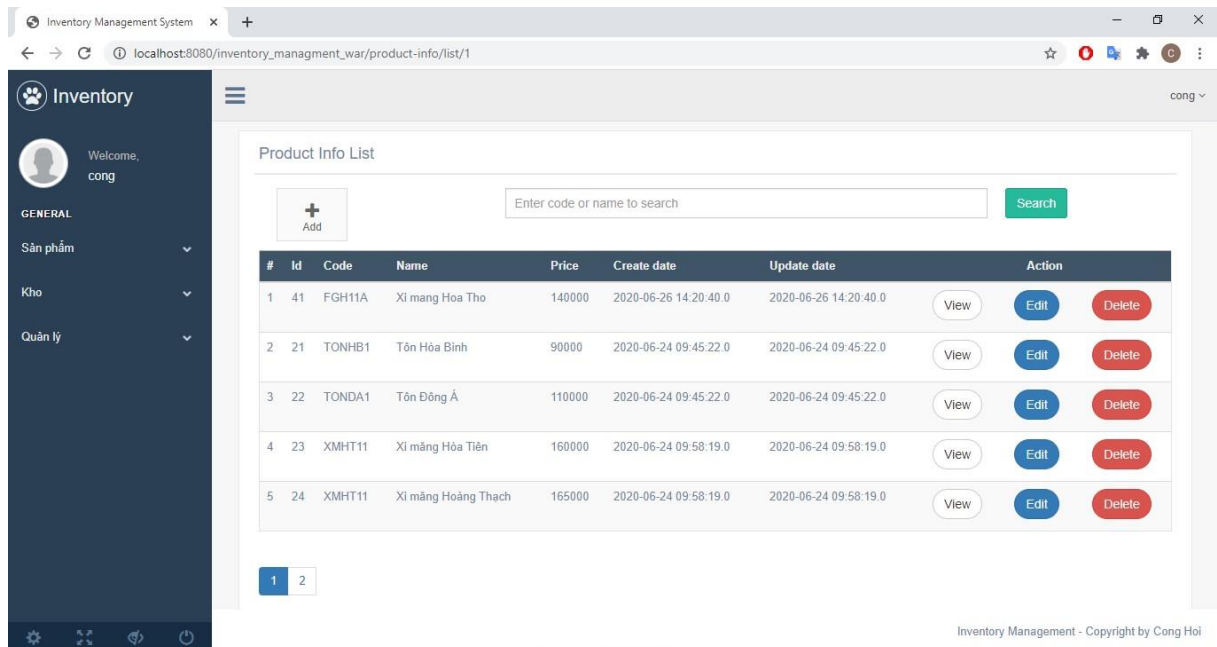
#### d. Product In Stock



### Mô tả chi tiết các chức năng:

Product In Stock là màn hình hiển thị danh sách số lượng từng sản phẩm có trong kho. Người dùng có thể nhập code (Mã sản phẩm) hoặc name (Tên sản phẩm) để xem số lượng một sản phẩm cụ thể trong kho.

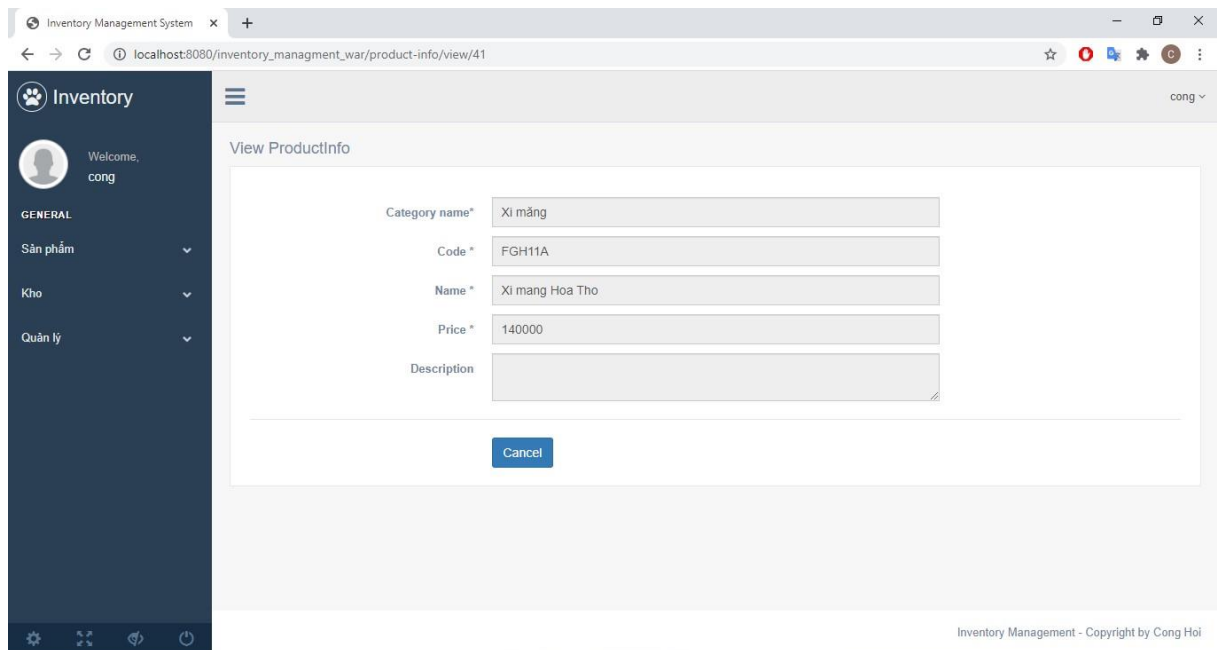
#### e. Product Info List



#### Mô tả chi tiết các chức năng:

Màn hình Product Info List hiển thị danh sách các sản phẩm. Người dùng có thể thao tác trên các sản phẩm như: View (Thông tin chi tiết sản phẩm), Edit (Sửa sản phẩm), Delete (Xóa sản phẩm).

#### f. Product View

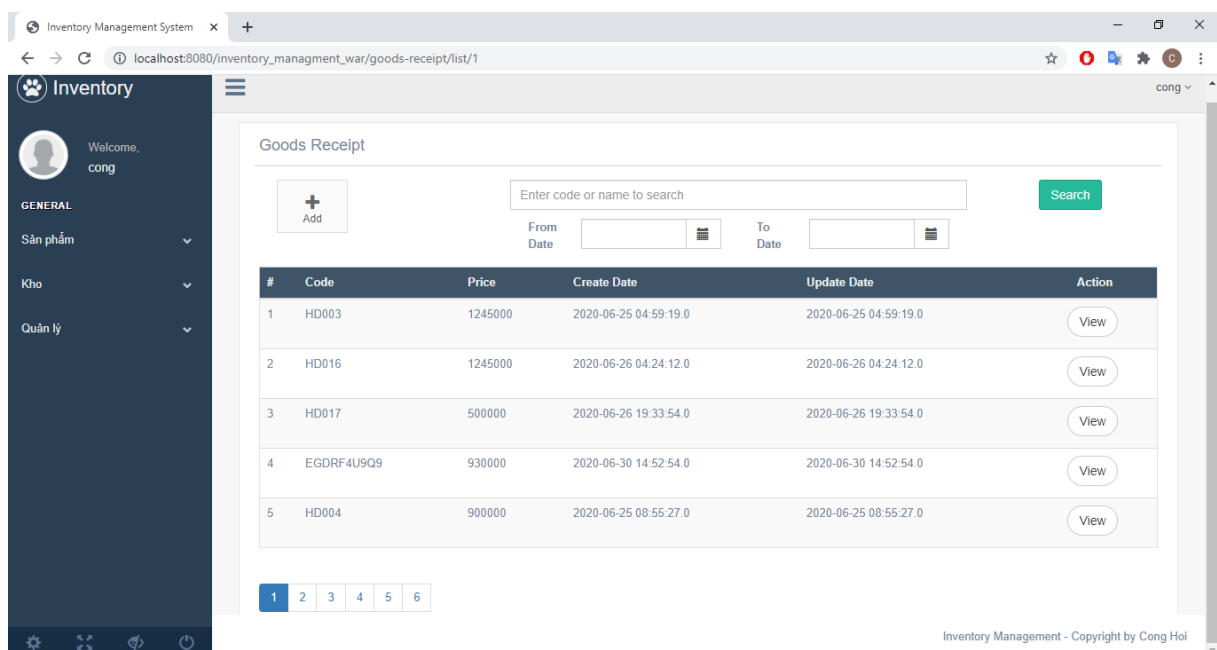


### Mô tả chi tiết các chức năng:

Màn hình hiển thị thông tin chi tiết sản phẩm, bao gồm: Category name (Tên loại sản phẩm), Code (Mã sản phẩm), Name (Tên sản phẩm), Price (Giá sản phẩm), Description (Mô tả sản phẩm).

## 6.2.5. Nhập kho

### a. Màn hình Good Receipt



### Mô tả chi tiết các chức năng:



- Xem danh sách các records nhập kho.
- Dùng bộ lọc (From Date / To Date) để xem danh sách các records nhập kho theo khoản ngày nhất định.
- Tìm kiếm chính xác record nhập kho theo mã Code bằng chức năng Search
- Chuyển đến màn hình Add Goods Receipt bằng cách chọn nút Add
- Chuyển đến màn hình Edit Goods Receipt bằng cách chọn nút View

#### b. Màn hình Add Goods Receipt

The screenshot shows the 'Add Goods Receipt' interface. On the left is a dark sidebar with the 'Inventory' logo and a user profile 'Welcome, cong'. The main area has a header 'Add Goods Receipt' and a form with the following fields:

- Product Code:** A text input field with the placeholder 'Enter product code'.
- Product Name:** A dropdown menu currently showing 'Xi mang Hoa Tho'.
- Quantity:** A text input field with the value '0'.

Below the form is a table with the following data:

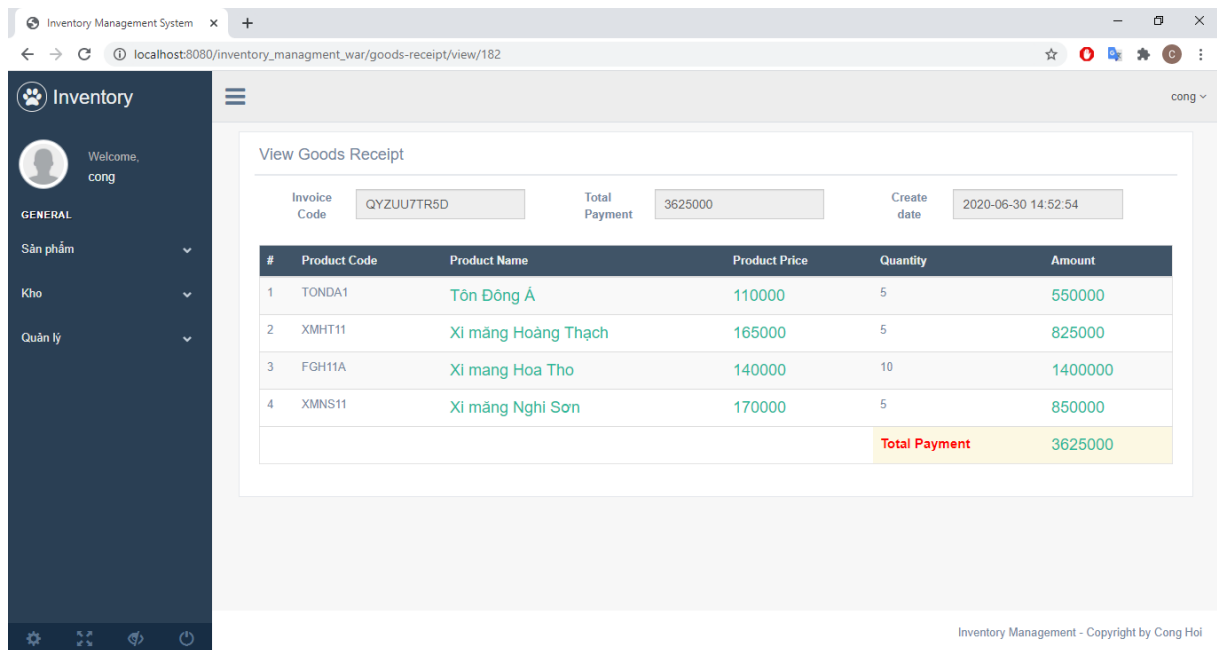
#	Product Code	Product Name	Product Price	Quantity	Amount
1	TONHB1	Tôn Hòa Bình	90000	10	900000
2	TONDA1	Tôn Đông Á	110000	3	330000
3	XMHT11	Xi măng Hòa Tiến	160000	5	800000
<b>Total Payment</b>					<b>2030000</b>

A green 'Add to Invoice' button is positioned at the bottom left of the form area.

#### Mô tả chi tiết các chức năng:

- Chọn sản phẩm trong Product Name, chọn số lượng sản phẩm trong Quantity để thêm sản phẩm vào danh sách sản phẩm trong một record nhập hàng.
- Xem danh sách sản phẩm trong một record nhập hàng vừa thêm.

#### c. Màn hình View Goods Receipt

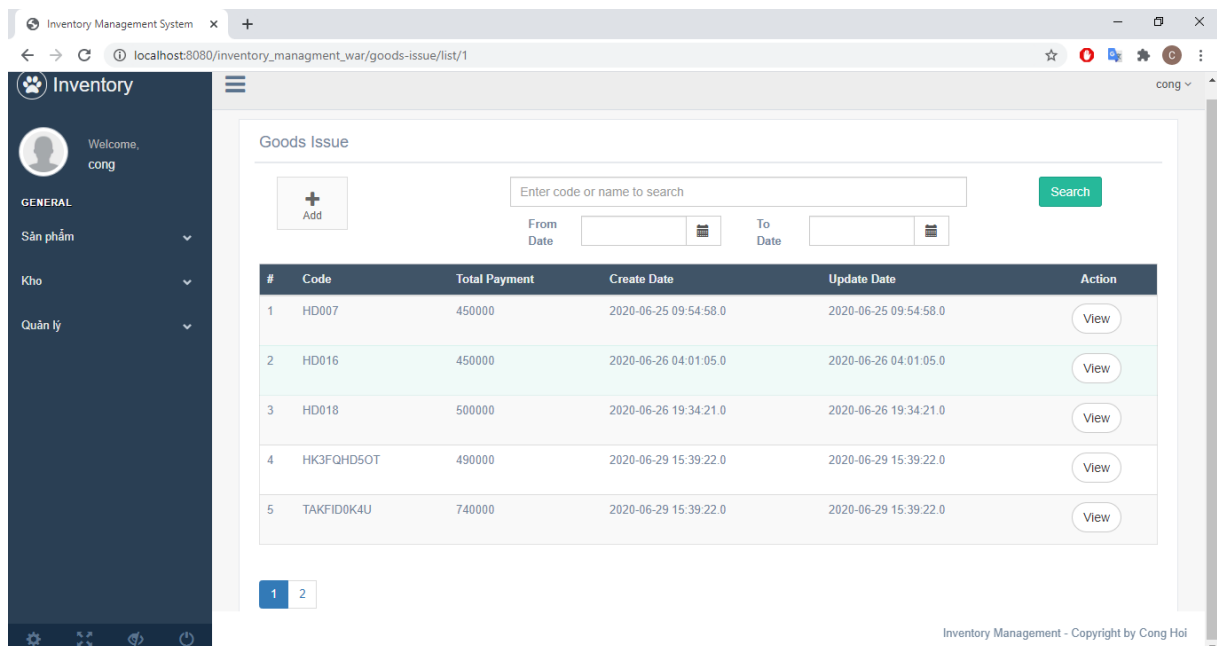


## Mô tả chi tiết các chức năng:

- Xem thông tin chi tiết một record nhập kho.

### 6.2.6. Xuất kho

#### a. Màn hình Good Issue



## Mô tả chi tiết các chức năng:

- Xem danh sách các records xuất kho.

- Dùng bộ lọc (From Date / To Date) để xem danh sách các records xuất kho theo khoảng ngày nhất định.
- Tìm kiếm chính xác record xuất kho theo mã Code bằng chức năng Search
- Chuyển đến màn hình Add Goods Issue bằng cách chọn nút Add
- Chuyển đến màn hình Edit Goods Issue bằng cách chọn nút View

#### b. Add Goods Issue

Inventory Management System

localhost:8080/inventory\_management\_war/goods-issue/add

Welcome, cong

GENERAL

Sản phẩm

Kho

Quản lý

Add Goods Issue

Product Code: Enter product code

Product Name: Tôn Đông Á

Product Price:

Quantity: 0

Add to Invoice

#	Product Code	Product Name	Product Price	Quantity	Amount
1	TONDA1	Tôn Đông Á	110000	10	1100000
2	XMHT11	Xi măng Hòa Tiên	160000	10	1600000
3	XMXT00	Xi măng Xuân Thành	160000	10	1600000
Total Payment					4300000

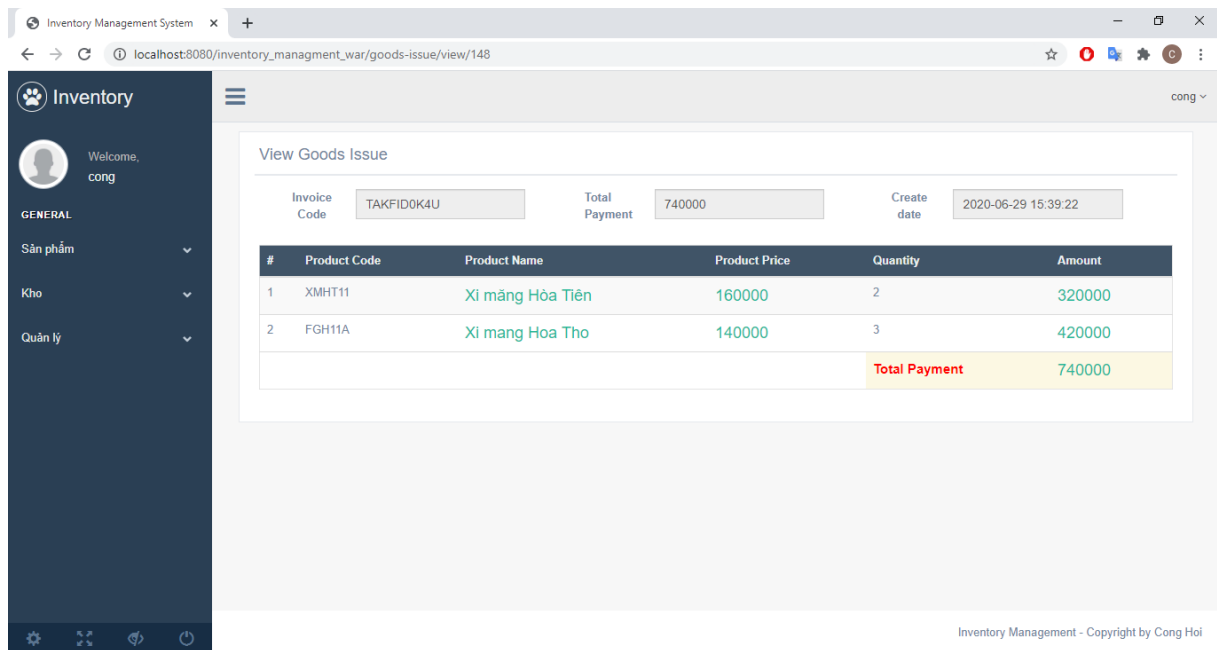
Add to Invoice

Inventory Management - Copyright by Cong Hoi

#### Mô tả chi tiết các chức năng:

- Chọn sản phẩm trong Product Name, chọn số lượng sản phẩm trong Quantity để thêm sản phẩm vào danh sách sản phẩm trong một record xuất kho.
- Xem danh sách sản phẩm trong một record xuất kho vừa thêm.

#### c. View Goods Issue

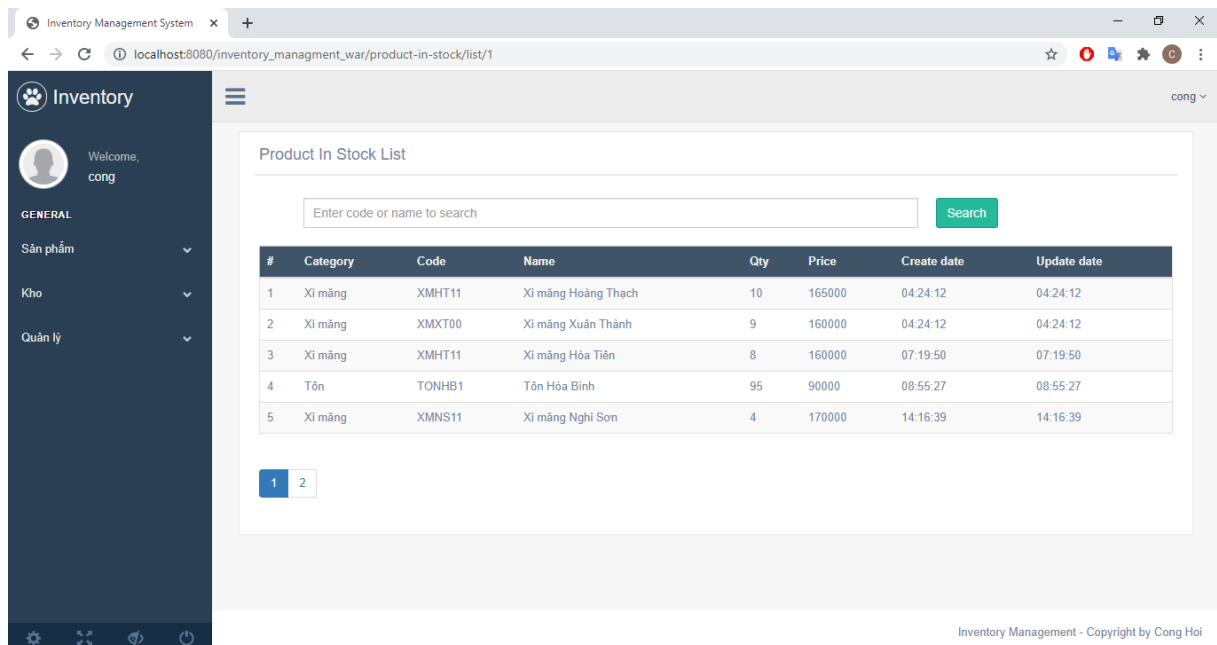


### Mô tả chi tiết các chức năng:

- Xem thông tin chi tiết một record nhập kho.

### 6.2.7. Sản phẩm trong kho

#### a. Màn hình Product In Stock List

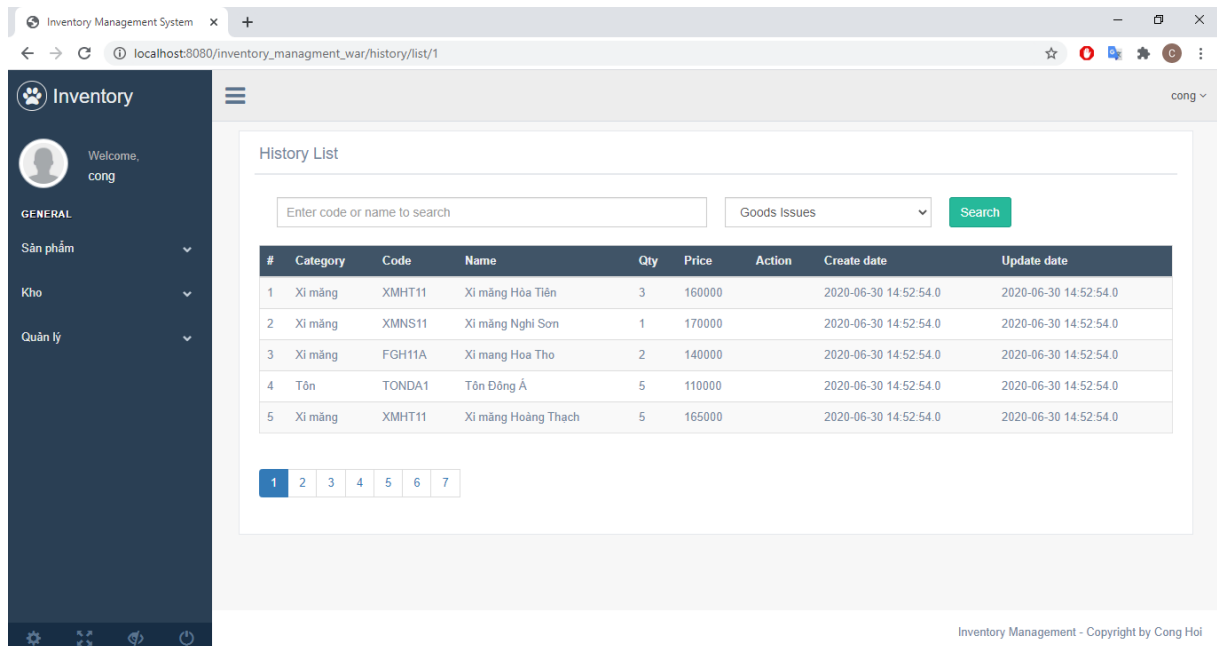


### Mô tả chi tiết các chức năng:

- Xem thông tin chi tiết danh sách sản phẩm có trong kho
- Tìm kiếm sản phẩm bằng Code hoặc Name bằng chức năng Search

## 6.2.8. Lịch sử kho

### a. Màn hình History List

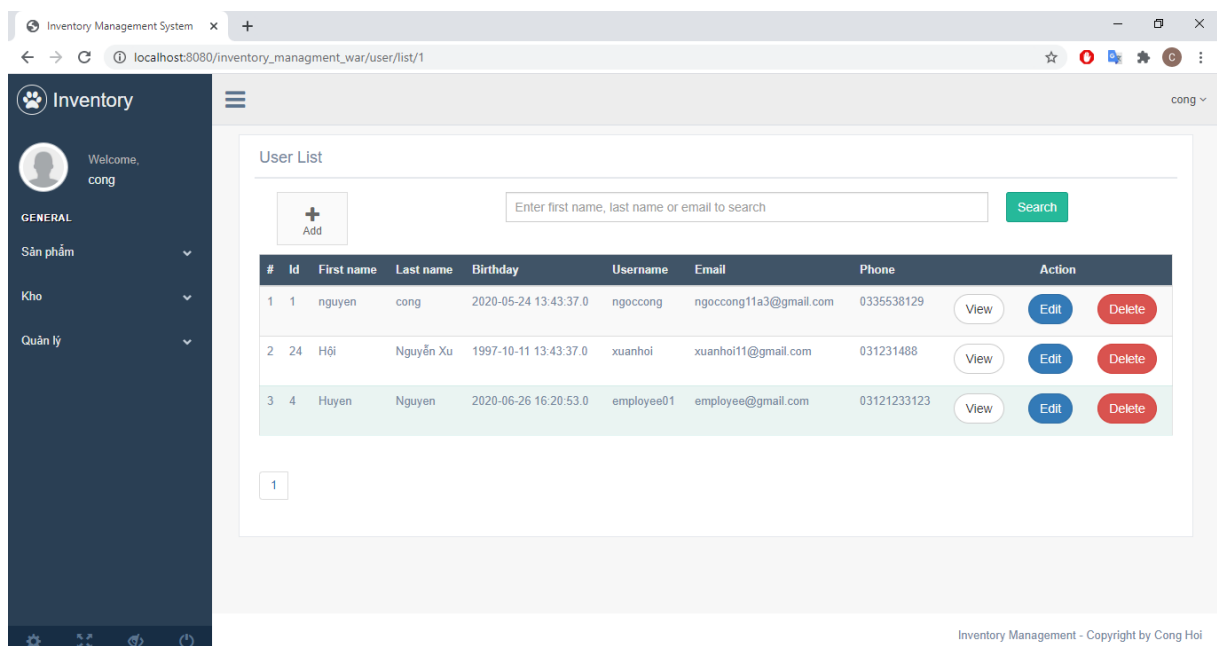


### Mô tả chi tiết các chức năng:

- Xem thông tin chi tiết danh sách các record xuất / nhập kho
- Tìm kiếm sản phẩm bằng Code hoặc Name bằng chức năng Search

## 6.2.9. User

### a. Màn hình User List



### Mô tả chi tiết các chức năng:

- Xem danh sách người dùng trong ứng dụng
- Tìm kiếm người dùng bằng cách nhập first name, lastname hoặc search bằng chức năng Search
- Chuyển đến màn hình Add Users bằng cách chọn nút Add
- Chuyển đến màn hình Edit User bằng cách chọn nút Edit
- Chuyển đến màn hình View User bằng cách chọn nút View
- Xóa người dùng bằng cách chọn nút Delete, pop-up delete records sẽ hiện ra

#### b. Màn hình Edit User

Inventory Management System

localhost:8080/inventory\_management\_war/user/edit/1

Inventory

Welcome, cong

GENERAL

Sản phẩm

Kho

Quản lý

Edit User

First name\*

Last name\*

Birth day\*

Username\*

Password\*

Email\*

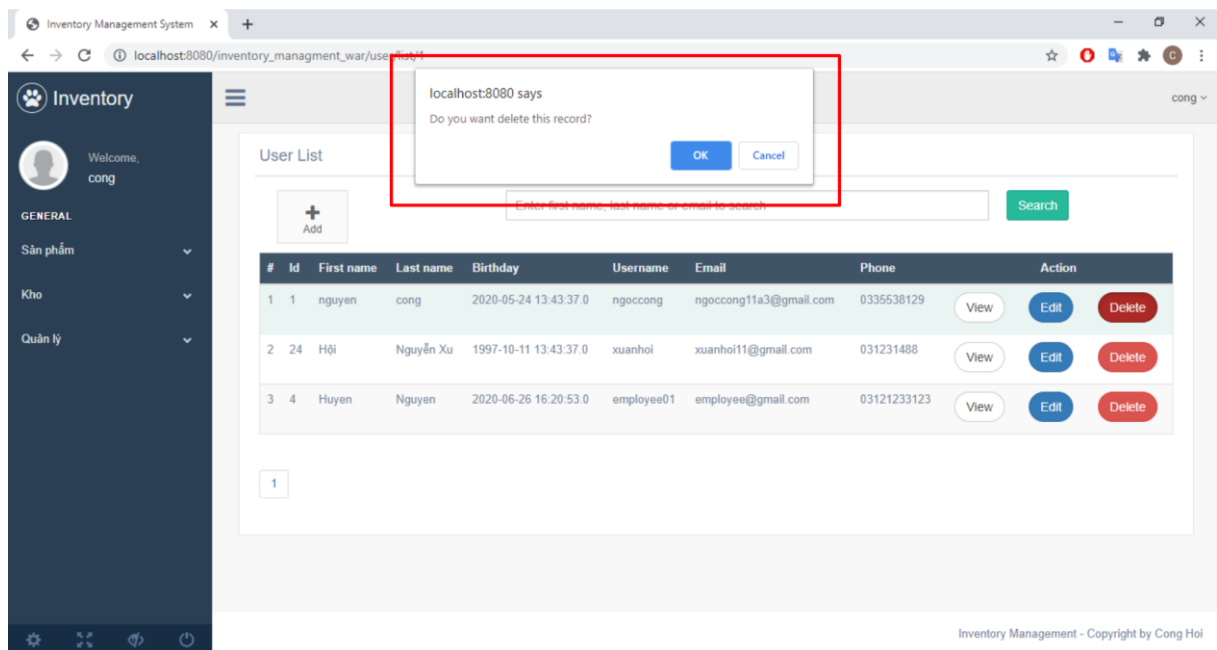
Phone\*

Inventory Management - Copyright by Cong Hoi

### Mô tả chi tiết các chức năng:

- Xóa thông tin hiện có trong các trường bằng nút Reset
- Chỉnh sửa người dùng bằng cách thêm thông tin cần chỉnh sửa vào các trường cần thiết rồi chọn Submit
- Chọn nút Cancel để hủy Edit User

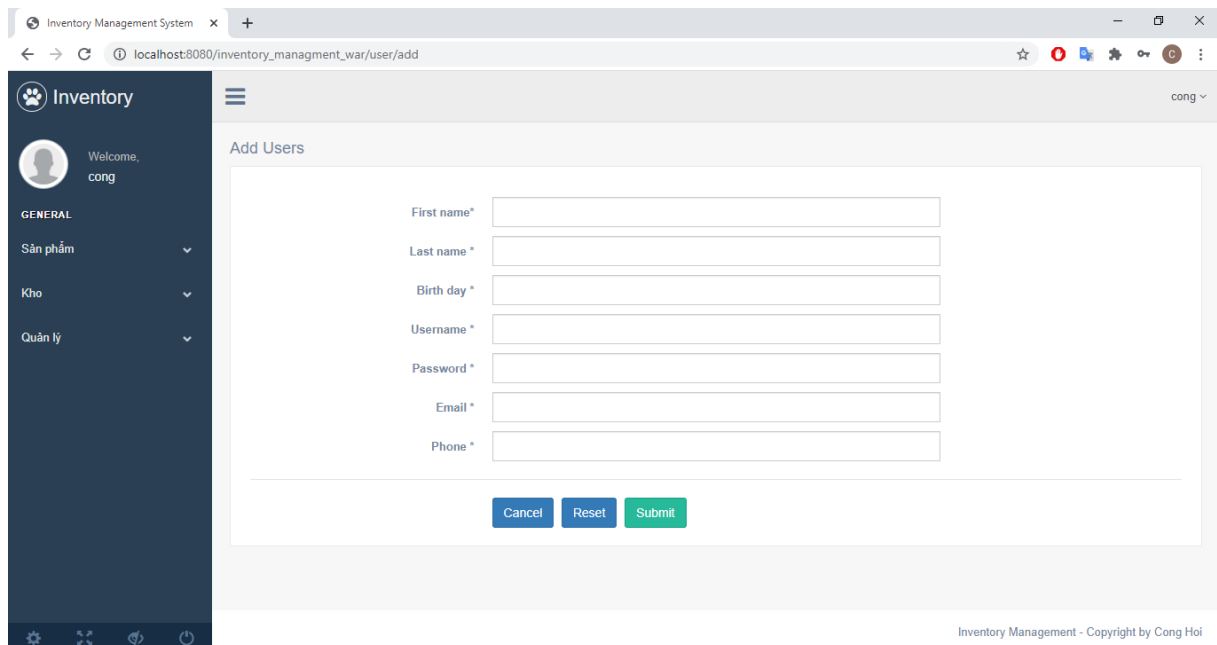
#### c. Pop-up delete records



### Mô tả chi tiết các chức năng:

- Xóa nhân viên bằng cách chọn nút Delete trong màn hình User List, sau đó pop-up delete records hiện ra, người dùng chọn button OK để xác nhận xóa, Cancel để hủy thao tác.

#### d. Màn hình Add Users



### Mô tả chi tiết các chức năng:

- Nhập các trường và thông tin cần thiết sau đó chọn Submit để thêm một người dùng.
- Chọn nút Reset để xóa những thông tin vừa nhập
- Chọn nút Cancel để hủy bỏ thao tác thêm người dùng

#### e. Màn hình View User

The screenshot shows a web application interface for an Inventory Management System. The main content area is titled 'View User' and contains a form with the following fields:

- First name\*: nguyen
- Last name\*: cong
- Birth day\*: 2020-05-24 13:43:37
- Username\*: ngoccong
- Password\*: \*\*\*\*\*
- Email\*: ngoccong11a3@gmail.com
- Phone\*: 0335538129

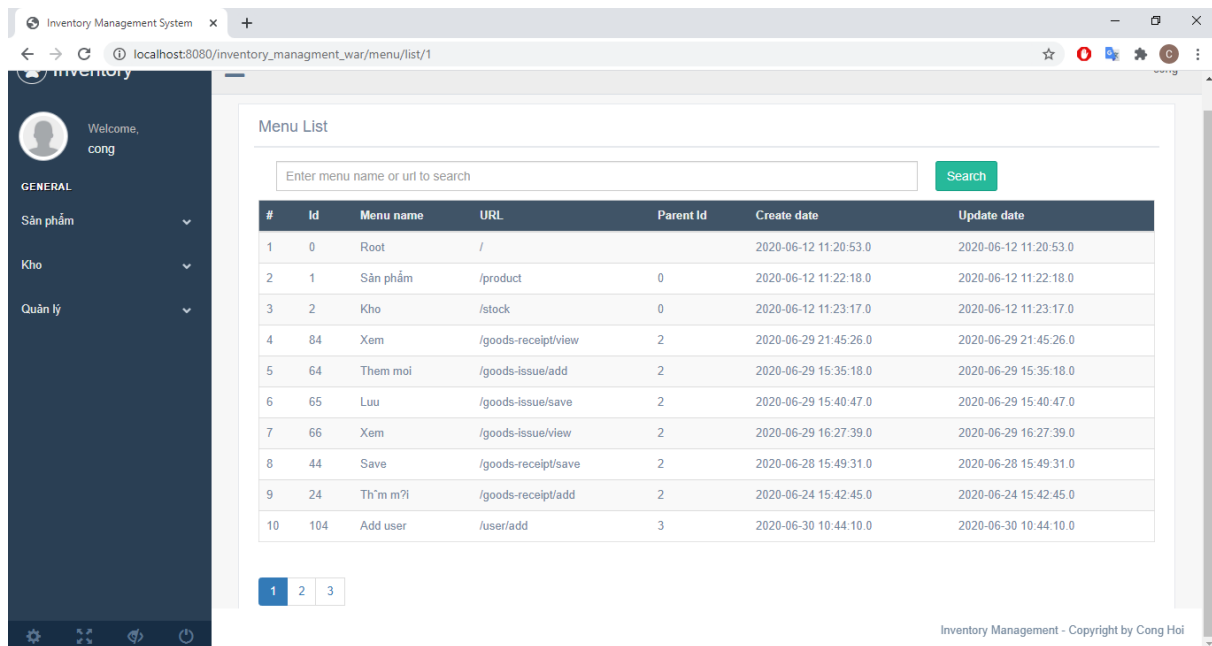
Below the form is a blue 'Cancel' button. The left sidebar is dark blue with the 'Inventory' logo and a user profile section showing 'Welcome, cong'. The sidebar menu includes 'GENERAL', 'Sản phẩm', 'Kho', and 'Quản lý'. The top navigation bar shows the user's name 'cong' and a dropdown arrow. The footer text reads 'Inventory Management - Copyright by Cong Hoi'.

#### Mô tả chi tiết các chức năng:

- Xem chi tiết thông tin người dùng.

#### 6.2.10. Menu

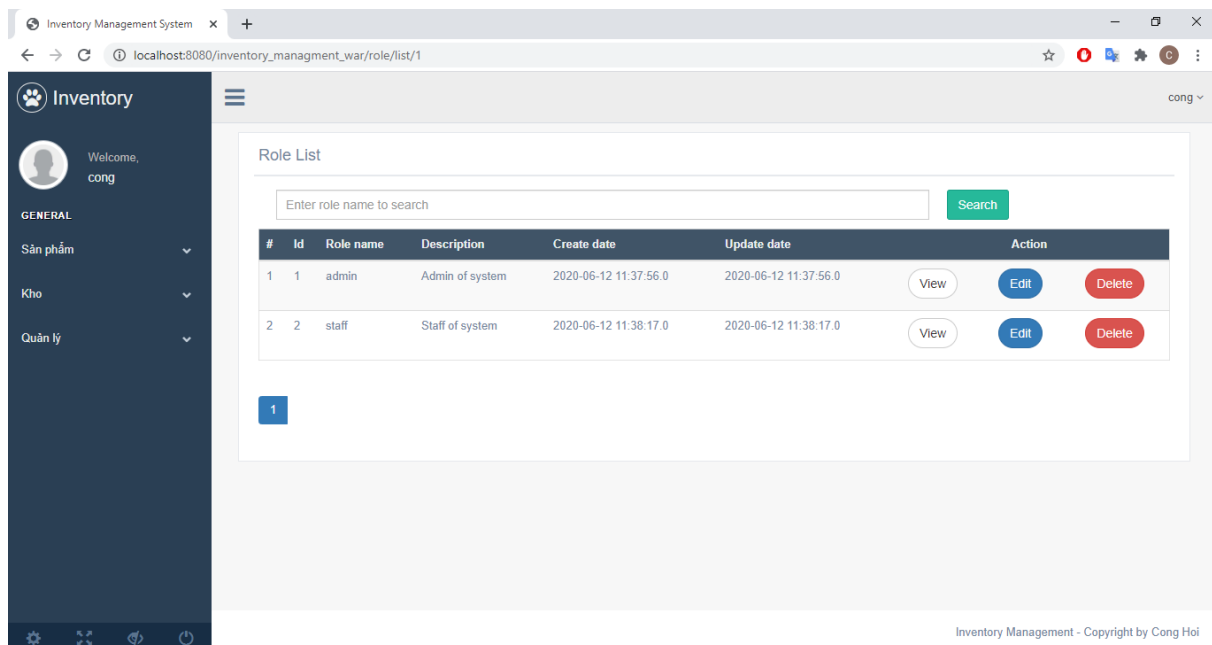




### Mô tả chi tiết các chức năng:

Menu List hiển thị các thông tin như id, tên menu, đường dẫn URL, ngày tạo, ngày sửa đổi cuối cùng.

#### 6.2.11. Quyền



### Mô tả chi tiết các chức năng:

Role List hiển thị danh sách các quyền, bao gồm thông tin như: Id, Role name, Description, Create date, Update date. Admin có thể xem, sửa, xóa các role.

## **CHƯƠNG 7. KẾT LUẬN**

### **7.1. Kết quả đạt được**

#### **7.1.1. Kiến thức**

Trong suốt quá trình làm đồ án vừa qua, nhóm đã đạt được những điều sau:

- Biết cách thiết kế, hoàn thiện một dự án nhỏ.
- Hiểu và vận dụng được kiến thức của môn hệ quản trị cơ sở dữ liệu, cụ thể là làm việc trên Oracle, như:
  - o Các cú pháp và cách suy nghĩ của ngôn ngữ PL/SQL.
  - o Cách thực hiện một Trigger, Function, Stored Procedure.
  - o Trang bị thêm kiến thức về khóa và các mức cô lập.
  - o Giải quyết được các trường hợp truy xuất đồng thời.

#### **7.1.2. Làm việc nhóm**

- Nắm được kỹ năng phân chia công việc, sắp xếp thời gian, làm việc nhóm.
- Hỗ trợ lẫn nhau để hoàn thành công việc.

#### **7.1.3. Ứng dụng Quản lý kho**

Xây dựng được phần mềm “Quản lý kho” với các chức năng sau:

- Chức năng đăng nhập/đăng xuất.
- Chức năng quản lý sản phẩm
- Chức năng quản lý loại sản phẩm
- Chức năng quản lý hàng tồn kho
- Chức năng quản lý nhập hàng
- Chức năng quản lý xuất hàng
- Chức năng quản lý nhân viên

### **7.2. Hạn chế**

Để hoàn thành được đồ án trên, trong quá trình thực hiện, nhóm đã gặp khá nhiều khó khăn và hạn chế:

- Giao diện không được chăm chút.
- Các tình huống truy xuất đồng thời có thể không hợp lí.
- Phân chia công việc có thể không đều.
- Thời gian chủ yếu là học online nên nhóm ít có cơ hội trao đổi trực tiếp.

### **7.3. Hướng phát triển**

- Phát triển thêm chức năng sao lưu cơ sở dữ liệu tại nhiều thời điểm và cho phép phục hồi cơ sở dữ liệu tại thời điểm mong muốn.
- Thêm chức năng phân quyền cho từng đối tượng cụ thể
- Tăng tính bảo mật và an toàn dữ liệu cho chương trình.
- Hướng tới sử dụng cho quản lý chuỗi nhiều kho bãi cùng lúc.
- Phát triển và chỉnh sửa giao diện để giúp người dùng dễ dàng hơn trong sử dụng.
- Cải tiến, tối ưu mã nguồn để phần mềm chạy nhanh hơn và hỗ trợ tốt người dùng.
- Hỗ trợ đa ngôn ngữ.
- Hoàn thiện, thêm các tính năng khác còn thiếu.
- Đưa vào sử dụng thực tế ở các doanh nghiệp.

## Phụ lục

### Phụ lục 1: Bảng phân chia công việc

STT	Đầu việc	Phân công		
		Công	Hội	Phóng
1	Phân tích bài toán, xác định chức năng	33%	33%	33%
2	Thiết kế & xây dựng CSDL	20%	40%	40%
3	Thiết kế giao diện ứng dụng	40%	40%	20%
4	Triển khai ứng dụng bằng Java Spring	80%	20%	0%
5	Viết các Trigger và Procedure	40%	30%	30%
6	Xử lý vấn đề truy xuất đồng thời	40%	30%	30%
7	Kiểm tra chức năng	25%	25%	50%
8	Soạn và chỉnh sửa báo cáo	30%	50%	20%

### Phụ lục 2: Tài liệu tham khảo

[1] Oracle Database Database Concepts, 12c Release 2 (12.2) - Primary Authors: Lance Ashdown, Tom Kyte

[2] Slide môn Hệ quản trị cơ sở dữ liệu – ThS. Đỗ Thị Minh Phụng, trường ĐH Công Nghệ Thông Tin

[3] Quản lý kho là gì?: <https://www.sapo.vn/blog/kinh-nghiem-quan-ly-kho-dat-hieu-qua/>

[4] Quy trình quản lý kho: <https://sec-warehouse.vn/quy-trinh-quan-ly-kho-hang.html>

[5] Kiến thức Oracle : <https://csc.edu.vn/lap-trinh-va-csdl/tin-tuc/kien-thuc-lap-trinh/hoc-oracle--kien-thuc-co-ban-cho-nguoi-moi-bat-dau-802>

[6] Hibernate framework: <https://hibernate.org/>

[7] Hibernate tutorial: [https://www.tutorialspoint.com/hibernate/hibernate\\_architecture.htm](https://www.tutorialspoint.com/hibernate/hibernate_architecture.htm)

[8] Tổng quan Spring framework: <https://viblo.asia/p/tong-quan-ve-spring-framework-YWOZryEyKQ0>

[9] Spring framework: <https://spring.io/>

[10] Github: <https://github.blog/>

[11]IntelliJ IDEA: <https://www.jetbrains.com/idea/>

