**ỦY BAN NHÂN DÂN THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC SÀI GÒN**

A blue button with white text

Description automatically generated

**BÁO CÁO NHÓM**

**MÔN: KIỂM THỬ PHẦN MỀM**

**Võ Thành Danh : 3122411024**

**Huỳnh Minh Quân : 3122411167**

**Huỳnh Duy Khang : 3122411088**

**Đỗ Phú Thành : 3122411189**

Mục Lục

[Bài 1. 1](#_Toc209084918)

[Bài 2: 2](#_Toc209084919)

[Bài 3 3](#_Toc209084920)

[Bài 4 6](#_Toc209084921)

[Bài 5. 8](#_Toc209084922)

[Bài 6. 8](#_Toc209084923)

[Câu 7. 10](#_Toc209084924)

[Bài tập ứng dụng 17](#_Toc209084925)

[Bài 1. 17](#_Toc209084926)

[Bài 2. 19](#_Toc209084927)

[Bài 3. 20](#_Toc209084928)

[Bài 4. 20](#_Toc209084929)

[Bài 5 22](#_Toc209084930)

[Bài 6. 23](#_Toc209084931)

[Bài 7. 24](#_Toc209084932)

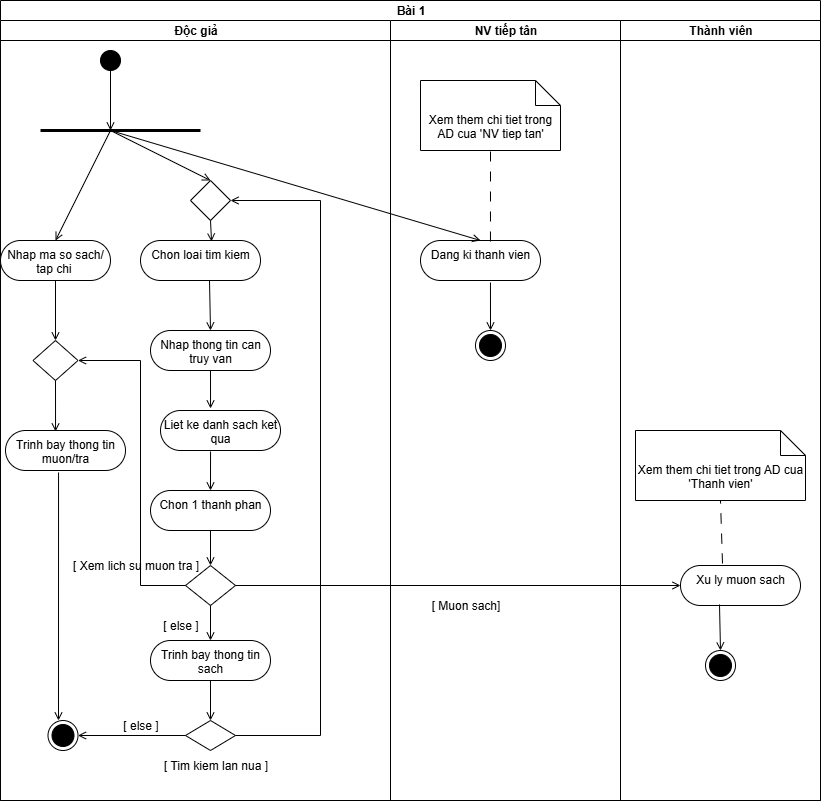
[Bài 8. 24](#_Toc209084933)

[Bài 9. 26](#_Toc209084934)

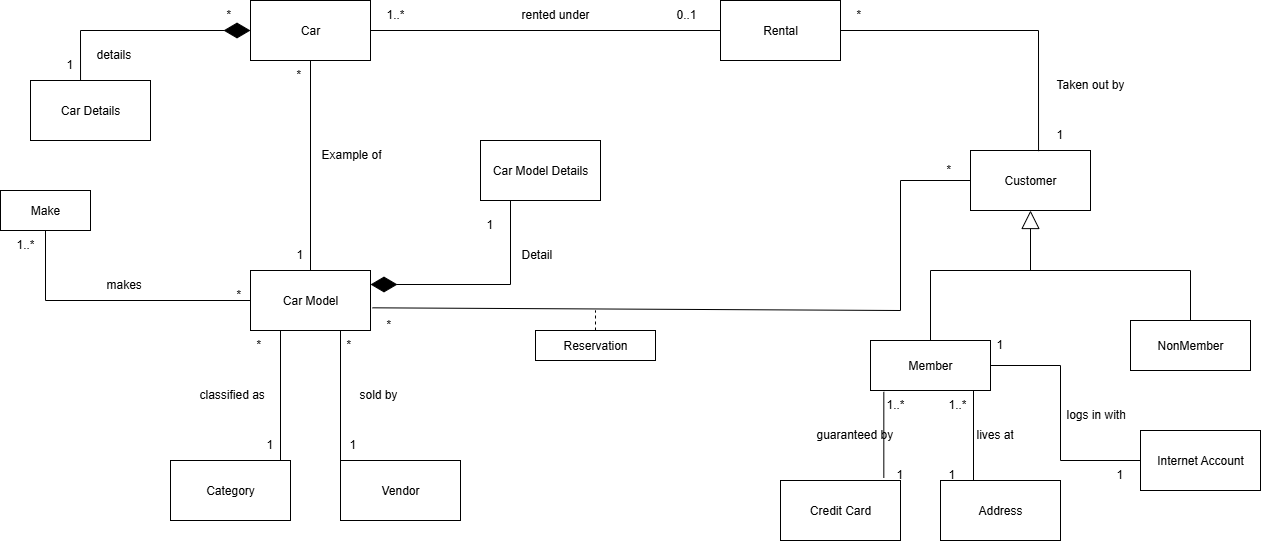
**Phân công công việc**

|  |  |
| --- | --- |
| Duy Khang | Bài 1 |
| Phú Thành | Bài 2 |
| Thành Danh | Bài 3 |
| Phú Thành | Bài 4 |
| Minh Quân | Bài 5 |
| Minh Quân | Bài 6 |
| Duy Khang | Bài 7 |
| Thành Danh | Bài ứng dụng 1 |
| Thành Danh | Bài ứng dụng 2 |
| Thành Danh | Bài ứng dụng 3 |
| Duy Khang | Bài ứng dụng 4 |
| Duy Khang | Bài ứng dụng 5 |
| Phú Thành | Bài ứng dụng 6 |
| Phú Thành | Bài ứng dụng 7 |
| Minh Quân | Bài ứng dụng 8 |
| Minh Quân | Bài ứng dụng 9 |

# Bài 1.



# Bài 2:



# Bài 3

A black background with white circles

AI-generated content may be incorrect.

a) Xác định và mô tả các tác nhân (Actors) xuất hiện trong sơ đồ trên.

- Student: đăng kí môn học

- Billing System: tiếp nhận các môn học được đăng kí

- Professor: chọn môn học để dạy

- Registrar:

+ Quản lý chương trình giảng dạy

+ Quản lý thông tin sinh viên

+ Quản lý thông tin giáo sư

+ Xác thực phòng đào tạo

b) Liệt kê và giải thích các trường hợp sử dụng (Use cases) được thể hiện trong sơ đồ.

- Register for Courses: Student đăng kí môn học và Billing System sẽ tiếp nhận các môn học được đăng kí từ Student

- Select Courses to Teach: Professor lựa chọn các môn học hiện có để giảng dạy

- Maintain Curriculum: Registrar trực tiếp quản lý các khóa học

- Maintain Student Information: Registrar tiến hành quản lý thông tin sinh viên

- Maintain Professor Information: Resgistrar tiến hành quản lý thông tin Professor

- Registrar Validation: được gọi bởi các Use Case Curriculum, Maintain Student Information, Maintain Professor Information

c) Phân tích các mối quan hệ giữa các use case (bao gồm cả mối quan hệ «uses» hoặc «include»).

1. Register for Courses

- Quan hệ:

+ Use case này gửi dữ liệu sang Billing System để xử lý học phí khi sinh viên đăng ký.

+ Đây là quan hệ liên kết trực tiếp (association).

2. Select Courses to Teach

- Quan hệ:

+ Professor chọn môn học để giảng dạy

+ Đây là quan hệ liên kết trực tiếp (association).

3. Maintain Student Information

- Quan hệ:

+ Registrar trực tiếp quản lý thông tin Student

+ Đây là quan hệ liên kết trực tiếp (association).

4. Maintain Professor Information

- Quan hệ:

+ Registrar trực tiếp quản lý thông tin Professor

+ Đây là quan hệ liên kết trực tiếp (association).

5. Maintain Curriculum

- Quan hệ:

+ Registrar trực tiếp quản lý các khóa học

+ Đây là quan hệ liên kết trực tiếp (association).

6. Registrar Validation

- Quan hệ:

+ Xác thực thông tin từ Maintain Curriculum, Maintain Professor Information, Maintain Student Information được Registrar đăng kí

+ Đây là quan hệ liên kết use được dùng cho 3 use case trên

d) Viết kịch bản chi tiết cho một trường hợp sử dụng cụ thể (ví dụ: “Register for Courses” hoặc“Maintain Student Information”).

|  |  |
| --- | --- |
| Use Case | Register for Courses |
| Description | Sinh viên đăng ký học phần cho học kỳ mới và đảm bảo thông tin học phí được gửi sang hệ thống thanh toán. |
| Users | Student, Billing System |
| Pre-condition | Sinh viên đã đăng nhập vào hệ thống quản lý đào tạo. |
| Post-condition | Thông tin sinh viên đã tồn tại trong cơ sở dữ liệu. |
| Trigger | Sinh viên chọn chức năng "Đăng ký môn học" từ giao diện hệ thống. |

|  |  |
| --- | --- |
| Step | Description |
| 1 | Sinh viên chọn chức năng “Register for Courses” từ hệ thống. |
| 2 | Hệ thống hiển thị danh sách các học phần mở trong học kỳ (kèm thông tin: mã môn, số tín chỉ, giảng viên, lịch học). |
| 3 | Sinh viên chọn các học phần muốn đăng ký. |
| 4 | Hệ thống ghi nhận các học phần mà sinh viên đã chọn. |
| 5 | Hệ thống gửi thông tin học phí tương ứng sang **Billing System**. |
| 6 | Hệ thống hiển thị thông báo thành công cho sinh viên và lưu kết quả đăng ký. |

e) Đề xuất cải tiến hoặc bổ sung thêm các use case mà bạn thấy cần thiết.

- Bổ sung các Use Case cho Student

+ Xem thời khóa biểu: sau khi đăng ký, sinh viên cần xem lịch học cá nhân.

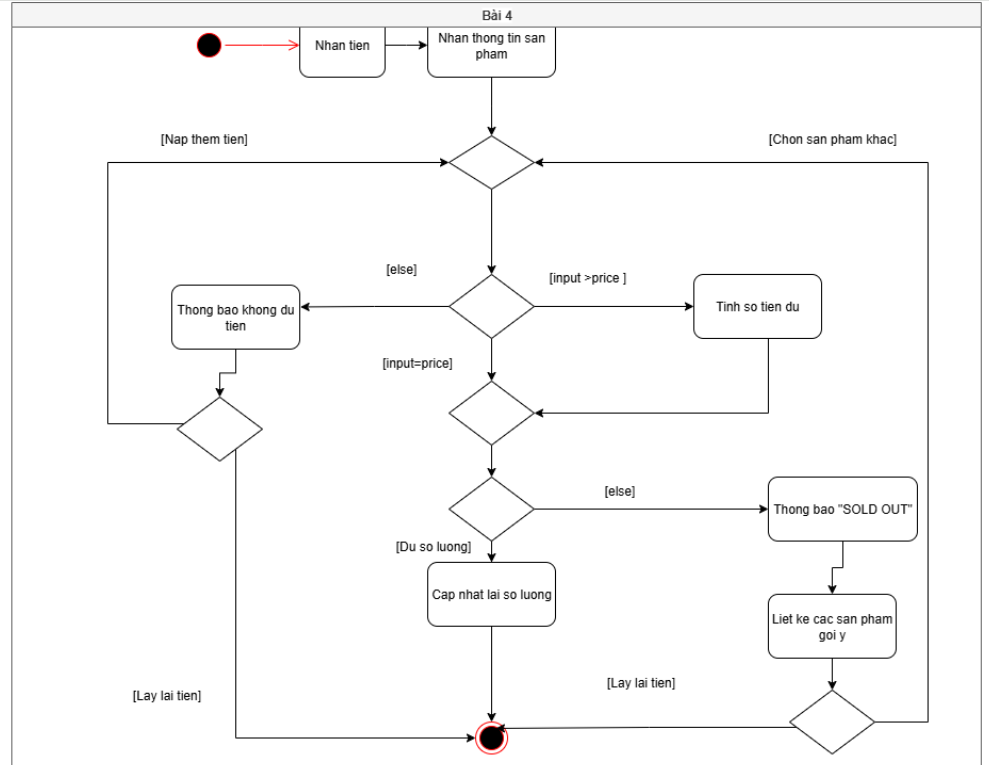
+ Hủy học phần: cho phép sinh viên hủy môn đã đăng ký trong thời hạn cho phép.

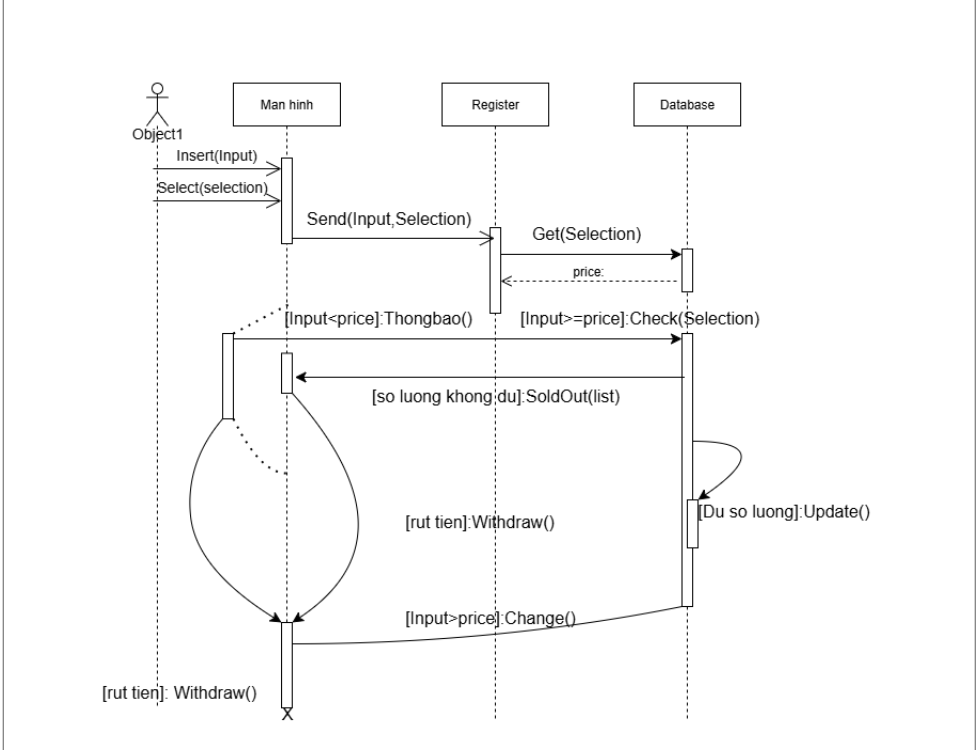
- Bổ sung cho Registrar :

+ Tạo báo cáo: ví dụ báo cáo số lượng sinh viên đăng ký, hủy môn.

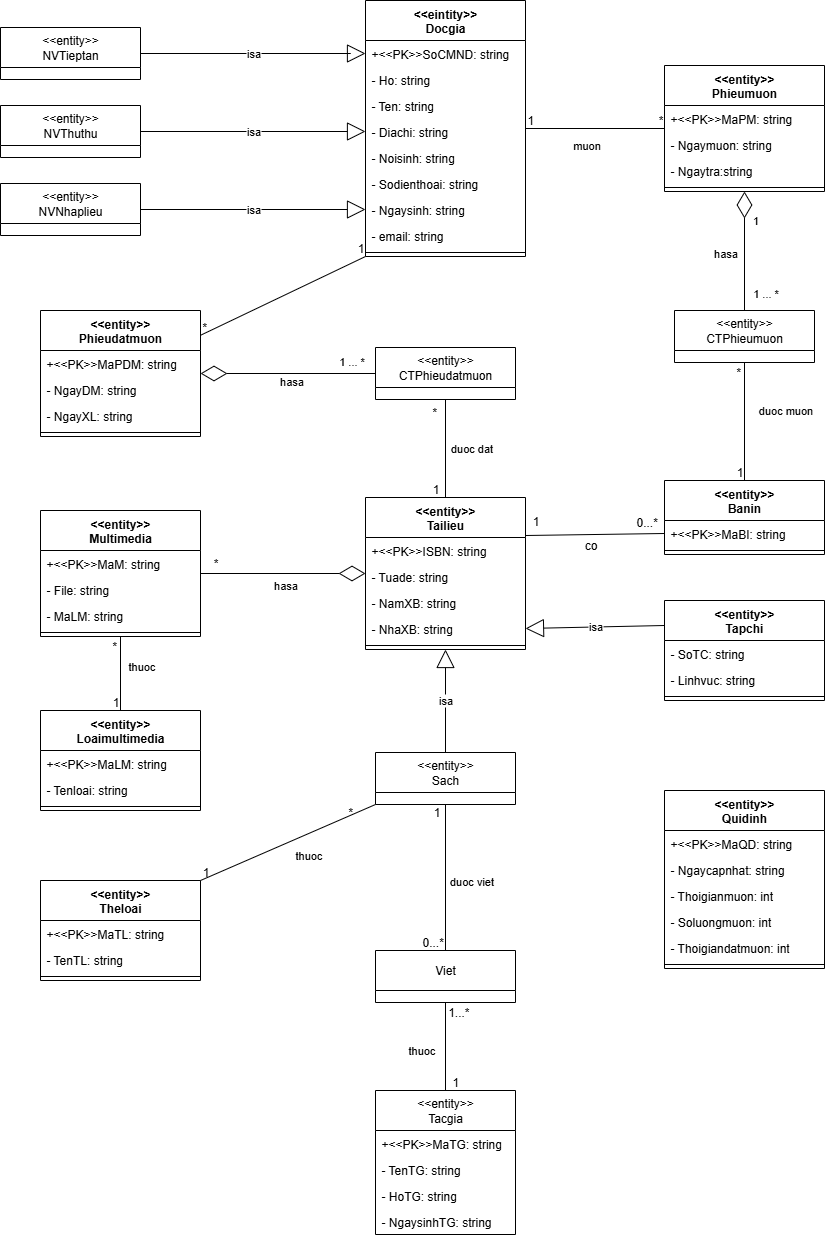
+ Quản lý lịch năm học: thiết lập ngày bắt đầu, kết thúc học kỳ, lịch thi.

# Bài 4

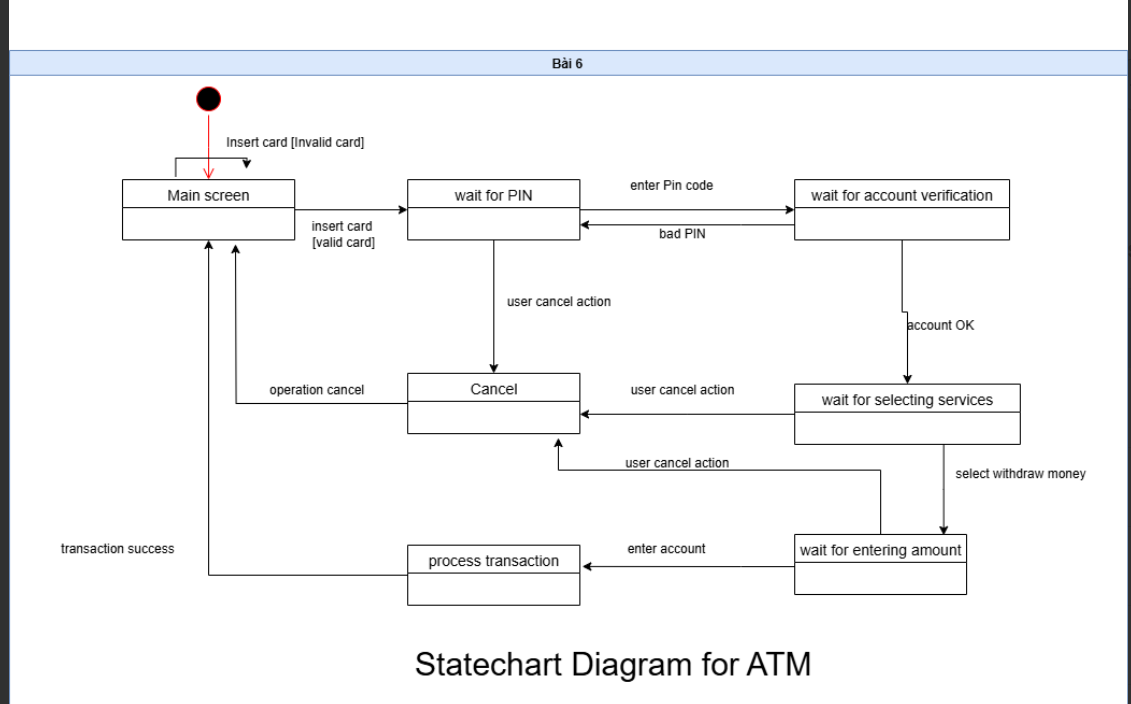


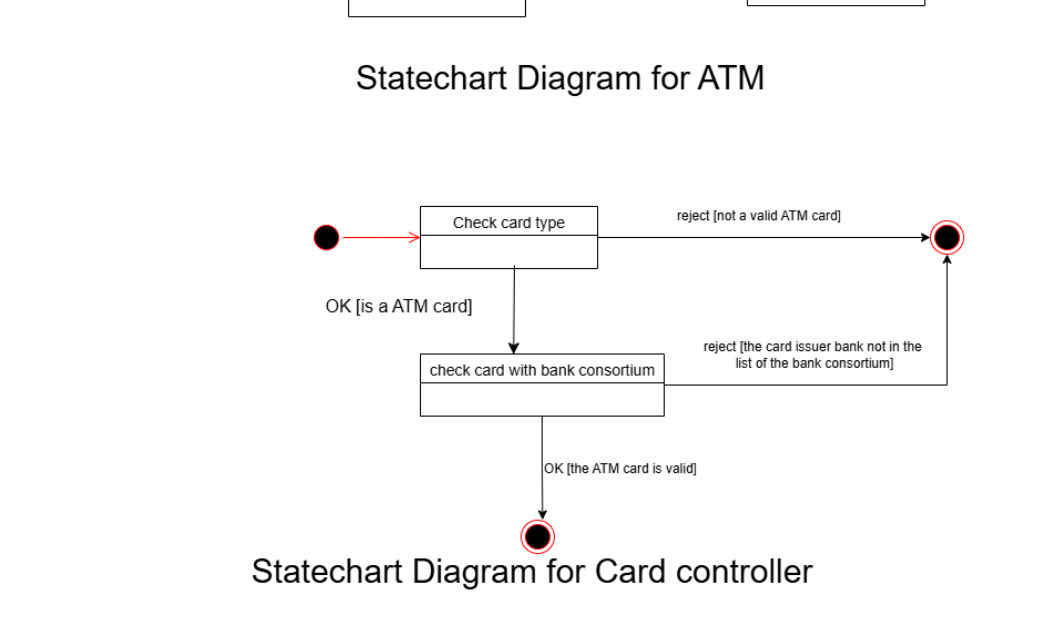


# Bài 5.

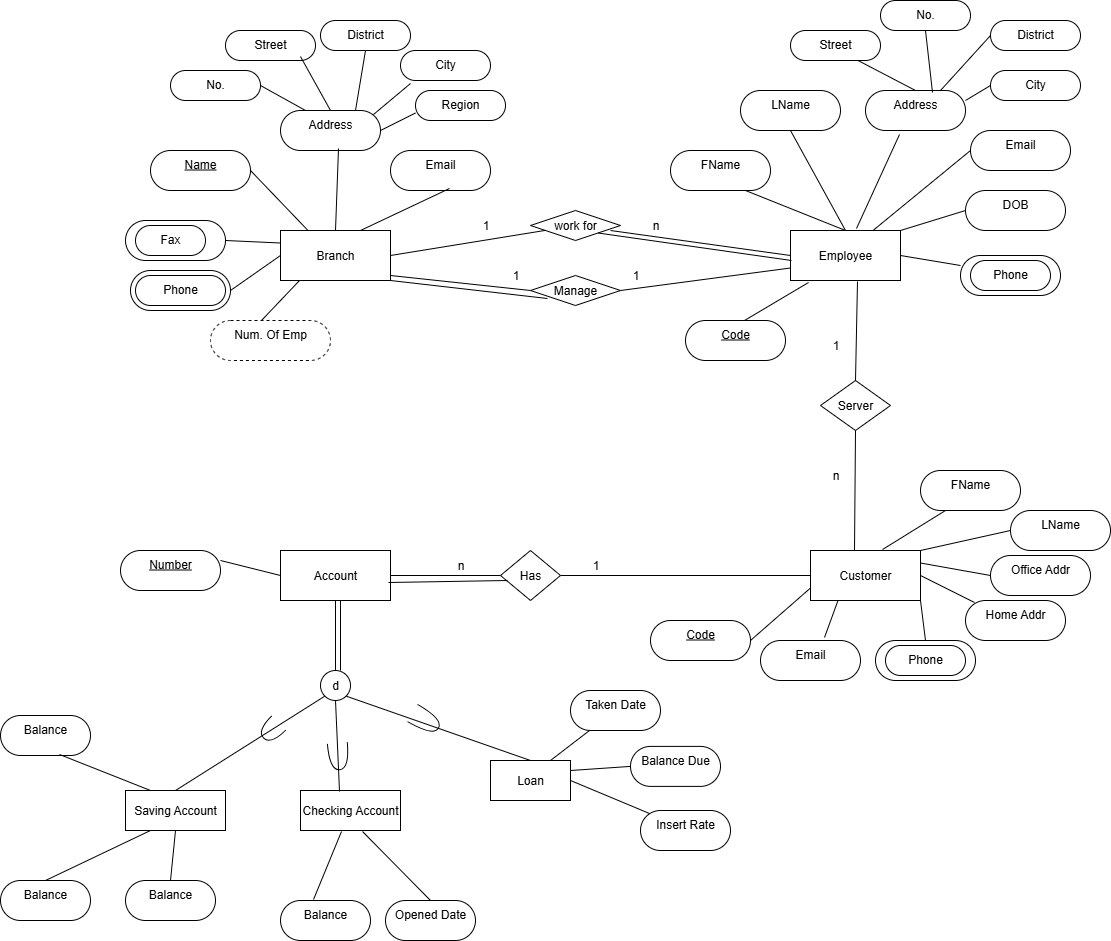


# Bài 6.





# Câu 7.



thiết kế database, viết truy vấn theo yêu cầu:

CREATE TABLE BRANCH (

Name VARCHAR(50) PRIMARY KEY,

AddressNo VARCHAR(20),

Street VARCHAR(50),

District VARCHAR(50),

City VARCHAR(50),

Region VARCHAR(50),

Email VARCHAR(50),

MngCode VARCHAR(10)

);

CREATE TABLE BRANCHPHONE (

BraName VARCHAR(50),

Phone VARCHAR(20),

PRIMARY KEY (BraName, Phone),

FOREIGN KEY (BraName) REFERENCES BRANCH(Name)

);

CREATE TABLE BRANCHFAX (

BraName VARCHAR(50),

Fax VARCHAR(20),

PRIMARY KEY (BraName, Fax),

FOREIGN KEY (BraName) REFERENCES BRANCH(Name)

);

CREATE TABLE EMPLOYEE (

Code VARCHAR(10) PRIMARY KEY,

FName VARCHAR(30),

LName VARCHAR(30),

AddressNo VARCHAR(20),

Street VARCHAR(50),

District VARCHAR(50),

City VARCHAR(50),

DOB DATE,

Email VARCHAR(50),

BraName VARCHAR(50),

FOREIGN KEY (BraName) REFERENCES BRANCH(Name)

);

CREATE TABLE EMPLOYEEPHONE (

EmpCode VARCHAR(10),

Phone VARCHAR(20),

PRIMARY KEY (EmpCode, Phone),

FOREIGN KEY (EmpCode) REFERENCES EMPLOYEE(Code)

);

CREATE TABLE CUSTOMER (

Code VARCHAR(10) PRIMARY KEY,

FName VARCHAR(30),

LName VARCHAR(30),

OfficeAddress VARCHAR(100),

HomeAddress VARCHAR(100),

Email VARCHAR(50),

EmpCode VARCHAR(10),

FOREIGN KEY (EmpCode) REFERENCES EMPLOYEE(Code)

);

CREATE TABLE CUSTOMERPHONE (

CusCode VARCHAR(10),

Phone VARCHAR(20),

PRIMARY KEY (CusCode, Phone),

FOREIGN KEY (CusCode) REFERENCES CUSTOMER(Code)

);

CREATE TABLE ACCOUNT (

Number VARCHAR(10) PRIMARY KEY,

CusCode VARCHAR(10),

FOREIGN KEY (CusCode) REFERENCES CUSTOMER(Code)

);

CREATE TABLE SAVINGACCOUNT (

AccNumber VARCHAR(10) PRIMARY KEY,

Balance DECIMAL(15,2),

InterestRate DECIMAL(5,2),

OpenedDate DATE,

FOREIGN KEY (AccNumber) REFERENCES ACCOUNT(Number)

);

CREATE TABLE CHECKINGACCOUNT (

AccNumber VARCHAR(10) PRIMARY KEY,

Balance DECIMAL(15,2),

OpenedDate DATE,

FOREIGN KEY (AccNumber) REFERENCES ACCOUNT(Number)

);

CREATE TABLE LOAN (

AccNumber VARCHAR(10) PRIMARY KEY,

BalanceDue DECIMAL(15,2),

InterestRate DECIMAL(5,2),

TakenDate DATE,

FOREIGN KEY (AccNumber) REFERENCES ACCOUNT(Number)

);

a. You are required to insert valid and meaningful data into the database. Each table has at least 4 rows.

INSERT INTO BRANCH VALUES ('BranchA','12','Main St','District 1','HCM','South','branchA@bank.com','NI1080');

INSERT INTO BRANCH VALUES ('BranchB','22','Hai Ba Trung','District 1','HCM','South','branchB@bank.com','NI1081');

INSERT INTO BRANCH VALUES ('BranchC','30','Nguyen Hue','District 5','HCM','South','branchC@bank.com','NI1082');

INSERT INTO EMPLOYEE VALUES

('NI1080','John','Smith','12','Main St','District 1','HCM','1980-05-12','john.smith@bank.com','BranchA'),

('NI1081','Anna','Taylor','45','Le Loi','District 3','HCM','1990-03-20','anna.taylor@bank.com','BranchA'),

('NI1082','Peter','Johnson','22','Hai Ba Trung','District 1','HCM','1985-07-10','peter.johnson@bank.com','BranchB'),

('NI1083','Mary','Brown','30','Nguyen Hue','District 5','HCM','1992-11-01','mary.brown@bank.com','BranchC');

INSERT INTO CUSTOMER VALUES

('C001','Peter','Johnson','Company A','12 Ly Thuong Kiet','peter.cus@bank.com','NI1082'),

('C002','David','Miller','Company B','45 Tran Hung Dao','david.miller@bank.com','NI1081'),

('C003','Sarah','Williams','Company C','99 Nguyen Hue','sarah.w@bank.com','NI1083');

INSERT INTO ACCOUNT VALUES

('A001','C001'),

('A002','C001'),

('A003','C002'),

('A004','C003');

INSERT INTO SAVINGACCOUNT VALUES

('A001',5000.00,3.5,'2020-01-10'),

('A002',8000.00,4.0,'2021-06-15');

INSERT INTO CHECKINGACCOUNT VALUES

('A003',2000.00,'2021-05-10'),

('A004',1500.00,'2022-07-20');

INSERT INTO LOAN VALUES

('A003',10000.00,6.5,'2021-07-01');

b. Change last name of an employee whose code is “N1080” to “Brown”.

UPDATE EMPLOYEE

SET LName = 'Brown'

WHERE Code = 'NI1080';

c. Delete the employee having code “N1080”. Explain what will happen to the customers who are severed by this employee.

DELETE FROM EMPLOYEE

WHERE Code = 'NI1080';

d. Retrieve all account numbers of a customer whose name is Peter Johnson.

SELECT A.Number

FROM ACCOUNT A

JOIN CUSTOMER C ON A.CusCode = C.Code

WHERE C.FName = 'Peter' AND C.LName = 'Johnson';

e. Find all employees who have their birthdays in March 2010.

SELECT \*

FROM EMPLOYEE

WHERE YEAR(DOB) = 2010 AND MONTH(DOB) = 3;

f. Retrieve the total balance of all saving accounts owned by a customer named “Peter Johnson”

(Supposing that he has many saving accounts)

SELECT SUM(S.Balance) AS TotalSavingBalance

FROM CUSTOMER C

JOIN ACCOUNT A ON C.Code = A.CusCode

JOIN SAVINGACCOUNT S ON A.Number = S.AccNumber  
WHERE C.FName = ‘Peter’ AND C.LNAME = ‘Johnson’

# Bài tập ứng dụng

## Bài 1.

Yêu cầu chức năng:

Danh mục sản phẩm (Product Catalog):

* Hiển thị danh sách sản phẩm với khả năng lọc và sắp xếp theo tên và giá
* Xem chi tiết sản phẩm (tên, mô tả, số lượng tồn kho, thông tin kho, trạng thái nổi bật, đánh giá)
* Quản lý sản phẩm bởi SysAdmin
* Gán sản phẩm vào kho hiện có

Giỏ hàng (Shopping Cart):

* Thêm sản phẩm vào giỏ hàng từ danh sách hoặc trang chi tiết
* Xóa sản phẩm khỏi giỏ hàng
* Hiển thị bảng tổng hợp (tổng chi phí, chi phí sau khuyến mãi, phí vận chuyển, tiết kiệm)
* Cập nhật tự động khi thay đổi giỏ hàng
* SysAdmin có thể xem và quản lý giỏ hàng của người dùng

Quy trình thanh toán (Payment Process):

* Xác thực thông tin sản phẩm
* Xử lý thanh toán
* Gửi email thông báo cho người mua

Tồn kho (Inventory):

* Quản lý kho hàng bởi SysAdmin
* Theo dõi số lượng tồn kho
* Quản lý thông tin kho (địa chỉ, trạng thái)

Đánh giá (Rating):

* Người mua có thể đánh giá sản phẩm (1-5 sao)
* Hiển thị đánh giá trên trang chi tiết sản phẩm

Kiểm soát truy cập (Access Control):

* Đăng nhập/đăng xuất cho người mua và SysAdmin
* Phân quyền theo vai trò (Buyer → trang sản phẩm, SysAdmin → trang quản trị)

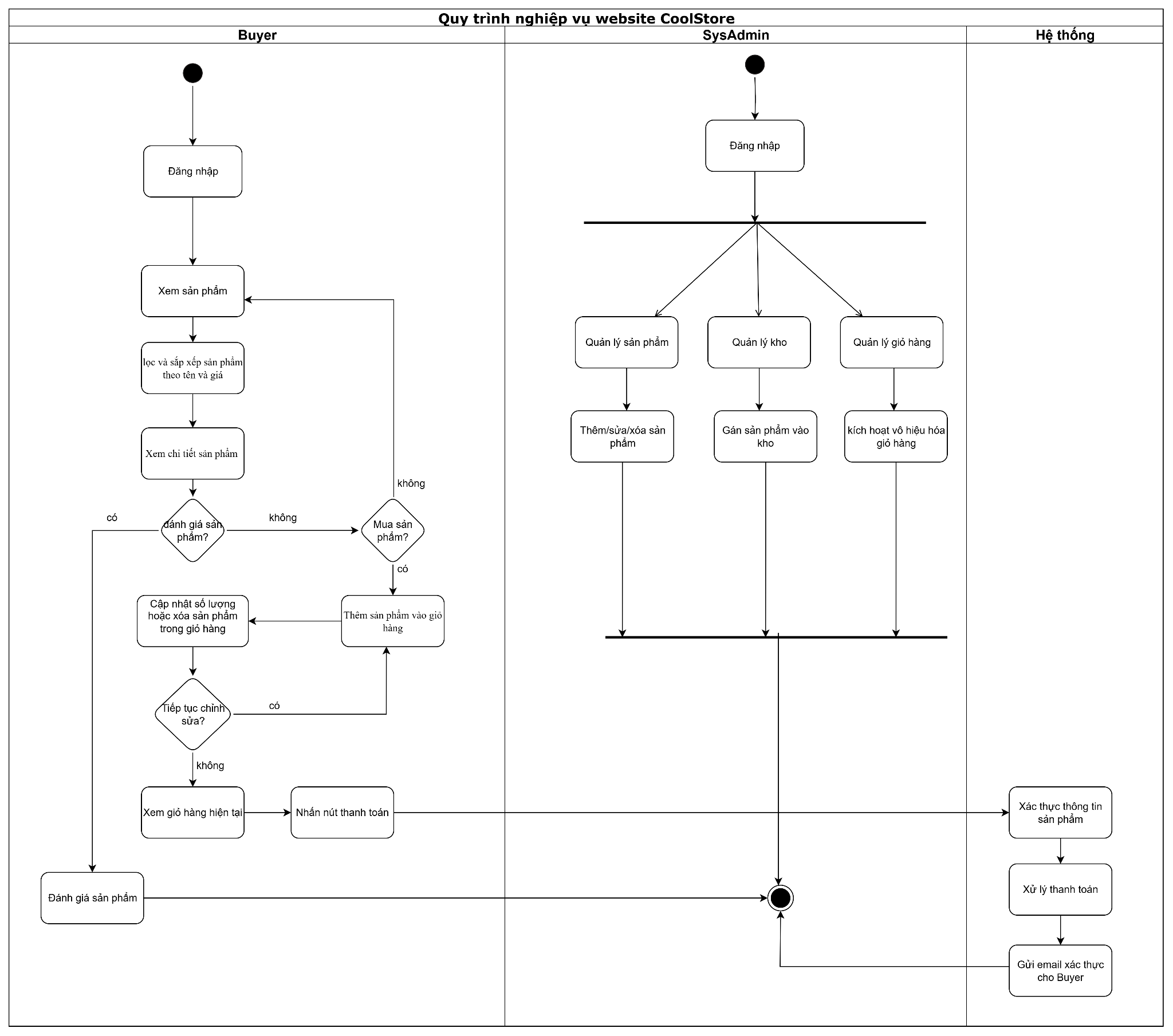
Thiết lập ban đầu:

* Tạo tài khoản SysAdmin và 2 tài khoản Buyer
* Tạo dữ liệu mẫu cho sản phẩm, kho, và đánh giá

Yêu cầu phi chức năng:

* Cập nhật giỏ hàng theo thời gian thực
* Xử lý đồng thời nhiều người dùng
* Xác thực và phân quyền an toàn
* Kiểm soát truy cập theo vai trò
* Giao diện thân thiện, dễ sử dụng
* Đảm bảo tính toàn vẹn dữ liệu giỏ hàng
* Backup và khôi phục dữ liệu

## Bài 2.



## Bài 3.

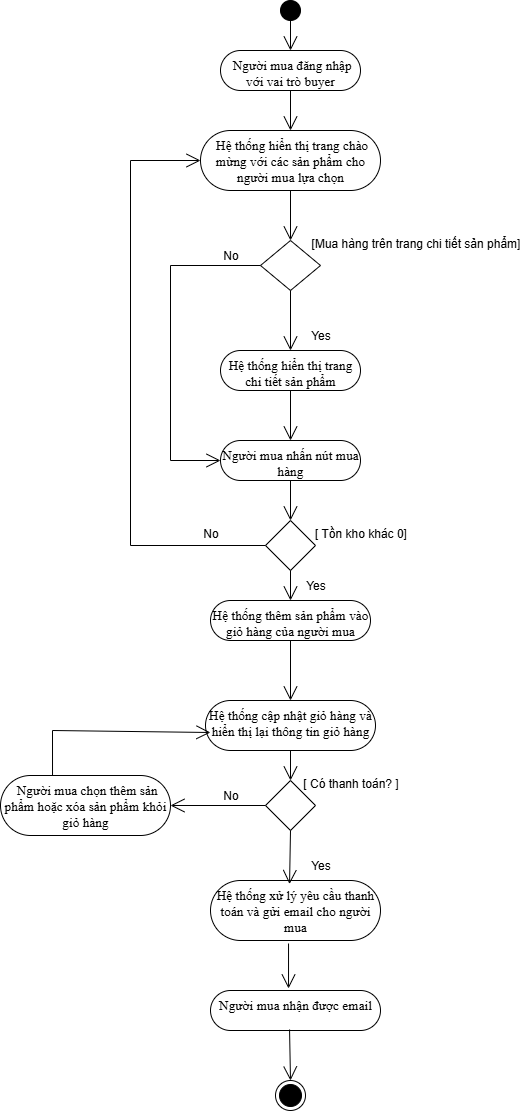
A group of white ovals with black text

AI-generated content may be incorrect.

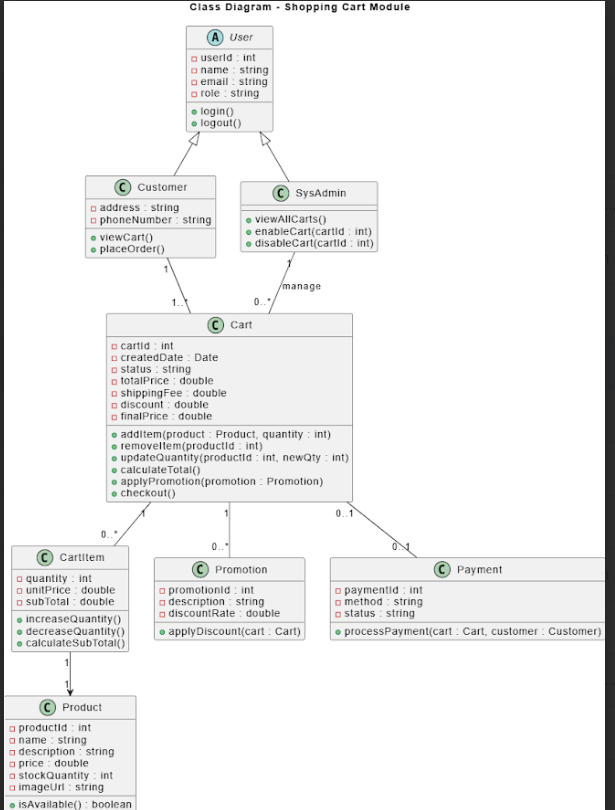
## Bài 4.

|  |  |  |
| --- | --- | --- |
| **Use Case số:** | UC1 | |
| **Tên Use-case:** | Mua sản phẩm từ trang chi tiết sản phẩm | |
| **Tác nhân chính:** | *Người mua* | |
| **Mục tiêu:** | Người mua chọn sản phẩm cụ thể từ trang chi tiết sản phẩm và thêm nó vào giỏ hàng để tiến hành mua sắm. | |
| **Tiền điều kiện:** | Người mua đã đăng nhập vào hệ thống với vai trò Buyer. | |
| **Luồng chính** | **Hành động của tác nhân** | **Hệ thống phản hồi** |
| 1. Người mua đăng nhập vào hệ thống với vai trò buyer |  |
|  | **2.** Hệ thống xác thực thông tin đăng nhập**.** |
|  | 3. Hệ thống hiển thị trang chào mừng với các sản phẩm cho người mua lựa chọn . |
| 4. Người mua chọn xem chi tiết sản phẩm |  |
|  | 5 Hệ thống hiển thị trang chi tiết sản phẩm với thông tin: tên, mô tả, số lượng tồn kho, địa chỉ kho, trạng thái “sản phẩm nổi bật” (nếu có), và đánh giá |
| 6. Người mua duyệt danh mục sản phẩm và nhấn nút "Mua hàng" trên trang chi tiết sản phẩm. **E1** |  |
|  | 7. Hệ thống thêm sản phẩm được chọn vào giỏ hàng của người mua. |
|  | 8. Hệ thống cập nhật giỏ hàng và hiển thị lại thông tin giỏ hàng: danh sách sản phẩm, tổng chi phí giỏ hàng, khuyến mãi, chi phí vận chuyển, tổng giá trị đơn hàng |
| **9**. Người mua xem và tiến hành thanh toán. **A1** |  |
|  | 10.Hệ thống sẽ bắt đầu xác thực thông tin sản phẩm, xử lý thanh toán, và sau đó gửi email cho người mua. |
| **11.** Người mua nhận được email. Kết thúc |  |
|  | |
| **Luồng thay thế** | **A1.** Người mua chọn thêm sản phẩm hoặc xóa sản phẩm khỏi giỏ hàng, quay lại **bước 8** của luồng chính | |
| **Luồng ngoại lệ** | **E1.{** **Sản phẩm hết hàng}** Khi người mua nhấn "Mua hàng", hệ thống kiểm tra tồn kho. Nếu số lượng tồn kho = 0, hệ thống hiển thị thông báo “Sản phẩm đã hết hàng” và không thêm vào giỏ hàng. Quay lại **bước 3** trong luồng chính | |
| **Hậu điền kiện:** | * Giỏ hàng được cập nhật với đầy đủ thông tin tổng hợp và người mua nhận được email | |
| **Ghi chú** | * Người mua có thể thêm nhiều sản phẩm từ nhiều trang chi tiết khác nhau. * Quản trị viên (SysAdmin) có thể xem giỏ hàng của người mua và xử lý nếu có dấu hiệu bất thường. | |

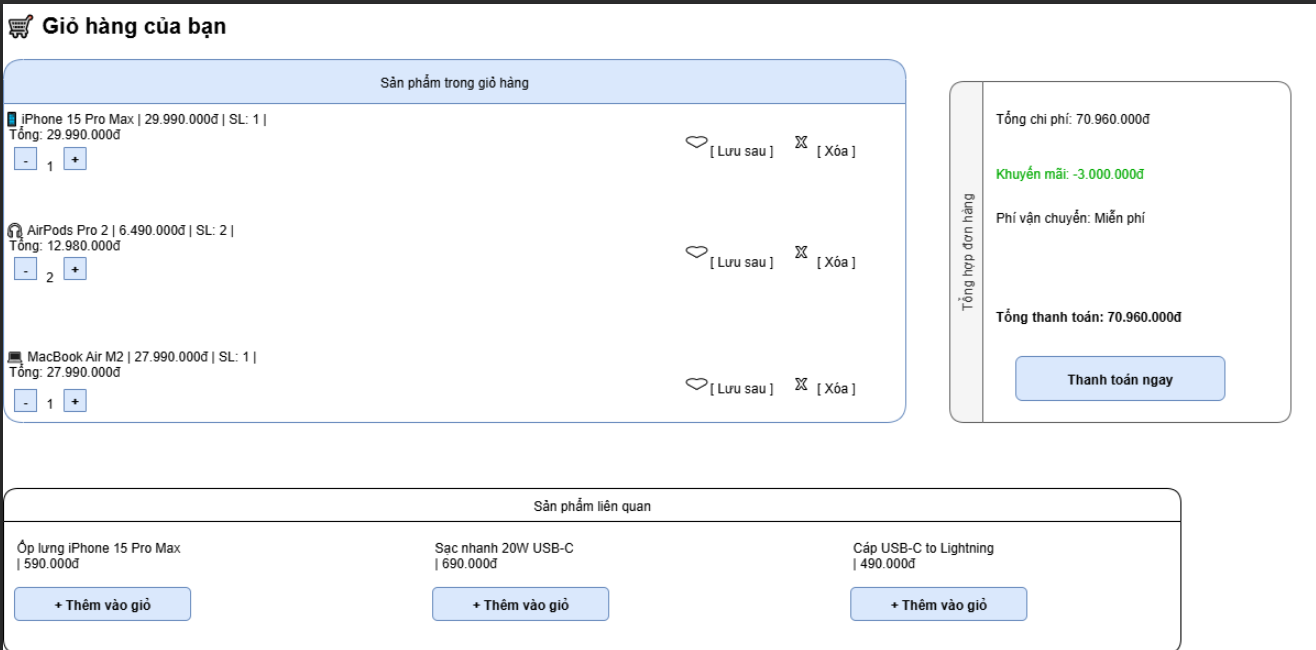
## Bài 5



## Bài 6.



## Bài 7.



## Bài 8.

**Kiến trúc phần mềm theo mô hình 3-tier**

**Lớp Presentation (UI/Web)**

* Web App:
  + Trang Danh mục: lọc/sắp xếp theo tên, giá; xem chi tiết nhanh (quick view).
  + Trang Chi tiết: tên, mô tả, tồn kho, thông tin kho (địa chỉ), “sản phẩm nổi bật”, đánh giá, nút “Mua hàng”.
  + Trang Giỏ hàng: danh sách dòng hàng + bảng tổng hợp (tổng chi phí, khuyến mãi, vận chuyển, tiết kiệm, tổng đơn).
  + Trang Thanh toán: xác thực thông tin + nút “Xác nhận thanh toán”.
  + Trang Quản trị: quản lý sản phẩm/kho, gán sản phẩm vào kho, xem/khóa giỏ hàng.
  + Trang Đăng nhập/đăng xuất: điều hướng Buyer → Catalog, SysAdmin → Admin.
* UI: chỉ xử lý hiển thị/validate cơ bản (required, định dạng), gọi API lớp Application.

**Lớp Application (Service/Business)**

* API Gateway (tuỳ chọn): gom các API, rate limit, auth.
* Các dịch vụ ứng dụng (có thể triển khai trong 1 service theo module hoặc tách nhỏ dần):
  1. CatalogService
     + Trả về thông tin hiển thị + tồn kho tổng hợp (lấy từ InventoryService).
  2. CartService
     + Thêm, cập nhật số lượng, xoá ...
     + Tự tính bảng tổng hợp: tổng trước KM, khuyến mãi tiết kiệm, phí ship, tổng đơn.
     + Gọi Pricing/PromotionService để tính khuyến mãi, InventoryService để reserve tạm khi thêm vào giỏ (tuỳ chính sách) hoặc chỉ kiểm tra khả dụng.
  3. CheckoutService
     + Xác thực tồn kho và giá, gọi PaymentService (mock/gateway), sau đó phát sự kiện “OrderPlaced”.
     + Gửi email qua NotificationService.
  4. InventoryService
     + CRUD kho; quản trị gán sản phẩm vào kho.
     + API kiểm tra/giảm tồn khi đặt hàng, hoàn trả khi huỷ.
  5. RatingService
     + Trung bình đánh giá sản phẩm.
  6. AuthService / AccessControl
     + Đăng nhập/đăng xuất; cấp JWT; phân quyền Buyer/SysAdmin (RBAC).
  7. AdminService
     + CRUD sản phẩm, gán sản phẩm vào kho hiện có.
     + Xem tất cả giỏ hàng, vô hiệu hoá giỏ không hợp lệ.
  8. NotificationService
     + Gửi email (SMTP/Email API) khi checkout.
  9. Pricing/PromotionService (nếu đã có logic KM/ship tách biệt)
     + Tính toán khuyến mãi, phí vận chuyển.
* Luồng chính:
  + UI → CartService: thêm sản phẩm → CartService kiểm tra hợp lệ + (tuỳ chọn) reserve tồn qua InventoryService → tính tổng hợp qua Pricing/PromotionService → trả giỏ + bảng tổng hợp.
  + Checkout: UI → CheckoutService → xác thực giá/tồn → PaymentService → nếu OK: Inventory trừ tồn, tạo Order, Notification gửi email.
* Tích hợp & giao tiếp:
  + Đồng bộ: REST/GraphQL giữa các service.
  + Bất đồng bộ: message queue (để gửi email/hậu xử lý sau thanh toán).

**Lớp Data (Persistence/Infra)**

* CSDL quan hệ (ví dụ PostgreSQL/MySQL):
  + Bảng: Products, Warehouses, ProductWarehouses (map sản phẩm-kho + tồn), Carts, CartItems, Orders, OrderItems, Users, Roles, Ratings, Promotions (tuỳ chọn).
* ORM/Repository: Repository + Unit of Work; transaction khi checkout.
* Indexing: chỉ mục theo name, price, productId, warehouseId, cartId.
* Seed Data lúc khởi động: tạo SysAdmin, 2 Buyer, dữ liệu mẫu sản phẩm/kho/đánh giá.
* Cache: Redis cho catalog và session giỏ khách (guest cart).
* Search mở rộng (tuỳ chọn): Elasticsearch/PG trigram cho tìm kiếm.

Luồng điển hình:

Thêm sản phẩm vào giỏ  
 UI → POST /cart/items (productId, qty) → CartService:

* Validate input & quyền.
* Gọi InventoryService kiểm tra tồn (theo kho mặc định hoặc best-match).
* Gộp dòng nếu đã tồn tại; cập nhật Cart + tính lại Summary (gọi Pricing/Promotion).
* Lưu DB; trả về giỏ + summary mới.

Thanh toán  
 UI → POST /checkout → CheckoutService:

* Xác thực giỏ & tồn kho lần cuối (lock/transaction).
* Trừ kho, tạo Order, gọi cổng thanh toán.

Phát sự kiện; Worker gửi email.

## Bài 9.

**Test-case cho chức năng “Thêm sản phẩm vào giỏ hàng”**

Giả định: API POST /carts/{userId}/items với payload { productId, quantity }. Hệ thống kiểm tra tồn kho khả dụng, cập nhật giỏ, và trả về giỏ + bảng tổng hợp.

**Test case 1 — Thêm sản phẩm hợp lệ**

* Dữ liệu đầu vào:
  + userId = U1 (Buyer hợp lệ, giỏ đang “active”)
  + productId = P1001 tồn kho khả dụng ≥ 5, quantity = 2
* Bước thực hiện:

Gọi POST /carts/U1/items với { productId: P1001, quantity: 2 }.

* Kết quả mong đợi:
  + HTTP 200.
  + cart.items có mục cho P1001 với qty = 2.
  + Bảng tổng hợp được cập nhật chính xác:
    - totalBeforeDiscount = price(P1001) \* 2
    - discount, shippingFee tính theo rule hiện hành
    - grandTotal = totalBeforeDiscount - discount + shippingFee
  + (Tuỳ chính sách) tồn kho không bị trừ ngay, hoặc reserve tạm được ghi nhận.
  + Log/audit tạo record “ADD\_ITEM”.

**Test case 2 — Vượt quá tồn kho khả dụng**

* Dữ liệu đầu vào:
  + userId = U1
  + productId = P1002 chỉ còn inventory = 3, yêu cầu quantity = 5
* Bước thực hiện:

POST /carts/U1/items với { productId: P1002, quantity: 5 }.

* Kết quả mong đợi:
  + HTTP 409 (Conflict) hoặc 400 kèm thông báo “Số lượng vượt quá tồn kho”.
  + Giỏ hàng không thay đổi.
  + Bảng tổng hợp giữ nguyên.
  + Không tạo reserve/không trừ tồn.

**Test case 3 — Sản phẩm không tồn tại/đã bị vô hiệu hoá**

* Dữ liệu đầu vào:
  + userId = U1
  + productId = P9999 (không tồn tại hoặc is\_active = false)
  + quantity = 1
* Bước thực hiện:

POST /carts/U1/items với { productId: P9999, quantity: 1 }.

* Kết quả mong đợi:
  + HTTP 404 (Not Found) hoặc 400 với thông báo “Sản phẩm không hợp lệ”.
  + Không có thay đổi nào trong giỏ.

**Test case 4 — Cộng dồn khi thêm trùng sản phẩm**

* Dữ liệu đầu vào:
  + Trong giỏ đã có P1001 với qty = 1.
  + Thêm { productId: P1001, quantity: 2 }.
* Bước thực hiện:

POST /carts/U1/items với { P1001, 2 }.

* Kết quả mong đợi:
  + Mục P1001 được cộng dồn thành qty = 3 nếu tổng ≤ tồn kho.
  + Bảng tổng hợp cập nhật chính xác, khuyến mãi (nếu bậc thang số lượng) được áp lại.

**Test case 5 — Kiểm thử quyền truy cập giỏ**

* Dữ liệu đầu vào:
  + userId = U2 cố gắng thêm vào giỏ của U1.
* Bước thực hiện:

POST /carts/U1/items với token của U2.

* Kết quả mong đợi:
  + HTTP 403 (Forbidden). Không thay đổi dữ liệu.