**TRƯỜNG ĐẠI HỌC SÀI GÒN**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**BÁO CÁO ĐỒ ÁN MÔN HỌC:**

**KIỂM THỬ PHẦN MỀM**

**Năm học 2025-2026**

**Tên đề tài:**

**Bài Tập Lab2**

**Giảng viên hướng dẫn: TS. Đỗ Như Tài**

Các thành viên trong nhóm:

1. Nguyễn Hoàn Báu - 3122411017

2. Võ Phương Liên Chi - 3122411022

3. Thái Thị Huỳnh Như - 3122411143

**Thành phố Hồ Chí Minh, năm 2025**

# **Bảng phân chia công việc**

|  |  |  |
| --- | --- | --- |
| **STT** | **Họ và tên** | **Công việc cần làm** |
| **1** | Nguyễn Hoàn Báu | **-** Bài 1.  - Bài 4.  - Bài 7:   * Phân tích yêu cầu, xác định thực thể và mối quan hệ. * Thiết kế CSDL mức Khái niệm (ERD).   - Bài tập ứng dụng:   * Câu 3: Xây dựng Use Case Diagram. * Câu 6: Thiết kế Class Diagram cho module Giỏ hàng. * Câu 7: Thiết kế giao diện màn hình "Quản lý giỏ hàng". |
| **2** | Võ Phương Liên Chi | - Bài 2.  - Bài 5.  - Bài 7:   * Chuyển sang CSDL mức Luận lý (Logical Model). * Viết mô tả bảng chi tiết (khóa chính, khóa ngoại).   - Bài tập ứng dụng:   * Câu 1: Liệt kê yêu cầu chức năng & phi chức năng. * Câu 4: Mô tả chi tiết Use Case "Mua sản phẩm từ trang chi tiết sản phẩm". * Câu 8: Đề xuất kiến trúc phần mềm 3 lớp. |
| **3** | Thái Thị Huỳnh Như | - Bài 3.  - Bài 6.  - Bài 7:   * Thiết kế CSDL mức Vật lý (Physical Model). * Thực hiện các câu lệnh Query (Insert, Update, Delete, Select) theo yêu cầu.   - Bài tập ứng dụng:   * Câu 2: Vẽ quy trình nghiệp vụ cho website. * Câu 5: Vẽ Activity Diagram quy trình mua hàng. * Câu 9: Viết 3 test case cho chức năng "Thêm sản phẩm vào giỏ hàng". |

**MỤC LỤC**

[**Bảng phân chia công việc** 2](#_Toc208854824)

[**Bài 1:** Vẽ lại quy trình nghiệp vụ sau (Độc giả, NV Tiếp tân, Thành viên) 5](#_Toc208854825)

[**Bài 2:** Vẽ mô hình khái niệm sau 6](#_Toc208854826)

[**Bài 3:** Vẽ lại business use case sau: 7](#_Toc208854827)

[**Bài 4:** Vẽ sơ đồ hoạt động và tương tác sau và tóm tắt ý nghĩa sơ đồ 10](#_Toc208854828)

[**Bài 5:** Vẽ lược đồ lớp cho bài toán Quản lý thư viện 12](#_Toc208854829)

[**Bài 6:** Vẽ lược đồ sau: 13](#_Toc208854830)

[**Bài 7:** 15](#_Toc208854831)

[**1. Thiết kế Cơ sở dữ liệu ở mức Khái niệm, Luận lý và Vật lý** 15](#_Toc208854832)

[**2. Query: Insert, Update, Delete, Select** 22](#_Toc208854833)

[**BÀI TẬP ỨNG DỤNG:** 25](#_Toc208854834)

[**1)Hãy liệt kê các yêu cầu chức năng và phi chức năng của hệ thống CoolStore dựa trên mô tả kịch bản nghiệp vụ.** 25](#_Toc208854835)

[**2) Vẽ quy trình nghiệp vụ cho website CoolStore.** 26](#_Toc208854836)

[**3) Xây dựng use case diagram cho website CoolStore.** 27](#_Toc208854837)

[**4) Mô tả chi tiết một Use Case "Mua sản phẩm từ trang chi tiết sản phẩm". Trình bày theo mẫu chuẩn gồm: Tên use case, Tác nhân chính, Mục tiêu, Tiền điều kiện, Luồng chính, Luồng thay thế, Hậu điều kiện, Ghi chú (nếu có).** 27](#_Toc208854838)

[**5) Sử dụng sơ đồ Activity Diagram để mô tả quy trình mua hàng từ khi người dùng nhấn “Mua hàng” cho đến khi hoàn tất thanh toán.** 30](#_Toc208854839)

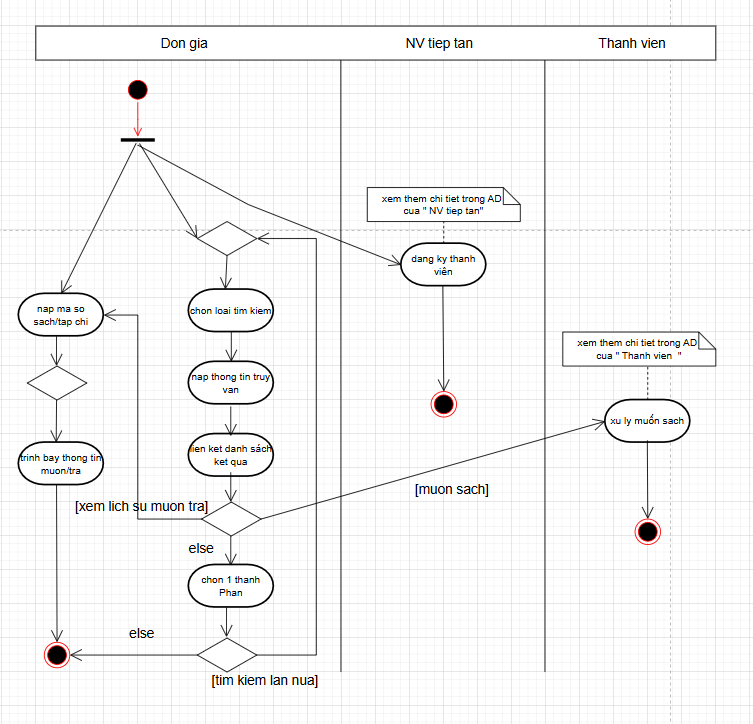
[**6) Thiết kế sơ đồ Class Diagram cho module Giỏ hàng.** 31](#_Toc208854840)

[**7) Thiết kế giao diện màn hình cho chức năng “Quản lý giỏ hàng” (có thể thiết kế một hoặc nhiều màn hình) gồm các thông tin:** 31](#_Toc208854841)

[**8) Đề xuất cách tổ chức kiến trúc phần mềm cho hệ thống CoolStore theo mô hình kiến trúc ba lớp (3-tier architecture).** 32](#_Toc208854842)

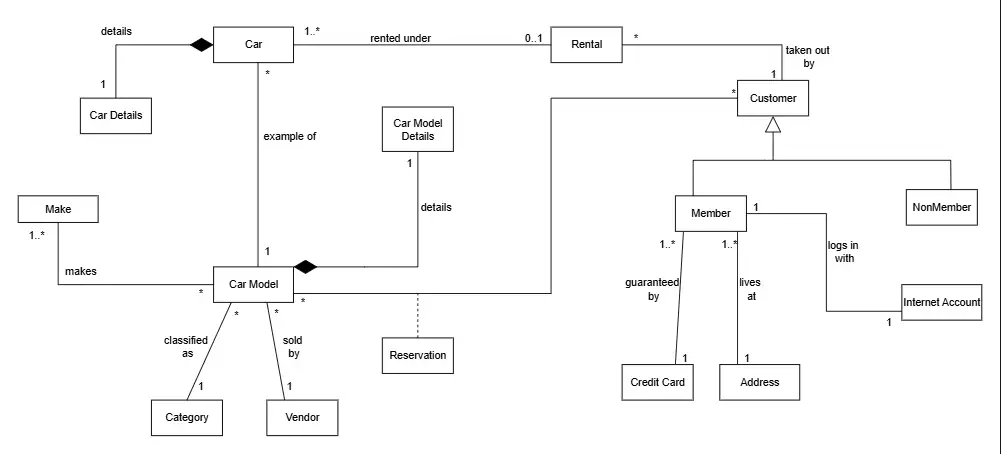
[**9) Viết ít nhất 3 ca kiểm thử (test cases) cho chức năng “Thêm sản phẩm vào giỏ hàng” – bao gồm dữ liệu đầu vào, bước thực hiện và kết quả mong đợi.** 33](#_Toc208854847)

**Bài 1:** Vẽ lại quy trình nghiệp vụ sau (Độc giả, NV Tiếp tân, Thành viên)

****

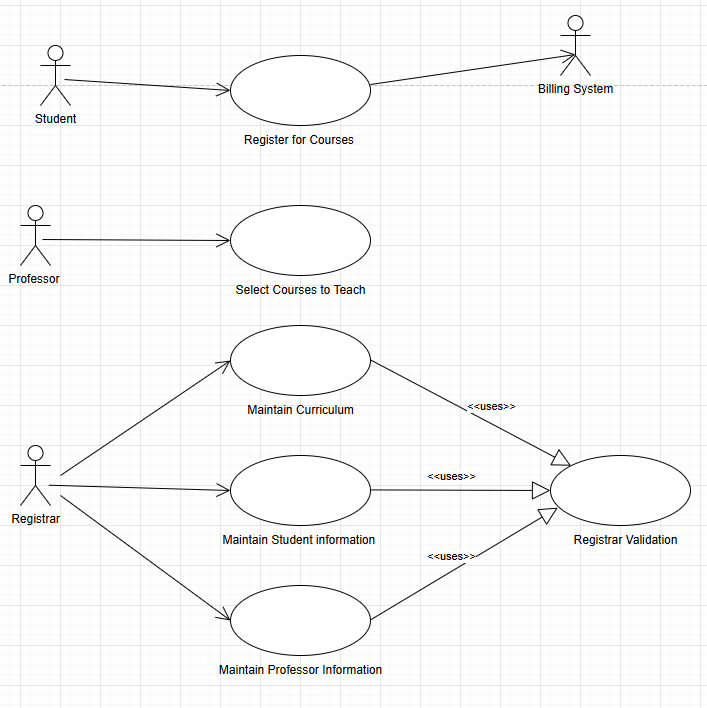
Sơ đồ này mô tả quy trình người dùng tra cứu và mượn sách trong thư viện. Người dùng có thể nhập mã sách để xem lịch sử mượn trả hoặc thực hiện tìm kiếm theo loại, nhập thông tin truy vấn và xem kết quả. Nếu chưa tìm thấy thì có thể tìm lại. Nhân viên tiếp tân có thể đăng ký thành viên mới, sau đó thành viên được quyền mượn sách.

# **Bài 2:** Vẽ mô hình khái niệm sau

****

Sơ đồ mô tả các lớp trong hệ thống thuê xe. Xe (Car) gắn với mẫu xe (Car Model), nhà sản xuất (Make), loại xe (Category), nhà cung cấp (Vendor) và chi tiết xe. Việc thuê xe (Rental) gắn với khách hàng (Customer), có thể là thành viên hoặc không. Thành viên có thông tin địa chỉ, thẻ tín dụng và tài khoản internet, đồng thời có thể đặt chỗ (Reservation).

# **Bài 3:** Vẽ lại business use case sau:

****

**a) Các tác nhân (Actors):**

- Student: Người dùng hệ thống với mục đích đăng ký môn học.

- Professor Người dùng hệ thống để chọn môn học sẽ giảng dạy và quản lý thông tin giảng dạy.

- Registrar: là tác nhân quản lý chương trình đào tạo, thông tin sinh viên, thông tin giảng viên, và thực hiện xác thực dữ liệu.

- Billing System: là hệ thống thanh toán bên ngoài, xử lý thanh toán học phí.

**b) Các Use-case:**

**-** Register for Courses: Sinh viên đăng ký môn học muốn học, hệ thống kết nối với Billing System để tính học phí của sinh viên.

- Select Courses to Teach: Giáo viên chọn môn học sẽ giảng dạy.

- Maintain Curriculum: Phòng đào tạo cập nhật, chỉnh sửa và quản lý nội dung chương trình học.

- Maintain Student Information: Cập nhật và quản lý dữ liệu sinh viên.

- Maintain Professor Information: Cập nhật và quản lý hồ sơ, thông tin giảng viên.

- Registrar Validation: Kiểm tra và xác nhận dữ liệu sinh viên, giảng viên và chương trình học khi có sự thay đổi.

**c) Các mối quan hệ:**

- Register for Courses có mối quan hệ với Billing System để tính và xử lý học phí.

- Maintain Student Information có quan hệ <<uses>> với Registrar Validation khi cập nhật thông tin sinh viên, cần xác thực tính hợp lệ.

- Maintain Professor Information cũng có quan hệ <<uses>> với Registrar Validation có nghĩa là khi thêm hoặc sửa thông tin giảng viên, dữ liệu sẽ được xác nhận hợp lệ trước khi lưu.

- Maintain Curriculum có quan hệ <<uses>> với Registrar Validation nghĩa là mỗi khi cập nhật chương trình đào tạo, hệ thống sẽ bắt buộc gọi chức năng kiểm tra hoặc xác thực để đảm bảo chương trình hợp lệ.

**d) Viết kịch bản chi tiết cho một trường hợp sử dụng cụ thể (ví dụ: "Register for Courses"):**

- Tên use case: Register for Courses.

- Tác nhân chính: Student.

- Tác nhân phụ: Billing System.

- Luồng chính (Main Flow):

1. Sinh viên đăng nhập vào hệ thống quản lý đào tạo.

2. Hệ thống hiển thị danh sách các môn học.

3. Sinh viên chọn các môn muốn đăng ký.

4. Hệ thống kiểm tra điều kiện (nếu có) của môn học.

5. Nếu hợp lệ, hệ thống sẽ gửi thông tin đăng ký đến Billing System để tính tiền học phí.

6. Billing System sẽ trả về kết quả tính toán.

7. Hệ thống xác nhận đăng ký thành công và lưu dữ liệu.

8. Sinh viên nhận được thông báo đăng ký hoàn tất cùng chi tiết học phí.

- Luồng thay thế (Alternative Flow):

+ Nếu sinh viên chọn môn học nhưng không đủ điều kiện (thiếu môn tiên quyết) thì hệ thống hiển thị thông báo lỗi.

+ Nếu Billing System gặp lỗi khi tính học phí, hệ thống sẽ thông báo cho sinh viên thử lại sau.

**e) Đề xuất cải tiến hoặc bổ sung thêm các use case:**

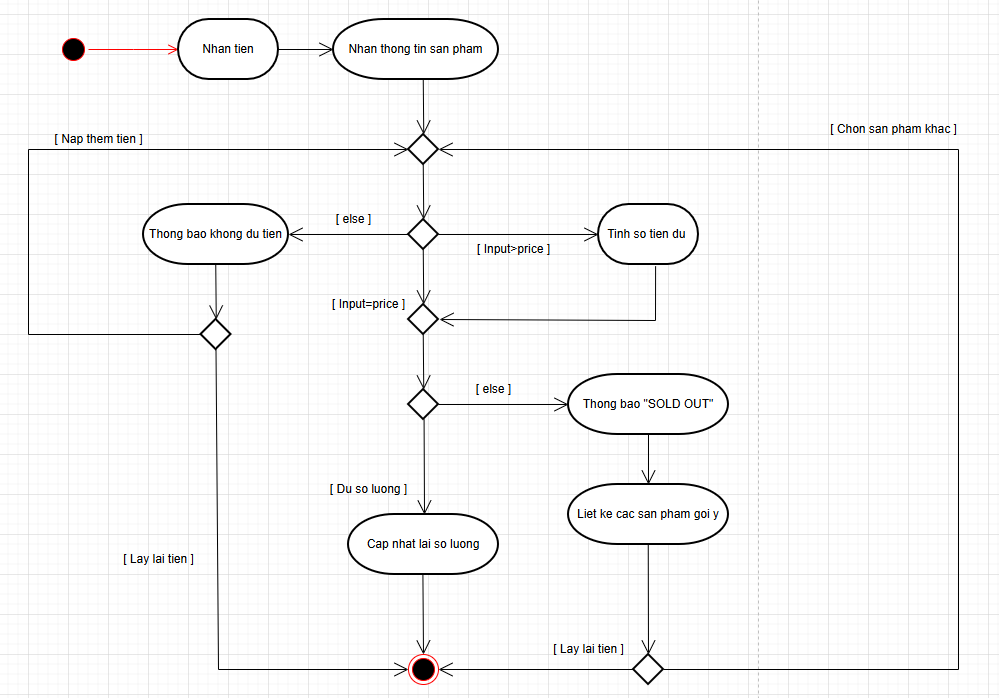
- Quản lý điểm và bảng điểm: Giảng viên nhập điểm cho sinh viên theo từng môn học. Sinh viên có thể xem điểm, in bảng điểm hoặc gửi yêu cầu phúc khảo.

**-** Báo cáo thống kê:

+ Hệ thống tự động tổng hợp báo cáo về số lượng sinh viên đăng ký, tình trạng môn học, tỷ lệ hoàn thành.

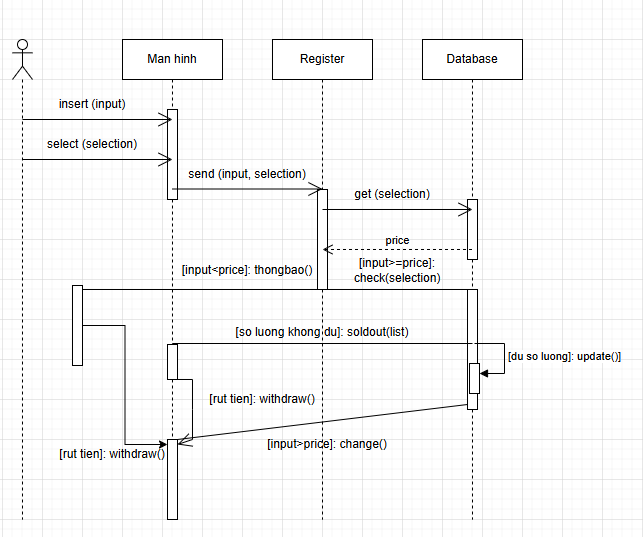
+ Hỗ trợ ban quản trị đưa ra quyết định điều chỉnh chương trình đào tạo.

# **Bài 4:** Vẽ sơ đồ hoạt động và tương tác sau và tóm tắt ý nghĩa sơ đồ

****

Tóm tắt ý nghĩa:

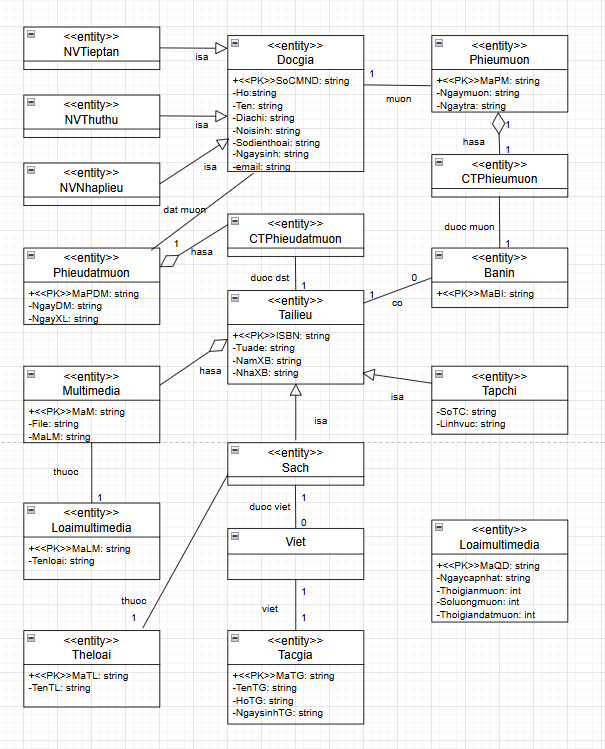
Trong sơ đồ hoạt động, đầu tiên khi người dùng nạp tiền vào và chọn sản phẩm. Hệ thống sẽ kiểm tra số tiền: nếu tiền không đủ thì thông báo lỗi và cho phép nạp thêm hoặc rút tiền. Còn nếu như đủ thì tính toán số tiền dư. Sau đó, hệ thống sẽ kiểm tra tồn kho sản phẩm, nếu sản phẩm đã hết hàng thì gửi thông báo “SOLD OUT”, sau đó liệt kê danh sách sản phẩm thay thế và cho phép người dùng chọn lại hoặc rút tiền. Và nếu còn hàng thì sẽ cập nhật lại số lượng, trừ tiền và hoàn tất giao dịch. Tóm lại, sơ đồ trên mô tả các nhánh xử lý khác nhau từ lúc nhập tiền cho tới khi giao dịch kết thúc.



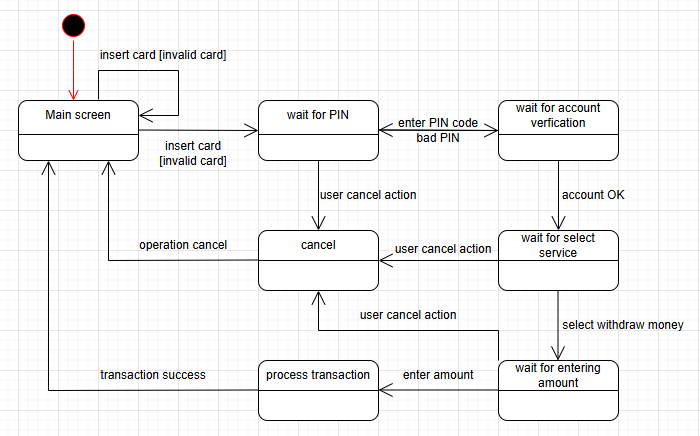
Tóm tắt ý nghĩa sơ đồ:

Trong sơ đồ tương tác, người dùng thao tác qua màn hình giao diện để nhập tiền và chọn sản phẩm. Màn hình gửi dữ liệu đến Register, Register sẽ truy vấn Database để lấy thông tin giá và số lượng sản phẩm. Từ kết quả đó, Register kiểm tra và phản hồi: báo “không đủ tiền” nếu số tiền thấp hơn giá, báo “SOLD OUT” nếu sản phẩm hết, hoặc cập nhật lại cơ sở dữ liệu khi sản phẩm còn hàng. Kết quả cuối cùng được trả về màn hình để hiển thị cho người dùng, đồng thời hỗ trợ rút lại tiền khi cần.

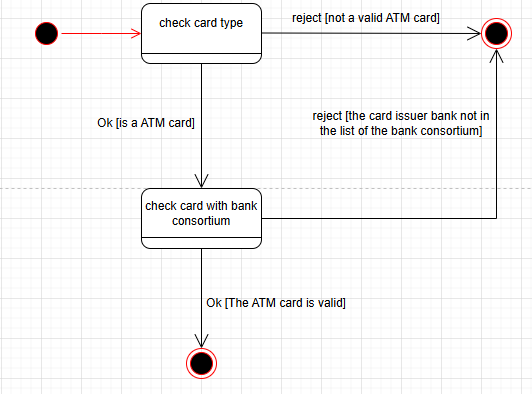
# **Bài 5:** Vẽ lược đồ lớp cho bài toán Quản lý thư viện



# **Bài 6:** Vẽ lược đồ sau:



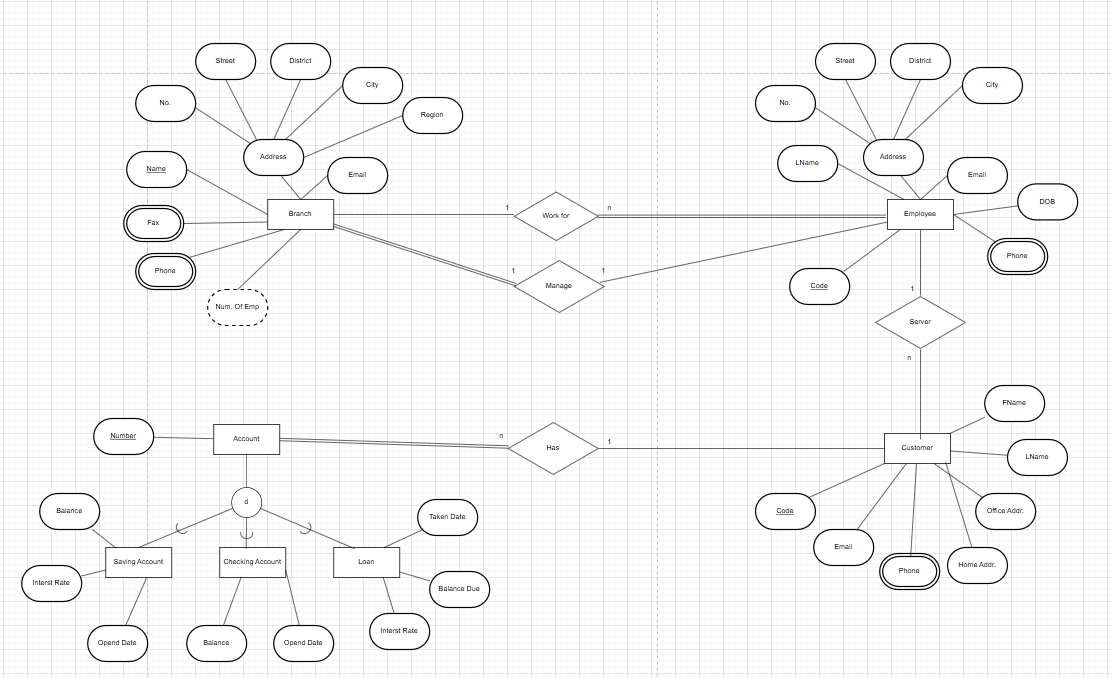
Sơ đồ thể hiện các bước khi khách hàng thực hiện giao dịch ATM: từ màn hình chính, người dùng đưa thẻ vào, nhập mã PIN, hệ thống xác minh tài khoản, chọn dịch vụ (rút tiền), nhập số tiền, xử lý giao dịch và trả kết quả. Người dùng có thể hủy bất kỳ lúc nào, hoặc thẻ/mã PIN không hợp lệ thì giao dịch bị hủy.



Sơ đồ mô tả quá trình kiểm tra tính hợp lệ của thẻ ATM: hệ thống xác định loại thẻ, nếu không hợp lệ thì từ chối. Nếu đúng thẻ ATM, hệ thống tiếp tục kiểm tra với liên minh ngân hàng. Nếu ngân hàng phát hành không thuộc liên minh thì từ chối, nếu hợp lệ thì thẻ được chấp nhận.

# **Bài 7:**

## **Thiết kế Cơ sở dữ liệu ở mức Khái niệm, Luận lý và Vật lý**



**Mức khái niệm (Conceptual Schema)**

* Thực thể và thuộc tính chính :

Branch (Chi nhánh): Name, AddressNo, Street, District, City, Region, Email, NumOfEmp.

BranchPhone: Phone (một chi nhánh có nhiều số điện thoại).

BranchFax: Fax (một chi nhánh có nhiều số fax).

Employee (Nhân viên): Code, FName, LName, DOB, Email, AddressNo, Street, District, City.

EmployeePhone: Phone (một nhân viên có nhiều số điện thoại).

Customer (Khách hàng): Code, FName, LName, OfficeAddress, HomeAddress, Email.

CustomerPhone: Phone (một khách hàng có nhiều số điện thoại).

Account (Tài khoản): Number.

SavingAccount: Balance, InterestRate, OpenedDate.

CheckingAccount: Balance, OpenedDate.

Loan: BalanceDue, InterestRate, TakenDate.

* Quan hệ và bội số (Cardinality) :

+ Branch – Employee:

* 1 chi nhánh có nhiều nhân viên (1–N).
* Mỗi nhân viên chỉ làm việc cho 1 chi nhánh.

+ Branch – Manager (Employee):

* 1 chi nhánh có đúng 1 nhân viên quản lý (1–1).
* Nhân viên quản lý chỉ quản lý chi nhánh của mình.

+ Employee – Customer:

* 1 nhân viên có thể phục vụ nhiều khách hàng (1–N).
* Mỗi khách hàng chỉ được phục vụ bởi 1 nhân viên.

+ Customer – Account:

* 1 khách hàng có thể mở nhiều tài khoản (1–N).
* Mỗi tài khoản chỉ thuộc về 1 khách hàng.

+ Account – SavingAccount / CheckingAccount / Loan:

* Mỗi tài khoản là một trong ba loại (1–1 chuyên biệt hóa).
* Một tài khoản chỉ thuộc đúng 1 loại: Tiết kiệm, Thanh toán hoặc Vay.

**Mức luận lý (Logical Schema)**

**-** Các bảng và ràng buộc quan hệ

1. BRANCH(Name, AddressNo, Street, District, City, Region, Email, NumOfEmp, MngCode)

* PK: Name
* FK: MngCode → EMPLOYEE(Code)

2. BRANCHPHONE(BraName, Phone)

* PK: (BraName, Phone)
* FK: BraName → BRANCH(Name)

3. BRANCHFAX(BraName, Fax)

* PK: (BraName, Fax)
* FK: BraName → BRANCH(Name)

4. EMPLOYEE(Code, FName, LName, AddressNo, Street, District, City, DOB, Email, BraName)

* PK: Code
* FK: BraName → BRANCH(Name)

5. EMPLOYEEPHONE(EmpCode, Phone)

* PK: (EmpCode, Phone)
* FK: EmpCode → EMPLOYEE(Code)

6. CUSTOMER(Code, FName, LName, OfficeAddress, HomeAddress, Email, EmpCode)

* PK: Code
* FK: EmpCode → EMPLOYEE(Code)

7. CUSTOMERPHONE(CusCode, Phone)

* PK: (CusCode, Phone)
* FK: CusCode → CUSTOMER(Code)

8. ACCOUNT(Number, CusCode)

* PK: Number
* FK: CusCode → CUSTOMER(Code)

9. SAVINGACCOUNT(AccNumber, Balance, InterestRate, OpenedDate)

* PK: AccNumber
* FK: AccNumber → ACCOUNT(Number)

10. CHECKINGACCOUNT(AccNumber, Balance, OpenedDate)

* PK: AccNumber
* FK: AccNumber → ACCOUNT(Number)

11. LOAN(AccNumber, BalanceDue, InterestRate, TakenDate)

* PK: AccNumber
* FK: AccNumber → ACCOUNT(Number)

**Mức Vật Lý ((Physical Schema)**

* **Bảng chi nhánh**

CREATE TABLE BRANCH (

Name VARCHAR(100) PRIMARY KEY,

AddressNo VARCHAR(20),

Street VARCHAR(100),

District VARCHAR(100),

City VARCHAR(100),

Region VARCHAR(100),

Email VARCHAR(150),

NumOfEmp INT DEFAULT 0,

MngCode VARCHAR(20) NULL

);

* **Điện thoại và fax chi nhánh**

CREATE TABLE BRANCHPHONE (

BraName VARCHAR(100),

Phone VARCHAR(30),

PRIMARY KEY (BraName, Phone),

FOREIGN KEY (BraName) REFERENCES BRANCH(Name) ON DELETE CASCADE

);

CREATE TABLE BRANCHFAX (

BraName VARCHAR(100),

Fax VARCHAR(30),

PRIMARY KEY (BraName, Fax),

FOREIGN KEY (BraName) REFERENCES BRANCH(Name) ON DELETE CASCADE

);

* **Nhân viên**

CREATE TABLE EMPLOYEE (

Code VARCHAR(20) PRIMARY KEY,

FName VARCHAR(50),

LName VARCHAR(50),

AddressNo VARCHAR(20),

Street VARCHAR(100),

District VARCHAR(100),

City VARCHAR(100),

DOB DATE,

Email VARCHAR(150),

BraName VARCHAR(100) NOT NULL,

FOREIGN KEY (BraName) REFERENCES BRANCH(Name)

);

* **Điện thoại nhân viên**

CREATE TABLE EMPLOYEEPHONE (

EmpCode VARCHAR(20),

Phone VARCHAR(30),

PRIMARY KEY (EmpCode, Phone),

FOREIGN KEY (EmpCode) REFERENCES EMPLOYEE(Code) ON DELETE CASCADE

);

* **Khách hàng**

CREATE TABLE CUSTOMER (

Code VARCHAR(20) PRIMARY KEY,

FName VARCHAR(50),

LName VARCHAR(50),

OfficeAddress VARCHAR(200),

HomeAddress VARCHAR(200),

Email VARCHAR(150),

EmpCode VARCHAR(20) NULL,

FOREIGN KEY (EmpCode) REFERENCES EMPLOYEE(Code) ON DELETE SET NULL

);

* **Điện thoại khách hàng**

CREATE TABLE CUSTOMERPHONE (

CusCode VARCHAR(20),

Phone VARCHAR(30),

PRIMARY KEY (CusCode, Phone),

FOREIGN KEY (CusCode) REFERENCES CUSTOMER(Code) ON DELETE CASCADE

);

* **Tài khoản**

CREATE TABLE ACCOUNT (

Number VARCHAR(20) PRIMARY KEY,

CusCode VARCHAR(20) NOT NULL,

FOREIGN KEY (CusCode) REFERENCES CUSTOMER(Code)

);

* **Các loại tài khoản**

CREATE TABLE SAVINGACCOUNT (

AccNumber VARCHAR(20) PRIMARY KEY,

Balance DECIMAL(18,2) NOT NULL,

InterestRate DECIMAL(5,2) NOT NULL,

OpenedDate DATE NOT NULL,

FOREIGN KEY (AccNumber) REFERENCES ACCOUNT(Number) ON DELETE CASCADE

);

CREATE TABLE CHECKINGACCOUNT (

AccNumber VARCHAR(20) PRIMARY KEY,

Balance DECIMAL(18,2) NOT NULL,

OpenedDate DATE NOT NULL,

FOREIGN KEY (AccNumber) REFERENCES ACCOUNT(Number) ON DELETE CASCADE

);

CREATE TABLE LOAN (

AccNumber VARCHAR(20) PRIMARY KEY,

BalanceDue DECIMAL(18,2) NOT NULL,

InterestRate DECIMAL(5,2) NOT NULL,

TakenDate DATE NOT NULL,

FOREIGN KEY (AccNumber) REFERENCES ACCOUNT(Number) ON DELETE CASCADE

);

* **Liên kết Manager với chi nhánh**

ALTER TABLE BRANCH

ADD CONSTRAINT fk\_branch\_manager

FOREIGN KEY (MngCode) REFERENCES EMPLOYEE(Code);

## **2. Query: Insert, Update, Delete, Select**

**a) Dữ liệu mẫu (>= 4 bản ghi)**

INSERT INTO BRANCH VALUES

('District 1','12','Le Loi','District 1','Ho Chi Minh','South','d1@abcbank.com',0,NULL),

('District 3','88','Vo Van Tan','District 3','Ho Chi Minh','South','d3@abcbank.com',0,NULL),

('Danang','25','Bach Dang','Hai Chau','Da Nang','Central','dn@abcbank.com',0,NULL),

('Hanoi-CauGiay','7','Tran Thai Tong','Cau Giay','Ha Noi','North','hn@abcbank.com',0,NULL);

INSERT INTO EMPLOYEE VALUES

('N1001','Anh','Nguyen','10','Tran Hung Dao','District 1','Ho Chi Minh','1988-03-15','anh.nguyen@abcbank.com','District 1'),

('N1002','Binh','Tran','20','Nguyen Thi Minh Khai','District 3','Ho Chi Minh','1990-07-21','binh.tran@abcbank.com','District 3'),

('N1003','Chi','Le','5','Le Duan','Hai Chau','Da Nang','1985-03-02','chi.le@abcbank.com','Danang'),

('N1080','Dung','Pham','12','Cau Giay','Cau Giay','Ha Noi','1992-03-30','dung.pham@abcbank.com','Hanoi-CauGiay');

INSERT INTO CUSTOMER VALUES

('C2001','Peter','Johnson','Bitexco, D1','12 Nguyen Trai, D1','peter.j@ex.com','N1001'),

('C2002','Lan','Pham','Etown, Tn.Binh','1 Ba Thang Hai, D10','lan.pham@ex.com','N1002'),

('C2003','Minh','Ho','FHome, DN','25 Hoang Dieu, DN','minh.ho@ex.com','N1003'),

('C2004','Hoa','Le','Keangnam, HN','7 Xuan Thuy, CG','hoa.le@ex.com','N1080');

INSERT INTO ACCOUNT VALUES

('A3001','C2001'),('A3002','C2001'),

('A3003','C2002'),('A3004','C2003');

**b) Đổi họ nhân viên N1080 thành “Brown”**

UPDATE EMPLOYEE

SET LName = 'Brown'

WHERE Code = 'N1080';

**c) Xóa nhân viên N1080**

DELETE FROM EMPLOYEE

WHERE Code = 'N1080';

**d) Liệt kê số tài khoản của khách hàng Peter Johnson**

SELECT a.Number

FROM ACCOUNT a

JOIN CUSTOMER c ON c.Code = a.CusCode

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

**f) Tổng số dư tất cả SavingAccount của Peter Johnson**

SELECT SUM(s.Balance) AS TotalBalance

FROM SAVINGACCOUNT s

JOIN ACCOUNT a ON s.AccNumber = a.Number

JOIN CUSTOMER c ON a.CusCode = c.Code

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

# **BÀI TẬP ỨNG DỤNG:**

## **1)Hãy liệt kê các yêu cầu chức năng và phi chức năng của hệ thống CoolStore dựa trên mô tả kịch bản nghiệp vụ.**

Yêu cầu chức năng:

- Quản lý người dùng.

- Danh mục sản phẩm (Product Catalog).

- Giỏ hàng (Shopping Cart).

- Quy trình thanh toán (Payment Process).

- Tồn kho (Inventory).

- Đánh giá (Rating).

- Thiết lập ban đầu khi khởi động.

Yêu cầu phi chức năng:

**-** Hiệu năng.

- Độ sẵn sàng.

- Bảo mật.

- Khả năng mở rộng.

- Khả năng phục hồi.

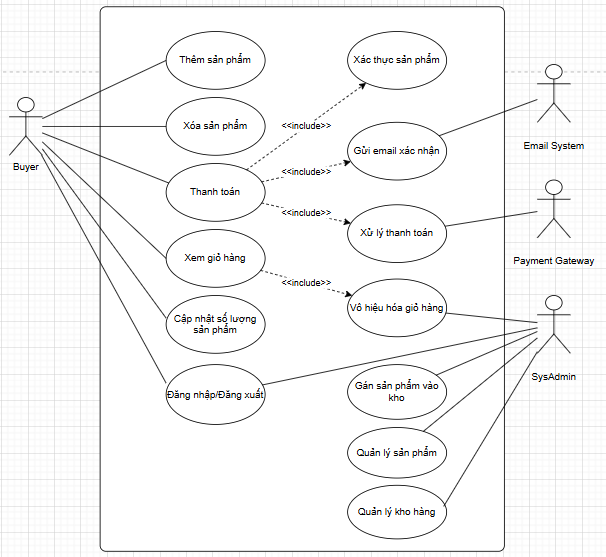
- Khả năng thao tác/UX.

- Bảo trì/Logging.

## **2) Vẽ quy trình nghiệp vụ cho website CoolStore.**

****

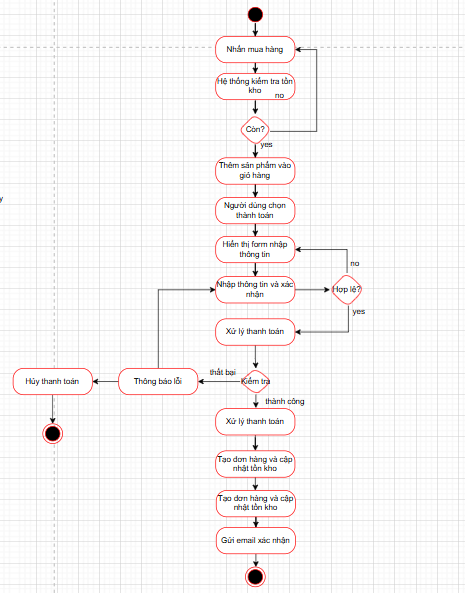
## **3) Xây dựng use case diagram cho website CoolStore.**

****

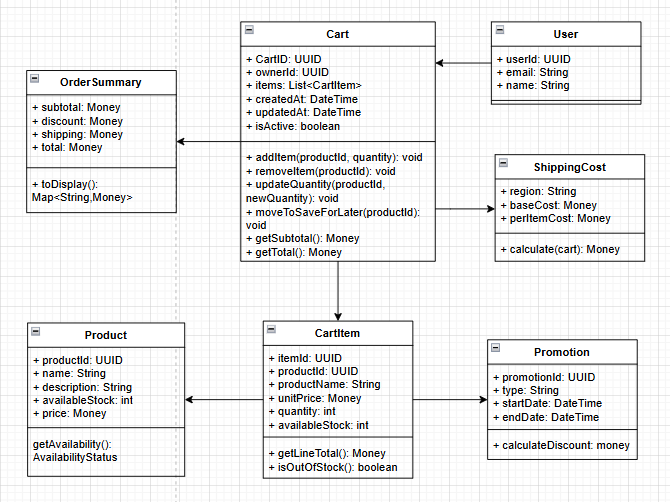
## **4) Mô tả chi tiết một Use Case "Mua sản phẩm từ trang chi tiết sản phẩm". Trình bày theo mẫu chuẩn gồm: Tên use case, Tác nhân chính, Mục tiêu, Tiền điều kiện, Luồng chính, Luồng thay thế, Hậu điều kiện, Ghi chú (nếu có).**

|  |  |  |
| --- | --- | --- |
| Tên Use Case: | Mua sản phẩm từ trang chi tiết sản phẩm | |
| Tác nhân chính: | Người mua hàng (đã đăng nhập) | |
| Mục tiêu: | Người mua chọn một sản phẩm cụ thể từ trang chi tiết, thêm vào giỏ hàng và tiến hành thanh toán để hoàn tất việc mua. | |
| Tiền điều kiện: | - Người mua đã đăng nhập vào hệ thống.  - Sản phẩm cần mua còn hàng.  - Hệ thống giỏ hàng và thanh toán đơn hàng đang hoạt động. | |
| Luồng chính | Actor action | System response |
| 1. Buyer truy cập vào trang chi tiết sản phẩm. |  |
|  | 2. Hệ thống hiển thị thông tin sản phẩm (tên, mô tả, giá, hình ảnh, tồn kho, đánh giá). |
| 3. Buyer chọn số lượng mong muốn. |  |
| 4. Buyer nhấn nút “Thêm vào giỏ hàng”. |  |
|  | 5. Hệ thống kiểm tra tồn kho:  - Nếu còn đủ hàng, sản phẩm được thêm vào giỏ hàng.  - Nếu không đủ hàng → chuyển sang luồng thay thế (A1). |
| 6. Buyer mở giỏ hàng và chọn “Thanh toán”. |  |
|  | 7. Hệ thống hiển thị màn hình nhập thông tin giao hàng và phương thức thanh toán. |
| 8. Buyer nhập đầy đủ thông tin, xác nhận thanh toán. |  |
|  | 9. Hệ thống xử lý thanh toán thành công (qua cổng Payment). A2 |
|  | 10. Hệ thống tạo đơn hàng, giảm tồn kho tương ứng và gửi thông báo (email/xác nhận trên web). |
| 11. Buyer nhận được thông báo đơn hàng đã đặt thành công. |  |
| Luồng thay thế | A1 – Sản phẩm hết hàng hoặc không đủ tồn kho. Hệ thống hiển thị hết hàng và quay lại bước 4  A2 – Thanh toán thất bại. Hệ thống hiển thị thông báo “Thanh toán thất bại” và cho phép Buyer chọn phương thức thanh toán khác, hoặc hủy đơn hàng. Quay lại bước 8. | |
| Hậu điều kiện | - Nếu thành công: Đơn hàng được tạo, sản phẩm giảm tồn kho, thông tin đơn hàng lưu trong hệ thống, Buyer nhận được xác nhận.  - Nếu thất bại: Không có đơn hàng nào được tạo, giỏ hàng giữ nguyên (trừ khi Buyer xóa). | |
| Ghi chú: | Trường hợp người mua chưa đăng nhập, hệ thống yêu cầu đăng nhập/đăng ký trước khi thực hiện mua hàng. | |

## **5) Sử dụng sơ đồ Activity Diagram để mô tả quy trình mua hàng từ khi người dùng nhấn “Mua hàng” cho đến khi hoàn tất thanh toán.**

****

## **6) Thiết kế sơ đồ Class Diagram cho module Giỏ hàng.**

****

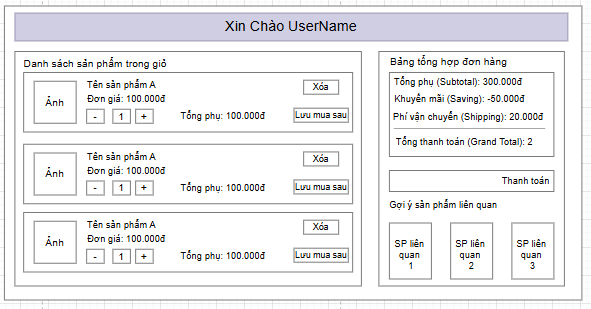
## **7) Thiết kế giao diện màn hình cho chức năng “Quản lý giỏ hàng” (có thể thiết kế một hoặc nhiều màn hình) gồm các thông tin:**

• Danh sách sản phẩm trong giỏ hàng: tên sản phẩm, hình ảnh, số lượng, giá từng món, tổng phụ.

• Chức năng thao tác nhanh: Tăng/giảm số lượng sản phẩm; Xóa sản phẩm khỏi giỏ; Lưu sản phẩm để mua sau.

• Bảng tổng hợp đơn hàng: Tổng chi phí giỏ hàng; Khuyến mãi tiết kiệm được; Phí vận chuyển; Tổng thanh toán cuối cùng.

• Hành động chính: Nút “Thanh toán”; Gợi ý sản phẩm liên quan để thêm vào giỏ.



## **8) Đề xuất cách tổ chức kiến trúc phần mềm cho hệ thống CoolStore theo mô hình kiến trúc ba lớp (3-tier architecture).**

## 1. Presentation Layer (Client):

- Chức năng: hiển thị danh mục, giỏ hàng, checkout, quản trị.

- Thành phần:

* Frontend Web: React.js (SPA) hoặc SSR Next.js.
* Mobile (nếu có): React Native / Flutter.
* Tương tác: gọi REST/GraphQL đến API Gateway.

- Yêu cầu phi chức năng:

* Responsive, accessible, tối ưu tốc độ, caching client-side (localStorage cho guest cart).

## 2. Application Layer (Business Logic / API)

- Chức năng: xử lý nghiệp vụ giỏ hàng, áp dụng promotion, tính phí vận chuyển, xác thực thanh toán, gửi email.

- Thành phần:

* API Gateway: phân phối request (ví dụ: Nginx / Kong / AWS API Gateway).
* Microservices / Monolith modular:
* Product Service (sản phẩm, kho)
* Cart Service (giỏ hàng) — lưu trạng thái tạm thời, session handling
* Order & Checkout Service (xác thực, tạo order, payment integration)
* Promotion Service
* User Service (auth, profile)
* Notification Service (email)

- Yêu cầu phi chức năng:

* Skalability (scale service Cart riêng), fault-tolerance, logging/monitoring, circuit-breaker cho external payment.

## 3. Data Layer

- Chức năng: lưu trữ sản phẩm, giỏ hàng (persistent), đơn hàng, kho, user.

- Thành phần:

* Relational DB: PostgreSQL hoặc MySQL cho dữ liệu chủ (product catalog, orders, users).
* NoSQL / Cache: Redis cho session & cart tạm thời.
* Search: Elasticsearch cho tính năng tìm kiếm,lọc,sắp xếp theo tên/giá.
* Backup & Replication: chiến lược backup, read-replicas cho tải đọc cao.

## - Giao tiếp giữa các lớp

* Frontend <-> API: HTTPS REST/GraphQL + JWT auth.
* Services nội bộ <-> DB: connection pooling, transactions khi cần (ví dụ reserve stock).
* Services <-> Queue: sự kiện asynchronous (OrderCreated -> InventoryService consume).

## **9) Viết ít nhất 3 ca kiểm thử (test cases) cho chức năng “Thêm sản phẩm vào giỏ hàng” – bao gồm dữ liệu đầu vào, bước thực hiện và kết quả mong đợi.**

### **Test Case 1: Thêm sản phẩm hợp lệ vào giỏ hàng**

- Dữ liệu đầu vào:

* Sản phẩm: Laptop A (productId = P1001)
* Giá: 15.000.000 VND
* Số lượng muốn thêm: 1 (trong khi còn tồn kho 10)

- Bước thực hiện:

* Người dùng đăng nhập hệ thống.
* Mở trang chi tiết sản phẩm Laptop A.
* Nhập số lượng = 1.
* Nhấn nút “Thêm vào giỏ hàng”.

- Kết quả mong đợi:

* Thông báo: “Đã thêm sản phẩm vào giỏ hàng thành công”.
* Giỏ hàng hiển thị Laptop A với số lượng = 1.
* Tổng tiền trong giỏ = 15.000.000 VND.

### **Test Case 2: Thêm sản phẩm vượt quá số lượng tồn kho**

- Dữ liệu đầu vào:

* Sản phẩm: Điện thoại B (productId = P2002)
* Tồn kho: 5
* Số lượng muốn thêm: 10

- Bước thực hiện:

* Người dùng đăng nhập.
* Mở trang chi tiết sản phẩm Điện thoại B.
* Nhập số lượng = 10.
* Nhấn nút “Thêm vào giỏ hàng”.

- Kết quả mong đợi:

* Hệ thống hiển thị thông báo lỗi: “Số lượng vượt quá tồn kho. Vui lòng chọn tối đa 5 sản phẩm”.
* Giỏ hàng không thêm sản phẩm mới.

### **Test Case 3: Nhập số lượng không hợp lệ (0 hoặc âm)**

- Dữ liệu đầu vào:

* Sản phẩm: Tai nghe Iphone (productId = P3003)
* Tồn kho: 20
* Số lượng muốn thêm: 0 (hoặc -2)

- Bước thực hiện:

* Người dùng đăng nhập.
* Mở trang chi tiết sản phẩm Tai nghe C.
* Nhập số lượng = 0 (hoặc -2).
* Nhấn nút “Thêm vào giỏ hàng”.

- Kết quả mong đợi:

* Hệ thống hiển thị thông báo: “Số lượng sản phẩm phải lớn hơn 0”.
* Giỏ hàng không thêm sản phẩm.