**TRƯỜNG ĐẠI HỌC SÀI GÒN**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**🙠🙣⚪🙡🙢**

**HỌC PHẦN: KIỂM THỬ PHẦN MỀM**

**PHÂN TÍCH THIẾT KẾ HỆ THỐNG COOLSTORE**

Giảng viên hướng dẫn: Đỗ Như Tài

Thành phố Hồ Chí Minh, ngày 11 tháng 9 năm 2024

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ tên** | **Mã số sinh viên** | **Công việc** |
| 1 | Nguyễn Phú Thành | 3122411191 | Bài tập minh họa: bài 1, 3, 6  Bài tập ứng dụng: câu 2, 7  Trình bày word |
| 2 | Nguyễn Hữu Vĩnh Phát | 3122411147 | Bài tập minh họa: bài 3, 4 |
| 3 | Liêu Tuấn Kiệt | 3122411105 | Bài tập minh họa: bài 2, 5 |
| 4 | Nguyễn Ngọc Hiếu | 3122411055 | Bài tập minh họa: bài 7  Bài tập ứng dụng: câu 1, 3, 4, 5, 6, 8, 9 |

# 

MỤC LỤC

[Phần 1: Bài tập minh họa 4](#_Toc208903828)

[Bài 1 4](#_Toc208903829)

[Bài 2 5](#_Toc208903830)

[Bài 3 5](#_Toc208903831)

[Bài 4 8](#_Toc208903832)

[Bài 5 9](#_Toc208903833)

[Bài 6 11](#_Toc208903834)

[Bài 7 11](#_Toc208903835)

[Thiết kế cơ sở dữ liệu ở mức khái niệm, mức luận lí, mức vật lí 11](#_Toc208903836)

[Query: insert, update, delete 14](#_Toc208903837)

[Phần 2: Bài tập ứng dụng 16](#_Toc208903838)

[Câu 1 16](#_Toc208903839)

[Câu 2 17](#_Toc208903840)

[Câu 3 17](#_Toc208903841)

[Câu 4 18](#_Toc208903842)

[Câu 5 19](#_Toc208903843)

[Câu 6 21](#_Toc208903844)

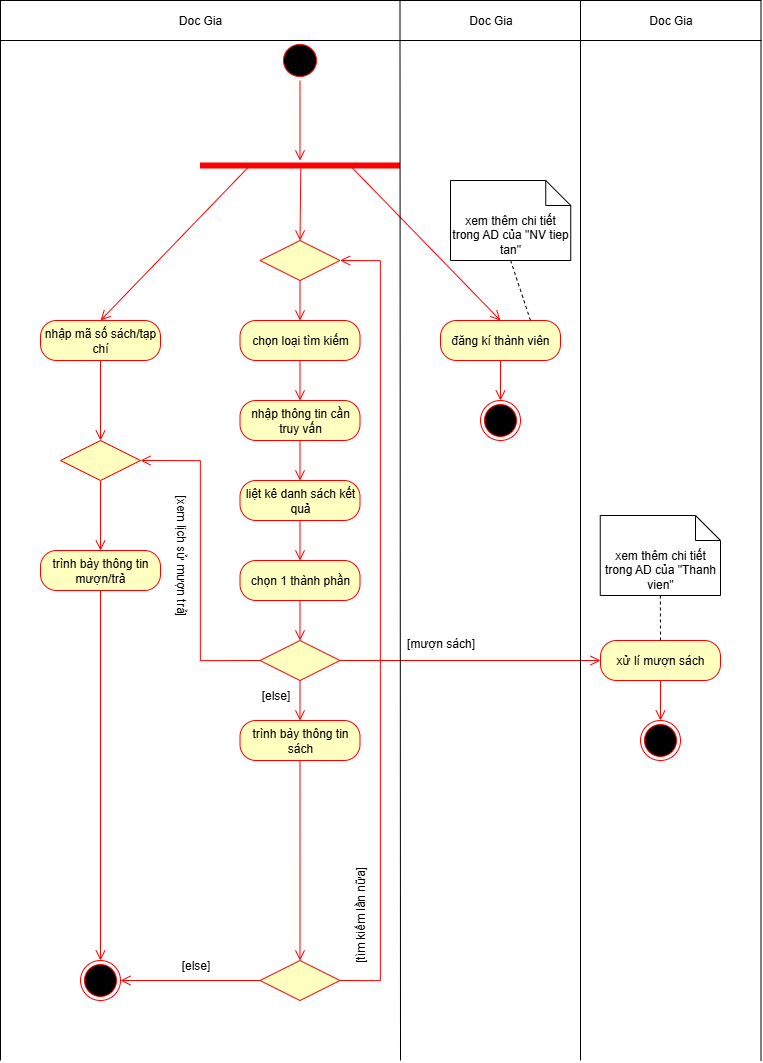
[Câu 7 21](#_Toc208903845)

[Câu 8 22](#_Toc208903846)

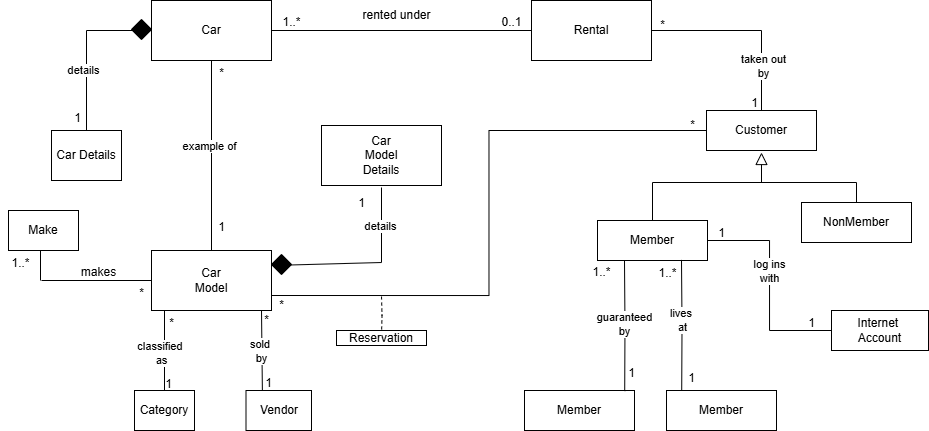
[Câu 9 23](#_Toc208903847)

# Phần 1: Bài tập minh họa

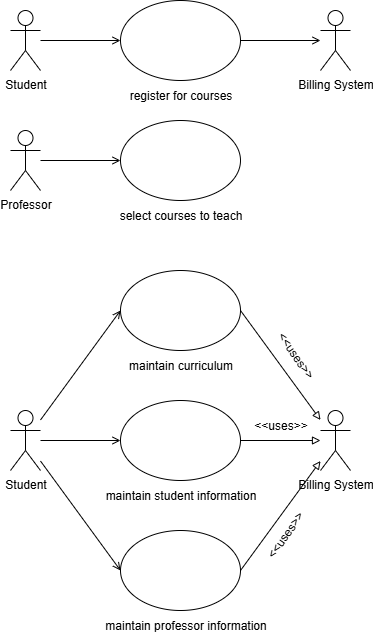
## Bài 1

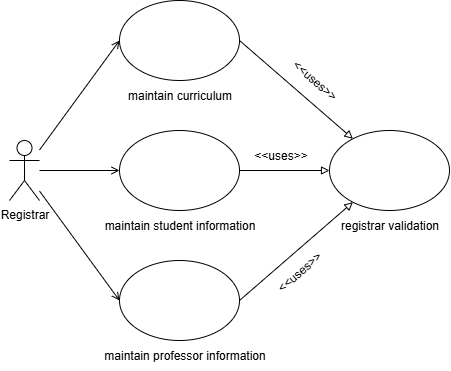


## Bài 2



## Bài 3





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

Có 4 actor:

* **Student**: Sinh viên, người thực hiện việc đăng ký học phần (Register for Courses).
* **Professor**: Giảng viên, người chọn các môn để dạy (Select Courses to Teach) và có thể liên quan đến việc duy trì chương trình học (Maintain Curriculum).
* **Registrar**: Nhân viên phòng đào tạo, người chịu trách nhiệm quản lý chương trình học (Maintain Curriculum), quản lý thông tin sinh viên (Maintain Student Information) và thông tin giảng viên (Maintain Professor Information).
* **Billing System**: Hệ thống thanh toán học phí, nhận dữ liệu từ quá trình đăng ký học phần để xử lý học phí.

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

* **Register for Courses**: sinh viên đăng kí các học phần
* **Select Courses to Teach**: giảng viên chọn các học phần mà mình sẽ giảng dạy
* **Maintain Curriculum**: cập nhật chương trình học (môn học, tín chỉ, yêu cầu tiên quyết, ...)
* **Maintain Student Information**: cập nhật, quản lí thông tin sinh viên
* **Maintain Professor Information**: cập nhật, quản lí thông tin giảng viên
* **Registrar Validation**: Hệ thống xác thực thông tin bởi nhân viên phòng đào tạo

1. Phân tích các mối quan hệ giữa các use case (bao gồm cả mối quan hệ <<uses>> và <<include>>)

* **Maintain Curriculum** <<uses>> **Registrar Validation** → Trước khi lưu thay đổi, chương trình học cần được xác thực.
* **Maintain Student Information** <<uses>> **Registrar Validation** → Trước khi lưu thay đổi, thông tin sinh viên cần được xác thực.
* Các use case khác hoạt động độc lập, có liên kết với các Actor khác nhau (Student, Professor, Billing System).

1. Viết kịch bản chi tiết cho 1 trường hợp sử dụng cụ thể (ví dụ: “Register for Courses” hoặc “Maintain Student Information”)

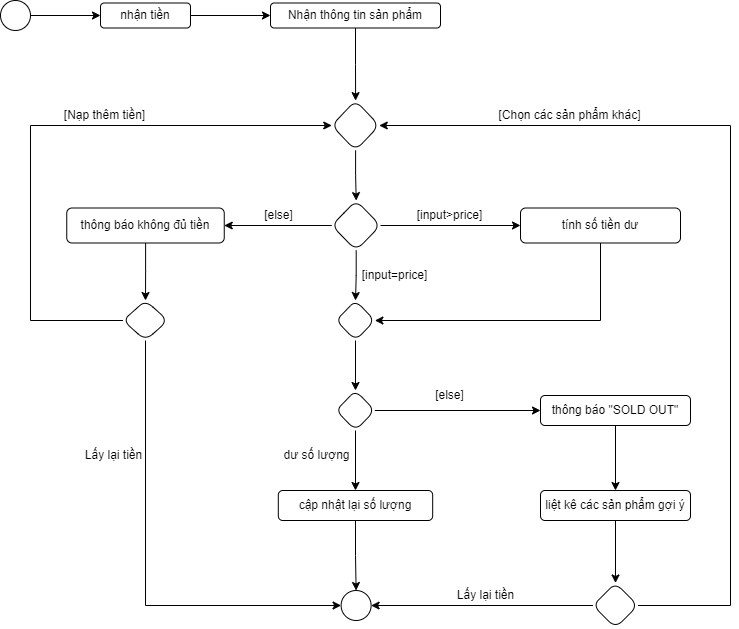
Use case: Register for Courses

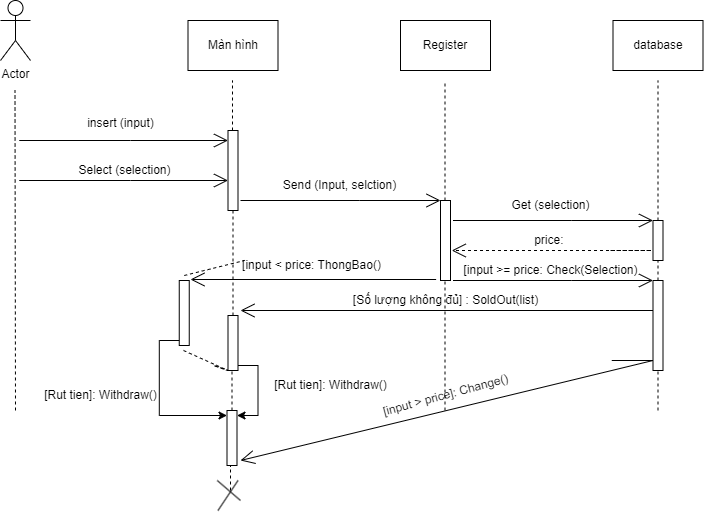
* Actor: Student
* Mục tiêu: Sinh viên đăng kí các học phần cho học kì mới
* Tiền điều kiện:
* Sinh viên đã đăng nhập hệ thống
* Học kì mới đã được mở để đăng kí
* Luồng sự kiện chính:
* Sinh viên chọn “Register for courses”
* Hệ thống hiển thị danh sách các học phần có thể đăng kí
* Sinh viên chọn các học phần muốn đăng kí
* Hệ thống kiểm tra điều kiện (Sinh viên đã hoàn thành học phần tiên quyết chưa? Có trùng lịch với học phần khác không?)
* Nếu hợp lệ, hệ thống ghi nhận đăng kí
* Hệ thống gửi đến Billing System để tính học phí
* Hệ thống báo cho sinh viên kết quả đăng kí thành công
* Luồng thay thế:
* Nếu học phần bị trùng lịch hoặc không thỏa điều kiện tiên quyết, thì hệ thống báo lỗi và yêu cầu chọn lại
* Kết quả sau cùng:
* Học phần được thêm vào thời khóa biểu của sinh viên
* Học phí được cập nhật

1. Đề xuất cải tiến hoặc bổ sung them các use case mà bạn thấy cần thiết

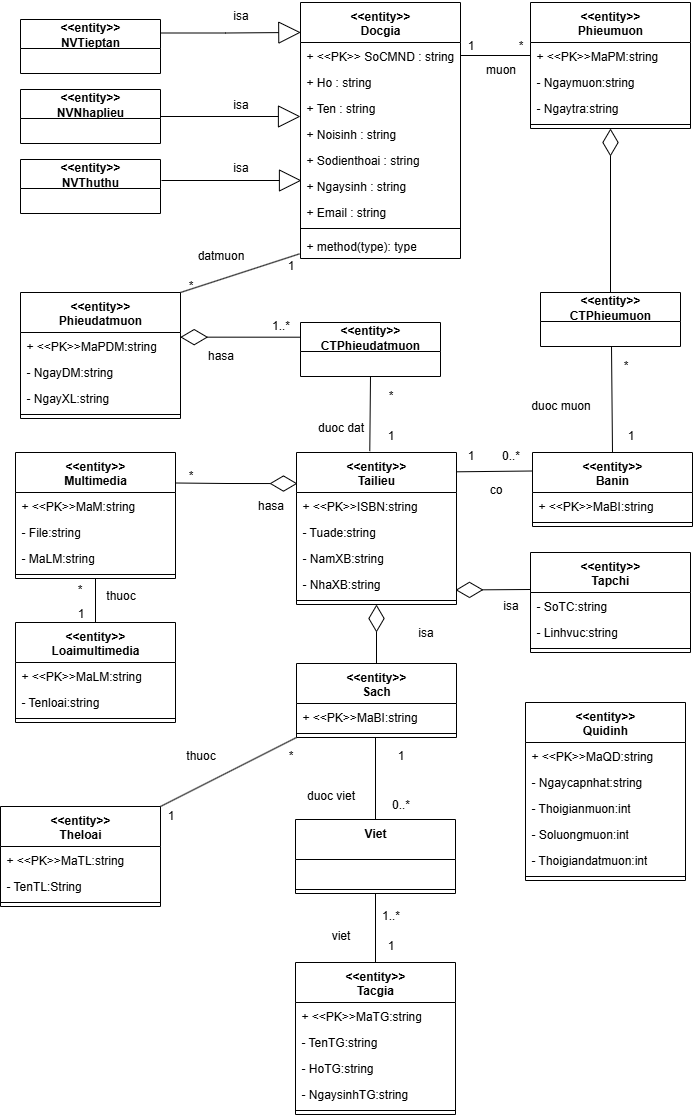
* Thêm use case **View Course Schedule** (Sinh viên xem thời khóa biểu mà mình đã đăng ký).
* Thêm use case **Drop Courses** (Sinh viên hủy học phần).
* Thêm use case **Approve Courses** (Cố vấn học tập duyệt các học phần mà sinh viên đăng kí).
* Thêm use case **Generate Reports** (Phòng đào tạo xuất báo cáo danh sách sinh viên, giảng viên, lớp học).

## Bài 4

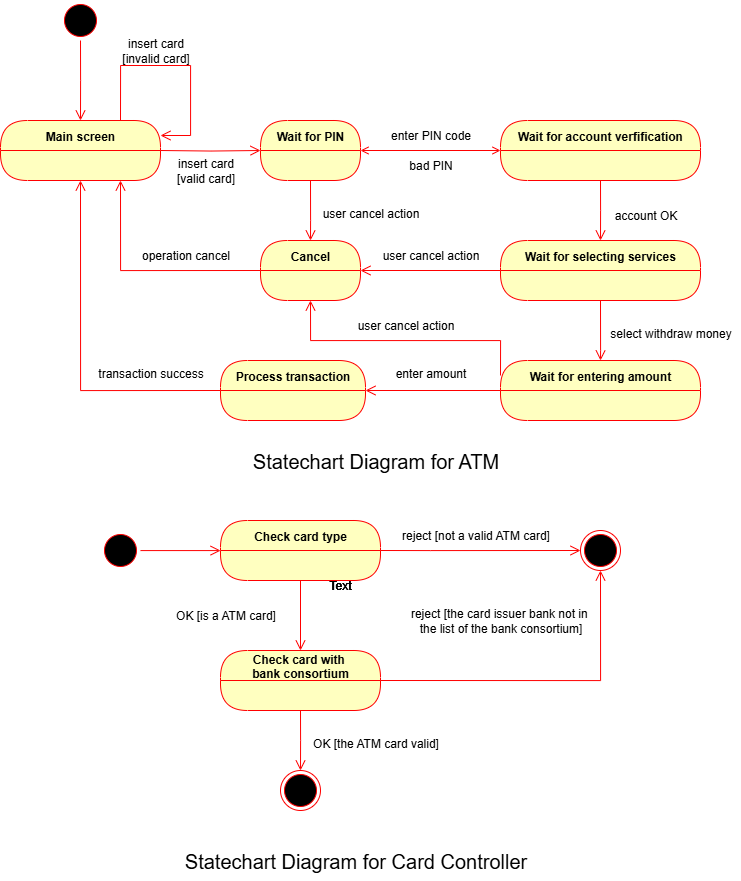




## Bài 5



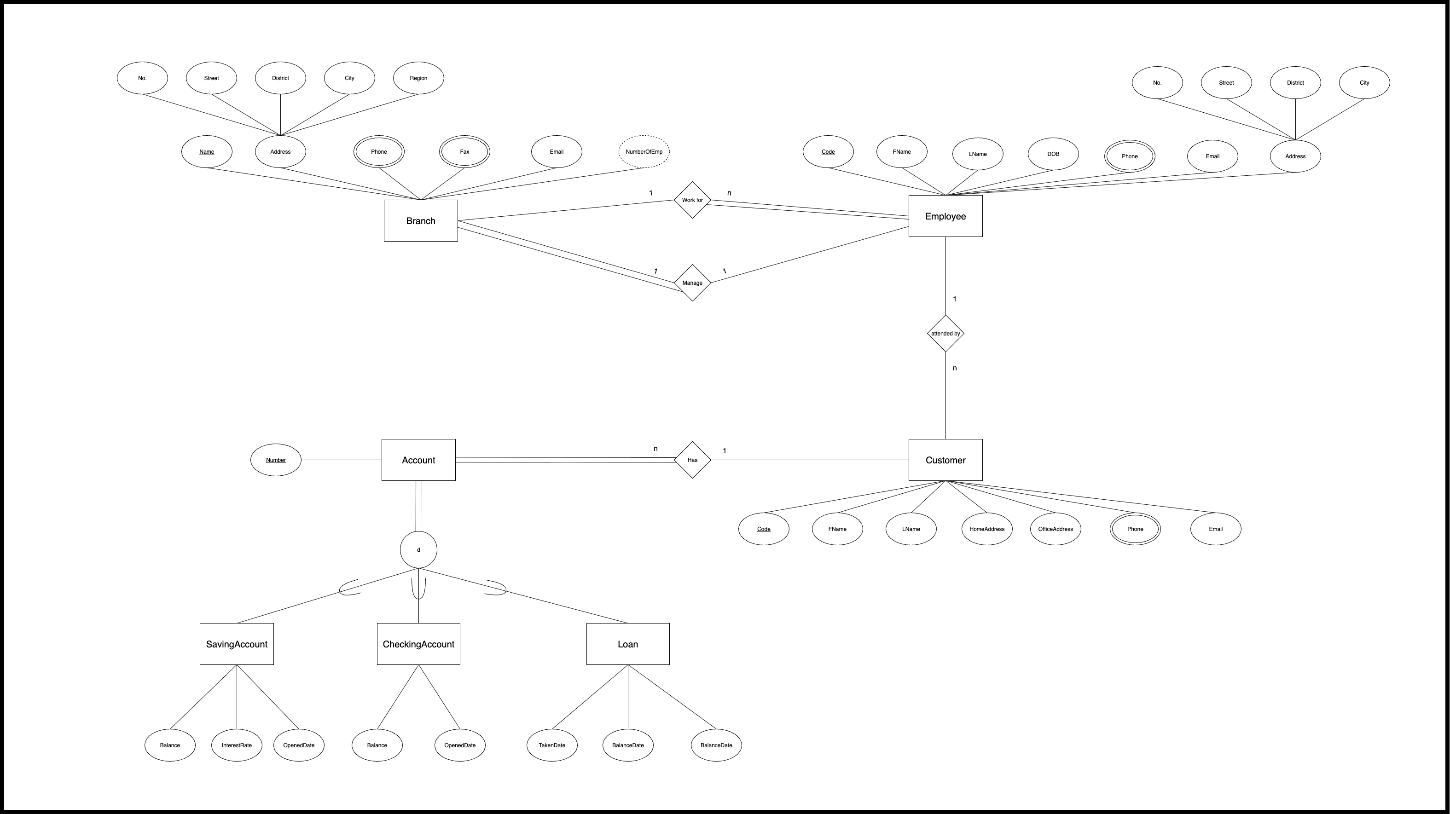
## Bài 6



## Bài 7

### Thiết kế cơ sở dữ liệu ở mức khái niệm, mức luận lí, mức vật lí

* Mức khái niệm



* Mức luận lí

Branch(**Name**, AddressNo, Street, District, City, Region, Email, MngCode)

BranchPhone(**BraName, Phone**)

BranchFax(**BraName, Fax**)

Employee(**Code**, FName, Lname, AddressNo, Street, District, City, DOB, Email, BraName)

EmployeePhone(**EmpCode, Phone**)

Customer(**Code**, FName, Lname, HomeAddress, OfficeAddress, Email, EmpCode)

CustomerPhone(**CusCode, Phone**)

Account(**Number**, CusCode)

SavingAccount(**AccNumber**, Balance, IntersestRate, OpenedDate)

CheckingAccount(**AccNumber**, Balance, OpenedDate)

Loan(**AccNumber**, BalanceDue, IntersestRate, TakenDate)

* Mức vật lí

Thực hiện cài đặt cơ sở dữ liệu bằng MySQL

-- Create database ABC\_Bank  
DROP DATABASE IF EXISTS ABC\_Bank;  
CREATE DATABASE ABC\_Bank;  
USE ABC\_Bank;  
  
-- Branch  
CREATE TABLE Branch (  
 Name VARCHAR(100) PRIMARY KEY,  
 AddressNo VARCHAR(20),  
 Street VARCHAR(100),  
 District VARCHAR(50),  
 City VARCHAR(50),  
 Region VARCHAR(50),  
 Email VARCHAR(100),  
 MngCode INT  
);  
  
-- Branch Phones  
CREATE TABLE BranchPhone (  
 BraName VARCHAR(100),  
 Phone VARCHAR(20),  
 PRIMARY KEY (BraName, Phone),  
 FOREIGN KEY (BraName) REFERENCES Branch(Name)  
);  
  
-- Branch Fax  
CREATE TABLE BranchFax (  
 BraName VARCHAR(100),  
 Fax VARCHAR(20),  
 PRIMARY KEY (BraName, Fax),  
 FOREIGN KEY (BraName) REFERENCES Branch(Name)  
);  
  
-- Employee  
CREATE TABLE Employee (  
 Code INT PRIMARY KEY,  
 FName VARCHAR(50),  
 LName VARCHAR(50),  
 AddressNo VARCHAR(20),  
 Street VARCHAR(100),  
 District VARCHAR(50),  
 City VARCHAR(50),  
 DOB DATE,  
 Email VARCHAR(100),  
 BraName VARCHAR(100),  
 FOREIGN KEY (BraName) REFERENCES Branch(Name)  
);  
  
-- Link Manager  
ALTER TABLE Branch  
ADD CONSTRAINT fk\_manager FOREIGN KEY (MngCode) REFERENCES Employee(Code);  
  
-- Employee Phones  
CREATE TABLE EmployeePhone (  
 EmpCode INT,  
 Phone VARCHAR(20),  
 PRIMARY KEY (EmpCode, Phone),  
 FOREIGN KEY (EmpCode) REFERENCES Employee(Code)  
);  
  
-- Customer  
CREATE TABLE Customer (  
 Code INT PRIMARY KEY,  
 FName VARCHAR(50),  
 LName VARCHAR(50),  
 HomeAddress VARCHAR(200),  
 OfficeAddress VARCHAR(200),  
 Email VARCHAR(100),  
 EmpCode INT,  
 FOREIGN KEY (EmpCode) REFERENCES Employee(Code)  
);  
  
-- Customer Phones  
CREATE TABLE CustomerPhone (  
 CusCode INT,  
 Phone VARCHAR(20),  
 PRIMARY KEY (CusCode, Phone),  
 FOREIGN KEY (CusCode) REFERENCES Customer(Code)  
);  
  
-- Account  
CREATE TABLE Account (  
 Number INT PRIMARY KEY,  
 CusCode INT,  
 FOREIGN KEY (CusCode) REFERENCES Customer(Code)  
);  
  
-- Saving Account  
CREATE TABLE SavingAccount (  
 AccNumber INT PRIMARY KEY,  
 Balance DECIMAL(15,2),  
 InterestRate DECIMAL(5,2),  
 OpenedDate DATE,  
 FOREIGN KEY (AccNumber) REFERENCES Account(Number)  
);  
  
-- Checking Account  
CREATE TABLE CheckingAccount (  
 AccNumber INT PRIMARY KEY,  
 Balance DECIMAL(15,2),  
 OpenedDate DATE,  
 FOREIGN KEY (AccNumber) REFERENCES Account(Number)  
);  
  
-- Loan  
CREATE TABLE Loan (  
 AccNumber INT PRIMARY KEY,  
 BalanceDue DECIMAL(15,2),  
 InterestRate DECIMAL(5,2),  
 TakenDate DATE,  
 FOREIGN KEY (AccNumber) REFERENCES Account(Number)  
);

### Query: insert, update, delete

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

Muốn insert phải gỡ bỏ rang buộc giữa MngCode và Employee

-- Branch  
INSERT INTO Branch VALUES  
('HCM01', '12', 'Le Loi', '1', 'HCM', 'South', 'hcm01@abc.com', 1001),  
('HN01', '25', 'Kim Ma', 'Ba Dinh', 'HN', 'North', 'hn01@abc.com', 1002),  
('DN01', '45', 'Tran Phu', 'Hai Chau', 'DN', 'Central', 'dn01@abc.com', 1003),  
('CT01', '78', 'Hoa Binh', 'Ninh Kieu', 'CT', 'West', 'ct01@abc.com', 1004);  
  
-- Employee  
INSERT INTO Employee VALUES  
(1001, 'John', 'Smith', '45', 'Nguyen Trai', '5', 'HCM', '1985-03-10', 'john.smith@abc.com', 'HCM01'),  
(1002, 'Anna', 'White', '22', 'Kim Ma', 'Ba Dinh', 'HN', '1990-07-15', 'anna.white@abc.com', 'HN01'),  
(1003, 'David', 'Johnson', '56', 'Tran Phu', 'Hai Chau', 'DN', '2010-03-20', 'david.j@abc.com', 'DN01'),  
(1004, 'Lisa', 'Green', '12', 'Hoa Binh', 'Ninh Kieu', 'CT', '2010-03-25', 'lisa.g@abc.com', 'CT01');  
  
-- Customer  
INSERT INTO Customer VALUES  
(2001, 'Peter', 'Johnson', 'FPT Tower HN', '12 Hoang Hoa Tham HN', 'peter.j@abc.com', 1002),  
(2002, 'Mary', 'Brown', 'Viettel HCM', '45 Tran Hung Dao HCM', 'mary.b@abc.com', 1001),  
(2003, 'Tom', 'Wilson', 'Sunrise DN', '56 Hai Phong DN', 'tom.w@abc.com', 1003),  
(2004, 'Laura', 'Nguyen', 'VinGroup HN', '89 Nguyen Chi Thanh HN', 'laura.n@abc.com', 1004);  
  
-- Account  
INSERT INTO Account VALUES  
(3001, 2001),  
(3002, 2001),  
(3003, 2002),  
(3004, 2003);  
  
-- Saving Account  
INSERT INTO SavingAccount VALUES  
(3001, 10000, 0.05, '2022-01-01'),  
(3002, 5000, 0.04, '2023-01-10'),  
(3003, 15000, 0.03, '2023-02-20'),  
(3004, 20000, 0.04, '2024-03-01');  
  
-- Checking Account  
INSERT INTO CheckingAccount VALUES  
(3002, 2000, '2023-01-10'),  
(3003, 1000, '2023-02-20'),  
(3004, 500, '2024-03-01'),  
(3001, 300, '2022-01-01');  
  
-- Loan  
INSERT INTO Loan VALUES  
(3001, 20000, 0.07, '2023-05-15'),  
(3002, 15000, 0.06, '2023-06-10'),  
(3003, 10000, 0.08, '2024-02-20'),  
(3004, 5000, 0.07, '2024-03-25');  
  
-- Branch Phone  
INSERT INTO BranchPhone VALUES  
('HCM01', '028-1111-2222'),  
('HN01', '024-3333-4444'),  
('DN01', '0236-5555-6666'),  
('CT01', '0292-7777-8888');  
  
-- Branch Fax  
INSERT INTO BranchFax VALUES  
('HCM01', '028-9999-1111'),  
('HN01', '024-8888-2222'),  
('DN01', '0236-7777-3333'),  
('CT01', '0292-6666-4444');  
  
-- Employee Phone  
INSERT INTO EmployeePhone VALUES  
(1001, '0901-111-111'),  
(1002, '0902-222-222'),  
(1003, '0903-333-333'),  
(1004, '0904-444-444');  
  
-- Customer Phone  
INSERT INTO CustomerPhone VALUES  
(2001, '0911-111-111'),  
(2002, '0912-222-222'),  
(2003, '0913-333-333'),  
(2004, '0914-444-444');

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

UPDATE Employee SET LName = 'BROWN' WHERE Code = 'N1080'

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 = 'N1080';

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

SELECT A.Number  
FROM `Account` AS A, Customer AS C  
WHERE C.LName = 'Johnson'  
AND C.FName = 'Peter'  
AND C.Code = A.CusCode

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

SELECT \*  
FROM Employee  
WHERE *YEAR*(Employee.DOB ) = 2010  
AND *MONTH*(Employee.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*(SV.Balance) AS 'Total balance of customer named Peter Johnson'  
FROM Account AS A, Customer AS C, SavingAccount AS SV  
WHERE C.FName = 'Peter' AND C.LName = 'Johnson'  
AND A.CusCode = C.Code  
AND SV.AccNumber = A.Number

# Phần 2: Bài tập ứng dụng

## Câu 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:

- Product Catalog: Xem danh mục các sản phẩm, lọc và sắp xếp theo tên/giá, phân trang

- Product Detail: Xem tên, mô tả, số lượng tồn kho, thông tin kho, trạng thái “Sản phẩm nổi bật” và đánh giá

- Admin product management: SysAdmin CRUD sản phẩm, gán sản phẩm vào kho hiện có

- Shopping Cart:

+ Thêm sản phẩm từ trang danh mục sản phẩm hoặc chi tiết sản phẩm

+ Xem bảng tổng hợp với thông tin: Tổng chi phí giỏ hàng, tổng chi phí sau khuyến mãi, chi phí vận chuyển, khuyến mãi tiết kiệm, tổng giá trị đơn hàng. (in realtime)

+ Thêm/Xoá sản phẩm khỏi giỏ hàng

- Checkout: Hệ thống xác thực thông tin sản phẩm, xử lý thanh toán, và gửi email cho người mua

- Rating: Buyer có thể gửi đánh giá (1–5 sao) cho sản phẩm.

- User Auth: Đăng ký/Đăng nhập/Đăng xuất

- Inventory: Quản lý kho hàng

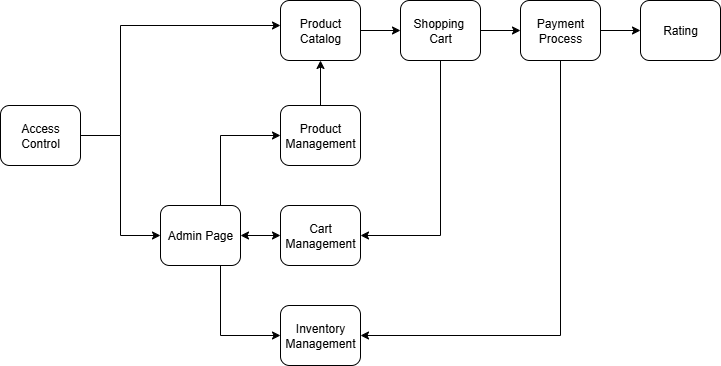
- Manage user’s carts: SysAdmin có thể xem bất kỳ giỏ hàng của bất kỳ người dùng nào, kích hoạt/vô hiệu hoá bất kỳ giỏ hàng nào không hợp lệ

* Yêu cầu phi chức năng:

- Initial Setup: Khi khởi động cần tạo một tài khoản admin và 2 người mua(Buyer), sản phẩm mẫu, kho và đánh giá cho các sản phẩm khởi tạo

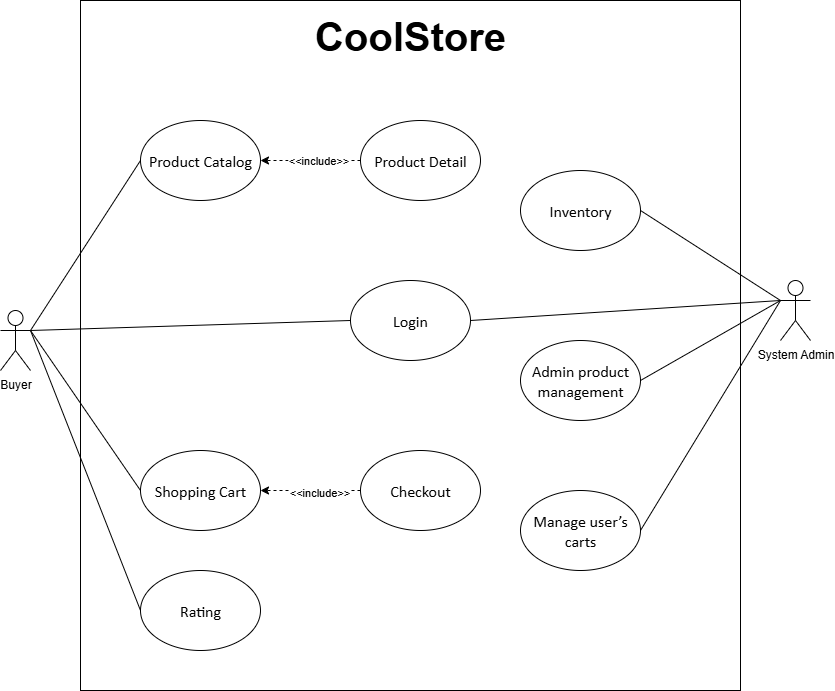
## Câu 2

Vẽ qui trình nghiệp vụ cho website CoolStore



## Câu 3

Xây dựng use case diagram cho website CoolStore



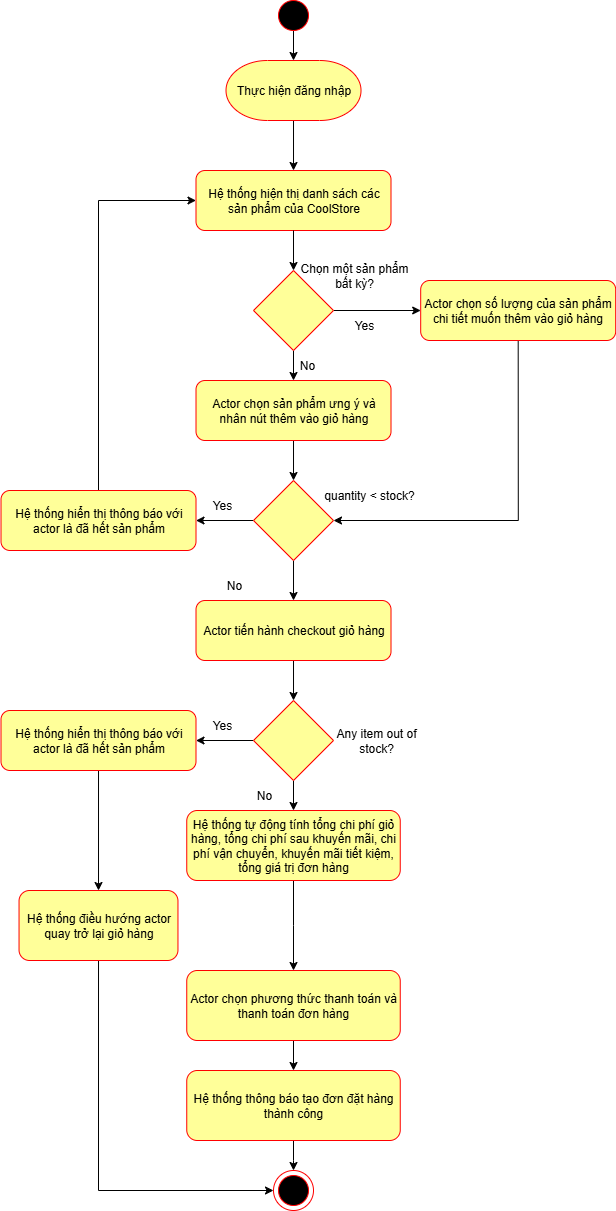
## Câu 4

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).

|  |  |  |
| --- | --- | --- |
|  | | |
| Use Case Number | UC1 | |
| Use Case Name | Mua sản phẩm từ trang chi tiết sản phẩm | |
| Actors | Buyer | |
| Maturity | Focused | |
| Summary | Tính năng này cho phép người mua có thể mua sản phẩm từ trang chi tiết sản phẩm | |
| Basic Course of Events | Actor Action | System Response |
| 1. Thực hiện xác thực đăng nhập |  |
|  | 2. Hệ thống hiển thị danh mục các sản phẩm của CoolStore Actor |
| 3. Use case bắt đầu khi Actor chọn một sản phẩm bất kỳ từ Product Catalog |  |
|  | 4. Hệ thống hiển thị chi tiết sản phẩm mà Actor chọn bao gồm tên, mô tả, số lượng tồn kho, thông tin kho, trạng thái “Sản phẩm nổi bật” và đánh giá |
| 5. Actor chọn số lượng cần mua và chọn thêm vào giỏ hàng. **A1 E1** |  |
|  | 6. Hệ thống thêm sản phẩm vào giỏ hàng và tự động tính tổng số tiền của giỏ hàng đó bao gồm subtotal |
| Alternative Paths | **A1.** Kiểm tra xem sản phẩm có đủ tồn kho để thêm vào giỏ hàng không. Nếu quantity > stock thì hệ thống sẽ báo lỗi và thông báo cho người dùng là không đủ hàng. Quay lại bước 4 của Basic Course of Events. | |
| Exception Paths | **E1**. Hệ thống kiểm tra xem số lượng mà người dùng nhập vào thêm vào phải là số dương hay không | |
| Extension Points | Không có | |
| Triggers | Người dùng muốn mua sản phẩm từ trang chi tiết sản phẩm | |
| Assumption | Đã có database của sản phẩm và sản phẩm chi tiết.  Người dùng đã có giỏ hàng lúc tạo tài khoản | |
| Pre-conditions | Không có | |
| Post Conditions | Nếu thành công thì sẽ đưa người dùng đến trang checkout để tiến hành thanh toán giỏ hàng.  Nếu thất bại thì không thêm sản phẩm vào giỏ hàng. | |
| Reference: Bunsiness Rules | Không có | |
| Author(s) | Hiếu | |
| Date | 15-09-2025 | |

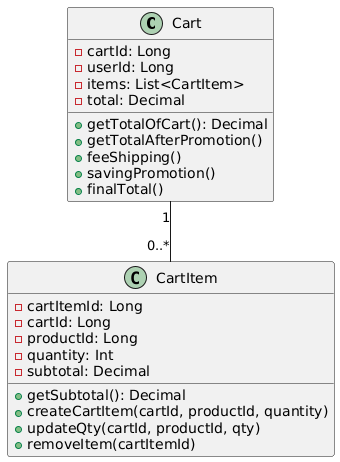
## Câu 5

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.



## Câu 6

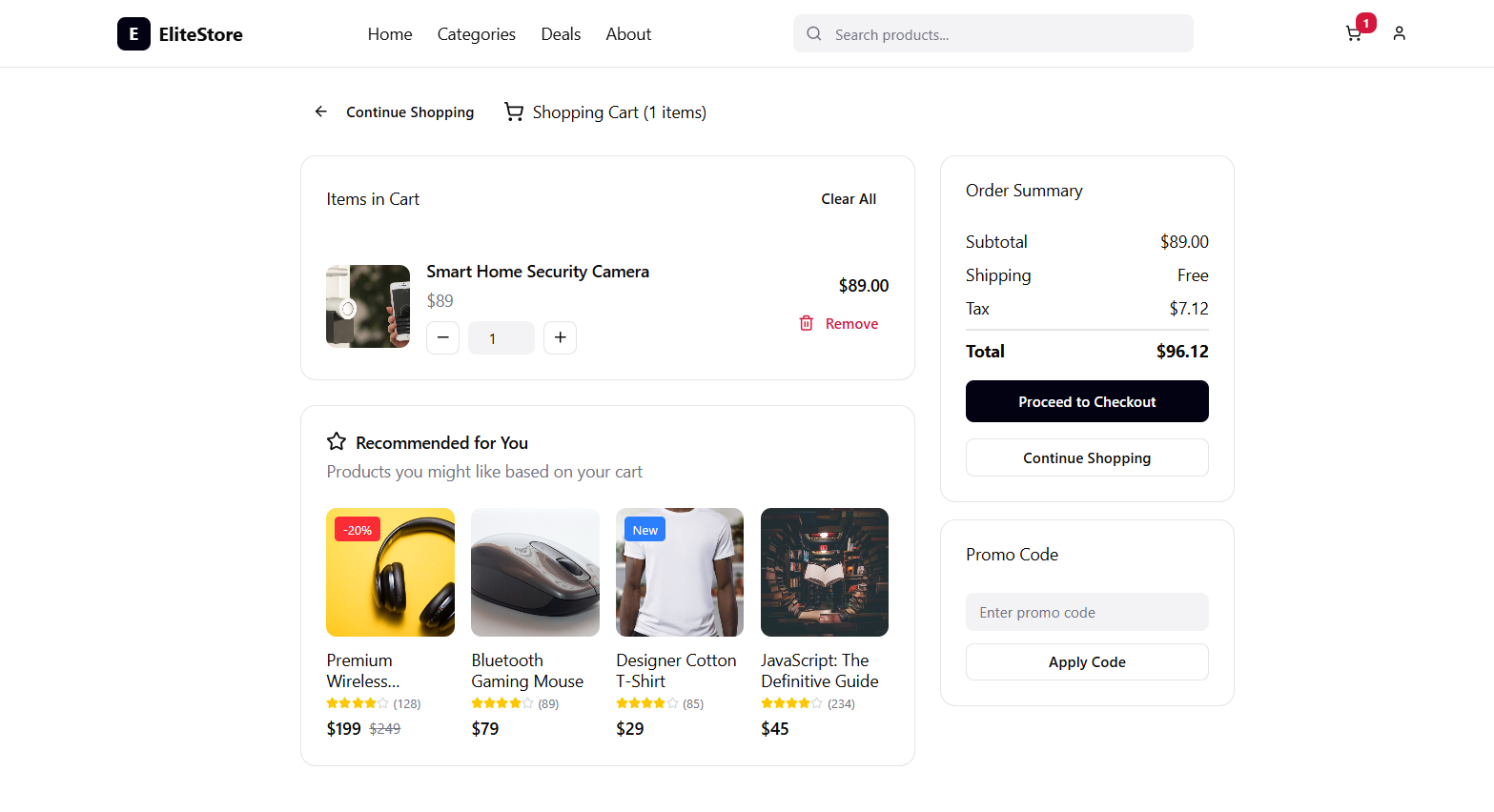
Thiết kế sơ đồ Class Diagram cho module Giỏ hàng.



## Câu 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 hang: Tổng chi phí giỏ hang; 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.



## Câu 8

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

**Layered 3-tier: Presentation (UI) → Application (Services) → Data (DB)**

* **Presentation Layer (Client)**
* React web app / mobile app.
* Responsibilities: render catalog, cart UI, validate forms, manage session, call APIs.
* **Application Layer (Backend API)**
* RESTful or gRPC microservices (or modular monolith):
* Catalog Service (product search/filter, product details, ratings).
* Cart Service (in-memory or persisted cart + merging, promotions).
* Checkout/Order Service (orchestrates inventory + payment + order creation).
* Inventory Service (manages stock per warehouse).
* User/Identity Service (authentication/authoriztation).
* Admin Service (product & warehouse management).
* Notification Service (send email).
* Use API Gateway for routing.
* Business logic, transactions, integration with PaymentGateway & WarehouseSystem.
* Use caching (Redis) for catalog hot data; use DB for persistent entities.
* **Data Layer**
* Relational DB for orders, carts (persisted), product metadata (MySQL/Postgres).
* NoSQL for product search/filters (Elasticsearch).
* Cache (Redis) for sessions & cart partial storage.
* Message queue (Kafka/RabbitMQ) for async tasks (send email, inventory updates).
* **Notes for consistency**
* Inventory & checkout need strong consistency: use transactions or SAGA pattern with compensation.
* For availability and scale: separate read models (CQRS) for catalog listing.

## Câu 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.

Mỗi test case gồm: ID, mục tiêu, dữ liệu đầu vào, các bước thực hiện, kết quả mong đợi.

TC1 – Thêm sản phẩm vào giỏ hàng khi đủ tồn kho (happy path)

* ID: TC\_ADD\_01
* Precondition: sản phẩm chi tiết PD300 có stock >= 5
* Test data: sản phẩm chi tiết PD300
* Steps:

1. Ở trang chi tiết sản phẩm PD300, click nút “Add to Cart” (quantity mặc định là 1)

2. Click icon cart để mở cart popup, điều chỉnh số lượng (ví dụ là quantity = 5)

* Expected:

1. Hệ thống trả về HTTP Code 200

2. Cart chứa CartItem với quantity = 5

3. Subtotal tăng tương ứng: unitPrice \* 5

4. Tồn kho vẫn giữ nguyên (chưa trừ tồn kho)

TC2 – Thêm sản phẩm vào giỏ hàng khi không đủ tồn kho

* ID: TC\_ADD\_02
* Precondition: sản phẩm chi tiết PD300 có stock = 1
* Test data: sản phẩm chi tiết PD300
* Steps:

1. Ở trang chi tiết sản phẩm, click nút “Add to Cart” (quantity default 1)

2. Click icon cart để mở cart popup, điều chỉnh số lượng (ví dụ là quantity = 5)

* Expected:

1. Hệ thống trả về HTTP code 4xx hoặc app-level error code

2. Hệ thống hiển thị thông báo lỗi: “Requested quantity exceeds available stock (only 1 left)”

3. Cart không chứa entry với quantity > 1. Cho phép thêm vào giỏ hàng với quantity tối đa là 1 hoặc từ chối hành động thêm vào giỏ hàng (tùy vào business rule)

4. Tồn kho vẫn giữ nguyên (chưa trừ tồn kho)

TC3 – Thêm cùng 1 sản phẩm nhiều lần (ví dụ là 2 lần)

* ID: TC\_ADD\_03
* Precondition: sản phẩm chi tiết PD300 có stock >= 2; cart chưa chứa PD300
* Test data: sản phẩm chi tiết PD300
* Steps:

1. Ở trang chi tiết sản phẩm, click nút “Add to Cart” (quantity default 1)

2. Thực hiện lại bước 1

* Expected:

1. Sau lần thêm 1: cart chứa PD300 với quantity = 1

2. Sau lần thêm 2: cart chứa PD300 với quantity = 2 (không nhân bản cart item)

3. Subtotal cập nhật tương ứng: 2 \* unitPrice