ỦY BAN NHÂN DÂN THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC SÀI GÒN

KHOA CÔNG NGHỆ THÔNG TIN



KIỂM THỬ PHẦN MỀM

BÀI TẬP 2: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

Sinh viên : Mssv:

Trần Đức Toản 3122411219

Đỗ Minh Triết 3122411224

Nguyễn Việt Đức 3122411044

Phạm Văn Tính 3122411214

Giảng viên hướng dẫn: Đỗ Như Tài

Thành phố Hồ Chí Minh,Tháng 9 năm 2025

Mục lục

[Phân công công việc](#_irl10f1ozjyc) 1

[Bài 1 : Qui trình nghiệp vụ](#_8m6te2neuy1) 2

[Bài 2 : Mô hình khái niệm](#_ufefbm2qn4jt) 3

[Bài 3 : Vẽ lại business use case](#_u70gkakqnml7) 4

[Bài 4 : Vẽ sơ đồ hoạt động và tương tác](#_gpg41ardpy6a) 7

[Bài 5 : Vẽ lược đồ lớp cho bài toán Quản lý thư viện](#_kikg5llzu2yd) 10

[Bài 6 : Vẽ lược đồ](#_5uublnesy4yu) 11

[Bài 7 Phân tích dữ liệu cho ABC Bank](#_m44hi0gss2w) 12

[Bài tập ứng dụng](#_n0ybq07p3hl5)**16**

[1) Liệt kê các yêu cầu chứ năng và phi chức năng của hệ thống](#_vdj6w4k7oum2) 16

[2) Quy trình nghiệp vụ](#_8xknbw42z3sy) 18

[3) Xây dựng use case diagram cho website CoolStore :](#_uylo9kktj1nn) 19

[4) Use-case mua hàng từ trang chi tiết sản phẩm](#_9bj3sorolvb2) 23

[5) Activity Diagram từ mua hàng tới thanh toán](#_nh00jbsukqyl) 26

[6) Thiết kế sơ đồ Class Diagram cho module giỏ hàng](#_cmu6206jwqpo) 27

[7) Thiết kế giao diện màn hình cho chức năng “Quản lý giỏ hàng” .](#_tsqoqfhgaruu) 27

[8) Đề xuất cách tổ chức kiến trúc phần mềm.](#_jfj3wpenotag) 28

[9) Viết test case kiểm thử](#_f1nxatb7mvyj) 32

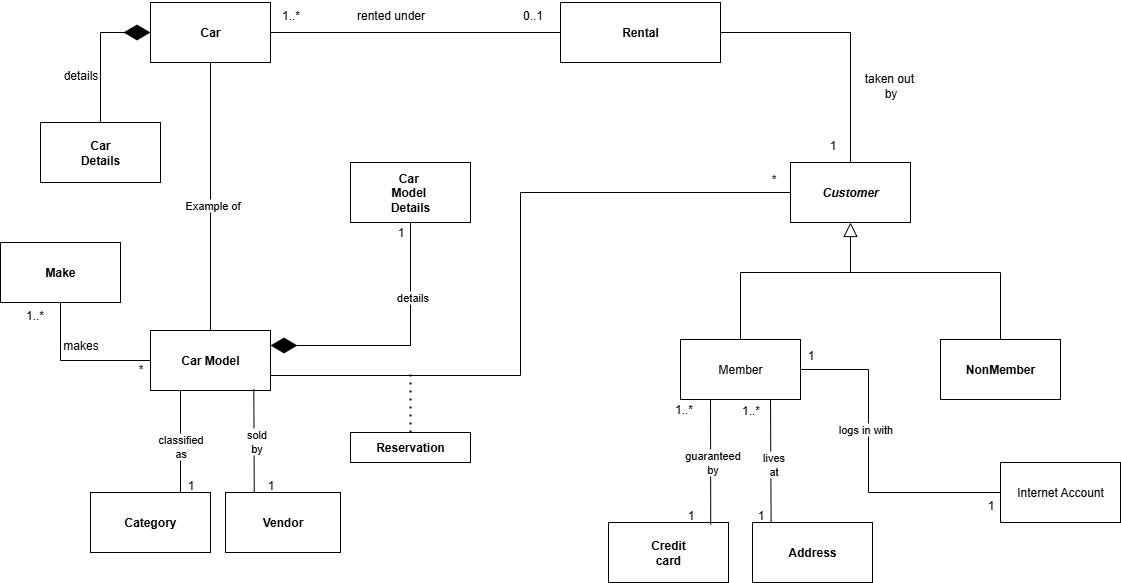
# 

# Phân công công việc

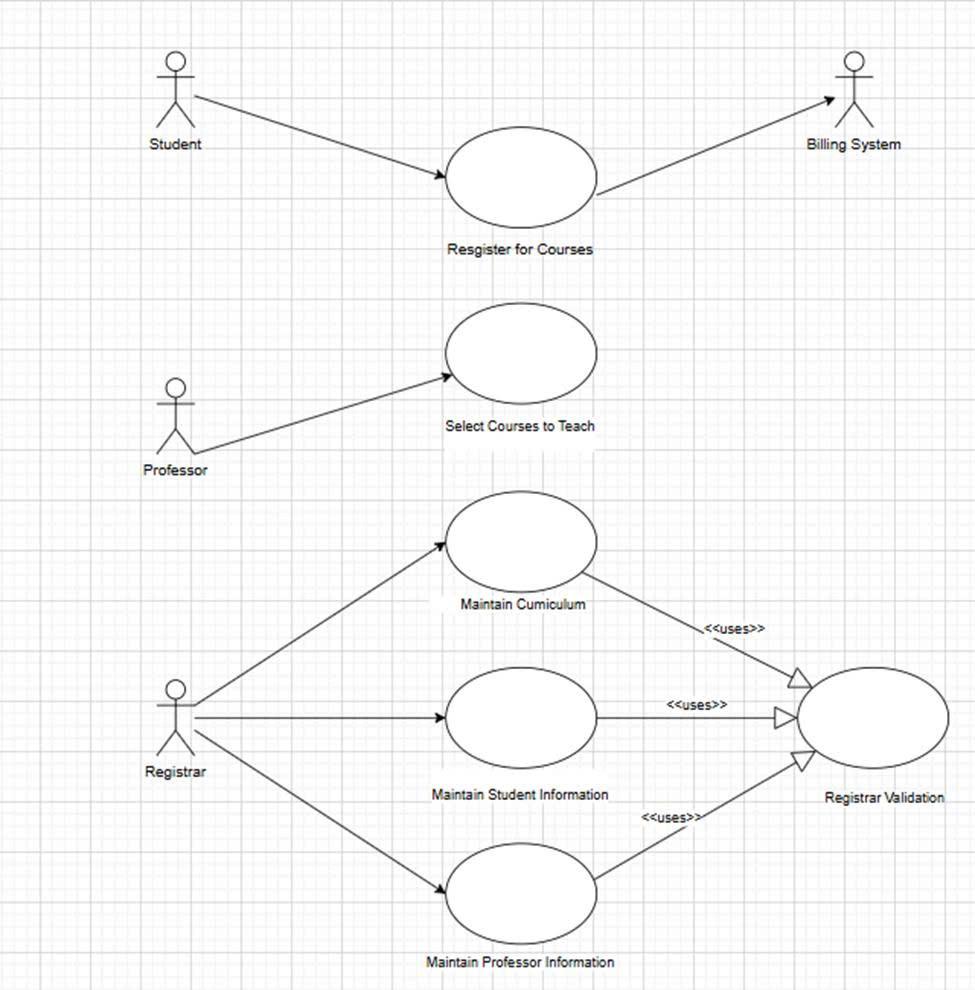
|  |  |  |  |
| --- | --- | --- | --- |
| Họ tên | MSSV | Công việc | Hoàn thành |
| Trần Đức Toản | 3122411219 | Bài 7, phần ứng dụng làm các ý: 6,7,9 | 100% |
| Đỗ Minh Triết | 3122411224 | Bài 5, bài 6, phần ứng dụng làm các ý: 4,5,9 | 100% |
| Nguyễn Việt Đức | 3122411044 | Bài 3, bài 4, phần ứng dụng làm các ý: 3,8,9 | 100% |
| Phạm Văn Tính | 3122411214 | Bài 1, bài 2, phần ứng dụng làm các ý: 1,2,9 | 100% |

# Bài 1 : Qui trình nghiệp vụ

# Bài 2 : Mô hình khái niệm



# Bài 3 : Vẽ lại business use case



a) Xác định và mô tả các tác nhân (Actors)

* Student (Sinh viên): đăng ký học phần, tương tác với hệ thống đăng ký môn học.
* Professor (Giảng viên): lựa chọn các môn học mà mình sẽ giảng dạy.
* Registrar (Phòng đào tạo/Quản lý đào tạo): quản lý chương trình học, thông tin sinh viên, thông tin giảng viên.
* Billing System (Hệ thống thanh toán): xử lý việc tính phí học phí khi sinh viên đăng ký môn học.

b) Liệt kê và giải thích các trường hợp sử dụng (Use Cases)

* Register for Courses: Sinh viên đăng ký môn học (liên quan đến hệ thống thanh toán).
* Select Courses to Teach: Giảng viên lựa chọn môn mình sẽ dạy.
* Maintain Curriculum: Phòng đào tạo quản lý chương trình học.
* Maintain Student Information: Quản lý hồ sơ, thông tin sinh viên.
* Maintain Professor Information: Quản lý hồ sơ, thông tin giảng viên.
* Registrar Validation: Xác nhận hợp lệ dữ liệu quản lý, được dùng chung bởi các chức năng Maintain Curriculum, Maintain Student Information, Maintain Professor Information.

c) Phân tích các mối quan hệ giữa các Use Case

* Register for Courses ↔ Billing System: Khi sinh viên đăng ký môn học, hệ thống phải kết nối với hệ thống thanh toán để xử lý học phí.
* Maintain Curriculum, Maintain Student Information, Maintain Professor Information đều có quan hệ «uses»/«include» với Registrar Validation → tức là mọi thao tác cập nhật dữ liệu đều cần qua bước xác nhận hợp lệ.

d) Viết kịch bản chi tiết cho một Use Case

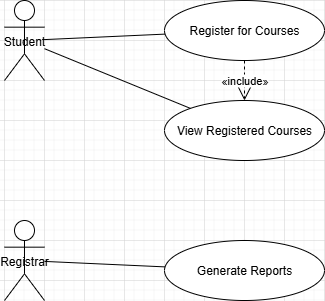
Ví dụ: Register for Courses

* Tác nhân chính: Student
* Mục tiêu: Đăng ký môn học trong học kỳ.
* Điều kiện tiên quyết: Sinh viên đã đăng nhập vào hệ thống.
* Luồng chính:
  + Sinh viên chọn “Register for Courses”.
  + Hệ thống hiển thị danh sách các môn học mở.
  + Sinh viên chọn môn muốn đăng ký.
  + Hệ thống kiểm tra điều kiện (tiên quyết, số tín chỉ, lịch trùng).
  + Nếu hợp lệ, hệ thống tạm lưu đơn đăng ký và gửi thông tin sang Billing System.
  + Billing System tính toán học phí, phản hồi kết quả.
  + Hệ thống hiển thị thông báo “Đăng ký thành công” cho sinh viên.
* Ngoại lệ:
  + Nếu môn đã đầy → thông báo lỗi.
  + Nếu không đủ điều kiện tiên quyết → từ chối đăng ký.
  + Nếu thanh toán thất bại → hủy đăng ký.

e) Đề xuất cải tiến hoặc bổ sung Use Case

* Thêm View Registered Courses: Sinh viên xem danh sách môn đã đăng ký.
* Thêm Drop Courses: Sinh viên hủy môn học đã đăng ký.
* Thêm Generate Reports: Phòng đào tạo xuất báo cáo thống kê sinh viên, giảng viên, môn học.
* Tích hợp Online Payment Gateway: thay vì chỉ kết nối với hệ thống nội bộ, có thể thêm các cổng thanh toán trực tuyến.

Ví dụ:



# 

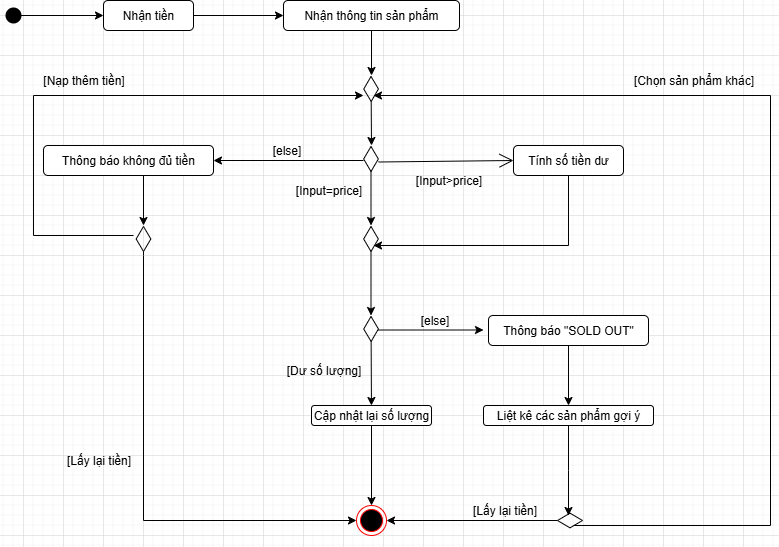
# 

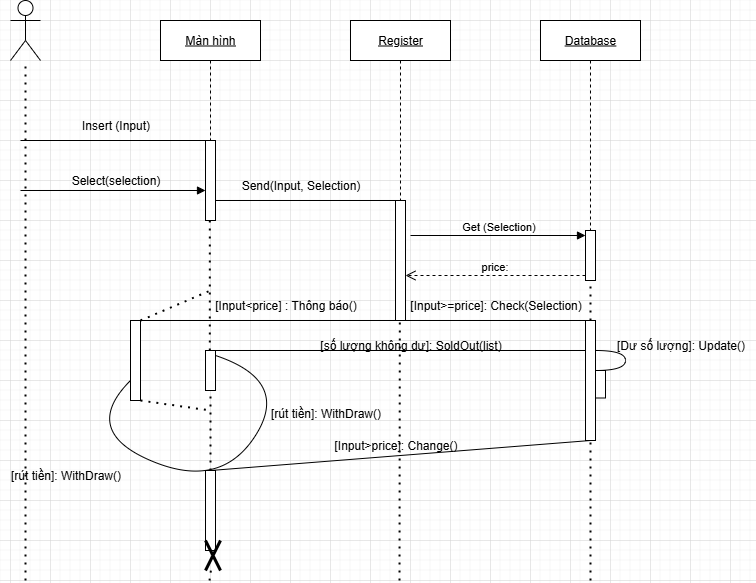
# 

# 

# Bài 4 : Vẽ sơ đồ hoạt động và tương tác

Vẽ lại sơ đồ và tương tác:





*Tóm tắt ý nghĩa của sơ đồ và tương tác:*

1. Sơ đồ hoạt động (Activity Diagram) – Hình 1

Phân tích sơ đồ:

* Đây là luồng hoạt động của máy bán hàng tự động.
* Các bước chính:
  + Người dùng nhận tiền → nhập thông tin sản phẩm.
  + Hệ thống kiểm tra số tiền:
    - Nếu tiền không đủ → thông báo lỗi, cho phép nạp thêm hoặc lấy lại tiền.
    - Nếu tiền đủ nhưng sản phẩm hết→ thông báo “SOLD OUT”, liệt kê sản phẩm gợi ý, hoặc cho chọn sản phẩm khác.
    - Nếu tiền đủ và còn hàng → cập nhật lại số lượng sản phẩm và giao dịch thành công.
* Điểm kết thúc: hoặc người dùng lấy lại tiền, hoặc hệ thống trừ sản phẩm thành công.

Ý nghĩa: Sơ đồ này mô tả quy trình xử lý nghiệp vụ của máy bán hàng tự động từ lúc nhận tiền đến khi bán sản phẩm hoặc trả tiền lại.

2. Sơ đồ tuần tự (Sequence Diagram) – Hình 2

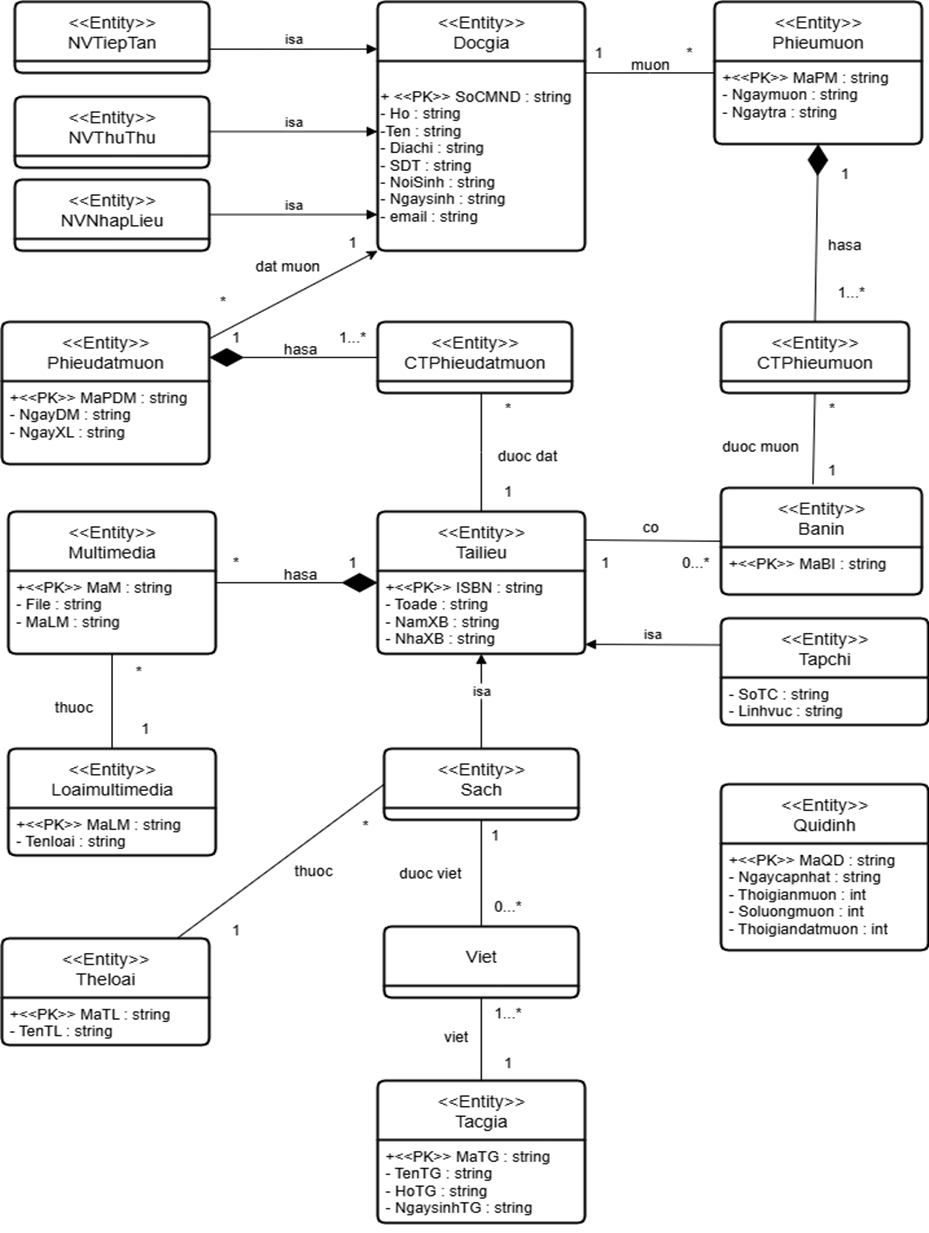
Phân tích sơ đồ:

* Các đối tượng tham gia: Màn hình, Register (bộ xử lý trung gian), Database.
* Luồng tương tác:
  + Người dùng thực hiện Insert(input) và Select(selection).
  + Hệ thống gửi yêu cầu từ Màn hình → Register → Database để kiểm tra.
  + Database trả về giá sản phẩm → Register kiểm tra điều kiện:
    - Nếu Input < Price → gửi thông báo không đủ tiền.
    - Nếu Input ≥ Price → kiểm tra số lượng:
      * Nếu hết hàng → trả về SoldOut(list).
      * Nếu đủ hàng → Database cập nhật số lượng sản phẩm (Update).
  + Register xử lý thêm:
    - Nếu cần rút tiền → Withdraw.
    - Nếu Input > Price → trả lại tiền thừa (Change).

Ý nghĩa: Sơ đồ tuần tự mô tả các thông điệp và sự trao đổi thông tin giữa các thành phần hệ thống khi người dùng mua hàng trên máy bán hàng tự động

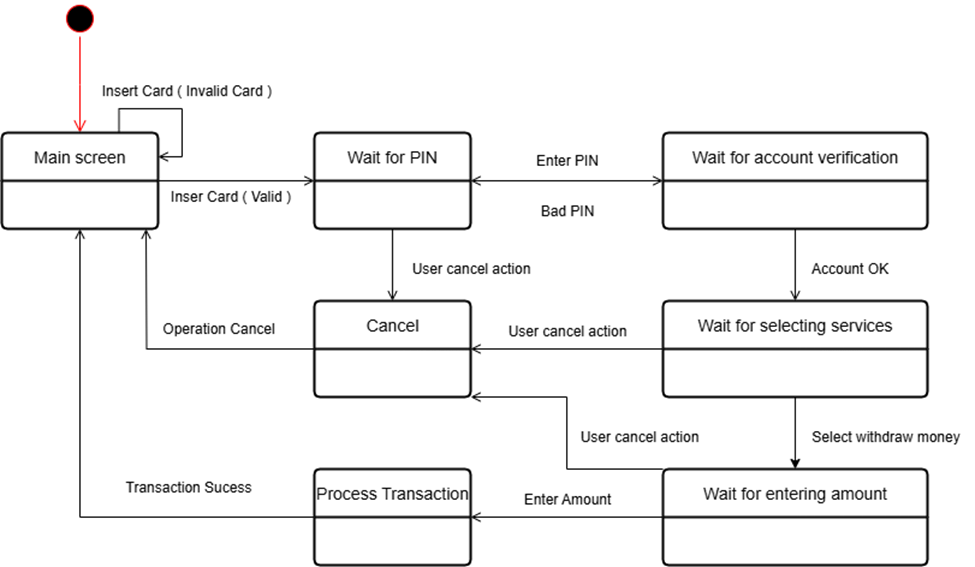
# 

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

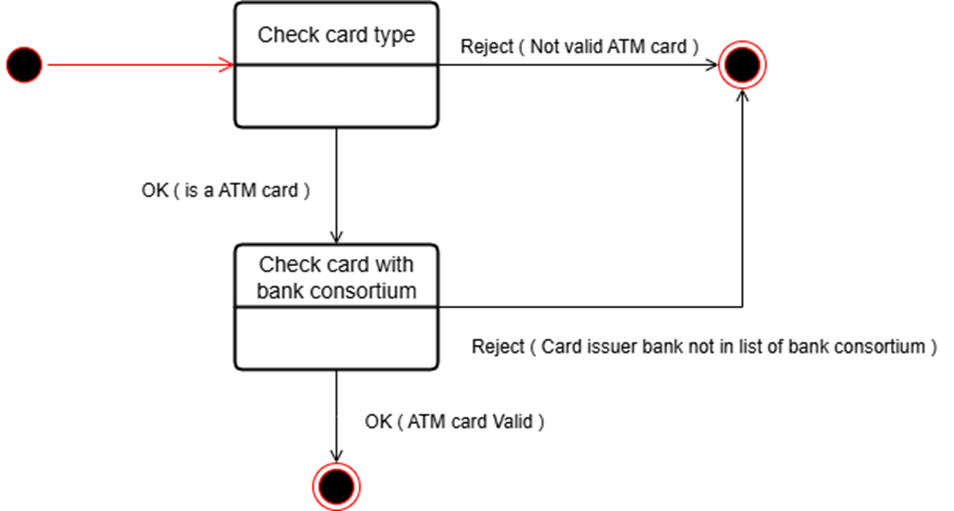


# Bài 6 : Vẽ lược đồ

6.1 ) State Chart for ATM



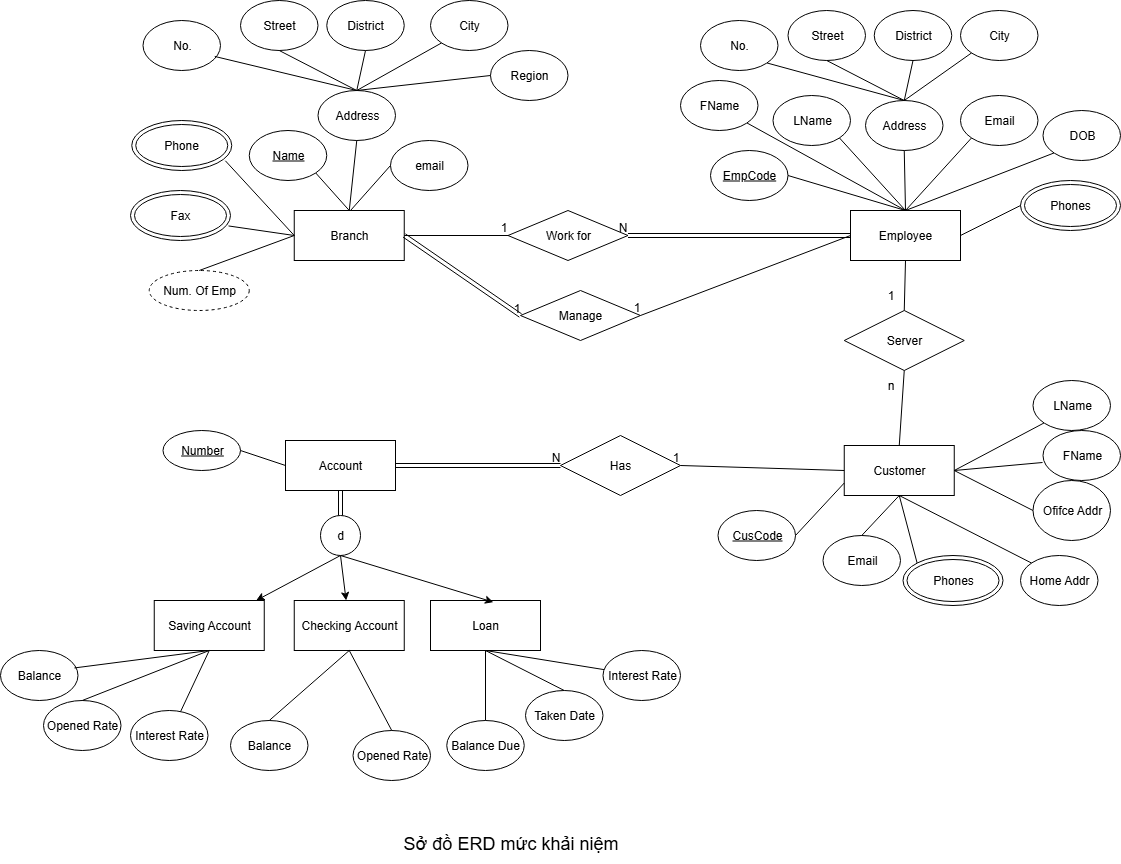
6.2 ) State Chart for Card Controller



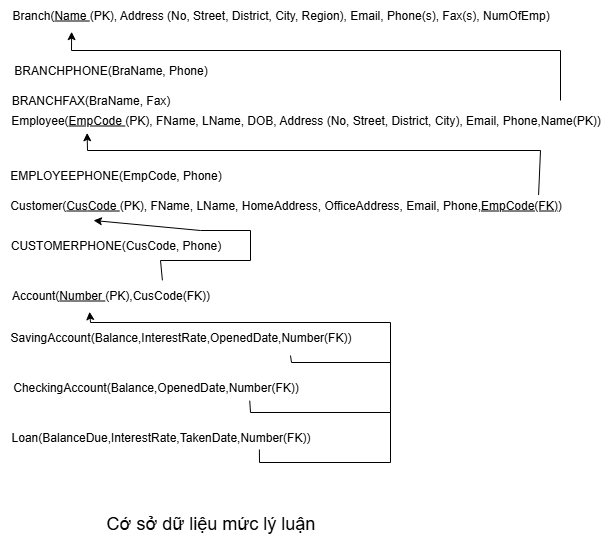
# Bài 7 Phân tích dữ liệu cho ABC Bank

1,Các sơ đồ

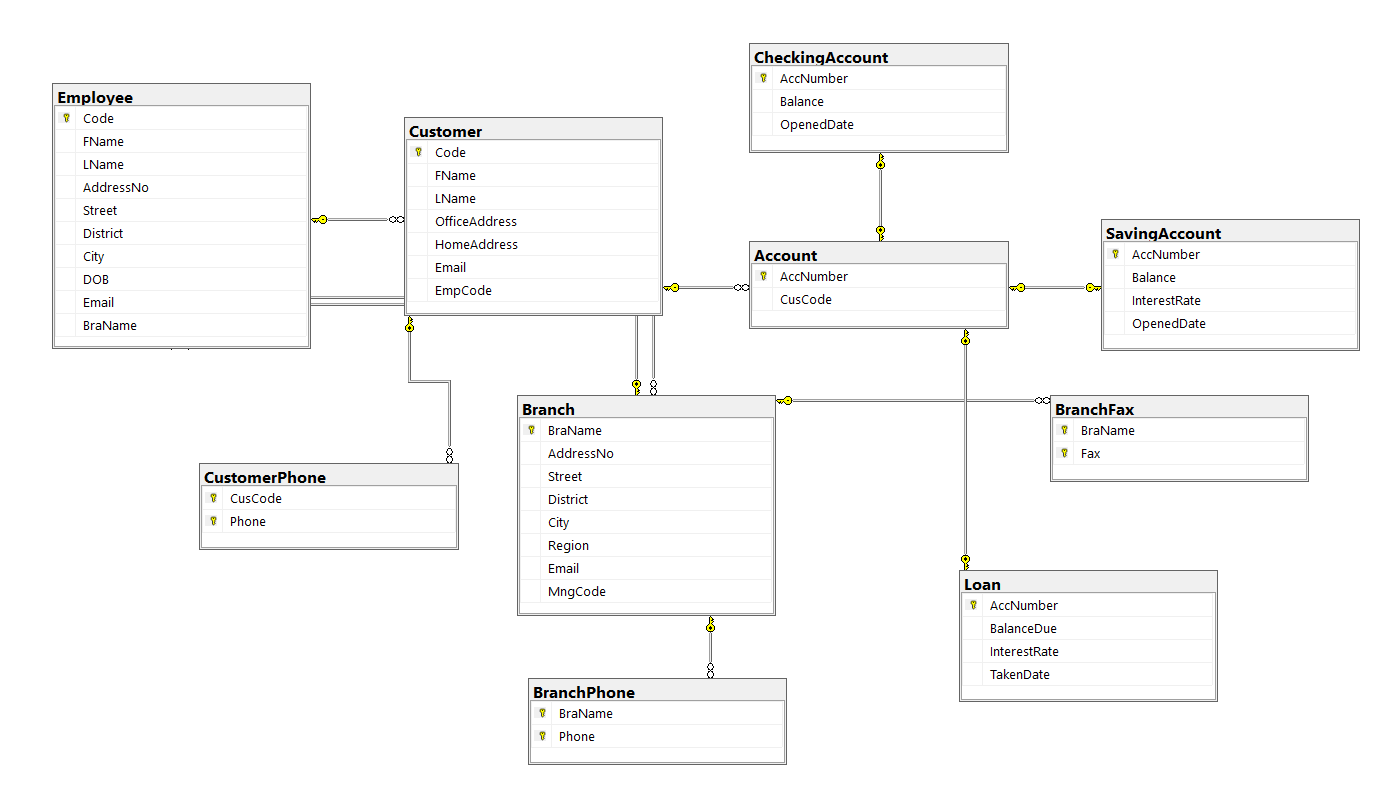
Mức khái niệm



Mức lý luận



Mức vật lý



2, Các câu truy vấn

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

--Thêm Branch

INSERT INTO Branch VALUES

('Hanoi', '12', 'Tran Hung Dao', 'Hoan Kiem', 'Hanoi', 'North', 'hanoi@abc.com', NULL),

('HCM', '45', 'Nguyen Hue', 'District 1', 'HCM', 'South', 'hcm@abc.com', NULL),

('Danang', '23', 'Le Duan', 'Hai Chau', 'Danang', 'Central', 'danang@abc.com', NULL),

('Hue', '89', 'Hung Vuong', 'Hue Center', 'Hue', 'Central', 'hue@abc.com', NULL);

-- Thêm Phone/Fax

INSERT INTO BranchPhone VALUES ('Hanoi', '0241111111'), ('Hanoi', '0242222222'),

('HCM', '0283333333'), ('Danang', '0236666666');

INSERT INTO BranchFax VALUES ('Hanoi', '0249999999'), ('HCM', '0288888888');

-- Thêm Employee

INSERT INTO Employee VALUES

('N1080', 'John', 'Smith', '1', 'Pham Van Dong', 'Cau Giay', 'Hanoi', '1985-03-10', 'john.smith@abc.com', 'Hanoi'),

('N1081', 'Anna', 'Tran', '22', 'Nguyen Trai', 'Thanh Xuan', 'Hanoi', '1990-07-22', 'anna.tran@abc.com', 'Hanoi'),

('N1082', 'Peter', 'Brown', '5', 'Nguyen Thi Minh Khai', 'District 1', 'HCM', '1992-03-15', 'peter.brown@abc.com', 'HCM'),

('N1083', 'Sarah', 'Le', '10', 'Le Loi', 'Hai Chau', 'Danang', '2010-03-05', 'sarah.le@abc.com', 'Danang');

-- Thêm Customer

INSERT INTO Customer VALUES

('C100', 'Peter', 'Johnson', 'Quan 1 - HCM', 'Quan 7 - HCM', 'peter.johnson@abc.com', 'N1080'),

('C101', 'Linda', 'Pham', 'Cau Giay - HN', 'Dong Da - HN', 'linda.pham@abc.com', 'N1081'),

('C102', 'Tom', 'Nguyen', 'Thanh Khe - DN', 'Hai Chau - DN', 'tom.nguyen@abc.com', 'N1083'),

('C103', 'Mai', 'Vo', 'Hue Center', 'Hue North', 'mai.vo@abc.com', 'N1082');

-- Thêm CustomerPhone

INSERT INTO CustomerPhone VALUES ('C100', '0901234567'), ('C101', '0912223333'), ('C102', '0931112222'), ('C103', '0945556666');

-- Thêm Account

INSERT INTO Account VALUES ('A001', 'C100'), ('A002', 'C100'), ('A003', 'C101'), ('A004', 'C102');

-- SavingAccount

INSERT INTO SavingAccount VALUES ('A001', 5000.00, 3.5, '2022-01-01'),

('A002', 7000.00, 4.0, '2023-05-01'),

('A003', 2000.00, 3.0, '2021-06-01');

-- CheckingAccount

INSERT INTO CheckingAccount VALUES ('A004', 1500.00, '2023-08-01');

-- Loan

INSERT INTO Loan VALUES ('A003', 10000.00, 7.5, '2022-10-10');

--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.AccNumber

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”

SELECT SUM(S.Balance) AS TotalBalance

FROM SavingAccount S

JOIN Account A ON S.AccNumber = A.AccNumber

JOIN Customer C ON A.CusCode = C.Code

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

# 

# Bài tập ứng dụng

## 1) Liệt kê các yêu cầu chứ năng và phi chức năng của hệ thống

Chức năng chính:

- Danh mục sản phẩm (Product catalog)

- Giỏ hàng (Shopping cart)

- Quy trình thanh toán (Payment Process)

- Quy trình thanh toán (Payment Process)

- Tồn kho (Inventory)

- Đánh giá (Rating)

- Kiểm soát truy cập (Access Control)

· Danh mục sản phẩm (Product Catalog)

o Người mua có thể duyệt danh sách sản phẩm với các chức năng lọc, sắp xếp theo tên sản phẩm, giá.

o Xem chi tiết sản phẩm (Tên, mô tả, số lượng tồn kho, thông tin kho như địa chỉ kho, trạng thái và đánh giá).

o Quản trị viên có quyền quản lý sản phẩm, gán sản phẩm vào kho hiện có.

· Giỏ hàng (Shopping cart)

o Người mua có thể thêm/xóa sản phẩm khỏi giỏ.

o Giỏ hàng tự động cập nhật tổng tiền, chi phí vận chuyển, khuyến mãi, vận chuyển, chiết khấu.

o Xem bảng tổng hợp giỏ hàng trước khi thanh toán.

o Quản trị viên có thể theo dõi bất kỳ người dùng nào để phát hiện bất thường.

· Quy trình thanh toán (Payment Process)

o Người mua thực hiện thanh toán qua nút “Thanh toán”.

o Hệ thống xác thực thông tin sản phẩm, xử lý thanh toán.

o Gửi email thông báo cho khách hàng sau khi thanh toán.

· Tồn kho (Inventory)

o Quản trị viên quản lý kho hàng.

o Cập nhật tồn kho khi sản phẩm được bán.

· Đánh giá (Rating)

o Người mua có thể đánh giá sản phẩm (theo thang điểm 1-5).

o Hiển thị điểm trung bình và nhận xét.

· Kiểm soát truy cập (Access Control)

o Người dùng/Quản trị viên có thể đăng nhập/đăng xuất hệ thống.

o Nếu là người mua - truy cập chức năng xem/mua hàng.

o Nếu là quản lý – truy cập chức năng quản lý.

* **Phi chức năng**

- Bảo mật:

o Xác thực người dùng qua đăng nhập/đăng xuất.

o Kiểm soát phân quyền (user vs admin).

- Hiệu năng:

o Tìm kiếm và lọc sản phẩm nhanh.

o Giỏ hàng và thanh toán phải xử lý tức thời.

- Khả dụng:

o Hệ thống hoạt động ổn định, sẵn sàng phục vụ nhiều người dùng cùng lúc.

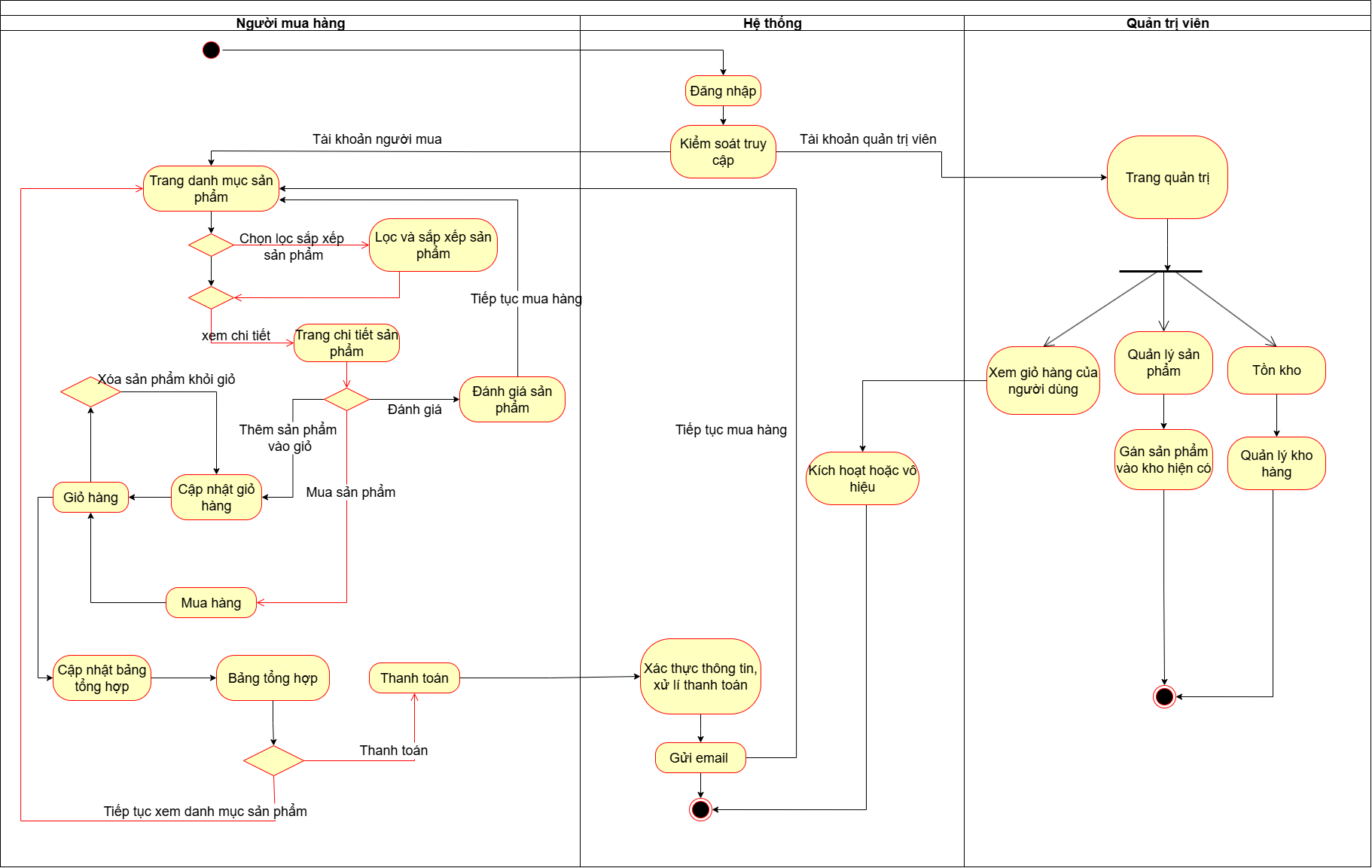
- Khả năng mở rộng:

o Cho phép thêm sản phẩm, kho, đánh giá mới mà không ảnh hưởng đến hệ thống.

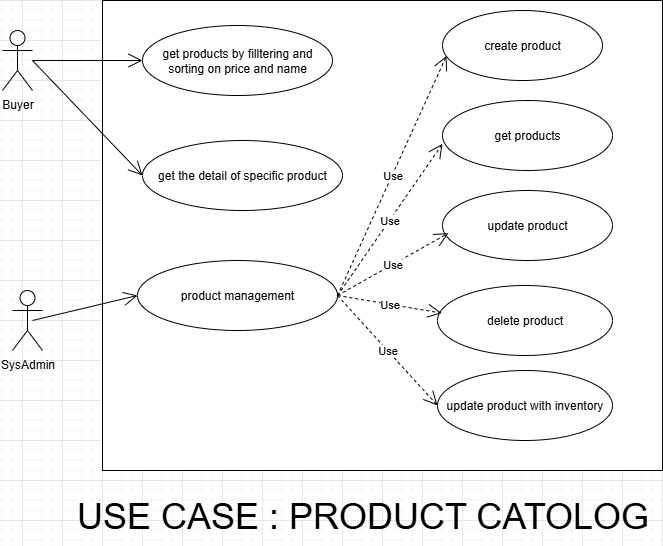
- Khả năng thông báo:

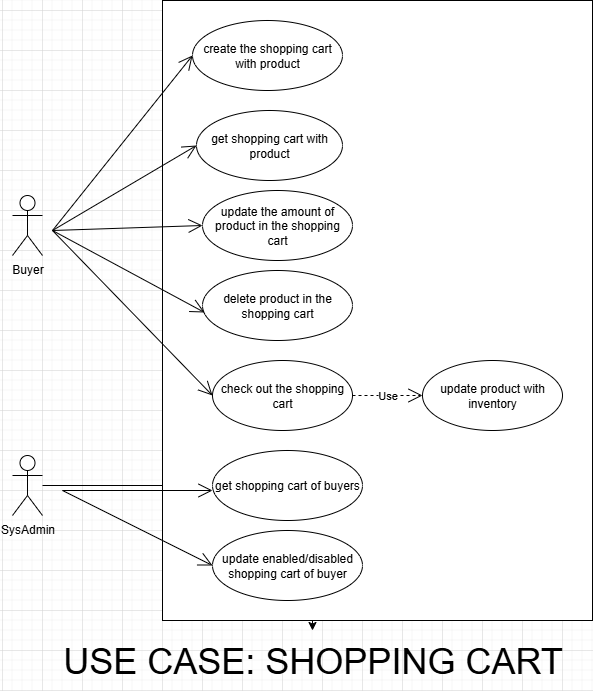
o Gửi email xác nhận thanh toán và trạng thái đơn hàng.

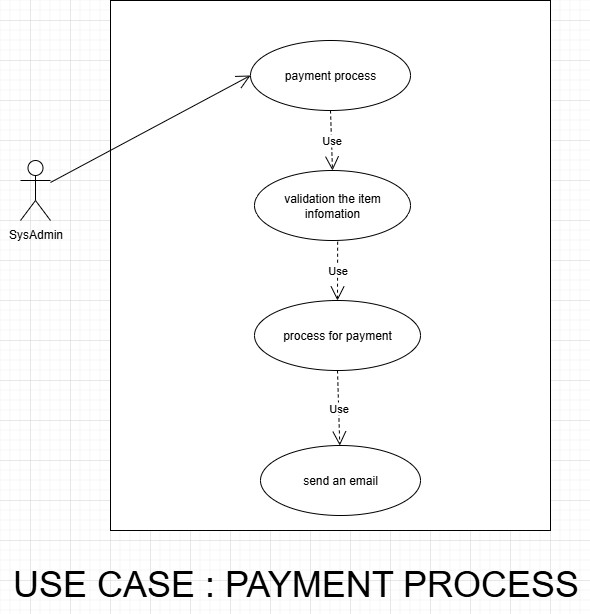
## 2) Quy trình nghiệp vụ

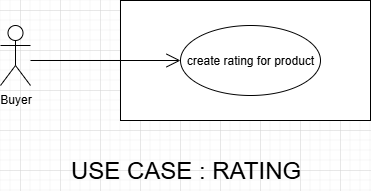


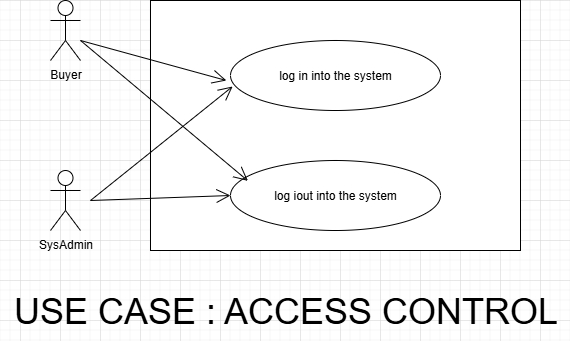
## 3) Xây dựng use case diagram cho website CoolStore :







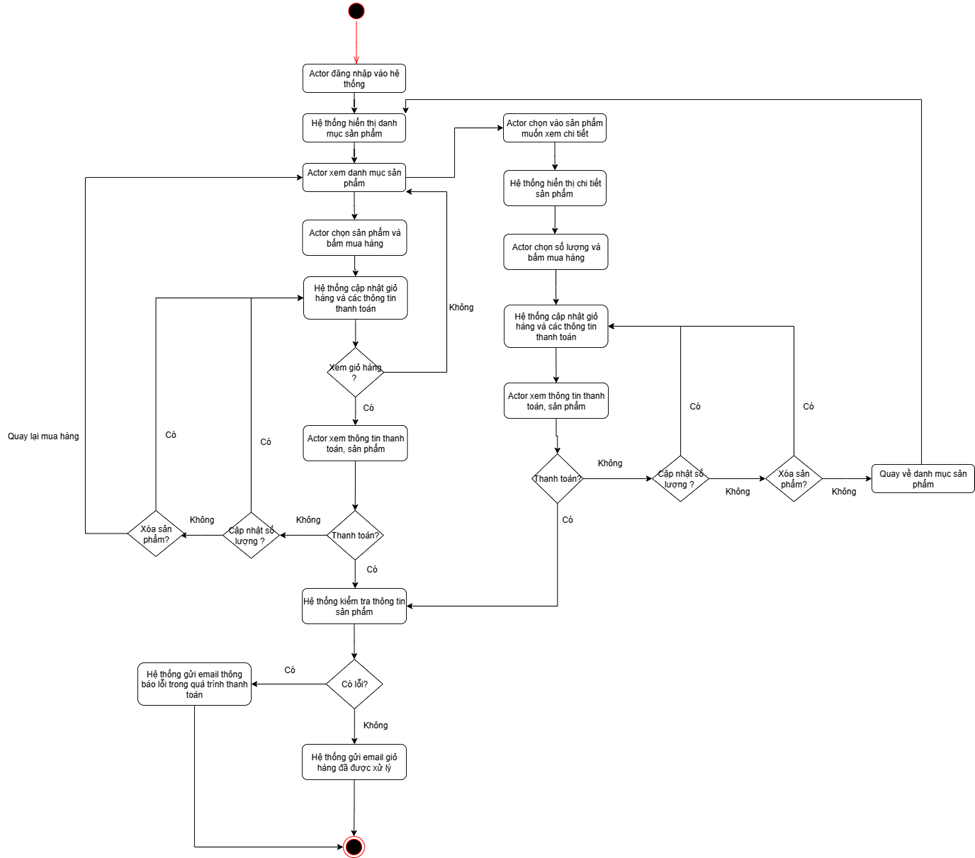




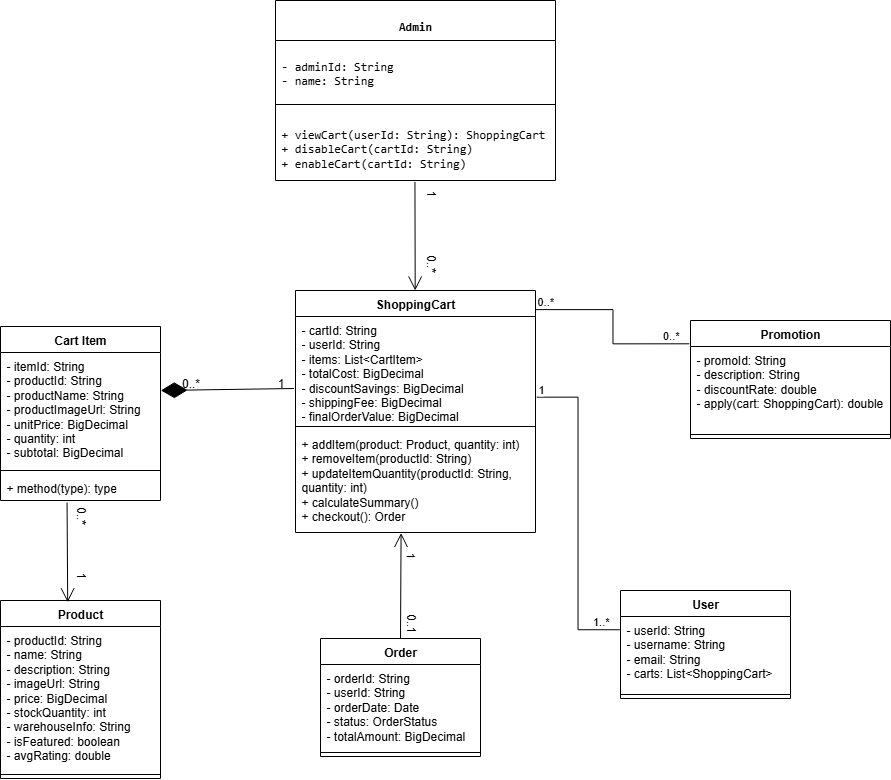
## 4) Use-case mua hàng từ trang chi tiết sản phẩm

|  |  |  |
| --- | --- | --- |
| **UC #01** | **Mua sản phẩm từ chi tiết sản phẩm** | **Độ phức tạp : High** |
| **Mô tả** | | Chức năng này cho phép người mua mua sản phẩm từ trang chi tiết sản phẩm |
| **Tác nhân** | **Chính** | Người mua |
| **Phụ** |  |
| **Tiền điều kiện** | | Actor đăng nhập thành công vào hệ thống bán hàng |
| **Hậu điều kiện** | **Thành công** | Hệ thống gửi email giỏ hàng thanh toán thành công đến actor |
| **Lỗi** | Hệ thống gửi email báo lỗi từ giỏ hàng thanh toán không thành công đến actor |
| **Đặc tả chức năng** | | |
| **Luồng sự kiện chính** | | |
| ·Chức năng bắt đầu khi actor lựa chọn một sản phẩm từ danh mục sản phẩm  Hệ thống hiển thị chi tiết sản phẩm và actor thực hiện các hành động sau :  - Mua hàng : Actor chọn số lượng và bấm mua hàng. Hệ thống sẽ cập nhật giỏ hàng và hiện các thông tin thanh toán của sản phẩm, giỏ hàng  - Chỉnh sửa: Actor chỉnh sửa giỏ hàng :  1. Thay đổi số lượng sản phẩm trong giỏ hàng  2. Xóa sản phẩm trong giỏ hàng  - Sau khi chỉnh sửa hệ thống sẽ cập nhật giỏ hàng và hiện lại thông tin thanh toán của sản phẩm , giỏ hàng  - Actor có thể quay về danh mục sản phẩm  - Actor chọn thanh toán giỏ hàng  - Hệ thống kiểm tra thông tin hợp lệ hay không và sẽ gửi email tới actor để báo lỗi hoặc thông báo thành công | | |
|  | | |

## 5) Activity Diagram từ mua hàng tới 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” .

Bao gồm các thông tin:

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

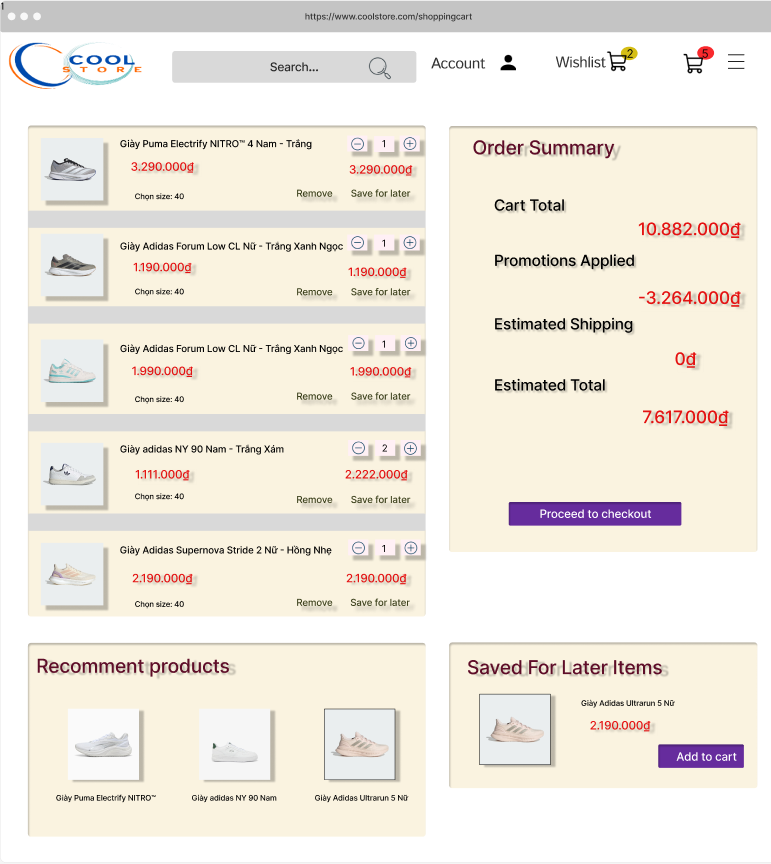
• Chức năng thao tác nhanh: Tăng/giảm số lượng sản phẩm; Xóa sản phẩm khỏi giỏ; Lưu sản phẩm để

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.

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

1. Tầng Trình Bày (Presentation Tier - Frontend)

Đây là "bộ mặt" của ứng dụng, chịu trách nhiệm hiển thị giao diện và thu thập tương tác từ người dùng.

Công nghệ đề xuất:

* Sử dụng một framework JavaScript hiện đại như React.js, Angular, hoặc Vue.js để xây dựng một ứng dụng trang đơn (Single-Page Application - SPA). Lựa chọn này giúp tạo ra trải nghiệm người dùng mượt mà, nhanh chóng và có tính tương tác cao.
* HTML5, CSS3 (với pre-processor như SASS) và các thư viện UI (như Bootstrap, Material-UI) để xây dựng giao diện đẹp mắt và responsive (tương thích trên nhiều thiết bị).
* Trách nhiệm chi tiết:
* Hiển thị dữ liệu: Gọi các API từ Tầng Ứng dụng để lấy dữ liệu (danh sách sản phẩm, chi tiết giỏ hàng,...) và hiển thị chúng cho người dùng.
* Thu thập đầu vào: Xử lý các sự kiện từ người dùng như click chuột (vào nút "Mua hàng" ), nhập liệu (form đăng nhập, tìm kiếm),...
* Xác thực phía Client: Kiểm tra sơ bộ tính hợp lệ của dữ liệu người dùng nhập vào (ví dụ: email có đúng định dạng, mật khẩu có đủ mạnh không) trước khi gửi đi để giảm tải cho server.

Nhiệm vụ:

* Hiển thị danh mục sản phẩm, chi tiết sản phẩm.
* Quản lý giỏ hàng (thêm, xóa, xem tổng hợp).
* Thực hiện thanh toán (giao diện nhập thông tin, xác nhận).
* Đăng nhập/đăng xuất, đăng ký tài khoản.
* Trang quản trị cho Admin (quản lý sản phẩm, kho, giỏ hàng người dùng).

2. Tầng Ứng Dụng (Application Tier - Backend/Business Logic)

Đây là "bộ não" của hệ thống, nơi tất cả các quy tắc nghiệp vụ và logic xử lý được thực thi. Với CoolStore, việc áp dụng kiến trúc Microservices ở tầng này là cực kỳ phù hợp.

Công nghệ đề xuất:Ngôn ngữ: Java (với Spring Boot), .NET Core, Node.js (với Express.js), hoặc Golang.

Giao tiếp: Các microservice sẽ giao tiếp với nhau và với Tầng Trình bày thông qua các REST APIs hoặc GraphQL. Giao tiếp bất đồng bộ (asynchronous) qua một Message Broker như RabbitMQ hoặc Apache Kafka có thể được dùng cho các tác vụ như gửi email xác nhận đơn hàng, giúp hệ thống phản hồi nhanh hơn và đáng tin cậy hơn.

Chi tiết các Microservices:

* Access Control Service: Xử lý đăng ký, đăng nhập/đăng xuất bằng JWT (JSON Web Tokens). Phân quyền chi tiết: ví dụ, API
* DELETE /products/{id} chỉ cho phép SysAdmin truy cập, trong khi API GET /products cho phép cả Buyer và khách truy cập.
* Product Catalog Service: Cung cấp các API để thực hiện CRUD (Create, Read, Update, Delete) cho sản phẩm. Tích hợp với một công cụ tìm kiếm mạnh mẽ như
* Elasticsearch để hỗ trợ tìm kiếm toàn văn (full-text search), lọc sản phẩm theo nhiều tiêu chí (giá, kho hàng, trạng thái "nổi bật").
* Shopping Cart Service: Quản lý giỏ hàng cho cả người dùng đã đăng nhập (lưu trong database) và người dùng khách (lưu trong session hoặc Redis). Logic phức tạp hơn bao gồm: áp dụng mã khuyến mãi, tính toán các loại phí (vận chuyển, thuế) và đảm bảo tính nhất quán của giá sản phẩm trong giỏ hàng.
* Inventory Service: Cung cấp API GET /inventory/{productId} để kiểm tra số lượng tồn kho theo thời gian thực. Quan trọng nhất là phải có cơ chế "locking" (khóa) sản phẩm trong kho khi người dùng bắt đầu quy trình thanh toán để tránh tình trạng bán quá số lượng (overselling).
* Payment Service: Tích hợp với các cổng thanh toán bên thứ ba (ví dụ: Stripe, PayPal, MoMo) qua SDK của họ. Service này không lưu trữ thông tin nhạy cảm như số thẻ tín dụng mà chỉ lưu lại mã giao dịch và trạng thái thanh toán (thành công/thất bại).
* Rating Service: Cho phép người dùng đã mua hàng đánh giá sản phẩm (thang điểm 1-5 sao và bình luận). Có thể bao gồm các chức năng như kiểm duyệt bình luận và tính toán điểm đánh giá trung bình.

Nhiệm vụ:

* Quản lý danh mục sản phẩm (lọc, tìm kiếm, phân trang).
* Quản lý giỏ hàng (thêm, xóa, tính tổng giá trị, khuyến mãi, vận chuyển).
* Xử lý thanh toán (tích hợp API với cổng thanh toán, xác thực thông tin).
* Quản lý kho hàng (cập nhật tồn kho khi bán hàng).
* Quản lý đánh giá sản phẩm.
* Quản lý quyền truy cập (phân biệt Buyer và SysAdmin).
* Gửi email thông báo sau khi thanh toán.

3. Tầng Dữ Liệu (Data Tier - Database)

Tầng này chịu trách nhiệm lưu trữ và quản lý toàn bộ dữ liệu của hệ thống một cách an toàn và hiệu quả.

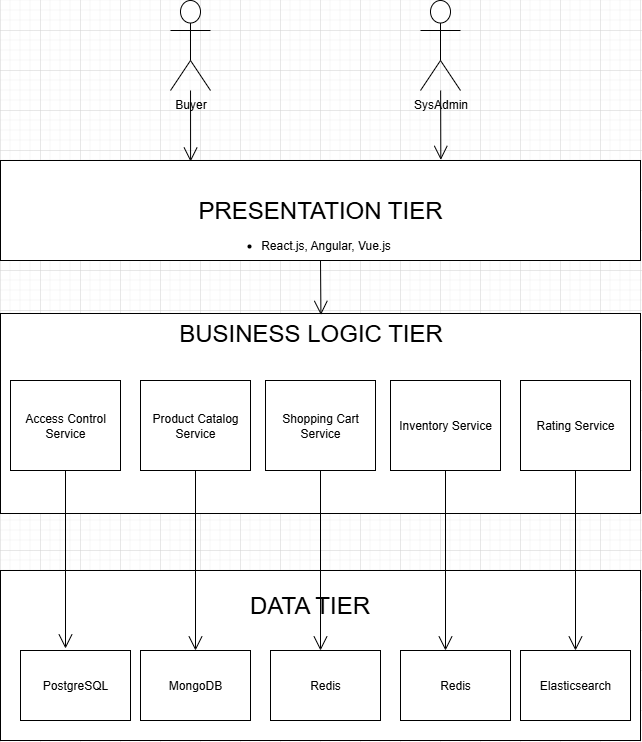
Mô hình đề xuất: Áp dụng mô hình Database-per-service. Mỗi microservice sẽ sở hữu và quản lý cơ sở dữ liệu riêng của mình. Điều này giúp các service hoàn toàn độc lập với nhau.

Lựa chọn công nghệ Database:

* PostgreSQL / MySQL (Cơ sở dữ liệu quan hệ): Dùng cho các service cần tính toàn vẹn dữ liệu và các giao dịch phức tạp như Access Control (lưu thông tin user), Product Catalog (lưu thông tin sản phẩm), và Payment (lưu lịch sử đơn hàng).
* MongoDB (Cơ sở dữ liệu NoSQL - Document): Phù hợp cho Rating Service để lưu trữ các đánh giá và bình luận, vì cấu trúc của chúng có thể linh hoạt.
* Redis (In-memory Key-Value Store): Dùng cho Shopping Cart Service để lưu trữ dữ liệu giỏ hàng của người dùng khách (guest carts) hoặc lưu cache, giúp truy xuất với tốc độ cực nhanh.
* Elasticsearch (Search Engine): Như đã đề cập, dùng để phục vụ chức năng tìm kiếm cho Product Catalog Service

Các bảng dữ liệu chính:

* Users: thông tin đăng nhập, phân quyền (Buyer, SysAdmin).
* Products: tên, mô tả, giá, trạng thái, tồn kho.
* Inventory: quản lý số lượng trong kho.
* Cart: thông tin giỏ hàng theo từng người dùng.
* Orders: đơn hàng, trạng thái thanh toán, chi tiết mua hàng.
* Reviews: đánh giá sản phẩm.



## 9) Viết test case kiểm thử

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: Kịch bản thành công - Người dùng đã đăng nhập

|  |  |
| --- | --- |
| Mục | Chi tiết |
| Mã Test Case | TC\_ADD\_001 |
| Tên Test Case | Thêm thành công sản phẩm còn hàng vào giỏ hàng từ trang danh mục. |
| Mức độ ưu tiên | Cao (High) |
| Điều kiện tiên quyết | 1. Người dùng buyer01 đã đăng nhập. <b 2. Sản phẩm SP001 ("Giày Cool") có tồn kho là 10.  3. Giỏ hàng của buyer01 đang trống. |
| Các bước thực hiện | 1. Dùng tài khoản buyer01 truy cập trang Danh mục sản phẩm.  2. Tìm đến SP001 sản phẩm.  3. Nhấn vào nút "Thêm vào giỏ hàng". |
| Dữ liệu kiểm thử | Tài khoản: buyer01/password123, Mã sản phẩm: SP001, Số lượng: 1. |
| Kết quả mong đợi | 1. Giao diện: Hiển thị một thông báo pop-up/toast "Đã thêm 'Áo Thun Cool' vào giỏ hàng". Biểu tượng giỏ hàng trên header cập nhật số lượng thành 1.  2. API**:** Lời gọi POST/api/v1/carts trả về mã trạng thái 201 Created.  3. Database**:** Bảng shopping\_cart\_items có một bản ghi mới với user\_id của buyer01, product\_id là SP001, và quantity là 1. |

Test Case 2: Kịch bản biên - Thêm sản phẩm cuối cùng trong kho

|  |  |
| --- | --- |
| Mục | Chi tiết |
| Mã Test Case | TC\_ADD\_002 |
| Tên Test Case | Thêm thành công sản phẩm cuối cùng trong kho. |
| Mức độ ưu tiên | Trung bình (Medium) |
| Điều kiện tiên quyết | 1. Người dùng buyer02 đã đăng nhập.  2. Sản phẩm SP002 ("Giày Addias") có tồn kho là 1. |
| Các bước thực hiện | 1. Dùng tài khoản buyer02 truy cập trang chi tiết của sản phẩm SP002.  2. Nhấn nút "Mua hàng". |
| Dữ liệu kiểm thử | Tài khoản: buyer02/password123, Mã sản phẩm: SP002. |
| Kết quả mong đợi | 1. Giao diện: Thêm vào giỏ hàng thành công. Sau khi tải lại trang chi tiết sản phẩm SP002, nút "Mua hàng" bị vô hiệu hóa và có dòng chữ "Hết hàng".  2. Database: Tồn kho của SP002 trong bảng inventory được cập nhật thành 0. |

Test Case 3: Kịch bản thất bại - Thêm sản phẩm đã hết hàng

|  |  |
| --- | --- |
| Mục | Chi tiết |
| Mã Test Case | TC\_ADD\_003 |
| Tên Test Case | Cố gắng thêm một sản phẩm đã hết hàng. |
| Mức độ ưu tiên | Cao (High) |
| Điều kiện tiên quyết | 1. Người dùng buyer01 đã đăng nhập.  2. Sản phẩm SP003 ("Giày Sneaker") có tồn kho là 0. |
| Các bước thực hiện | 1. Dùng tài khoản buyer01 truy cập trang chi tiết của sản phẩm SP003. |
| Dữ liệu kiểm thử | Tài khoản: buyer01/password123, Mã sản phẩm: SP003. |
| Kết quả mong đợi | 1. Giao diện: Nút "Mua hàng" trên giao diện bị xám đi (disabled) và không thể nhấn được. Có một thông báo rõ ràng như "Sản phẩm hiện đã hết hàng".  2. API**:** Nếu cố tình gọi API POST /api/v1/carts với productId: "SP003", hệ thống trả về lỗi 400 Bad Request với thông điệp "Sản phẩm đã hết hàng".  3. Database**:** Không có bản ghi mới nào được tạo trong bảng shopping\_cart\_items. |

Test Case 4: Kịch bản cho người dùng khách (Guest User)

|  |  |
| --- | --- |
| Mục | Chi tiết |
| Mã Test Case | TC\_ADD\_004 |
| Tên Test Case | Người dùng chưa đăng nhập thêm sản phẩm vào giỏ hàng. |
| Mức độ ưu tiên | Cao (High) |
| Điều kiện tiên quyết | 1. Người dùng chưa đăng nhập.  2. Sản phẩm SP001 ("Giày Cool") có tồn kho là 10. |
| Các bước thực hiện | 1. Truy cập trang web CoolStore mà không đăng nhập.  2. Vào trang danh mục và nhấn "Mua hàng" tại sản phẩm SP001.  3. Đăng nhập bằng tài khoản buyer01 (giỏ hàng đang trống). |
| Dữ liệu kiểm thử | Tài khoản: buyer01/password123,Mã sản phẩm: SP001. |
| Kết quả mong đợi | 1. Trước khi đăng nhập: Sản phẩm được thêm vào giỏ hàng thành công. Thông tin giỏ hàng được lưu trong Local Storage hoặc Session của trình duyệt.  2. Sau khi đăng nhập: Hệ thống tự động đồng bộ giỏ hàng từ session vào tài khoản buyer01. Giỏ hàng của buyer01 giờ đây có sản phẩm SP001. |