Thẻ 1

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

**TRƯỜNG ĐẠI HỌC SÀI GÒN**

------- \*\*\* -------

**BÀI TẬP NHÓM 2**

**Môn: Kiểm thử phần mềm**

***Họ và tên:***  *Đỗ Minh Quân - 3122411166*

*Lê Thị Mỹ Hương - 3122411077*

*Trần Bùi Ty Ty - 3122411241*

*Trần Vĩnh Huy - 3122411072*

***Lớp:***  *DCT122C2*

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

**MỤC LỤC**

[PHÂN CÔNG CÔNG VIỆC 2](#_rddgrvdyjm77)

[Bài 1. Vẽ lại quy trình nghiệp vụ sau (Độc giả, NV Tiếp tân, Thành viên) 4](#_6yful4jsqh45)

[Bài 2: Vẽ mô hình khái niệm sau 6](#_ic53dz9peyx4)

[Bài 3: Vẽ lại business use case sau: 7](#_srvg789ey65q)

[a) Xác định và mô tả các tác nhân (Actors) xuất hiện trong sơ đồ trên. 8](#_hu63y5h4ww71)

[b) Liệt kê và giải thích các trường hợp sử dụng (Use cases) được thể hiện trong sơ đồ. 9](#_emg3z1b14r7t)

[c) Phân tích các mối quan hệ giữa các use case (bao gồm cả mối quan hệ «uses» hoặc «include»). 10](#_ixi6u06lz7vv)

[d) Viết kịch bản chi tiết cho một trường hợp sử dụng cụ thể (ví dụ: “Register for Courses” hoặc “Maintain Student Information”). 11](#_3ion8m53fdfr)

[e) Đề xuất cải tiến hoặc bổ sung thêm các use case mà bạn thấy cần thiết. 15](#_ignmzpe7vwto)

[Bài 4: Vẽ sơ đồ hoạt động và tương tác sau và tóm tắt ý nghĩa sơ đồ 15](#_7uvlmcfkju2i)

[Bài 5. Vẽ lược đồ lớp cho bài toán Quản lý thư viện 18](#_m5oy6318ie0d)

[Bài 6. Vẽ lược đồ sau: 21](#_klc77xpikpyo)

[Bài 7: Phân tích dữ liệu cho ABC Bank 22](#_rpcx8yiyuevl)

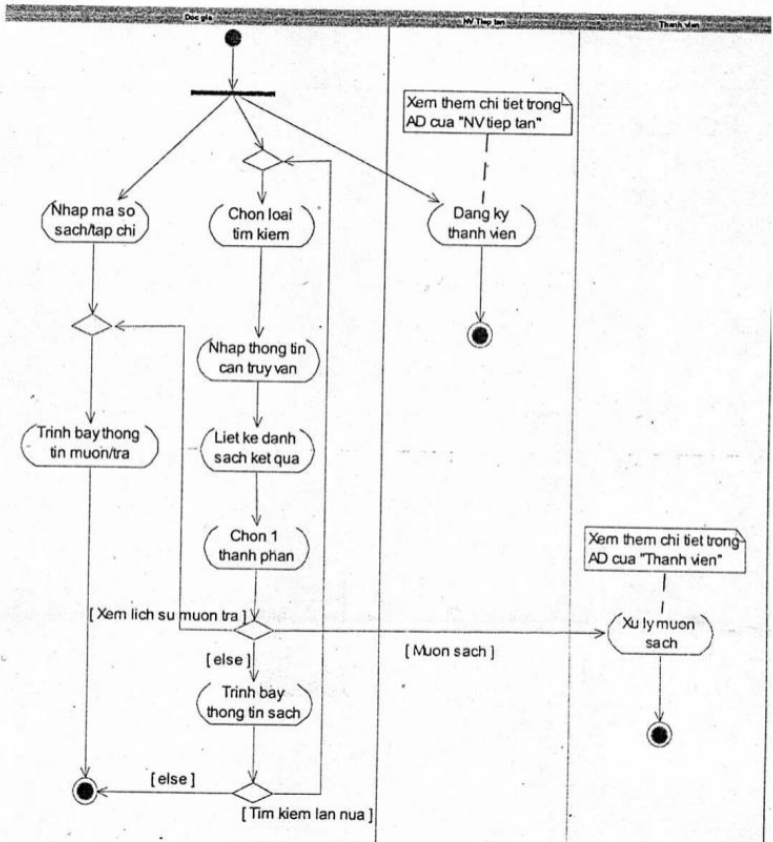
[BÀI TẬP ỨNG DỤNG 31](#_rjpki7wto7in)

# **PHÂN CÔNG CÔNG VIỆC**

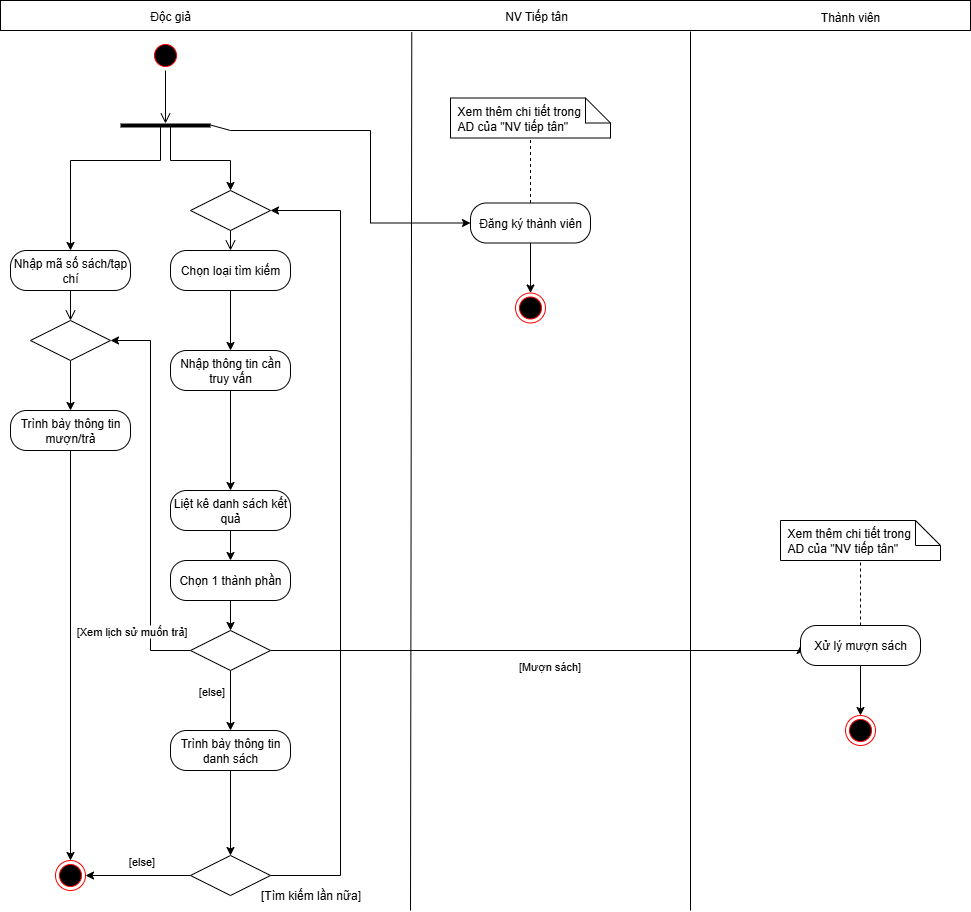
| Thành viên | Nhiệm vụ chính | Chi tiết công việc | Ghi chú |
| --- | --- | --- | --- |
| Quân | Phân tích nghiệp vụ & Use case | - Bài 1: Vẽ lại quy trình nghiệp vụ (Độc giả, NV tiếp tân, Thành viên)  - Bài 2: Vẽ mô hình khái niệm  - Bài 3:  • Xác định Actors  • Liệt kê Use cases  • Phân tích quan hệ Use case  • Viết kịch bản 1 Use case cụ thể  • Đề xuất Use case mới | *Drawio:*  https://drive.google.com/file/d/17QMaL4Bq35nEMAF8bMJ0ZR86WCQvcvvo/view?usp=sharing |
| Huy | Mô hình hành vi & cấu trúc hệ thống | - Bài 4: Vẽ sơ đồ hoạt động và sơ đồ tương tác, giải thích ý nghĩa  - Bài 5: Vẽ lược đồ lớp cho bài toán Quản lý thư viện  - Bài 6: Vẽ lược đồ được yêu cầu | *Drawio:* https://drive.google.com/file/d/1QJdmxgXKm\_U9po5ilLgleEgpG6CCSKrc/view?usp=sharing |
| Ty | Phân tích dữ liệu & thiết kế CSDL | - Bài 7 (ABC Bank):  • Thiết kế CSDL mức Khái niệm, Luận lý, Vật lý  • Tạo các bảng (BRANCH, EMPLOYEE, CUSTOMER, ACCOUNT, …)  • Viết các truy vấn Insert, Update, Delete, Select theo yêu cầu (a → f) | SQL và thiết kế CSDL |
| Hương | Ứng dụng CoolStore & kiểm thử | - Bài 8 (CoolStore):  • Xác định yêu cầu chức năng & phi chức năng  • Vẽ quy trình nghiệp vụ  • Xây Use case diagram  • Viết Use case “Mua sản phẩm…”  • Vẽ Activity Diagram  • Thiết kế Class Diagram module Giỏ hàng  • Thiết kế giao diện “Quản lý giỏ hàng”  • Đề xuất kiến trúc 3-tier  • Viết 3 test cases cho chức năng “Thêm sản phẩm vào giỏ” | Bao quát kiến thức thiết kế hệ thống, giao diện và kiểm thử |

**PHÂN TÍCH THIẾT KẾ HỆ THỐNG**

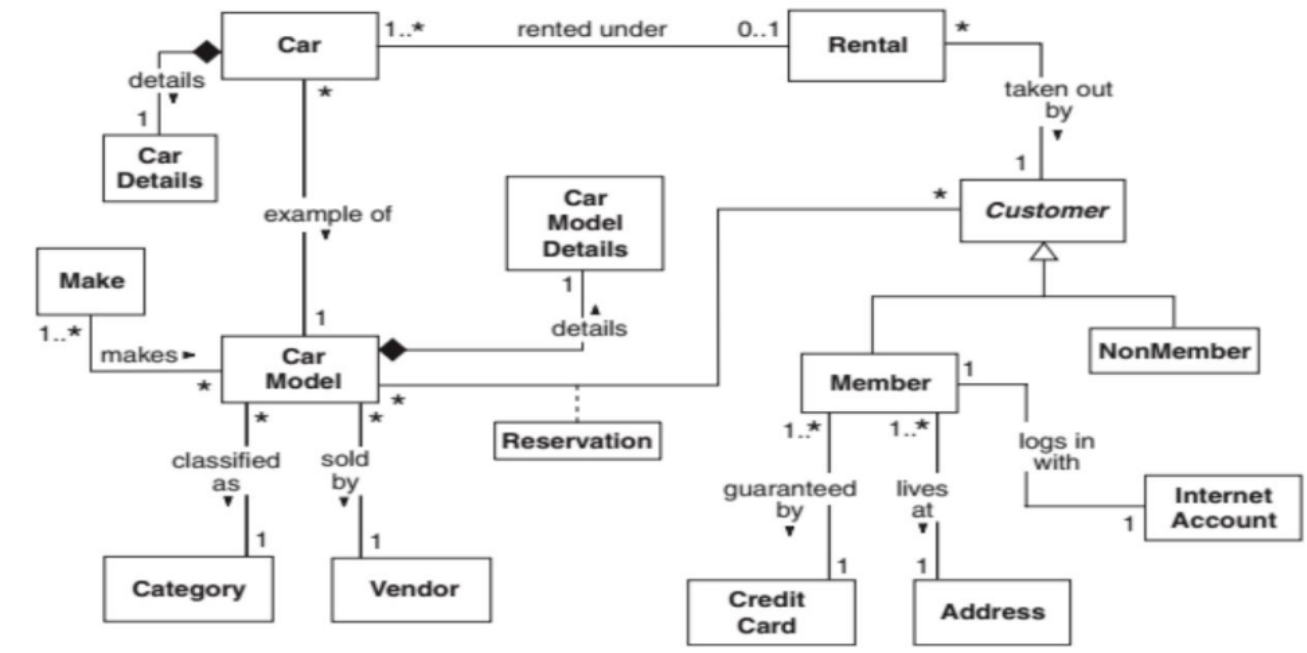
# **Bài 1. Vẽ lại quy trình nghiệp vụ sau (Độc giả, NV Tiếp tân, Thành viên)**



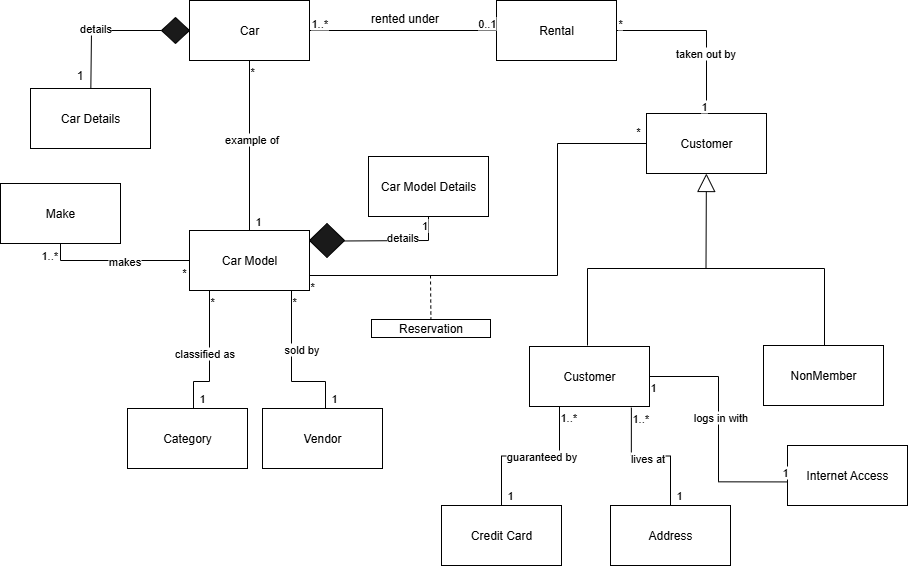
**Trả lời:**



# **Bài 2: Vẽ mô hình khái niệm sau**



Trả lời:



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

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

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

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

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

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

**Trả lời:**

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

Trong sơ đồ trên, có ba tác nhân (bao gồm Student, Professor và Registrar) tham gia vào hệ thống:

* Student (Sinh viên): Là người sử dụng chính của hệ thống với chức năng đăng ký môn học. Khi thực hiện đăng ký, sinh viên có liên quan trực tiếp đến hệ thống thanh toán để xử lý học phí.
* Professor (Giảng viên): Là tác nhân có vai trò chọn các môn học mà mình sẽ giảng dạy trong từng học kỳ.
* Registrar (Phòng đào tạo/Quản trị viên): Quản lý toàn bộ dữ liệu học vụ, bao gồm việc duy trì và cập nhật chương trình đào tạo, quản lý thông tin sinh viên cũng như thông tin giảng viên.
* Billing System (Hệ thống thanh toán): Là một tác nhân bên ngoài hệ thống, có nhiệm vụ xử lý các khoản học phí phát sinh sau khi sinh viên đăng ký môn học thành công.

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

Sơ đồ trên có 6 Use Case. Trong đó:

1. Register for Courses (Đăng ký môn học): Sinh viên đăng ký học phần, hệ thống kiểm tra điều kiện và gửi thông tin sang Billing System để xử lý học phí.
2. Select Courses to Teach (Chọn môn để giảng dạy): Giảng viên chọn các môn mà mình sẽ phụ trách.
3. Maintain Curriculum (Quản lý chương trình đào tạo): Registrar cập nhật, điều chỉnh chương trình học.
4. Maintain Student Information (Quản lý thông tin sinh viên): Registrar quản lý hồ sơ, thông tin học tập của sinh viên.
5. Maintain Professor Information (Quản lý thông tin giảng viên): Registrar quản lý thông tin, hồ sơ của giảng viên.
6. Registrar Validation (Xác nhận của Registrar): Giúp hỗ trợ kiểm tra và xác nhận dữ liệu cho các chức năng mà Registrar thực hiện.

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

Trong sơ đồ, các Use Case có những mối quan hệ sau:

* Register for Courses → Billing System: Use case *Register for Courses* của sinh viên có mối quan hệ giao tiếp với tác nhân ngoài hệ thống là *Billing System*. Điều này có nghĩa là sau khi sinh viên thực hiện đăng ký môn học, hệ thống sẽ gửi thông tin đăng ký sang hệ thống thanh toán để xử lý học phí. Đây là mối quan hệ trao đổi dữ liệu (communication association) giữa một use case và một actor bên ngoài.
* Maintain Curriculum, Maintain Student Information, Maintain Professor Information → Registrar Validation: Ba use case Registrar thực hiện gồm quản lý chương trình đào tạo, quản lý thông tin sinh viên, và quản lý thông tin giảng viên đều có mối quan hệ «uses» đến use case *Registrar Validation*. Quan hệ này cho thấy mỗi khi Registrar chỉnh sửa hoặc cập nhật dữ liệu trong hệ thống bắt buộc phải thực hiện bước xác nhận/kiểm tra hợp lệ. Nhờ vậy, hệ thống đảm bảo dữ liệu đầu vào được chính xác và hợp lệ trước khi lưu trữ hoặc tạo.

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

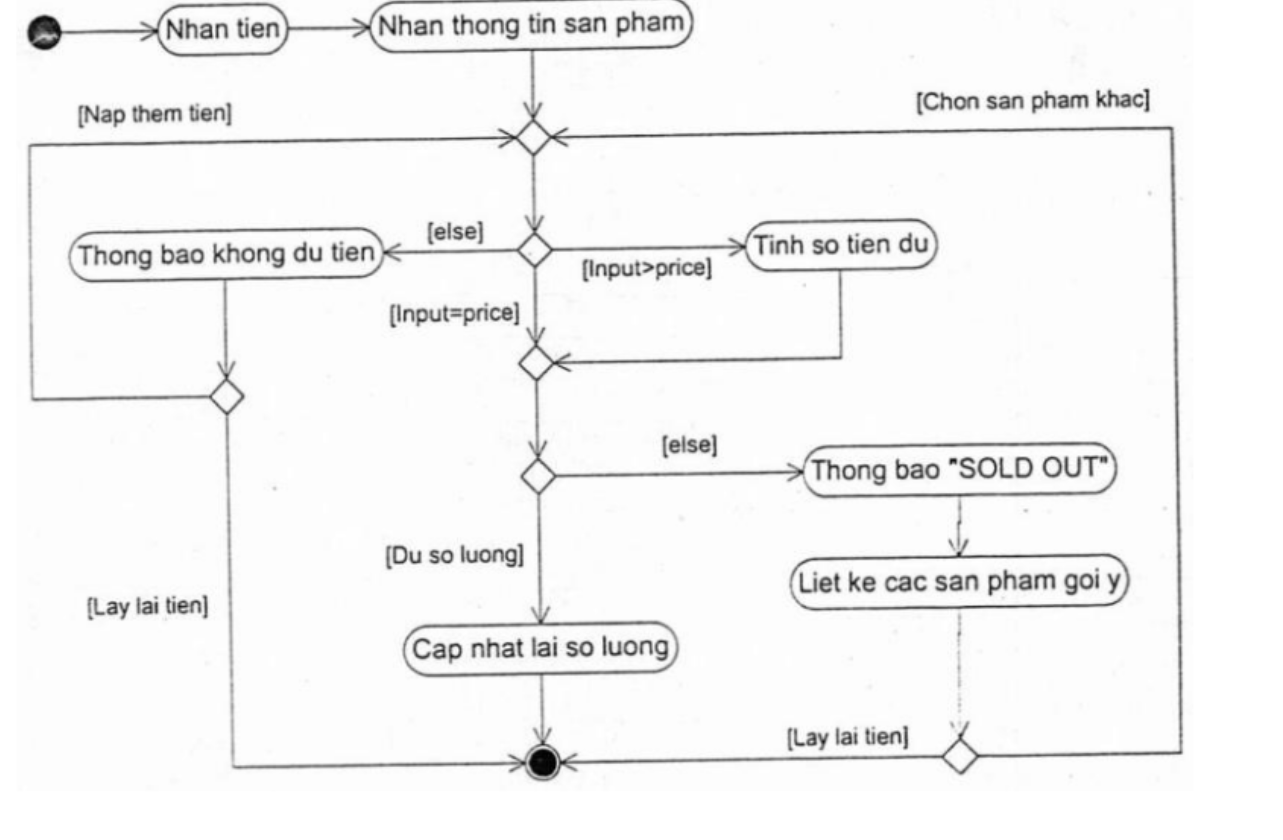
|  | | |
| --- | --- | --- |
| Use Case Number: | UC1 | |
| Use Case Name: | Đăng ký học phần (Register for Courses) | |
| Actor (s): | *Student* | |
| Maturity: | Focused | |
| Summary: | *Sinh viên thực hiện đăng ký học phần cho kỳ học sắp tới. Hệ thống kiểm tra điều kiện tiên quyết, cập nhật cơ sở dữ liệu đăng ký, và gửi thông tin đến Hệ thống thanh toán để xử lý học phí.* | |
| Basic Course of Events: | Actor Action | System Response |
| 1. Actor thực hiện {Đăng nhập hệ thống} |  |
|  | 2. Hệ thống hiển thị trang chào mừng với chức năng chính để Actor chọn. |
| 3. Actor chọn “Đăng ký học phần”. |  |
|  | 4. Hệ thống hiển thị danh sách học phần có thể đăng ký trong kỳ. Actor chọn học phần (có thể lặp lại nhiều lần: chọn học phần, kỳ, ngày, giờ). |
| 5. Sinh viên nhấn “Lưu đăng ký” A1 A2. |  |
|  | 6. Hệ thống cập nhật dữ liệu đăng ký học phần vào cơ sở dữ liệu (E1). |
|  | 7. Hệ thống gửi thông tin đăng ký sang Hệ thống thanh toán để tính học phí. |
|  | 8. Hệ thống thông báo cho Sinh viên rằng “đăng ký thành công và vui lòng đóng học phí đúng hạn” |
|  | |
| Alternative Paths: | A1. Trong bước 5, Sinh viên có thể xem thông tin chi tiết của môn học từ danh mục. | |
| A2. Trong bước 5, Sinh viên có thể chọn “Reset” để chọn lại học phần→ chuyển về bước 4 | |
| Exception Paths: | E1. Tại bước 7, nếu vi phạm điều kiện (chưa đủ tiên quyết, quá số môn, đã học qua môn) → hệ thống báo lỗi và quay lại bước 4. | |
| Extension Points: | Kiểm tra học phần: Đảm bảo sinh viên đáp ứng điều kiện tiên quyết, không vượt quá số môn tối đa và không đăng ký trùng môn đã qua. | |
| Triggers: | Actor wishes to input *student preferences* for future semesters. | |
| Assumptions: | Hệ thống thanh toán sẵn sàng hoạt động tại thời điểm đăng ký. | |
| Preconditions: | Sinh viên đã đăng nhập vào hệ thống. | |
| Post Conditions: | Dữ liệu đăng ký học phần được lưu trữ và thông tin thanh toán đang chờ xử lý. | |
| Reference: Business Rules | None | |
| Author(s): | XXXXX | |
| Date: | 11-14-07 | |
| Activity Diagram: Next Page | | |

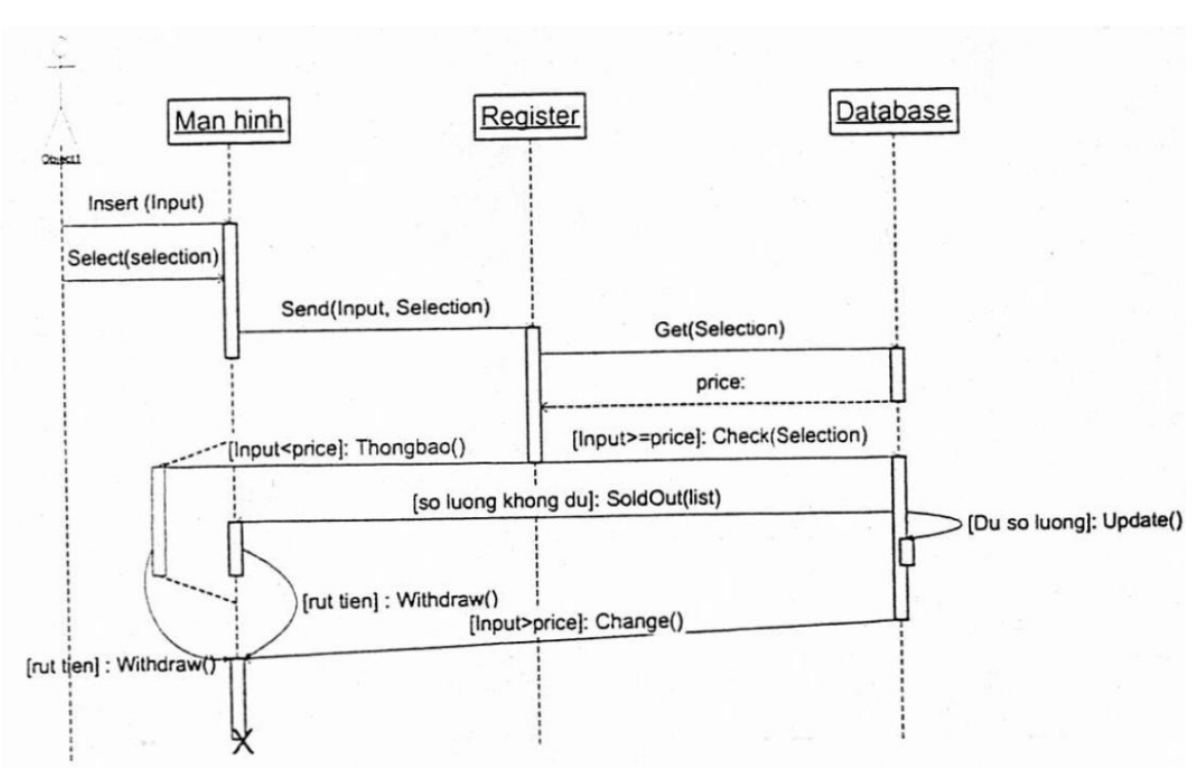
## Đề xuất cải tiến hoặc bổ sung thêm các use case mà bạn thấy cần thiết.

Để cải tiến hệ thống, có thể bổ sung thêm một số use case quan trọng. Chẳng hạn như:

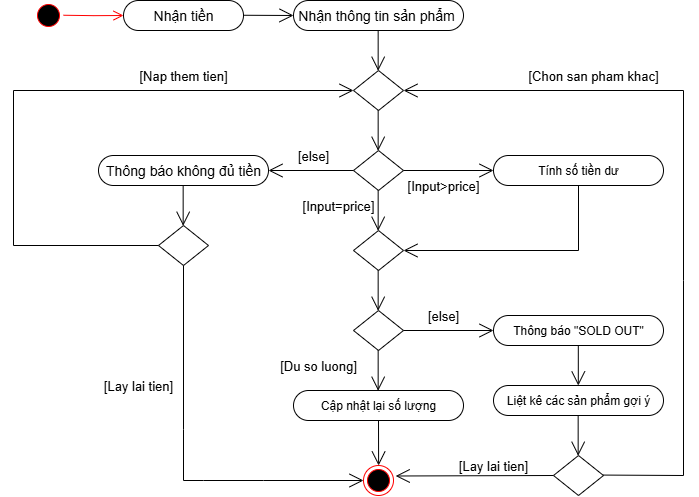
* Student – View Course Schedule: Use case này cho phép sinh viên xem toàn bộ lịch học, bao gồm môn học, thời gian, địa điểm và giảng viên phụ trách. Nhờ đó, sinh viên có thể quản lý thời gian hiệu quả hơn, sắp xếp kế hoạch học tập hợp lý và chuẩn bị tốt cho từng buổi học.
* Professor – Grade Assignments / Exams: Use case này cho phép professor nhập và quản lý điểm cho các bài tập, kiểm tra hoặc kỳ thi. Giúp đảm bảo đánh giá kết quả học tập của sinh viên được thực hiện chính xác, đồng thời lưu trữ đầy đủ thông tin trong hệ thống.

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





**Trả lời:**

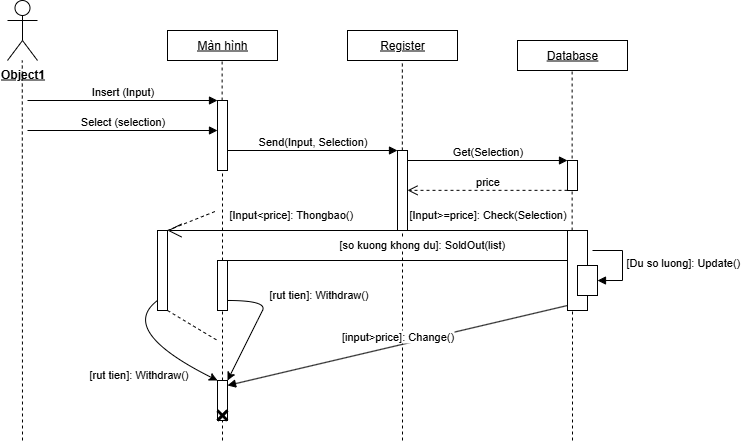


Sơ đồ này mô tả quy trình hoạt động của một máy bán hàng tự động.

Trước hết, người dùng sẽ nạp tiền vào máy rồi chọn sản phẩm muốn mua. Máy sẽ kiểm tra số tiền đã nạp: nếu chưa đủ, máy sẽ thông báo và yêu cầu người dùng nạp thêm; nếu số tiền nhiều hơn giá sản phẩm, máy sẽ tính số tiền dư để trả lại.

Tiếp theo, máy kiểm tra xem sản phẩm còn hàng hay không. Nếu đã hết, máy sẽ thông báo “SOLD OUT” và gợi ý các sản phẩm khác. Ngược lại, nếu vẫn còn hàng, máy sẽ cập nhật lại số lượng sau khi bán.

Cuối cùng, máy sẽ trả lại tiền dư (nếu có) và hoàn tất giao dịch. Người dùng có thể chọn tiếp sản phẩm khác hoặc kết thúc mua hàng.



Sơ đồ trên là sơ đồ tuần tự (sequence diagram) mô tả cách các thành phần trong hệ thống máy bán hàng tự động tương tác với nhau trong quá trình mua hàng.

Người dùng (Object1) sẽ nạp tiền (Insert) và chọn sản phẩm (Select) trên màn hình.

Màn hình gửi thông tin tiền và sản phẩm đã chọn sang Register (bộ xử lý trung gian), rồi Register lấy thông tin sản phẩm từ Database để biết giá và số lượng.

Nếu số tiền nhỏ hơn giá sản phẩm, màn hình sẽ thông báo không đủ tiền cho người dùng.

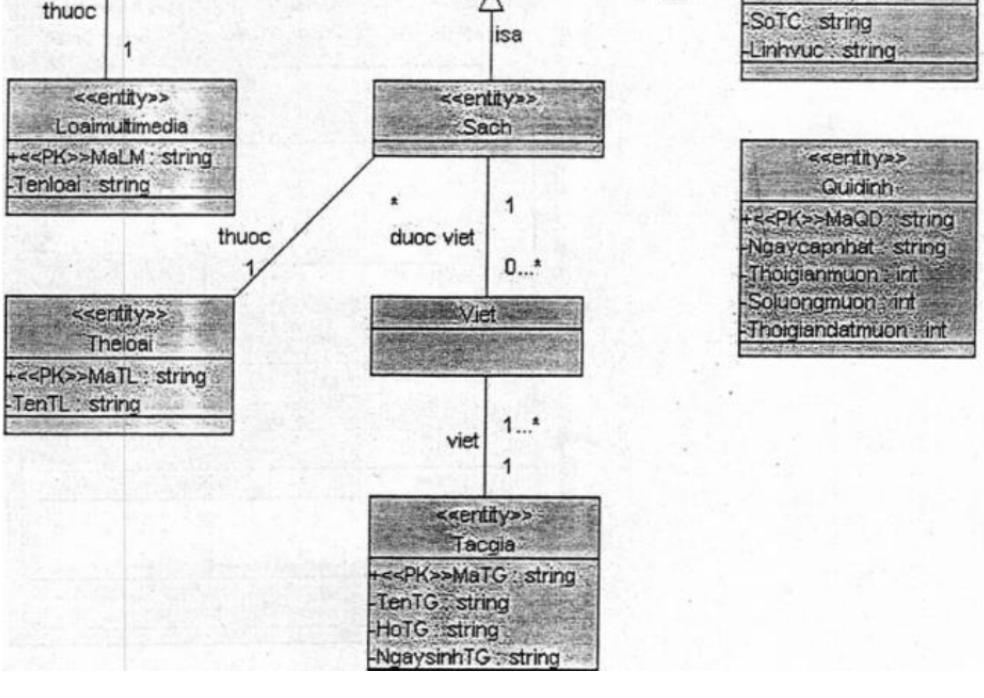
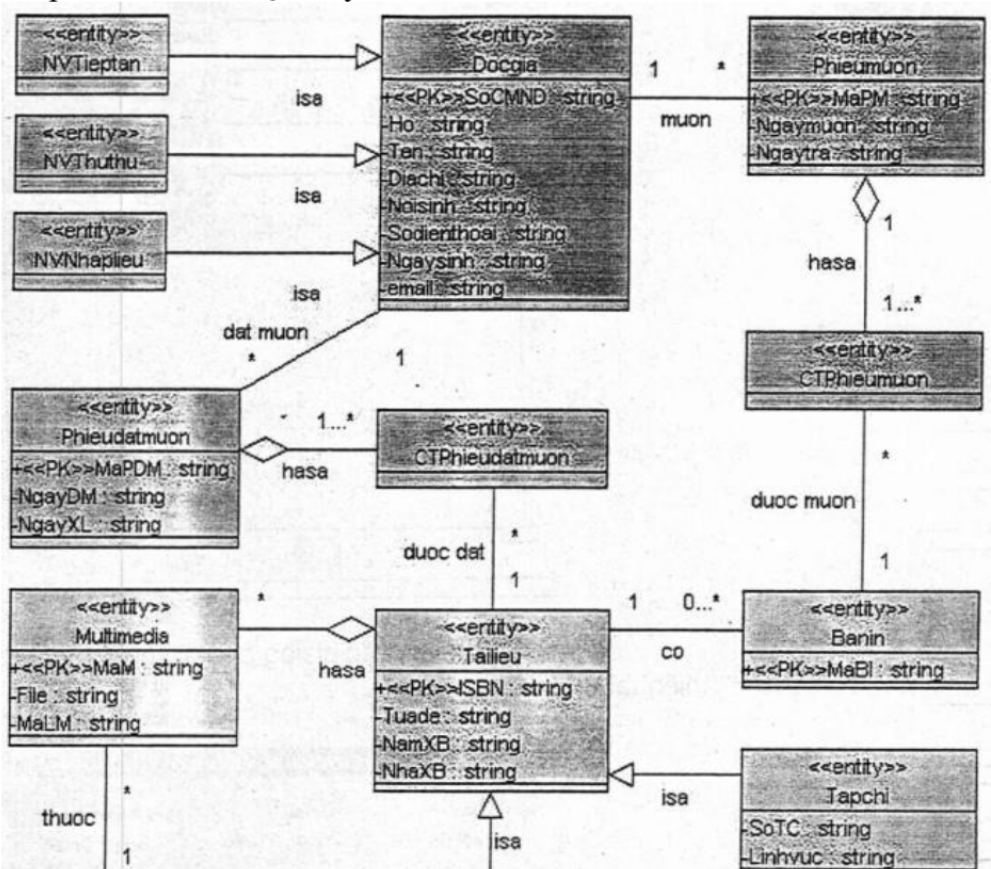
Nếu số tiền lớn hơn hoặc bằng giá sản phẩm, Register sẽ: Kiểm tra số lượng sản phẩm trong Database.

Nếu hết hàng, sẽ trả về danh sách sản phẩm gợi ý (SoldOut). Nếu còn hàng, sẽ cập nhật lại số lượng sản phẩm (Update) trong Database.

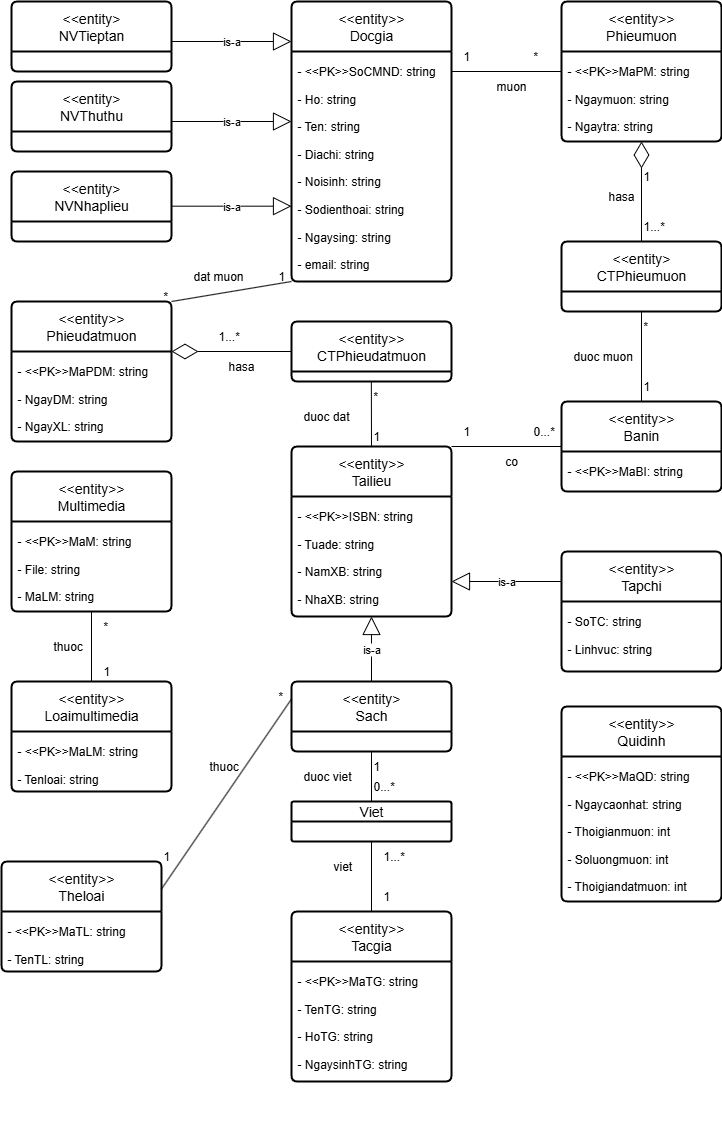
Xuất hàng cho người dùng (Withdraw). Nếu có dư tiền, hệ thống sẽ trả lại tiền thừa (Change).

Cuối cùng, giao dịch kết thúc và người dùng nhận được sản phẩm cùng tiền thừa (nếu có).

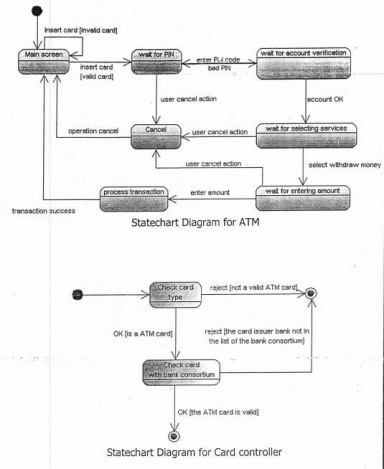
# **Bài 5. Vẽ lược đồ lớp cho bài toán Quản lý thư viện**



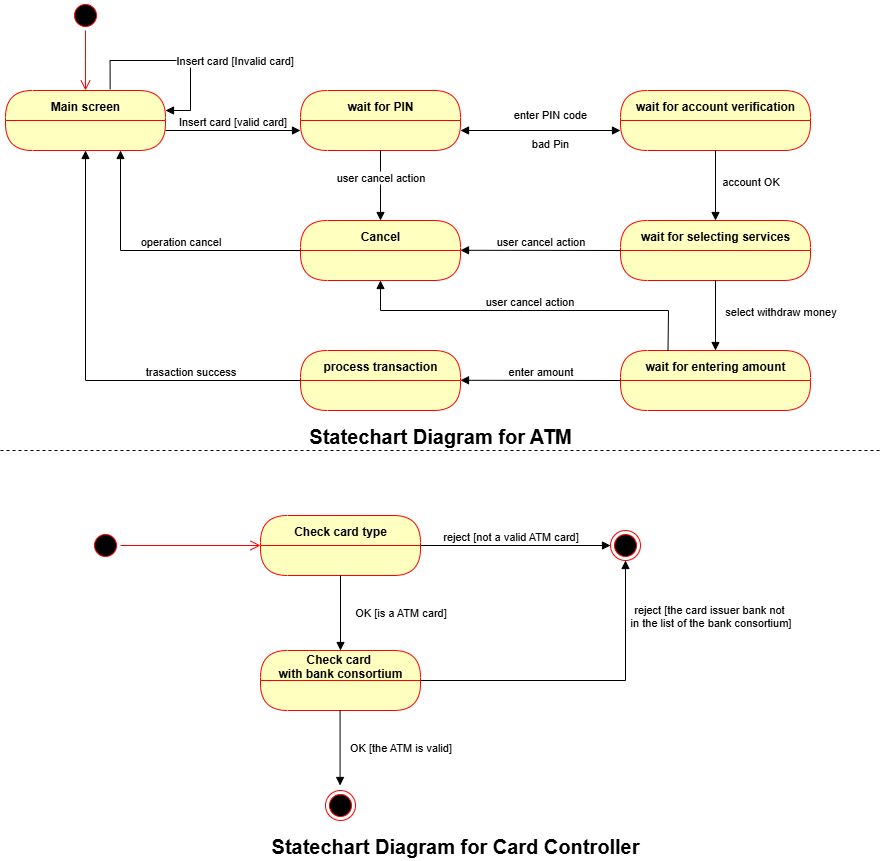
**Trả lời:**



# **Bài 6. Vẽ lược đồ sau:**



Trả lời:

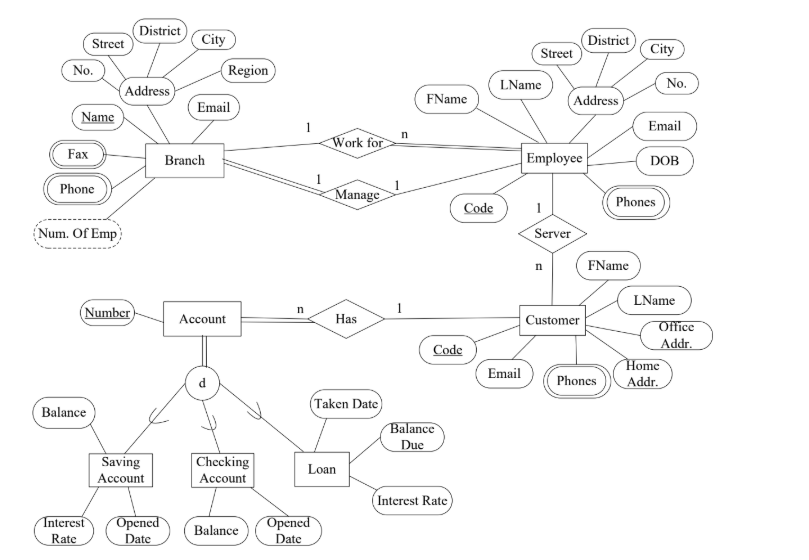


# **Bài 7: Phân tích dữ liệu cho ABC Bank**

ABC Bank has many branches. Each branch has a unique name, an address (including No., Street,

District, City, and Region), phone numbers, fax numbers, email, and total number of employees working there. Each branch has many employees and an employee must work at a branch. There is always one employee assigned to manage that branch. The employee can only manage the office to which he or she is assigned. For each employee, we need to store a unique code, first name, last name, date of birth, phone numbers, email, and home address (including No., Street, District, and City). The bank issues three different types of bank accounts for their customers. For Savings Accounts, the bank must keep track of the account’s balance, interest rate, and the date the account was opened. Checking Accounts pay no interest, so the bank keeps track of just the balance and date opened. The third type of account, Loans, requires tracking the date the loan was taken, the balance due, and the interest rate of the loan. Each customer may have any number of bank accounts, and all accounts have a unique account number. The bank saves the following information of each customer: a unique code, first name, last name, home address, office address, phone numbers, email. Each customer is attended by an employee and an employee can serve many customers.

Solution Example:



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, OfficeAddress, HomeAddress, Email, *EmpCode*) CUSTOMERPHONE (CusCode, Phone)

ACCOUNT (Number, *CusCode*)

SAVINGACCOUNT (AccNumber, Balance, InterestRate, OpenedDate)

CHECKINGACCOUNT (AccNumber, Balance, OpenedDate)

LOAN (AccNumber, BalanceDue, InterestRate, TakenDate)

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

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

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

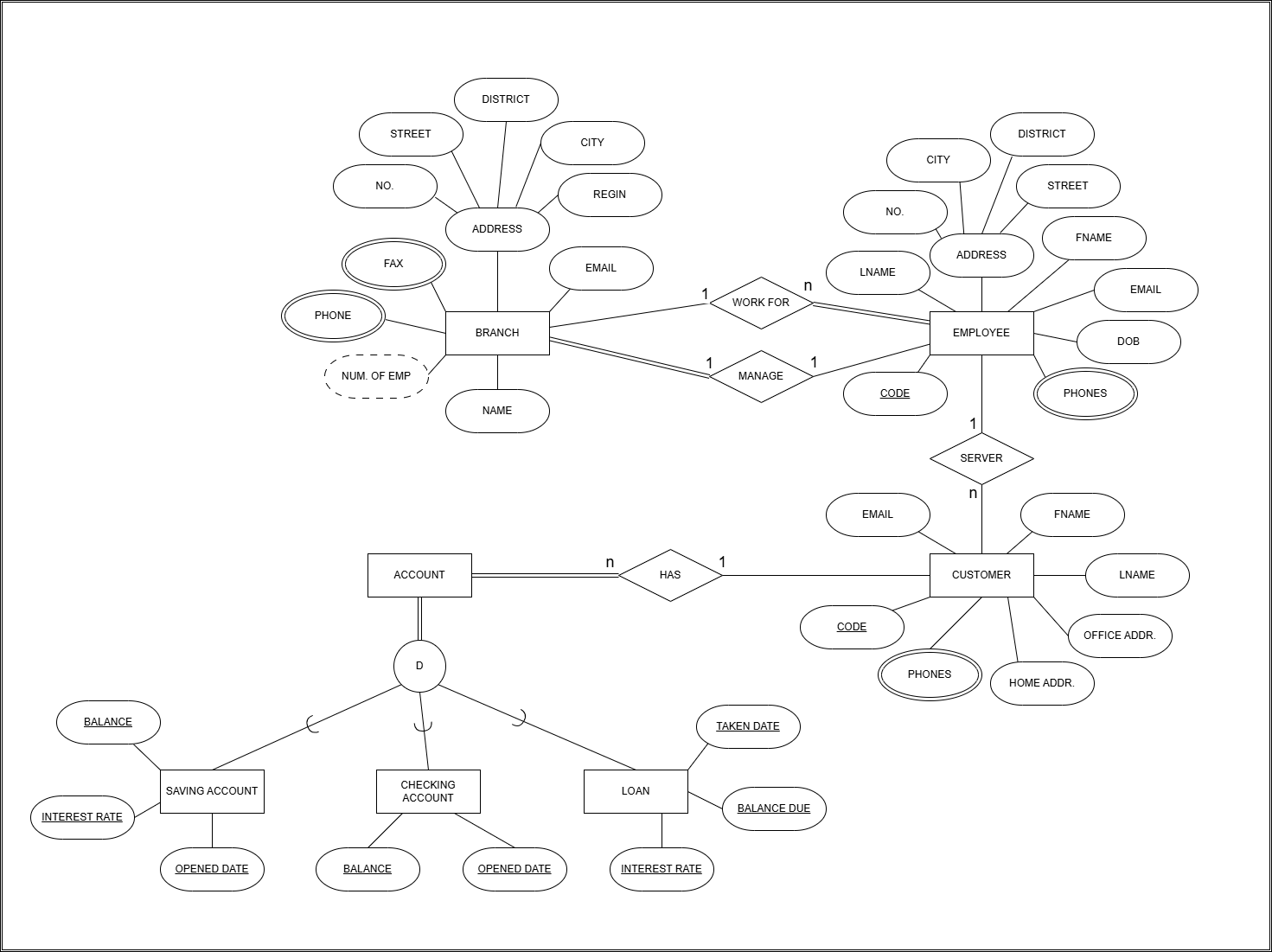
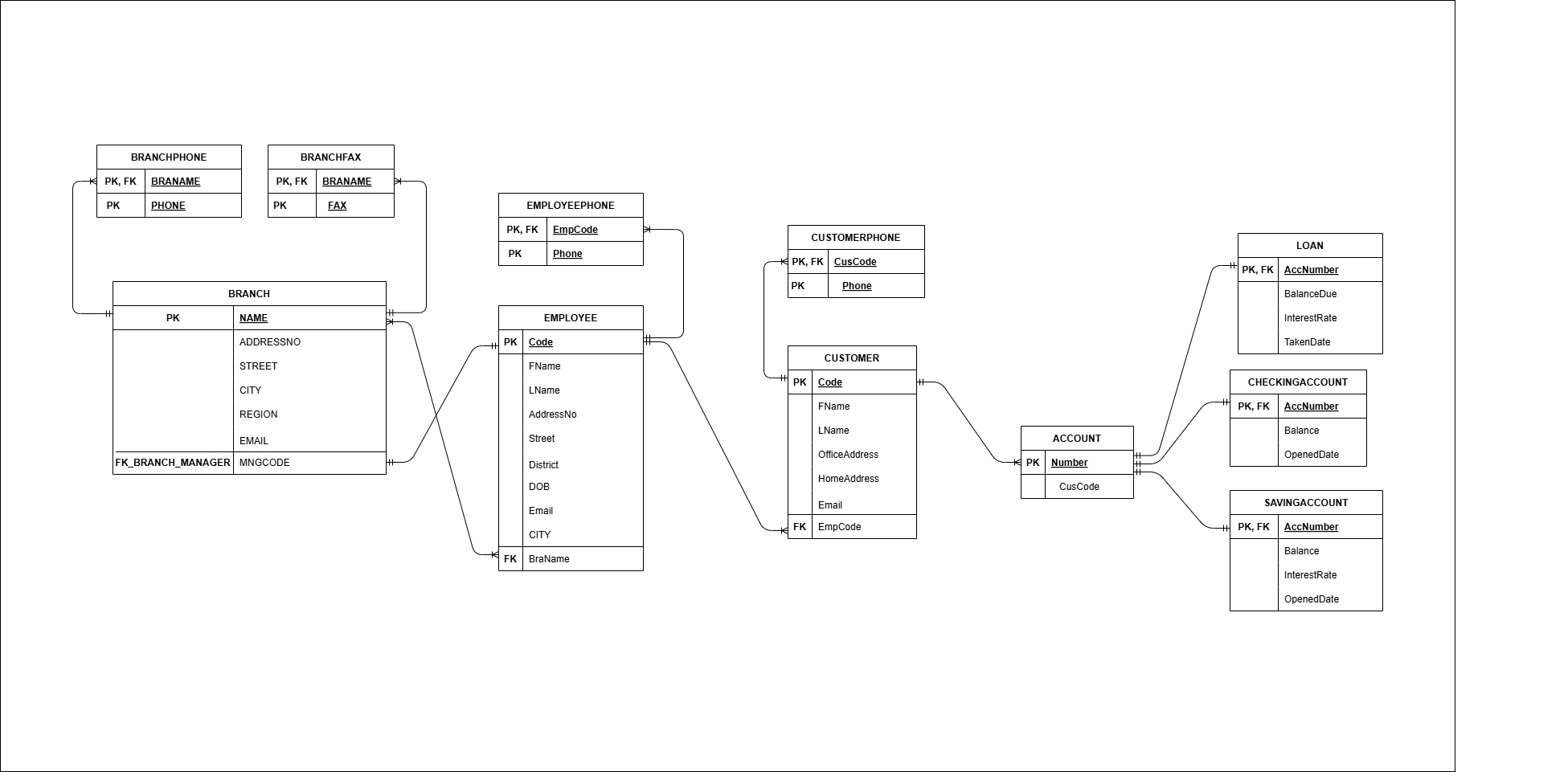
**c. Delete the employee having code “N1080”. Explain what will happen to the customers who are served by this employee.**

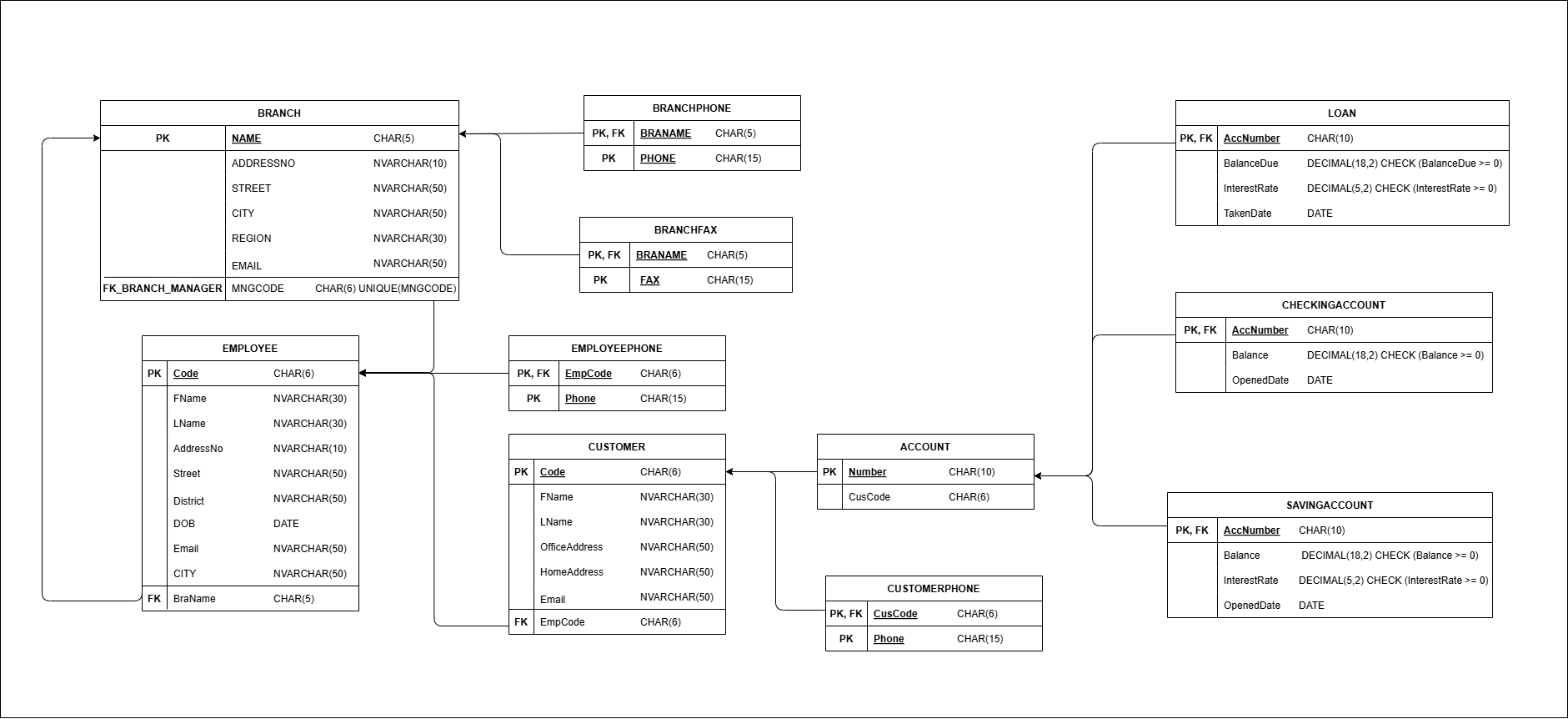
**d. Retrieve all account numbers of a customer whose name is Peter Johnson.**

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

**f. Retrieve the total balance of all saving accounts owned by a customer named “Peter Johnson” (Supposing that he has many saving accounts)**

Trả lời:

* CDM 
* LDM
* PDM



2. Query: Insert, Update, Delete, Select

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

-- BRANCH

INSERT INTO BRANCH (Name, AddressNo, Street, District, City, Region, Email, MngCode)

VALUES ('B001', '12', 'Nguyen Trai', 'Thanh Xuan', 'Ha Noi', 'North', 'b001@bank.com', NULL),

('B002', '45', 'Le Loi', '1', 'HCM', 'South', 'b002@bank.com', NULL),

('B003', '23', 'Hai Ba Trung', '3', 'Da Nang', 'Central', 'b003@bank.com', NULL),

('B004', '67', 'Tran Hung Dao', 'Hai Chau', 'Da Nang', 'Central', 'b004@bank.com', NULL);

-- EMPLOYEE

INSERT INTO EMPLOYEE VALUES

('N1001', 'John', 'Smith', '12', 'Ly Thuong Kiet', 'Hoan Kiem', 'Ha Noi', '1985-05-10', 'john@bank.com', 'B001'),

('N1002', 'Anna', 'White', '45', 'Pasteur', '1', 'HCM', '1990-03-20', 'anna@bank.com', 'B002'),

('N1003', 'Peter', 'Brown', '56', 'Nguyen Hue', '3', 'HCM', '2010-03-15', 'peter@bank.com', 'B002'),

('N1080', 'David', 'Miller', '78', 'Tran Phu', 'Hai Chau', 'Da Nang', '1988-07-22', 'david@bank.com', 'B004');

-- UPDATE BRANCH

UPDATE BRANCH SET MngCode = 'N1001' WHERE Name = 'B001';

UPDATE BRANCH SET MngCode = 'N1002' WHERE Name = 'B002';

UPDATE BRANCH SET MngCode = 'N1003' WHERE Name = 'B003';

UPDATE BRANCH SET MngCode = 'N1080' WHERE Name = 'B004';

-- CUSTOMER

INSERT INTO CUSTOMER VALUES

('C001', 'Peter', 'Johnson', 'CTY ABC', '12 Ba Trieu', 'peter@cus.com', 'N1001'),

('C002', 'Mary', 'Tran', 'CTY XYZ', '45 Dien Bien', 'mary@cus.com', 'N1002'),

('C003', 'Tom', 'Nguyen', 'CTY KLM', '78 Vo Thi Sau', 'tom@cus.com', 'N1003'),

('C004', 'Lucy', 'Pham', 'CTY DEF', '90 Tran Hung Dao', 'lucy@cus.com', 'N1080');

-- ACCOUNT

INSERT INTO ACCOUNT VALUES

('A001', 'C001'),

('A002', 'C001'),

('A003', 'C002'),

('A004', 'C003');

-- SAVINGACCOUNT

INSERT INTO SAVINGACCOUNT VALUES

('A001', 5000, 0.05, '2020-01-10'),

('A002', 3000, 0.04, '2021-06-15'),

('A003', 7000, 0.05, '2019-12-01'),

('A004', 2000, 0.03, '2021-11-20');

-- BRANCHPHONE

INSERT INTO BRANCHPHONE VALUES ('B001', '024-1111111');

INSERT INTO BRANCHPHONE VALUES ('B001', '024-2222222');

INSERT INTO BRANCHPHONE VALUES ('B002', '028-3333333');

INSERT INTO BRANCHPHONE VALUES ('B003', '0236-4444444');

-- BRANCHFAX

INSERT INTO BRANCHFAX VALUES ('B001', '024-9999999');

INSERT INTO BRANCHFAX VALUES ('B002', '028-8888888');

INSERT INTO BRANCHFAX VALUES ('B003', '0236-7777777');

INSERT INTO BRANCHFAX VALUES ('B004', '0236-6666666');

-- EMPLOYEEPHONE

INSERT INTO EMPLOYEEPHONE VALUES ('N1001', '090-1234567');

INSERT INTO EMPLOYEEPHONE VALUES ('N1001', '091-2222333');

INSERT INTO EMPLOYEEPHONE VALUES ('N1002', '093-4444555');

INSERT INTO EMPLOYEEPHONE VALUES ('N1003', '094-5555666');

-- CUSTOMERPHONE

INSERT INTO CUSTOMERPHONE VALUES ('C001', '098-1111111');

INSERT INTO CUSTOMERPHONE VALUES ('C001', '098-2222222');

INSERT INTO CUSTOMERPHONE VALUES ('C002', '097-3333333');

INSERT INTO CUSTOMERPHONE VALUES ('C003', '096-4444444');

-- CHECKINGACCOUNT

INSERT INTO CHECKINGACCOUNT VALUES ('A001', 1500, '2020-01-10');

INSERT INTO CHECKINGACCOUNT VALUES ('A002', 2500, '2021-06-15');

INSERT INTO CHECKINGACCOUNT VALUES ('A003', 1000, '2019-12-01');

INSERT INTO CHECKINGACCOUNT VALUES ('A004', 800, '2021-11-20');

-- LOAN

INSERT INTO LOAN VALUES ('A001', 20000, 0.07, '2022-01-15');

INSERT INTO LOAN VALUES ('A002', 15000, 0.06, '2021-09-10');

INSERT INTO LOAN VALUES ('A003', 10000, 0.05, '2023-03-20');

INSERT INTO LOAN VALUES ('A004', 5000, 0.04, '2024-05-01');

1. ***Change the last name of an employee whose code is “N1080” to “Brown”.***

-- EX B

SELECT LName FROM EMPLOYEE WHERE Code = 'N1080';

UPDATE EMPLOYEE

SET LName = 'Brown'

WHERE Code = 'N1080';

SELECT LName FROM EMPLOYEE WHERE Code = 'N1080';

1. ***Delete the employee having code “N1080”. Explain what will happen to the customers who are served by this employee.***

Ta có bảng CUSTOMER có cột EmpCode tham chiếu đến EMPLOYEE(Code) để xác định nhân viên đang phục vụ khách hàng. Khi muốn xóa nhân viên có mã N1080, ta cần chú ý đến ràng buộc khóa ngoại (Foreign Key Constraint):

* Trường hợp không khai báo ON DELETE  
   → Không thể xóa được nhân viên N1080.  
   Lý do: Vì trong bảng CUSTOMER vẫn còn dữ liệu khách hàng có EmpCode = N1080, nếu xóa thì sẽ vi phạm toàn vẹn tham chiếu.
* Trường hợp ON DELETE CASCADE  
   → Khi xóa nhân viên N1080 thì tất cả khách hàng do nhân viên này quản lý cũng sẽ bị xóa luôn.  
   Hệ quả: Dữ liệu nhân viên và khách hàng liên quan đều mất khỏi hệ thống.

ALTER TABLE CUSTOMER DROP CONSTRAINT FK\_CUSTOMER\_EMPLOYEE;

ALTER TABLE CUSTOMER

ADD CONSTRAINT FK\_CUSTOMER\_EMPLOYEE\_CASCADE

FOREIGN KEY (EmpCode) REFERENCES EMPLOYEE(Code)

ON DELETE CASCADE;

DELETE FROM EMPLOYEE WHERE Code = 'N1080';

SELECT FName FROM EMPLOYEE WHERE Code = 'N1080'

SELECT FName FROM CUSTOMER WHERE EmpCode = 'N1080'

* Trường hợp ON DELETE SET NULL  
   → Khi xóa nhân viên N1080 thì các bản ghi khách hàng vẫn giữ lại, nhưng cột EmpCode của những khách hàng đó sẽ được chuyển thành NULL.  
   Điều này cho thấy các khách hàng này hiện không còn nhân viên phụ trách.

ALTER TABLE CUSTOMER DROP CONSTRAINT FK\_CUSTOMER\_EMPLOYEE;

ALTER TABLE CUSTOMER

ADD CONSTRAINT FK\_CUSTOMER\_EMPLOYEE\_SETNULL

FOREIGN KEY (EmpCode) REFERENCES EMPLOYEE(Code)

ON DELETE SET NULL;

DELETE FROM EMPLOYEE WHERE Code = 'N1080';

SELECT FName FROM EMPLOYEE WHERE Code = 'N1080'

SELECT FName FROM CUSTOMER WHERE EmpCode = 'N1080'

Việc xóa nhân viên N1080 ảnh hưởng trực tiếp đến dữ liệu khách hàng. Tùy vào cách khai báo ràng buộc khóa ngoại (Restrict, Cascade hay Set Null) mà thao tác xóa có thể bị chặn lại, xóa cả khách hàng liên quan, hoặc chỉ làm mất liên kết giữa khách hàng với nhân viên.

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

-- EX D

SELECT A.Number

FROM ACCOUNT A

JOIN CUSTOMER C ON A.CusCode = C.Code

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

1. ***Find all employees who have their birthdays in March 2010.***

-- EX E

SELECT \*

FROM EMPLOYEE

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

1. *Retrieve the total balance of all saving accounts owned by a customer named “Peter Johnson” (Supposing that he has many saving accounts)*

-- EX F

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

Tham khảo: https://vietnam-devs.github.io/coolstore-microservices

Website CoolStore có kịch bản kinh doanh cơ bản cho Danh mục sản phẩm (Product Catalog), Giỏ hàng (Shopping Cart), Quy trình thanh toán (Payment Process), Tồn kho (Inventory), Đánh giá (Rating) và Kiểm soát truy cập (Access Control).

Với Danh mục sản phẩm, người mua (Buyer) có thể duyệt danh sách sản phẩm với các chức năng lọc và sắp xếp theo tên sản phẩm và giá. Người mua có thể xem chi tiết sản phẩm trên trang danh sách sản phẩm bằng cách nhấp vào sản phẩm đó. Trên trang chi tiết, người mua có thể xem tên, mô tả, số lượng tồn kho, thông tin kho như địa chỉ kho, trạng thái "sản phẩm nổi bật" (nếu có) và đánh giá. Quản trị viên (SysAdmin) trong hệ thống có thể quản lý sản phẩm và có quyền gán sản phẩm vào kho hiện có.

Với Giỏ hàng, người mua có thể mua bất kỳ sản phẩm nào trên danh sách sản phẩm thông qua nút "Mua hàng" trên mỗi sản phẩm. Ngoài ra, người mua cũng có thể mua sản phẩm tại trang chi tiết sản phẩm. Sau khi mua sản phẩm, người mua sẽ thấy các sản phẩm này trong giỏ hàng và bảng tổng hợp với thông tin cơ bản như tổng chi phí giỏ hàng, tổng chi phí sau khuyến mãi, chi phí vận chuyển, khuyến mãi tiết kiệm, tổng giá trị đơn hàng. Bất cứ khi nào người mua thêm sản phẩm hoặc xóa sản phẩm khỏi giỏ hàng, thì bảng tổng hợp và giỏ hàng sẽ được cập nhật. Sau đó, người mua có thể thực hiện quy trình thanh toán bằng cách nhấn nút Thanh toán trên trang giỏ hàng. Quản trị viên có thể xem toàn bộ giỏ hàng của bất kỳ người dùng nào để có thể kích hoạt hoặc vô hiệu hóa bất kỳ giỏ hàng không hợp lệ nào trên website CoolStore.

Với Quy trình thanh toán, sau khi người mua nhấn nút thanh toán, hệ thống sẽ bắt đầu xác thực thông tin sản phẩm, xử lý thanh toán, và sau đó gửi email cho người mua để người mua biết chuyện gì đang xảy ra.

Với Tồn kho, Quản trị viên có thể quản lý kho hàng.

Với Đánh giá, người mua có thể đánh giá bất kỳ sản phẩm nào mà cô ấy thấy tốt (thang điểm 1 -> 5 sao).

Với Kiểm soát truy cập, người mua hoặc quản trị viên có thể đăng nhập/đăng xuất hệ thống. Nếu là người mua, người đó sẽ được đưa đến trang danh mục sản phẩm, còn nếu là quản trị viên, sẽ được đưa đến trang quản trị.

Một số nhiệm vụ thiết lập ban đầu cần được thực hiện khi website CoolStore khởi động như: tạo người dùng quản trị, hai người dùng mua hàng, và dữ liệu mẫu cho sản phẩm, kho, đánh giá cho một vài sản phẩm.

**Câu hỏi**

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.

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).

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).

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.

Trả lời:

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

* Functional Requirement:
* Product Catalog

Buyer (Người mua) có thể:

* + Xem danh sách sản phẩm
  + Lọc và sắp xếp theo tên sản phẩm và giá
  + Xem chi tiết sản phẩm (tên, mô tả, số lượng, tồn kho, thông tin kho như địa chỉ kho, trạng thái “sản phẩm nổi bật” , và đánh giá- rating)

SysAdmin (Quản trị viên) có thể:

* + Quản lý sản phẩm
  + Gán sản phẩm vào kho hiện có
* Shopping Cart

Buyer (Người mua) có thể:

* + Mua sản phẩm thông qua nút “Mua hàng” trên mỗi sản phẩm hoặc tại trang chi tiết sản phẩm.
  + Xem giỏ hàng và bảng tổng hợp (tổng chi phí giỏ hàng, tổng chi phí sau khuyến mãi, chi phí vận chuyển, khuyến mãi tiết kiệm, tổng giá trị đơn hàng)
  + Thêm sản phẩm hoặc xóa sản phẩm khỏi giỏ hàng -> bảng tổng hợp và giỏ hàng được cập nhật.
  + Thực hiện thanh toán bằng cách nhấn nút “Thanh toán” trên trang giỏ hàng.

SysAdmin (Quản trị viên) có thể:

* + Xem toàn bộ giỏ hàng của mọi người dùng
  + Kích hoạt hoặc vô hiệu hóa các giỏ hàng không hợp lệ trên website
* Payment process

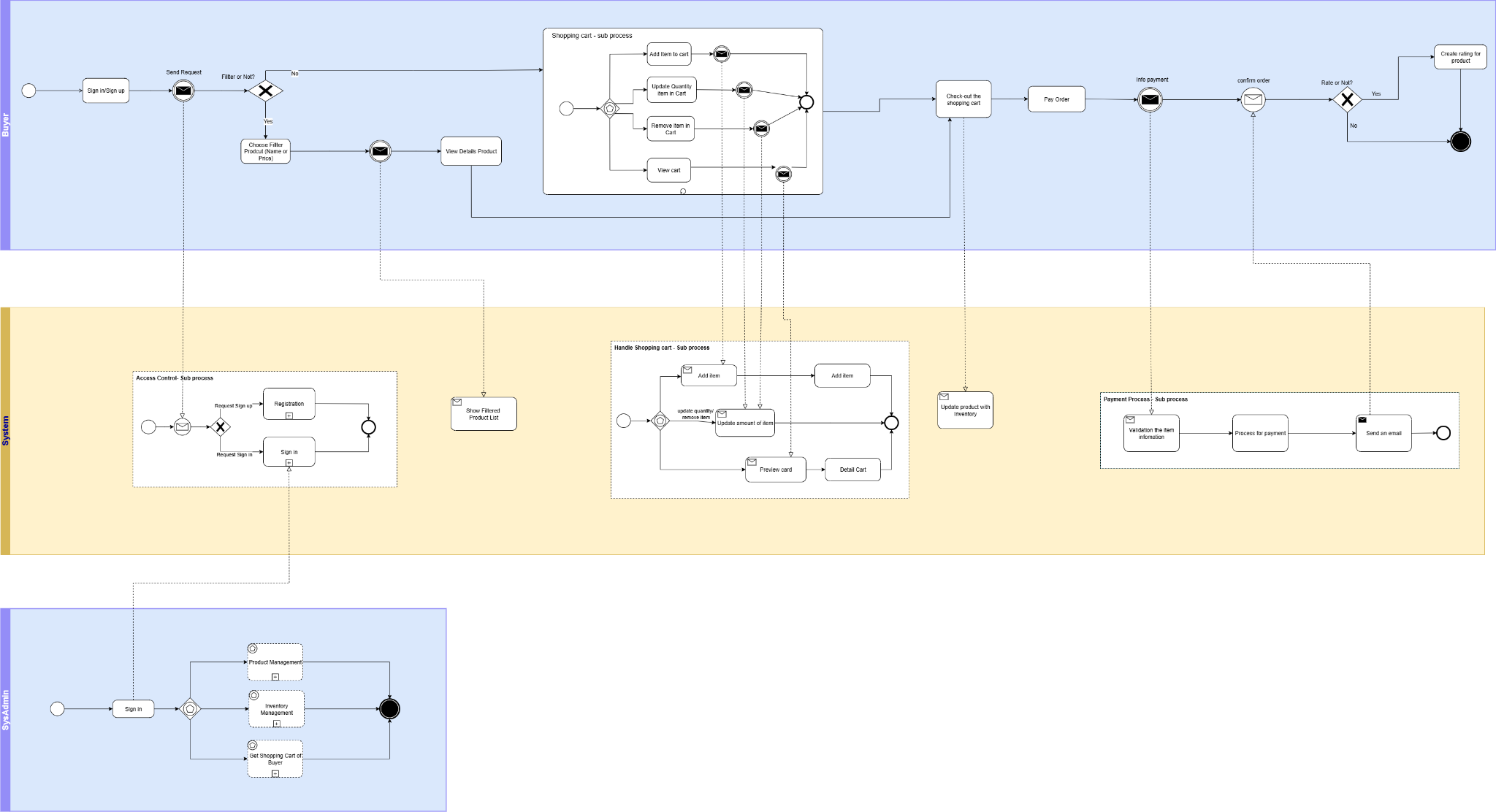
System sẽ:

* + Xác thực thông tin sản phẩm khi người dùng nhấn nút “Thanh toán”
  + Xử lý thanh toán
  + Gửi email xác nhận cho người mua
* Inventory
  + SysAdmin (Quản trị viên) có thể quản lý kho hàng
* Rating
  + Buyer (Người mua) có thể đánh giá sản phẩm (thang điểm từ 1 đến 5 sao)
* Access Control
  + Người dùng (Buyer và SysAdmin) có thể:
  + Đăng nhập/ đăng xuất.
  + Buyer được chuyển đến danh mục sản phẩm
  + SysAdmin được chuyển đến trang quản trị

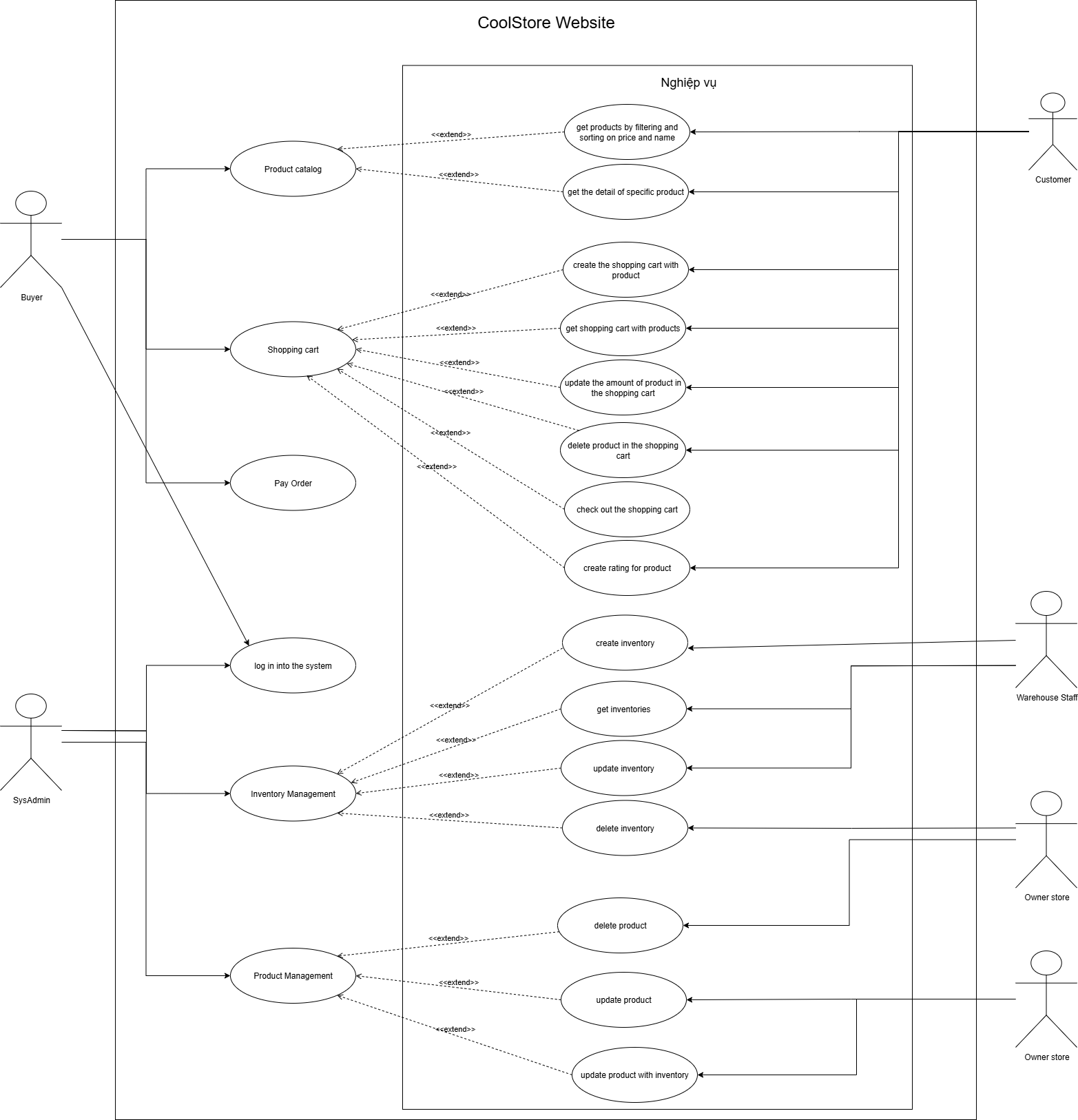
II. **Non-Functional**

* Availability (Tính khả dụng)
  + Hệ thống cần hoạt động ổn định, cho phép người dùng truy cập liên tục để mua hàng và quản lý
* Performance (Hiệu suất):
  + Chức năng lọc/ sắp xếp sản phẩm và cập nhật giỏ hàng cần được phản hồi nhanh
  + Thanh toán và email cần xử lý không trễ
* Security (Bảo mật)
  + Đăng nhập cần bảo mật
  + SysAdmin có quyền cao hơn Buyer
  + Các giỏ hàng không hợp lệ cần được quản lý để tránh gian lận
* Email Notification
  + Gửi email xác nhận tự động sau khi thanh toán

1. **Vẽ quy trình nghiệp vụ cho website CoolStore**



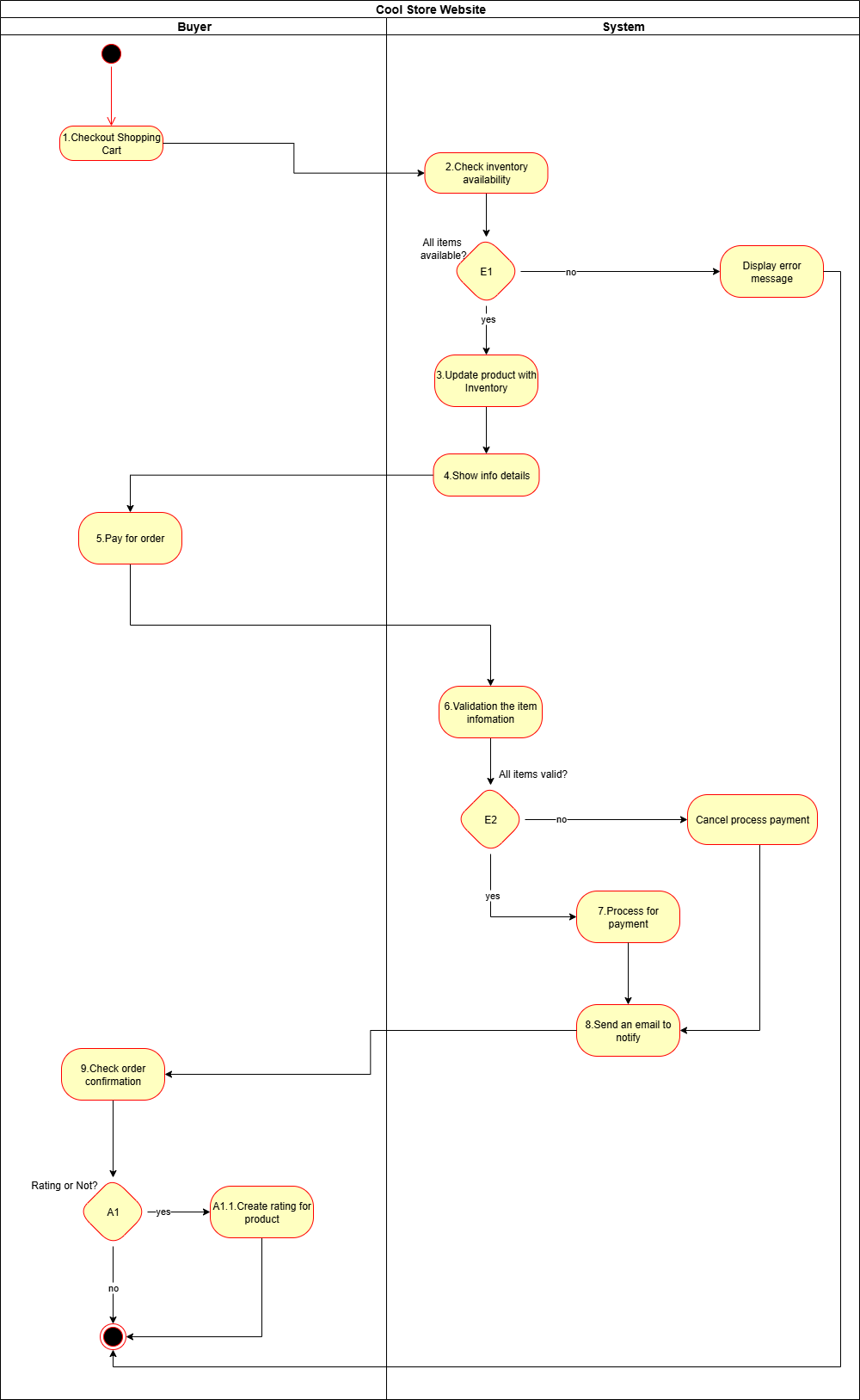
1. **Xây dựng use case diagram cho website CoolStore.**



1. **Kịch bản "Mua sản phẩm từ trang chi tiết sản phẩm”**

|  | | |
| --- | --- | --- |
| Use Case Number: | UC1 | |
| Use Case Name: | Mua sản phẩm từ trang chi tiết sản phẩm | |
| Actor (s): | *Buyer* | |
| Maturity: | Focused | |
| Summary: | *Người dùng chọn sản phẩm từ trang chi tiết, chọn thuộc tính (màu, size, số lượng), sau đó nhấn "Mua ngay" hoặc "Thêm vào giỏ hàng". Hệ thống xử lý thêm sản phẩm, kiểm tra tồn kho và chuyển đến bước thanh toán hoặc tiếp tục mua sắm.* | |
| Basic Course of Events: | Actor Action | System Response |
| 1. Actor chọn “Mua hàng” tại trang chi tiết sản phẩm |  |
|  | 2. Kiểm tra tình trạng hàng tồn kho. E1 |
|  | 3. Cập nhật sản phẩm với tồn kho |
|  | 4. Hiển thị thông tin chi tiết đơn hàng |
| 5. Nhấn “Thanh toán” |  |
|  | 6. Xác thực thông tin sản phẩm |
|  | 7. Thực hiện xử lý thanh toán. E2 |
|  | 8. Gửi email thông báo |
| 9. Kiểm tra xác nhận đơn hàng. Use case kết thúc tại đây. A1 |  |
| Alternative Paths: | A1. Buyer tạo đánh giá sản phẩm, Use case kết thúc tại đây | |
| Exception Paths: | E1. Nếu các sản phẩm không có sẵn, hệ thống thông báo lỗi, Use case kết thúc tại đây.  E2. Nếu thông tin sản phẩm mua hàng không hợp lệ (có thay đổi về khuyến mãi, số lượng của sản phẩm…), hệ thống hủy xử lý thanh toán, Use case tiếp tục tại bước 8 của luồng sự kiện chính. | |
| Extension Points: | None | |
| Triggers: | Người dùng chọn mua sản phẩm từ trang chi tiết. | |
| Assumptions: | Người dùng đã đăng nhập trước khi thực hiện thao tác mua hàng  Sản phẩm đang còn hoạt động và có sẵn để đặt mua | |
| Preconditions: | Người dùng đã đăng nhập và đang ở trang chi tiết sản phẩm. | |
| Post Conditions: | Đơn hàng đã được tạo và lưu vào hệ thống  Người dùng nhận thông báo email xác nhận mua hàng | |
| Reference: Business Rules | None | |
| Author(s): | XXXXX | |
| Date: | 17-09-25 | |
| Activity Diagram: | | |

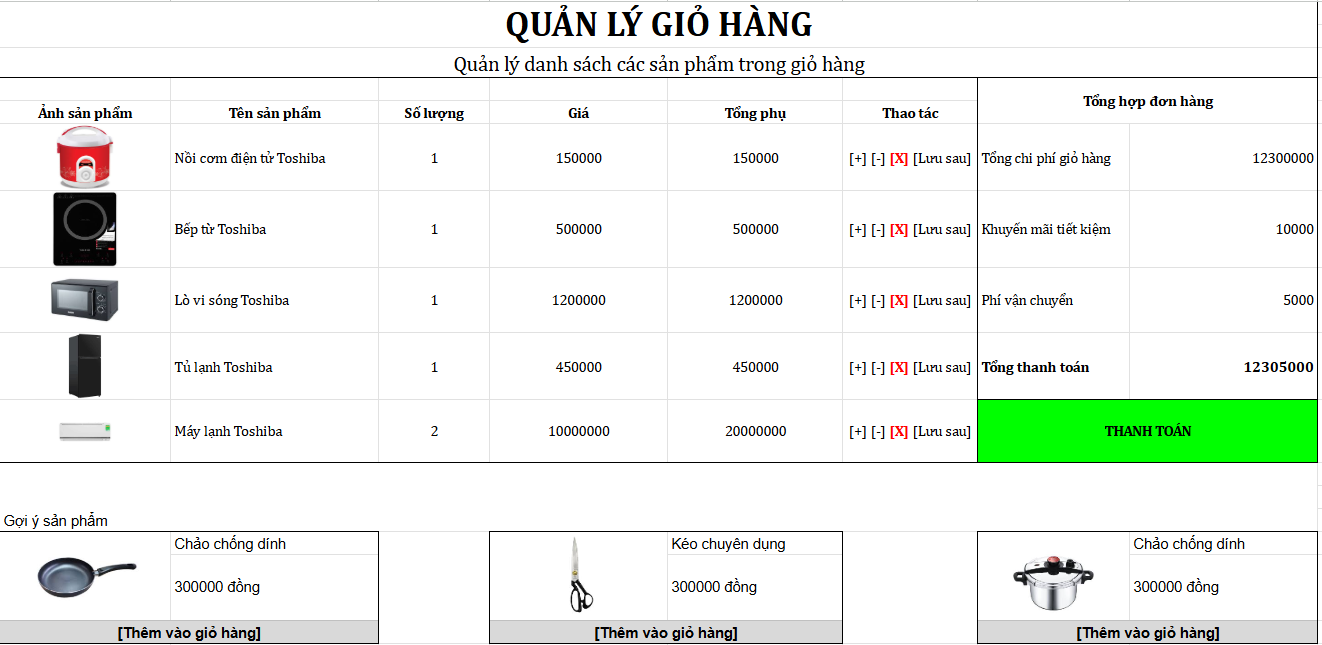
1. **Sơ đồ Activity Diagram để mô tả quy trình mua hàng từ khi người dùng nhấn “Mua hàng” cho đến khi hoàn tất thanh toán.**



1. **Thiết kế class diagram cho “Giỏ hàng”**

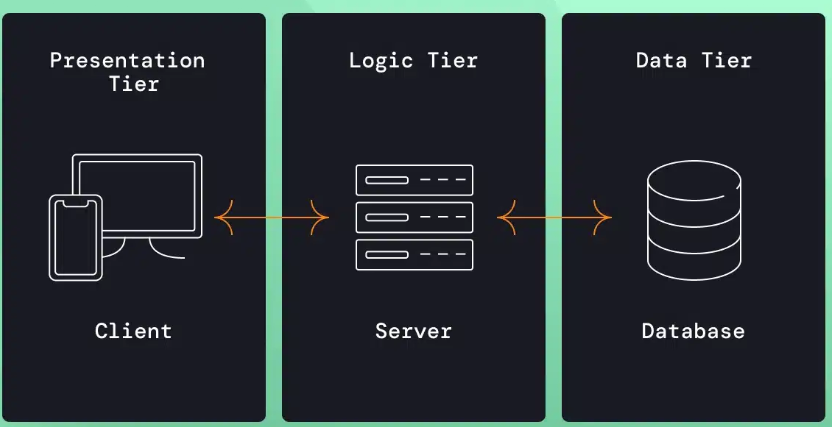


1. **Thiết kế giao diện màn hình cho chức năng “Quản lý giỏ hàng”**



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

* Nguồn tham khảo: <https://vfunction.com/blog/3-tier-application/#toc-heading-2>



* Khái niệm:

Là một kiến trúc kiểu client/server mà trong đó, các yếu tố như:

* + *Presentation Tier*: Giao diện người dùng nơi mà end-user tương tác với hệ thống. Nhiệm vụ chính là hiển thị dữ liệu và nhận dữ liệu từ người dùng.
  + *Logic Tier*: Tầng giữa của kiến ​​trúc, còn được gọi là tầng logic, xử lý các quy tắc kinh doanh, tính toán và xử lý cốt lõi của ứng dụng. Gồm các thành phần Business Logic Layer (BLL), Data Access Layer (DAL) và Data Transfer Object (DTO).
  + *Data Tier:* Việc lưu trữ dữ liệu được phát triển như những module độc lập, quản lý lưu trữ, truy xuất và thao tác dữ liệu của ứng dụng, thường sử dụng cơ sở dữ liệu

→ những yếu tố trên được duy trì trên các nền tảng độc lập. Mô hình 3 tầng (3-tiers) được coi là một kiến trúc phần mềm và là một mẫu thiết kế.

* Đề xuất cách tổ chức kiến trúc phần mềm cho hệ thống CoolStore:

Hệ thống sẽ bao gồm 3 phần :

1. Presentation Tier (Client Layer)

* Vai trò:
  + Giao diện để Buyer và SysAdmin tương tác với hệ thống.
  + Gửi yêu cầu (request) đến Logic Tier và hiển thị dữ liệu trả về.
* Thành phần:
  + Buyer UI (Product Catalog, Shopping Cart, Checkout, Rating).
  + SysAdmin UI (Quản lý sản phẩm, tồn kho, giỏ hàng).
* Công nghệ gợi ý: ReactJS / Angular / Vue.
* Chức năng:
  + Buyer: đăng nhập/đăng xuất, duyệt sản phẩm, xem chi tiết, thêm vào giỏ hàng, cập nhật/xóa sản phẩm trong giỏ, checkout, rating.
  + SysAdmin: đăng nhập/đăng xuất, quản lý sản phẩm, quản lý tồn kho, quản lý giỏ hàng của Buyer (bật/tắt).
* Lưu ý quan trọng:
  + UI không truy cập trực tiếp DB.
  + Mọi thao tác phải gọi đến Logic Tier → đảm bảo tính đúng đắn nghiệp vụ.

1. Logic Tier (server layer)

* Vai trò:
  + Xử lý toàn bộ nghiệp vụ của hệ thống.
  + Đóng vai trò "trung gian" giữa Presentation và Data Tier.
* Công nghệ gợi ý: .NET Core / Java Spring Boot / Node.js (RESTful API hoặc GraphQL).
* Các module nghiệp vụ chính:
  + Access Control Service: xác thực (login/logout), phân quyền (Buyer/SysAdmin).
  + Product Catalog Service: quản lý sản phẩm (CRUD, tìm kiếm, lọc/sắp xếp, hot product, rating).
  + Shopping Cart Service: thêm/xóa/sửa sản phẩm trong giỏ, tính toán tổng tiền/khuyến mãi, checkout.
  + Order & Payment Service: xử lý thanh toán, gửi email thông báo.
  + Inventory Service: quản lý tồn kho, gán sản phẩm cho kho.
* Đặc điểm:
  + Tất cả nghiệp vụ được xử lý tập trung tại đây.
  + Có thể mở rộng thành microservices nếu hệ thống phát triển lớn.
  + Có thể triển khai load balancing cho hiệu năng cao.
* Nguyên tắc chung: Mỗi service đều phải kiểm tra tính hợp lệ, các điều kiện ràng buộc của dữ liệu trước khi xử lý logic

1. Data Tier (Database Layer)

* Vai trò:
  + Lưu trữ dữ liệu bền vững của hệ thống.
  + Chỉ giao tiếp với Logic Tier, không cho Presentation Tier truy cập trực tiếp.
* Database chính:
  + ProductDB: sản phẩm, rating.
  + CartDB: giỏ hàng, trạng thái checkout.
  + OrderDB: đơn hàng, giao dịch thanh toán.
  + InventoryDB: kho hàng, vị trí, số lượng.
  + UserDB: thông tin Buyer & SysAdmin.
* Công nghệ gợi ý:
  + SQL Server / PostgreSQL cho quan hệ.
  + Redis (cache tạm thời giỏ hàng để tăng tốc).

1. **Test cases cho chức năng “Thêm sản phẩm vào giỏ hàng”**

Test Case 1 — Thêm sản phẩm có tồn kho đủ

* ID: TC\_ADD\_001
* Mục tiêu: Xác nhận thêm 1 sản phẩm có tồn kho >= số lượng yêu cầu.
* Dữ liệu đầu vào: Product A (stock=10), qty=2, user đã đăng nhập.
* Các bước:
  + Mở trang chi tiết Product A.
  + Chọn qty = 2.
  + Nhấn nút Thêm vào giỏ.
* Kết quả mong đợi:
  + Hệ thống trả về success.
  + Item xuất hiện trong giỏ với quantity=2.
  + lineTotal = unitPrice \* 2.
  + cart.subtotal được cập nhật tương ứng.
  + Không có lỗi tồn kho.

Test Case 2 — Thêm sản phẩm vượt quá tồn kho

* ID: TC\_ADD\_002
* Mục tiêu: Hệ thống báo lỗi khi thêm số lượng lớn hơn stock.
* Dữ liệu đầu vào: Product B (stock=3), qty=5, user đã đăng nhập.
* Các bước:
  + Mở trang chi tiết Product B.
  + Nhập qty = 5.
  + Nhấn Thêm vào giỏ.
* Kết quả mong đợi:
  + Hệ thống trả về thông báo lỗi: "Số lượng yêu cầu vượt quá tồn kho (chỉ còn 3)."
  + Không thêm item vào giỏ hoặc thêm tối đa số lượng còn lại theo cấu hình.
  + Không làm thay đổi cart.subtotal.

Test Case 3 — Thêm sản phẩm khi user chưa đăng nhập (guest)

* ID: TC\_ADD\_003
* Mục tiêu: Kiểm tra chức năng thêm giỏ cho guest (session-based).
* Dữ liệu đầu vào: Product C (stock=5), qty=1, user chưa đăng nhập.
* Các bước:
  + Ở trạng thái không đăng nhập, mở trang Product C.
  + Nhấn Thêm vào giỏ.
  + Mở mini-cart hoặc trang giỏ.
* Kết quả mong đợi:
  + Sản phẩm được thêm vào giỏ dựa trên session/cookie.
  + Khi user đăng nhập sau đó, giỏ được merge (nếu policy là merge).
  + cart hiển thị đúng item & subtotal.

Thẻ 2

**BÀI TẬP ỨNG DỤNG**

Tham khảo: https://vietnam-devs.github.io/coolstore-microservices

Website CoolStore có kịch bản kinh doanh cơ bản cho **Danh mục sản phẩm (Product Catalog), Giỏ hàng (Shopping Cart), Quy trình thanh toán (Payment Process), Tồn kho (Inventory), Đánh giá (Rating)** và **Kiểm soát truy cập (Access Control)**.

Với **Danh mục sản phẩm**, người mua (Buyer) có thể duyệt danh sách sản phẩm với các chức năng lọc và sắp xếp theo tên sản phẩm và giá. Người mua có thể xem **chi tiết sản phẩm** trên trang danh sách sản phẩm bằng cách nhấp vào sản phẩm đó. Trên trang chi tiết, người mua có thể xem tên, mô tả, số lượng tồn kho, thông tin kho như địa chỉ kho, trạng thái "sản phẩm nổi bật" (nếu có) và đánh giá. **Quản trị viên (SysAdmin)** trong hệ thống có thể **quản lý sản phẩm** và có quyền **gán sản phẩm vào kho hiện có**.

Với **Giỏ hàng**, người mua có thể **mua bất kỳ sản phẩm nào** trên danh sách sản phẩm thông qua nút "Mua hàng" trên mỗi sản phẩm. Ngoài ra, người mua cũng có thể mua sản phẩm tại **trang chi tiết sản phẩm**. Sau khi mua sản phẩm, người mua sẽ thấy các sản phẩm này trong giỏ hàng và **bảng tổng hợp** với thông tin cơ bản như tổng chi phí giỏ hàng, tổng chi phí sau khuyến mãi, chi phí vận chuyển, khuyến mãi tiết kiệm, tổng giá trị đơn hàng. Bất cứ khi nào người mua **thêm sản phẩm** hoặc **xóa sản phẩm** khỏi giỏ hàng, thì **bảng tổng hợp và giỏ hàng sẽ được cập nhật**. Sau đó, người mua có thể thực hiện **quy trình thanh toán** bằng cách nhấn nút Thanh toán trên trang giỏ hàng. **Quản trị viên có thể xem toàn bộ giỏ hàng của bất kỳ người dùng nào** để có thể **kích hoạt hoặc vô hiệu hóa** bất kỳ giỏ hàng không hợp lệ nào trên website CoolStore.

Với **Quy trình thanh toán**, sau khi người mua nhấn nút thanh toán, hệ thống sẽ bắt đầu **xác thực thông tin sản phẩm**, **xử lý thanh toán**, và sau đó **gửi email cho người mua** để người mua biết chuyện gì đang xảy ra.

Với **Tồn kho**, **Quản trị viên** có thể **quản lý kho hàng**.

Với **Đánh giá**, người mua có thể **đánh giá bất kỳ sản phẩm nào** mà cô ấy thấy tốt (thang điểm 1 -> 5 sao).

Với **Kiểm soát truy cập**, người mua hoặc quản trị viên có thể **đăng nhập/đăng xuất hệ thống**. Nếu là người mua, người đó sẽ được đưa đến **trang danh mục sản phẩm**, còn nếu là quản trị viên, sẽ được đưa đến **trang quản trị**.

Một số **nhiệm vụ thiết lập ban đầu** cần được thực hiện khi website CoolStore khởi động như: tạo **người dùng quản trị**, **hai người dùng mua hàng**, và **dữ liệu mẫu** cho **sản phẩm, kho, đánh giá** cho một vài sản phẩm.

**Câu hỏi**

**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.**

1. **Functional Requirement:**
2. **Product Catalog**

Buyer (Người mua) có thể:

* Xem danh sách sản phẩm
* Lọc và sắp xếp theo tên sản phẩm và giá
* Xem chi tiết sản phẩm (tên, mô tả, số lượng, tồn kho, thông tin kho như địa chỉ kho, trạng thái “sản phẩm nổi bật” , và đánh giá- rating)

SysAdmin (Quản trị viên) có thể:

* Quản lý sản phẩm
* Gán sản phẩm vào kho hiện có

1. **Shopping Cart**

Buyer (Người mua) có thể:

* Mua sản phẩm thông qua nút “Mua hàng” trên mỗi sản phẩm hoặc tại trang chi tiết sản phẩm.
* Xem giỏ hàng và bảng tổng hợp (tổng chi phí giỏ hàng, tổng chi phí sau khuyến mãi, chi phí vận chuyển, khuyến mãi tiết kiệm, tổng giá trị đơn hàng)
* Thêm sản phẩm hoặc xóa sản phẩm khỏi giỏ hàng -> bảng tổng hợp và giỏ hàng được cập nhật.
* Thực hiện thanh toán bằng cách nhấn nút “Thanh toán” trên trang giỏ hàng.

SysAdmin (Quản trị viên) có thể:

* Xem toàn bộ giỏ hàng của mọi người dùng
* Kích hoạt hoặc vô hiệu hóa các giỏ hàng không hợp lệ trên website

1. **Payment process**

System sẽ:

* Xác thực thông tin sản phẩm khi người dùng nhấn nút “Thanh toán”
* Xử lý thanh toán
* Gửi email xác nhận cho người mua

1. **Inventory**

SysAdmin (Quản trị viên) có thể quản lý kho hàng

1. **Rating**

Buyer (Người mua) có thể đánh giá sản phẩm (thang điểm từ 1 đến 5 sao)

1. **Access Control**

Người dùng (Buyer và SysAdmin) có thể:

* Đăng nhập/ đăng xuất.
* Buyer được chuyển đến danh mục sản phẩm
* SysAdmin được chuyển đến trang quản trị

**II. Non-Functional**

1. **Availability (Tính khả dụng)**

* Hệ thống cần hoạt động ổn định, cho phép người dùng truy cập liên tục để mua hàng và quản lý

1. **Performance (Hiệu suất)**

* Chức năng lọc/ sắp xếp sản phẩm và cập nhật giỏ hàng cần được phản hồi nhanh
* Thanh toán và email cần xử lý không trễ

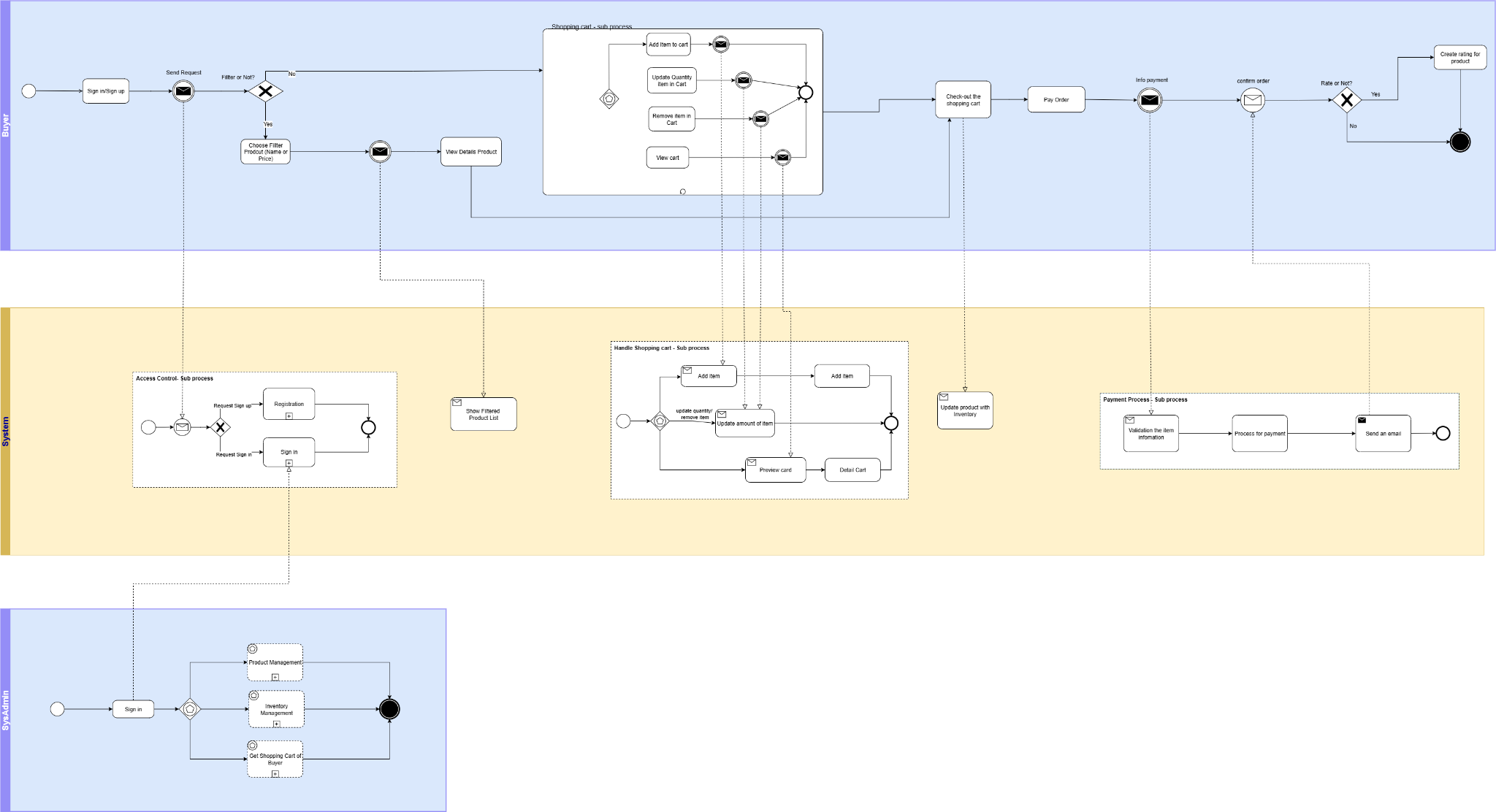
1. **Security (Bảo mật)**

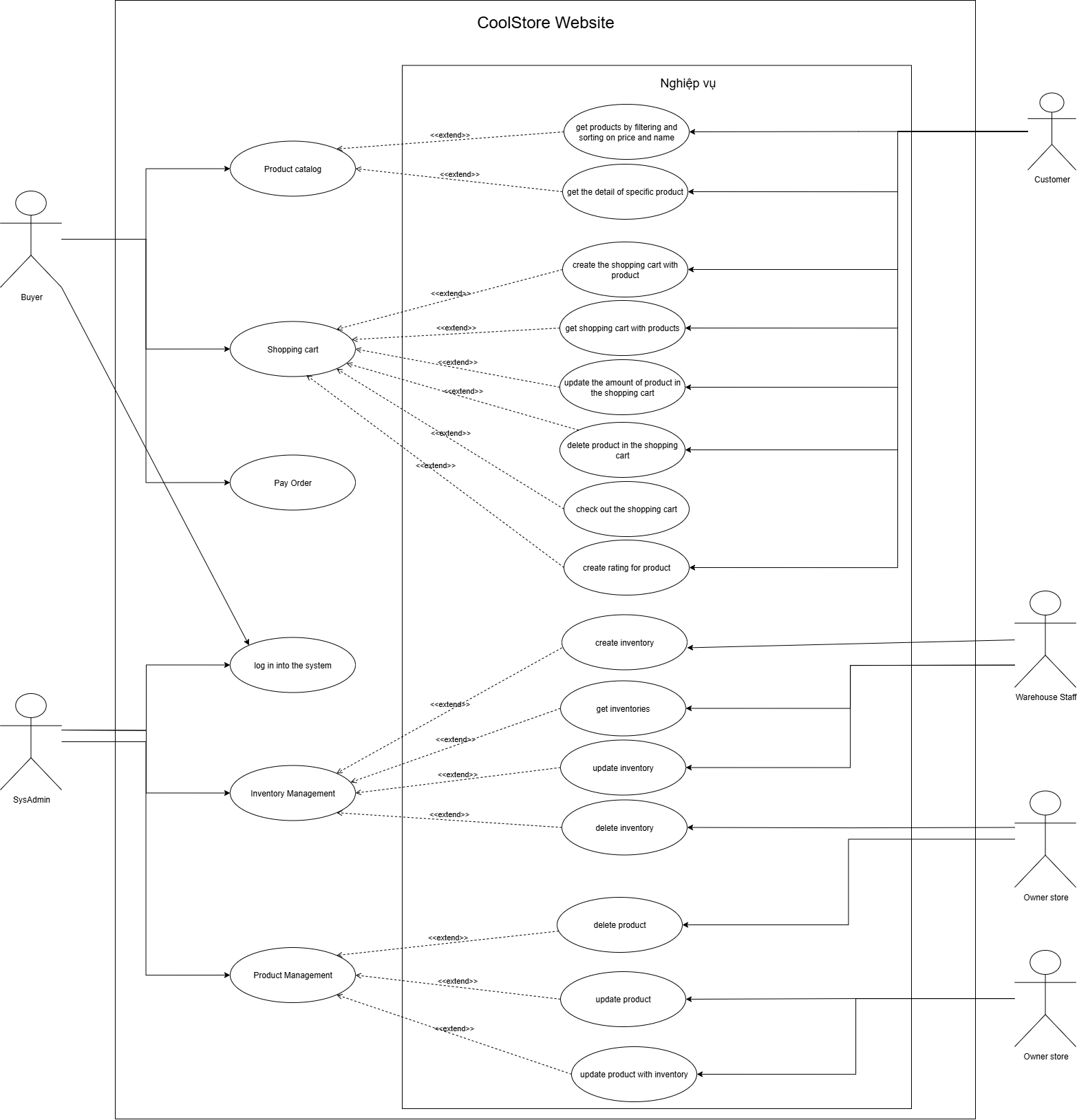
* Đăng nhập cần bảo mật
* SysAdmin có quyền cao hơn Buyer
* Các giỏ hàng không hợp lệ cần được quản lý để tránh gian lận

1. **Email Notification**

* Gửi email xác nhận tự động sau khi thanh toán

**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).**

**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.**

done

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



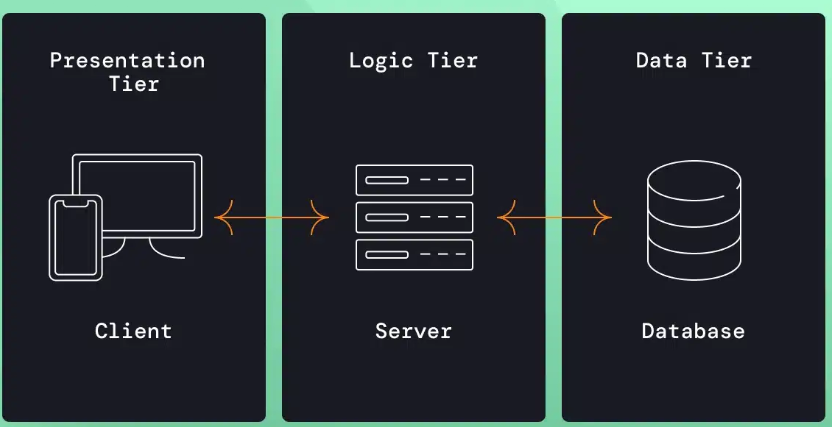
**7) Thiết kế giao diện màn hình cho chức năng “Quản lý giỏ hàng” *(có thể thiết kế một hoặc nhiều màn hình)* gồm các thông tin:**

* **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.**

**done**

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

Nguồn tham khảo: <https://vfunction.com/blog/3-tier-application/#toc-heading-2>



**Khái niệm:**

Là một kiến trúc kiểu client/server mà trong đó, các yếu tố như:

* *Presentation Tier*: Giao diện người dùng nơi mà end-user tương tác với hệ thống. Nhiệm vụ chính là hiển thị dữ liệu và nhận dữ liệu từ người dùng.
* *Logic Tier*: Tầng giữa của kiến ​​trúc, còn được gọi là tầng logic, xử lý các quy tắc kinh doanh, tính toán và xử lý cốt lõi của ứng dụng. Gồm các thành phần Business Logic Layer (BLL), Data Access Layer (DAL) và Data Transfer Object (DTO).
* *Data Tier:* Việc lưu trữ dữ liệu được phát triển như những module độc lập, quản lý lưu trữ, truy xuất và thao tác dữ liệu của ứng dụng, thường sử dụng cơ sở dữ liệu

Là những yếu tố được duy trì trên các nền tảng độc lập. Mô hình 3 tầng (3-tiers) được coi là một kiến trúc phần mềm và là một mẫu thiết kế.

**Đề xuất cách tổ chức kiến trúc phần mềm cho hệ thống CoolStore:**

Hệ thống sẽ bao gồm 3 phần :

1. Presentation Tier (Client Layer)

* Vai trò:
  + Giao diện để Buyer và SysAdmin tương tác với hệ thống.
  + Gửi yêu cầu (request) đến Logic Tier và hiển thị dữ liệu trả về.
* Thành phần:
  + Buyer UI (Product Catalog, Shopping Cart, Checkout, Rating).
  + SysAdmin UI (Quản lý sản phẩm, tồn kho, giỏ hàng).
* Công nghệ gợi ý: ReactJS / Angular / Vue.
* Chức năng:
  + Buyer: đăng nhập/đăng xuất, duyệt sản phẩm, xem chi tiết, thêm vào giỏ hàng, cập nhật/xóa sản phẩm trong giỏ, checkout, rating.
  + SysAdmin: đăng nhập/đăng xuất, quản lý sản phẩm, quản lý tồn kho, quản lý giỏ hàng của Buyer (bật/tắt).
* Lưu ý quan trọng:
  + UI không truy cập trực tiếp DB.
  + Mọi thao tác phải gọi đến Logic Tier → đảm bảo tính đúng đắn nghiệp vụ.

1. Logic Tier (server layer)

* Vai trò:
  + Xử lý toàn bộ nghiệp vụ của hệ thống.
  + Đóng vai trò "trung gian" giữa Presentation và Data Tier.
* Công nghệ gợi ý: .NET Core / Java Spring Boot / Node.js (RESTful API hoặc GraphQL).
* Các module nghiệp vụ chính:
  + Access Control Service: xác thực (login/logout), phân quyền (Buyer/SysAdmin).
  + Product Catalog Service: quản lý sản phẩm (CRUD, tìm kiếm, lọc/sắp xếp, hot product, rating).
  + Shopping Cart Service: thêm/xóa/sửa sản phẩm trong giỏ, tính toán tổng tiền/khuyến mãi, checkout.
  + Order & Payment Service: xử lý thanh toán, gửi email thông báo.
  + Inventory Service: quản lý tồn kho, gán sản phẩm cho kho.
* Đặc điểm:
  + Tất cả nghiệp vụ được xử lý tập trung tại đây.
  + Có thể mở rộng thành microservices nếu hệ thống phát triển lớn.
  + Có thể triển khai load balancing cho hiệu năng cao.
* Nguyên tắc chung: Mỗi service đều phải kiểm tra tính hợp lệ, các điều kiện ràng buộc của dữ liệu trước khi xử lý logic

1. Data Tier (Database Layer)

* Vai trò:
  + Lưu trữ dữ liệu bền vững của hệ thống.
  + Chỉ giao tiếp với Logic Tier, không cho Presentation Tier truy cập trực tiếp.
* Database chính:
  + ProductDB: sản phẩm, rating.
  + CartDB: giỏ hàng, trạng thái checkout.
  + OrderDB: đơn hàng, giao dịch thanh toán.
  + InventoryDB: kho hàng, vị trí, số lượng.
  + UserDB: thông tin Buyer & SysAdmin.
* Công nghệ gợi ý:
  + SQL Server / PostgreSQL cho quan hệ.
  + Redis (cache tạm thời giỏ hàng để tăng tốc).

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.

**Trả lời:**