# BT2 - Phân tích thiết kế hệ thống

Môn học: Kiểm thử phần mềm

Năm học: 2025-2026

Giảng viên: TS. Đỗ Như Tài

Thành viên:

Hồ Nguyên Minh 3122411175

Thông Trung Quyền 3122411123

Lục Trọng Khoa 3122411096

Mục lục:

[Mục lục: 2](#_Toc32581)

[Phân công công việc: 2](#_Toc21443)

[Bài tập minh họa 3](#_Toc15685)

[Bài 1. 3](#_Toc31999)

[Bài 2. 4](#_Toc30079)

[Bài 3. 4](#_Toc8099)

[Bài 4. 8](#_Toc30269)

[Bài 5. 10](#_Toc908)

[Bài 6. 11](#_Toc14264)

[Bài 7. 12](#_Toc6299)

[BÀI TẬP ỨNG DỤNG 19](#_Toc2225)

[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. 19](#_Toc8036)

[2) Vẽ qui trình nghiệp vụ cho website CoolStore 20](#_Toc2264)

[3) Xây dựng use case diagram cho website CoolStore. 21](#_Toc23680)

[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). 21](#_Toc10091)

[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. 22](#_Toc21301)

[6) Thiết kế sơ đồ Class Diagram cho module Giỏ hàng. 23](#_Toc20392)

[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: 23](#_Toc10893)

[8) Đề xuất cách tổ chức kiến trúc phần mềm cho hệ thống CoolStore theo mô hình kiến trúc ba lớp (3-tier architecture). 24](#_Toc31249)

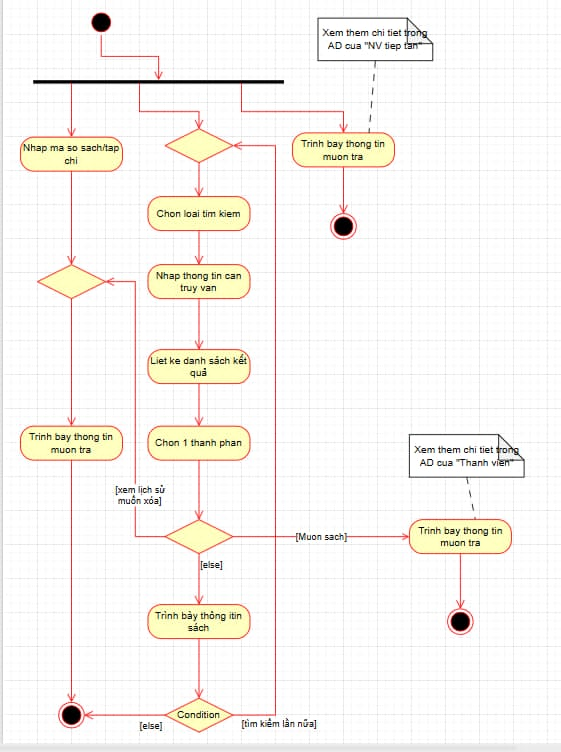
[9) Viết ít nhất 3 ca kiểm thử (test cases) cho chức năng “Thêm sản phẩm vào giỏ hàng” – bao gồm dữ liệu đầu vào, bước thực hiện và kết quả mong đợi. 25](#_Toc14367)

Phân công công việc:

|  |  |
| --- | --- |
| Thành viên | Công việc |
| Thông Trung Quyền | Làm phần bài tập ứng dụng |
| Lục Trọng Khoa | Vẽ hình, làm phần bài tập minh họa |
| Hồ Nguyên Minh | Tìm ý nghĩa các hình trong bài tập minh họa |

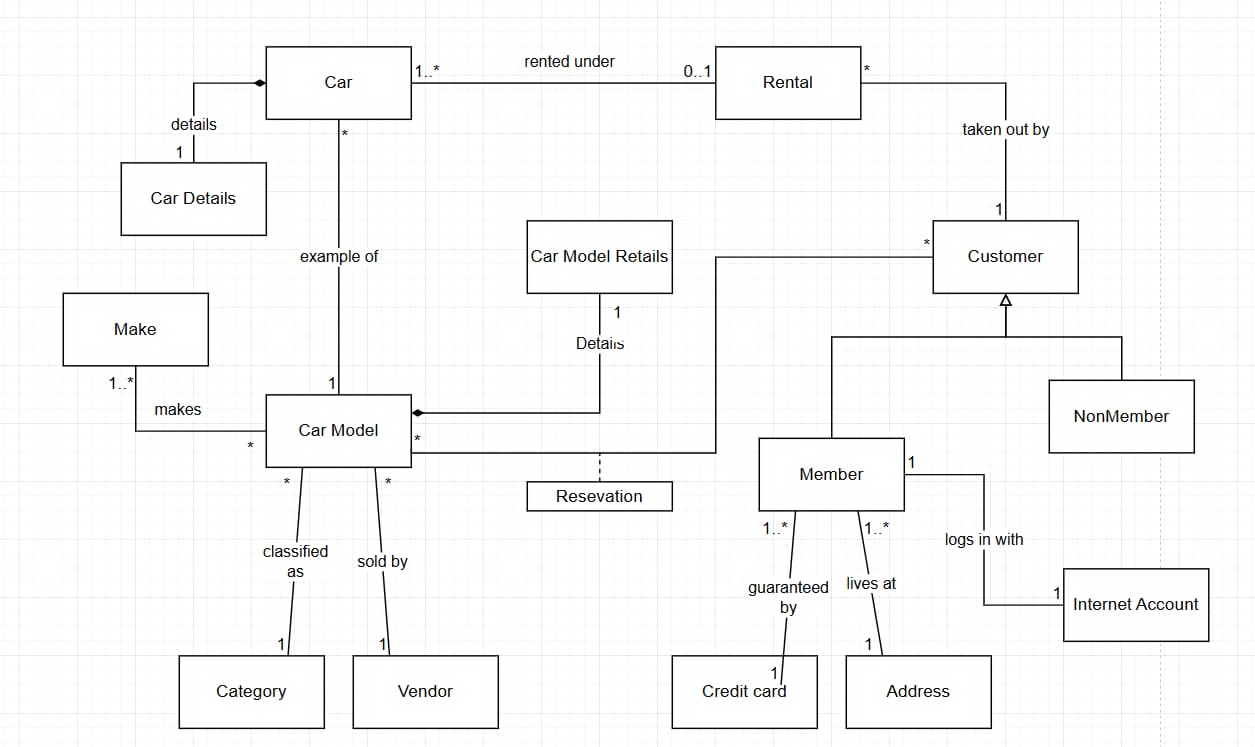
**Bài tập minh họa**

Bài 1.



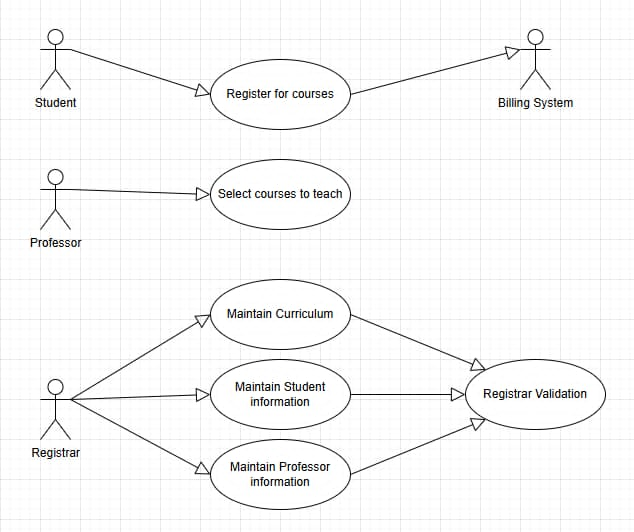
Ý nghĩa:

Bài 2.



Ý nghĩa:

Bài 3.



### a) Xác định và mô tả các tác nhân (Actors) trong sơ đồ

**Student (Sinh viên)** — người đăng ký học phần; tương tác chính với chức năng Register for Courses.

**Professor (Giảng viên)** — chọn môn/khóa để dạy; tương tác với Select Courses to Teach.

**Registrar (Phòng đào tạo / Cán bộ đăng ký)** — quản lý dữ liệu (curriculum, student info, professor info) và thực hiện các xác thực/duyệt.

**Billing System (Hệ thống thanh toán / Kế toán)** — hệ thống bên ngoài xử lý thanh toán/hoá đơn, liên kết với chức năng đăng ký học phần.

b) Liệt kê và giải thích các Use Cases có trên sơ đồ

**Register for Courses** — Sinh viên đăng ký các học phần/khoá học cho học kỳ.

**Select Courses to Teach** — Giảng viên chọn các khóa/môn họ sẽ giảng dạy.

**Maintain Curriculum** — Cán bộ quản lý chương trình/hệ thống môn học (thêm/sửa/xóa chương trình, môn).

**Maintain Student Information** — Cập nhật dữ liệu sinh viên (thông tin cá nhân, trạng thái học tập, học phí, …).

**Maintain Professor Information** — Cập nhật thông tin giảng viên (chi tiết liên hệ, bộ môn, quyền dạy).

**Registrar Validation** (Xác thực của phòng đăng ký) — chức năng kiểm tra/duyệt (ví dụ kiểm tra điều kiện, trạng thái học phí, giới hạn tín chỉ).

c) Phân tích các mối quan hệ giữa các use case (semantic)

Trong sơ đồ có biểu diễn mối quan hệ kiểu «uses»/«includes» giữa một vài use case với **Registrar Validation**. Giải thích ý nghĩa:

**Association (liên kết)**: actor ↔ use case (ví dụ Student — Register for Courses).

**<<include>> / <<uses>> (tái sử dụng hành vi chung)**: khi một use case luôn cần thực hiện một hành vi con chung, thì use case chính **include** hành vi con đó. Ở sơ đồ:

Maintain Student Information **uses/includes** Registrar Validation — tức việc thay đổi thông tin sinh viên phải qua bước xác thực/duyệt của phòng đăng ký.

Tương tự Maintain Curriculum và Maintain Professor Information đều **uses/includes** Registrar Validation

**<<extend>> (mở rộng điều kiện)**: không có use case dạng extend

Tóm lại: Registrar Validation là một use case tái sử dụng (include) cho nhiều use case quản trị nhằm đảm bảo mọi thay đổi đều phải được xác thực.

d) Viết kịch bản chi tiết cho một use case (**Register for Courses**)

|  |  |  |
| --- | --- | --- |
| Tên use case | Register for Courses | |
| Tác nhân chính | Student | |
| Bên liên quan | Billing System (thanh toán), Registrar (xác thực/duyệt), Hệ thống quản lý học phần. | |
| Mục tiêu | Sinh viên đăng ký thành công các học phần phù hợp cho học kỳ. | |
| Tiền điều kiện | Sinh viên đã đăng nhập hợp lệ vào hệ thống; Sinh viên đủ điều kiện để đăng ký (đang học, không bị cấm đăng ký). | |
| Luồng chính | Actor action | System Respone |
| 1. Sinh viên mở màn hình đăng ký học phần. |  |
|  | 2. Hệ thống hiển thị danh sách các học phần có sẵn (với mã, tên, số tín chỉ, giáo viên, thời khóa biểu, số chỗ còn lại, điều kiện tiền quyết). |
| 3. Sinh viên chọn một hoặc nhiều học phần muốn đăng ký. |  |
|  | 4. Hệ thống **tự động kiểm tra** xung đột thời khóa biểu, điều kiện tiền quyết, số tín chỉ tối đa, và yêu cầu học phí (gọi các use case/con-tác vụ: CheckPrerequisites, CheckScheduleConflict, CheckCreditLimit). **A1, A2** |
|  | 5. Hệ thống gửi yêu cầu xác thực đến **Registrar Validation** (include). |
|  | 6. Registrar validation phê duyệt (hoặc tự động phê duyệt nếu không cần can thiệp). **A3** |
|  | 7. Hệ thống chuyển sang **Billing System** để xử lý thanh toán/ghi nhận phí nếu học phần cần đóng phí. **A4** |
|  | 8. hệ thống ghi nhận đăng ký, cập nhật số chỗ còn lại, gửi thông báo xác nhận cho sinh viên (email/portal). **A5** |
|  | 9. Sinh viên đã được đăng ký học phần thành công. |
| Luồng thay thế | A1. Nếu có xung đột thời gian → hệ thống cảnh báo, đề xuất các học phần thay thế hoặc yêu cầu sinh viên bỏ chọn học phần xung đột.  A2. Nếu điều kiện tiền quyết không đạt → hệ thống từ chối đăng ký và hiển thị lý do; sinh viên có thể yêu cầu xét duyệt (đi vào luồng khác).  A3. Nếu Registrar từ chối → hiển thị lý do; có thể hướng dẫn làm khiếu nại/điều chỉnh.  A4. Nếu thanh toán thất bại → đơn đăng ký được giữ ở trạng thái chờ thanh toán; sinh viên có thể thử lại hoặc chọn hình thức khác.  A5. Nếu môn đã đầy trong lúc đăng ký → hệ thống có thể **extend** luồng bằng chức năng AddToWaitlist hoặc thông báo từ chối. | |
| Hậu điều kiện: | Nếu thành công: bản ghi đăng ký học phần tồn tại, sinh viên được tính tín chỉ, hóa đơn/ghi nhận thanh toán được tạo; Nếu thất bại: trạng thái tương ứng (bị từ chối, chờ thanh toán, trong danh sách chờ). | |

**Yêu cầu đặc biệt:**

Kiểm tra thời khóa biểu phải nhanh/real-time.

Bảo mật/ấn định quyền (chỉ sinh viên được đăng ký cho chính mình).

Giao tiếp an toàn với Billing System.

e) Đề xuất cải tiến hoặc bổ sung use cases cần thiết

**Use cases nên thêm:**

**Login / Authenticate** — xác thực người dùng (student/professor/registrar).

**View Academic Schedule / Timetable** — sinh viên/giảng viên xem thời khóa biểu cá nhân.

**Drop Course** — huỷ đăng ký một học phần.

**Check Prerequisites** — kiểm tra tự động điều kiện tiền quyết (nên **include** trong Register for Courses).

**Process Payment** — tách riêng chức năng thanh toán (Billing System là actor, Process Payment được include trong Register for Courses khi cần).

**Add to Waitlist** — nếu lớp đầy, thêm vào danh sách chờ (**extend** của Register for Courses).

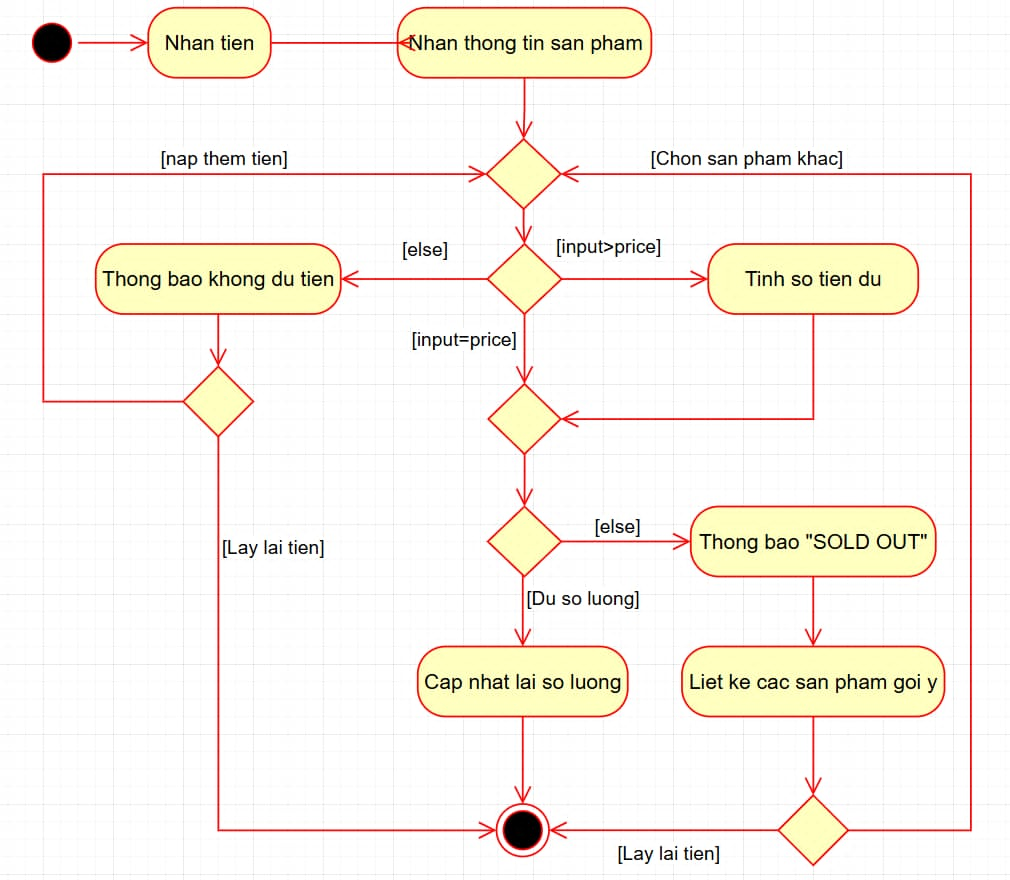
**Request Override / Special Approval** — khi sinh viên cần phép vượt tín chỉ hoặc đăng ký khi chưa đủ điều kiện (luồng mở rộng, liên hệ Registrar).

**Generate Transcript / Academic Report** — báo cáo học tập, điểm.

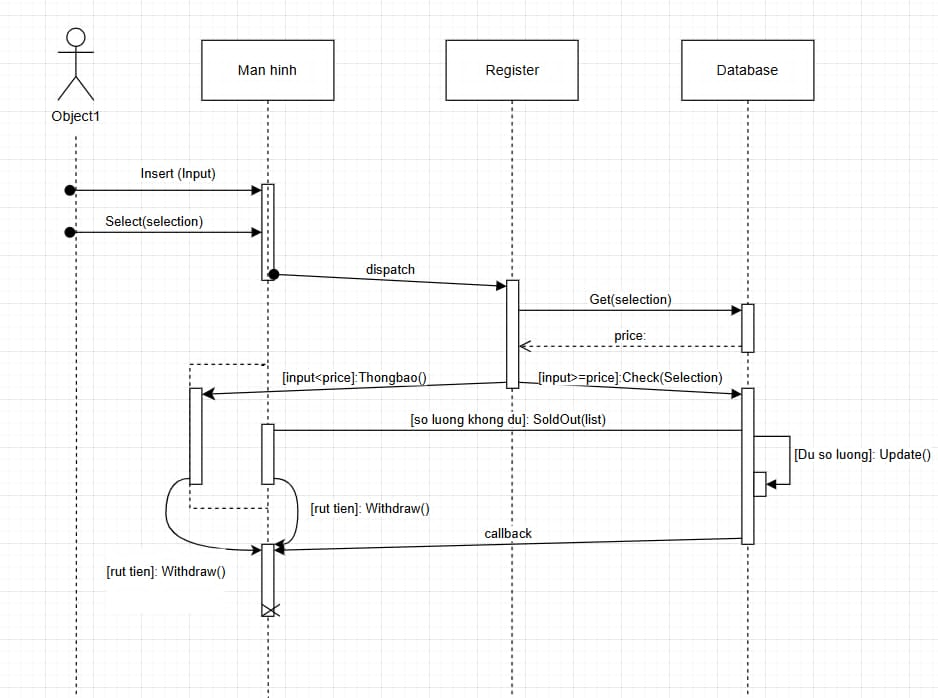
**Manage Rooms/Timeslot** — cho phòng đào tạo phân bố phòng/giờ (liên quan tới Maintain Curriculum).

**Notify / Announce** — gửi thông báo (thời hạn đăng ký, thay đổi lớp,…) — có thể là service chung.

Bài 4.



Ý nghĩa:



Ý nghĩa:

### Activity diagram — ý nghĩa

**Mô tả quy trình nghiệp vụ (flow)**: trình tự các bước của một giao dịch mua (ví dụ máy bán hàng): bắt đầu → nạp tiền → chọn sản phẩm → kiểm tra tiền đủ/không → kiểm tra tồn kho → cập nhật số lượng/hoạt động xuất sản phẩm → trả tiền thừa → kết thúc.

**Các điểm quyết định (decision nodes)**: kiểm tra tiền có đủ? và còn hàng? — hai nhánh chính dẫn tới luồng thành công hoặc luồng ngoại lệ.

**Luồng ngoại lệ/điều kiện thay thế**: khi **không đủ tiền** hệ thống báo và cho nạp thêm; khi **hết hàng** (SOLD OUT) hệ thống thông báo, gợi ý sản phẩm khác và trả tiền.

**Vòng lặp / lựa chọn lại**: cho phép người dùng nạp thêm tiền hoặc chọn sản phẩm khác trước khi tiếp tục.

**Mục đích**: minh họa chiến lược xử lý nghiệp vụ — ai làm gì, theo thứ tự nào, và các tình huống ngoại lệ cần xử lý.

### Sequence diagram — ý nghĩa

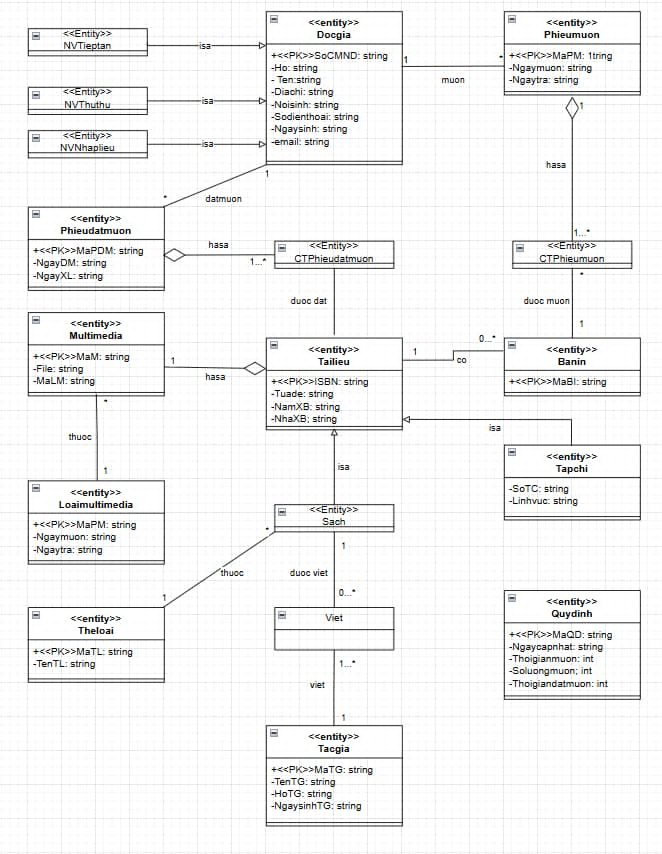
### ****Mô tả tương tác thời gian giữa các thành phần****: hiển thị các thông điệp (calls) theo thứ tự giữa ****Màn hình (UI)**** → ****Register (controller / business logic)**** → ****Database****.

**Trình tự cụ thể**: UI gửi Insert(input) và Select(selection) → Register gọi Get(selection) lên DB → DB trả price/quantity → Register so sánh input với price → nếu thiếu tiền: trả về thông báo; nếu đủ: kiểm tra tồn kho → nếu còn hàng: gọi Update() giảm số lượng và trả tiền thừa (Withdraw(change)); nếu hết hàng: trả SoldOut và yêu cầu hoàn tiền hoặc gợi ý.

**Cho thấy nơi xử lý logic**: kiểm tra giá, xác thực số lượng, cập nhật kho nằm ở Register/DB; UI chỉ hiển thị và nhận input.

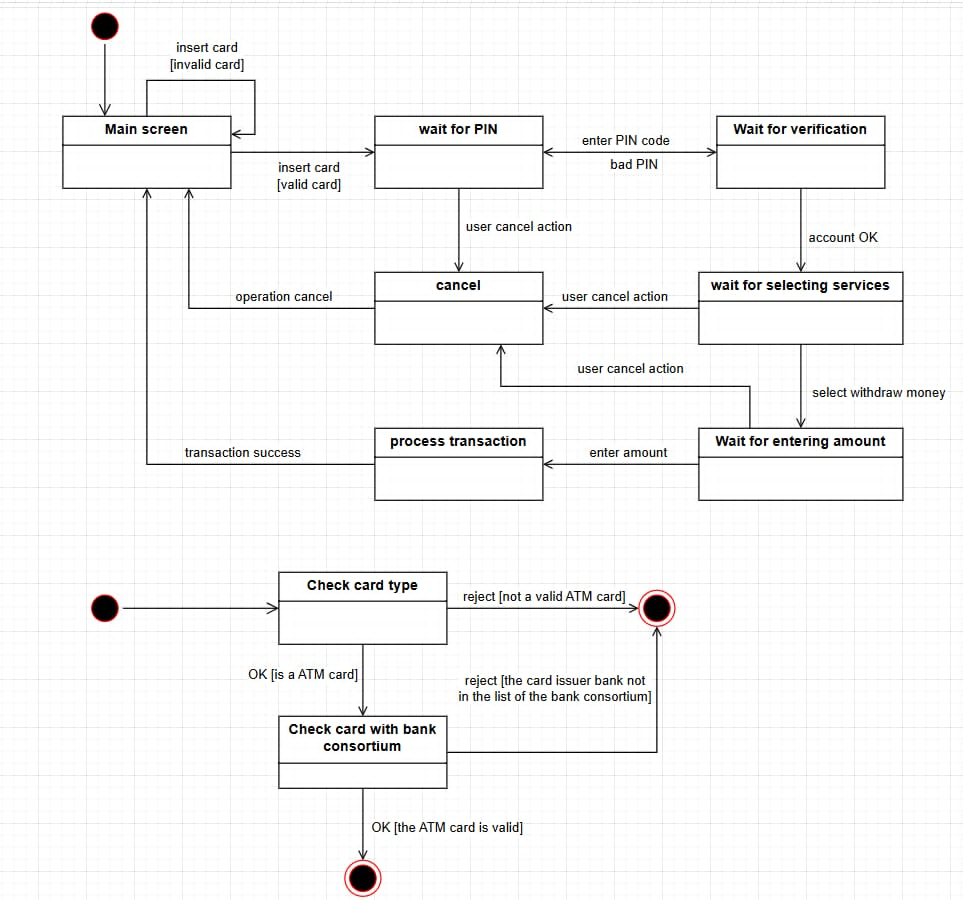
**Mục đích**: làm rõ ai gọi gì, khi nào, và theo thứ tự thời gian — rất quan trọng để cài đặt giao thức, gọi hàm, và xử lý lỗi/rollback.

Bài 5.



Ý nghĩa:

Bài 6.

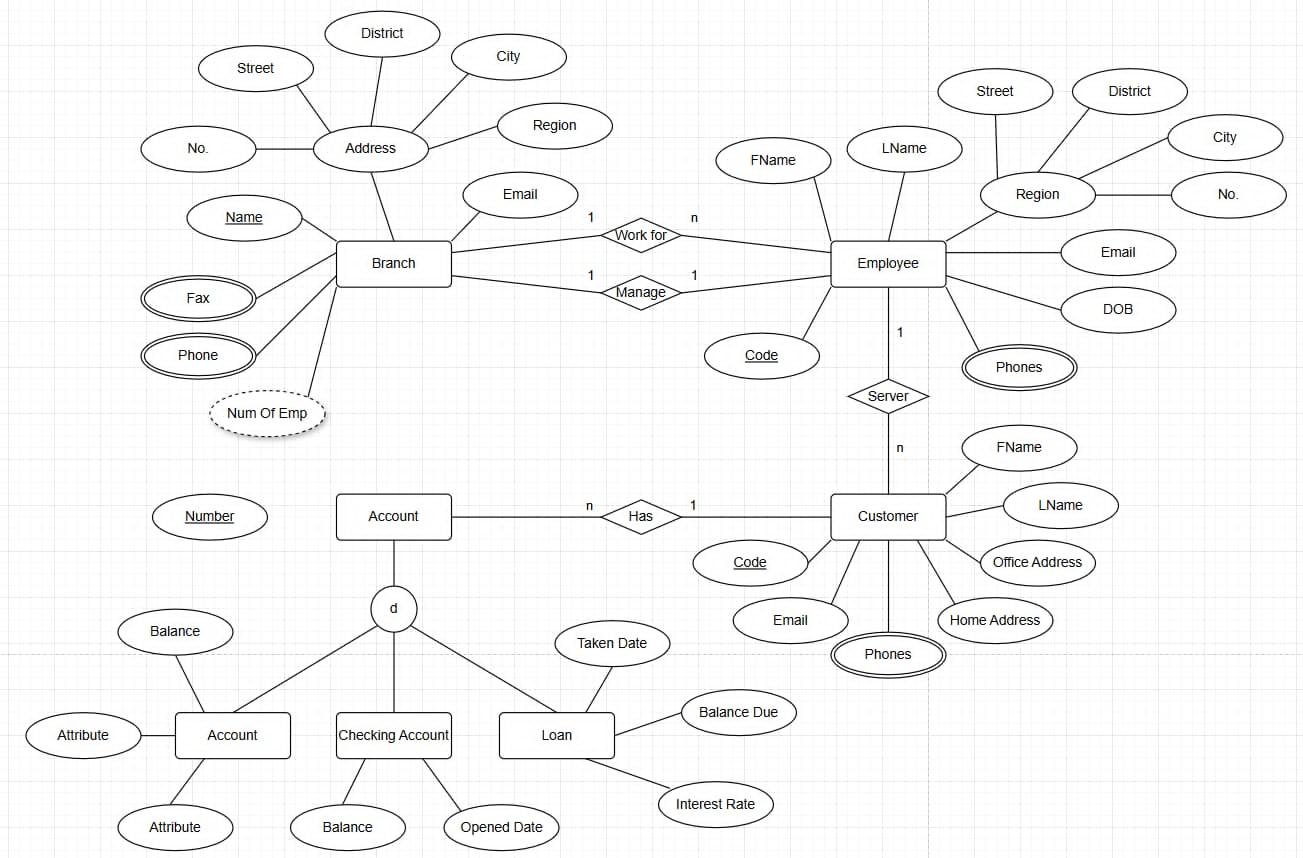


Ý nghĩa:

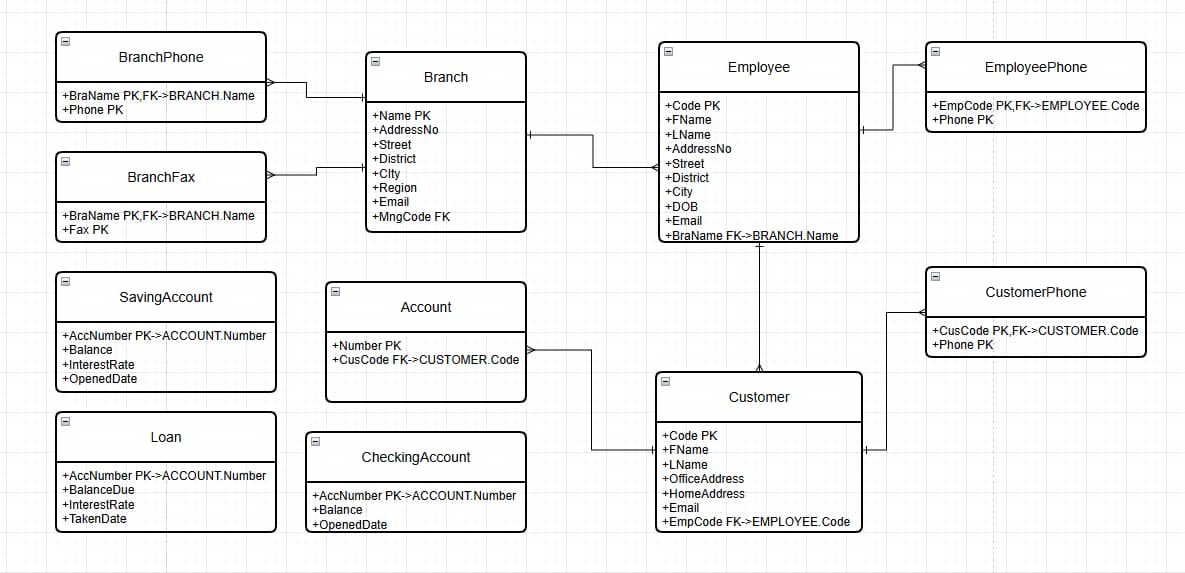
Bài 7.

1. Thiết kế Cơ sở dữ liệu ở mức Khái niệm, Luận lý và Vật lý

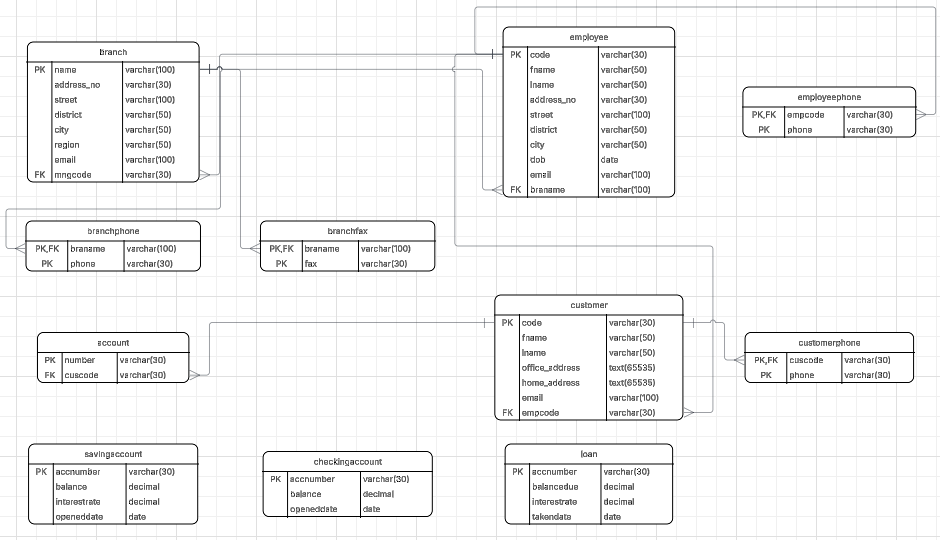
- Mức khái niệm:



- Mức luận lý:



- Mức vật lý:



1. Query: Insert, Update, Delete, Select

*a.*

-- BRANCH

INSERT INTO Branch(BranchID, Name, AddressNo, Street, District, City, Region, Email, NumEmployees, ManagerEmpCode)

VALUES

('B001','ABC Central','10','Hai Ba Trung','Dist1','HCMC','South','central@abc.com', 3, NULL),

('B002','ABC North','22','Nguyen Trai','Dist2','Hanoi','North','north@abc.com', 2, NULL),

('B003','ABC West','5','Le Loi','Dist3','Da Nang','Central','west@abc.com', 2, NULL),

('B004','ABC East','12','Tran Hung Dao','Dist4','Can Tho','South','east@abc.com', 1, NULL);

-- BRANCH\_PHONE

INSERT INTO BranchPhone(BranchPhoneID, BranchID, PhoneNumber) VALUES

(1,'B001','(028)1234567'),

(2,'B001','(028)7654321'),

(3,'B002','(024)1112222'),

(4,'B003','(0236)3334444');

-- BRANCH\_FAX

INSERT INTO BranchFax(BranchFaxID, BranchID, FaxNumber) VALUES

(1,'B001','(028)1234000'),

(2,'B002','(024)1112000'),

(3,'B003','(0236)3330000'),

(4,'B004','(0292)4440000');

-- EMPLOYEE - EmpCode là mã string

INSERT INTO Employee(EmpCode, FName, LName, DOB, Email, AddressNo, Street, District, City, Region, BranchID)

VALUES

('N1001','Alice','Smith','1985-07-10','alice.smith@abc.com','15','Le Loi','Dist3','Da Nang','Central','B003'),

('N1002','Bob','Lee','1978-11-05','bob.lee@abc.com','20','Nguyen Trai','Dist2','Hanoi','North','B002'),

('N1080','John','Doe','1982-03-15','john.doe@abc.com','3','Hai Ba Trung','Dist1','HCMC','South','B001'),

('N1004','Karen','Tran','1990-03-22','karen.tran@abc.com','8','Tran Hung Dao','Dist4','Can Tho','South','B004');

-- EMPLOYEE\_PHONE

INSERT INTO EmployeePhone(EmpPhoneID, EmpCode, PhoneNumber) VALUES

(1,'N1001','0908123456'),

(2,'N1002','0912345678'),

(3,'N1080','0987654321'),

(4,'N1004','0977123456');

-- CUSTOMER

INSERT INTO Customer(CusCode, FName, LName, HomeAddress, OfficeAddress, Email, EmpCode)

VALUES

('C001','Mai','Nguyen','HNo1 StA','Office A','mai.nguyen@x.com','N1001'),

('C002','Peter','Johnson','HNo2 StB','Office B','p.johnson@example.com','N1080'),

('C003','Tran','Vu','HNo3 StC','Office C','tran.vu@x.com','N1002'),

('C004','Linh','Pham','HNo4 StD','Office D','linh.pham@x.com','N1004');

-- CUSTOMER\_PHONE

INSERT INTO CustomerPhone(CusPhoneID, CusCode, PhoneNumber) VALUES

(1,'C001','0911000111'),

(2,'C002','0912000222'),

(3,'C003','0913000333'),

(4,'C004','0914000444');

-- ACCOUNT, AccountType có thể là 'SAVING','CHECKING','LOAN'

INSERT INTO Account(AccNumber, CusCode, AccountType, Balance, OpenedDate) VALUES

('A2001','C002','SAVING', 5000.00, '2019-06-01'), -- Peter Johnson saving

('A2002','C002','SAVING', 12000.50, '2020-01-15'), -- Peter Johnson another saving

('A3001','C001','CHECKING', 2500.00, '2018-09-10'),

('L4001','C003','LOAN', -15000.00, '2021-03-05');

-- SAVING\_ACCOUNT (matching AccNumber)

INSERT INTO SavingAccount(AccNumber, InterestRate, OpenedDate, Balance) VALUES

('A2001', 0.0150, '2019-06-01', 5000.00),

('A2002', 0.0200, '2020-01-15', 12000.50),

('A2003', 0.0180, '2017-04-20', 300.00), -- if extra saving account exists

('A2004', 0.0120, '2016-12-12', 1000.00);

-- CHECKING\_ACCOUNT

INSERT INTO CheckingAccount(AccNumber, OpenedDate, Balance) VALUES

('A3001','2018-09-10',2500.00),

('A3002','2019-05-05', 400.00),

('A3003','2017-07-07', 900.00),

('A3004','2020-11-11', 150.00);

-- LOAN

INSERT INTO Loan(AccNumber, BalanceDue, InterestRate, TakenDate) VALUES

('L4001',15000.00, 0.0750,'2021-03-05'),

('L4002',8000.00, 0.0900,'2019-08-10'),

('L4003',20000.00,0.0600,'2018-02-20'),

('L4004',5000.00, 0.1000,'2022-01-01');

*b.*

UPDATE Employee

SET LName = 'Brown'

WHERE EmpCode = 'N1080';

*c.*

DELETE FROM Employee

WHERE EmpCode = 'N1080';

*d.*

SELECT a.AccNumber

FROM Account a

JOIN Customer c ON a.CusCode = c.CusCode

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

*e.*

SELECT EmpCode, FName, LName, DOB

FROM Employee

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

*f.*

SELECT c.FName, c.LName, SUM(s.Balance) AS TotalSavingBalance

FROM Customer c

JOIN Account a ON c.CusCode = a.CusCode

JOIN SavingAccount s ON s.AccNumber = a.AccNumber

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

AND a.AccountType = 'SAVING'

GROUP BY c.CustCode, c.FName, c.LName;

BÀI TẬP ỨNG DỤNG

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

**Yêu cầu chức năng**

**Danh mục sản phẩm:**

**Người mua (Buyer):** Duyệt danh sách sản phẩm với các chức năng lọc và sắp xếp theo tên sản phẩm và giá. 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):** Quản lý sản phẩm; gán sản phẩm vào kho hàng.

**Giỏ hàng:**

**Người mua:** Thêm sản phẩm từ danh sách sản phẩm hoặc trang chi tiết sản phẩm.

Xem sản phẩm trên 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).

Giỏ hàng và bảng sẽ được cập nhật khi thêm hay xóa sản phẩm.

Nhấn nút Thanh toán trên trang giỏ hàng để thực hiện quy trình thanh toán.

**Quản trị viên**: Xem giỏ hàng bất kỳ người dùng và kích hoạt hoặc vô hiệu hóa giỏ hàng không hợp lệ của người dùng.

**Thanh toán:**

Hệ thống sẽ xác thực thông tin, xử lý thanh toán gửi email xác nhận cho người mua.

**Tồn kho:** Quản trị viên có thểquản lý kho hàng

**Đánh giá:**

Người mua có thể đánh giá bất kỳ sản phẩm nào (thang điểm 1 -> 5 sao)

**Kiểm soát truy cập:**

Người dùng (Người mua, Quản trị viên) có thể đăng nhập và đăng xuất.

Hệ thống điều hướng người dùng đến trang phù hợp sau khi đăng nhập (Người mua tới trang sản phẩm, Quản trị viên tới trang quản trị).

**Yêu cầu phi chức năng (Non-functional Requirements)**

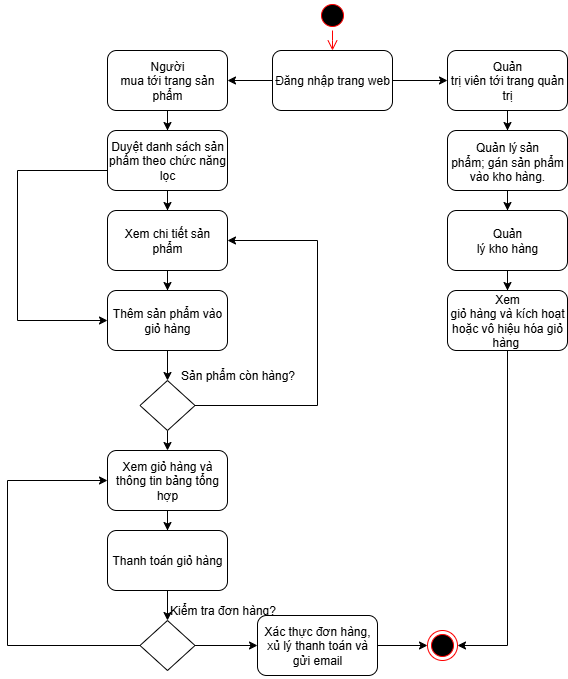
**Tính khả dụng (Usability):** Giao diện phải thân thiện, dễ sử dụng cho cả Người mua và Quản trị viên. Quy trình mua sắm phải trực quan và đơn giản.

**Hiệu năng (Performance):** Thời gian tải trang, đặc biệt là trang danh mục sản phẩm có chức năng lọc và sắp xếp, phải nhanh và chính xác. Giỏ hàng phải được cập nhật ngay lập tức.

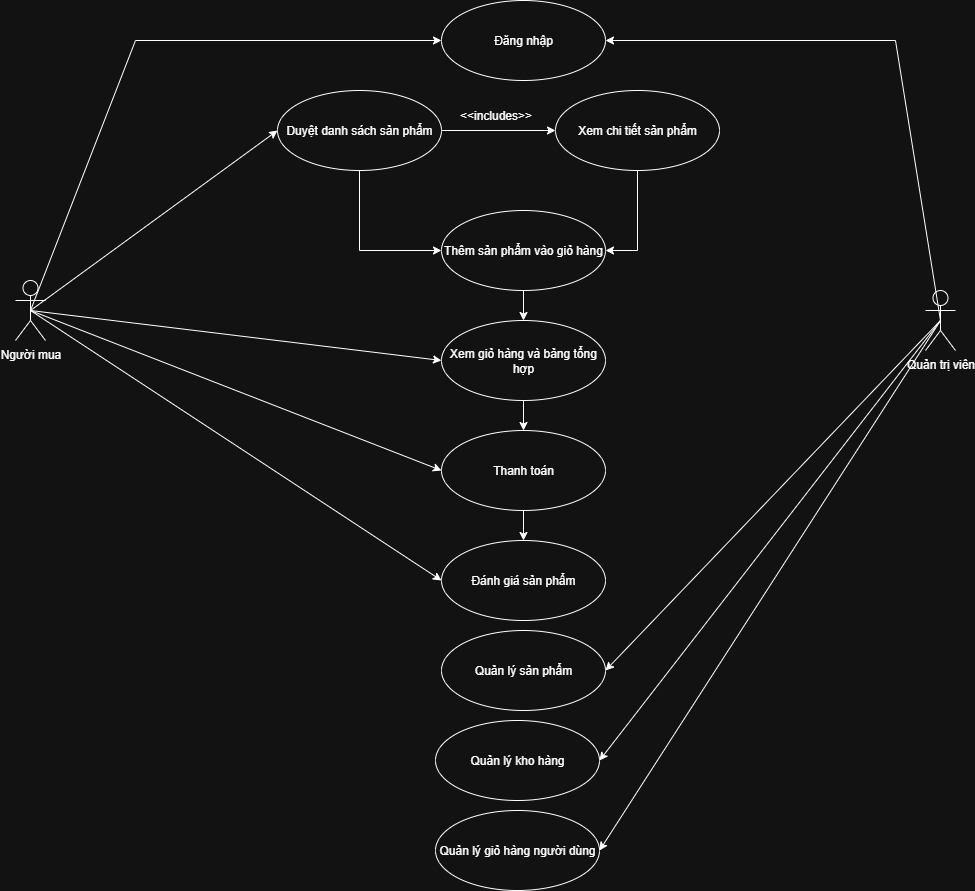
**Độ tin cậy (Reliability):** Hệ thống phải hoạt động ổn định, đặc biệt là quy trình thanh toán phải chính xác và không xảy ra lỗi. Dữ liệu tồn kho phải được cập nhật đồng bộ.

**Bảo mật (Security):** Hệ thống phải bảo mật thông tin tài khoản người dùng và thông tin thanh toán. Phải có sự phân quyền rõ ràng giữa Người mua và Quản trị viên.

2) Vẽ qui trình nghiệp vụ cho website CoolStore



3) Xây dựng use case diagram cho website CoolStore.

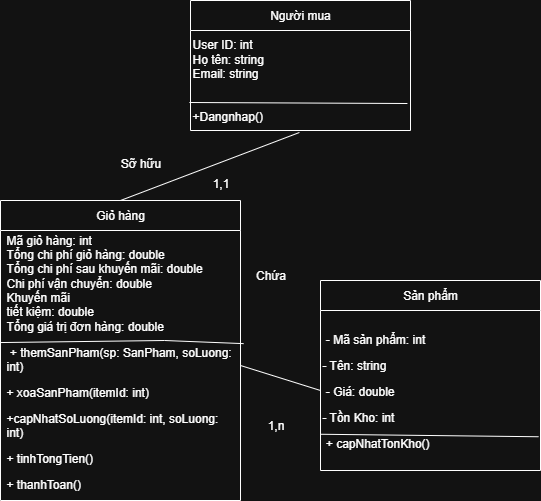


4) Mô tả chi tiết một Use Case "Mua sản phẩm từ trang chi tiết sản phẩm". Trình bày theo mẫu chuẩn gồm: Tên use case, Tác nhân chính, Mục tiêu, Tiền điều kiện, Luồng chính, Luồng thay thế, Hậu điều kiện, Ghi chú (nếu có).

|  |  |  |
| --- | --- | --- |
| Tên use case | Mua sản phẩm từ trang chi tiết sản phẩm | |
| Tác nhân chính | Người mua | |
| Mục tiêu | Cho phép người mua thêm một sản phẩm vào giỏ hàng từ trang hiển thị thông tin chi tiết của sản phẩm | |
| Tiền điều kiện | Sản phẩm còn hàng; Người mua đang ở trang duyệt sản phẩm | |
| Luồng chính | Actor action | System Respone |
| 1. Người mua mở trang chi tiết sản phẩm |  |
|  | 2. Hệ thống hiển thị thông tin sản phẩm (tên, mô tả, tồn kho, giá, đánh giá, trạng thái nổi bật). |
| 3. Người mua nhấn nút **“Mua hàng”**. |  |
|  | 4. Hệ thống kiểm tra tồn kho. **A1** |
|  | 5. Hệ thống thêm sản phẩm vào giỏ hàng. |
| Luồng thay thế | **A1** Nếu sản phẩm hết hàng, hệ thống hiển thị thông báo "Sản phẩm đã hết hàng" và không cho thêm vào giỏ. | |
| Hậu điều kiện: | Sản phẩm được thêm vào giỏ hàng thành công.  Giỏ hàng được cập nhật. | |

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.



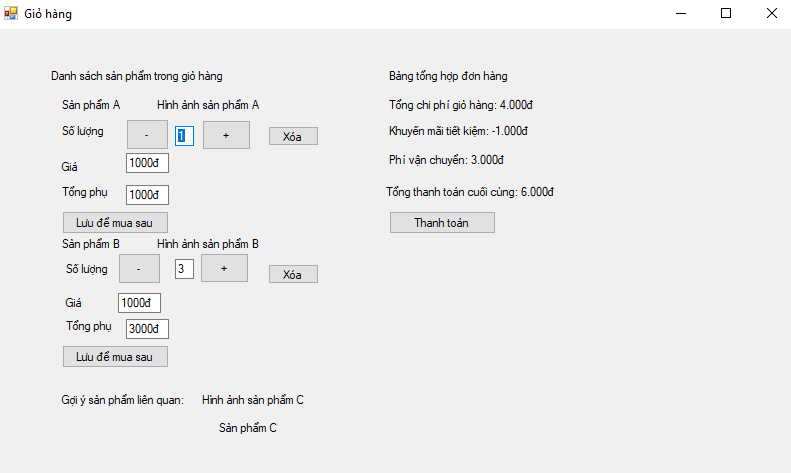
7) Thiết kế giao diện màn hình cho chức năng “Quản lý giỏ hàng” (có thể thiết kế một hoặc nhiều màn hình) gồm các thông tin:

• Danh sách sản phẩm trong giỏ hàng: tên sản phẩm, hình ảnh, số lượng, giá từng món, tổng phụ.Kiểm thử phần mềm Bài tập 2: Phân tích thiết kế hệ thống

• 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.



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

**Lớp View**

**Trách nhiệm chính**:

* Hiển thị giao diện người dùng (UI).
* Thu thập dữ liệu đầu vào từ người dùng (như click chuột, nhập liệu vào form).
* Gửi yêu cầu đến lớp Controller để xử lý.

**Các thành phần trong CoolStore**:

* **Trang danh mục sản phẩm**: Hiển thị danh sách sản phẩm cho người mua duyệt, lọc và sắp xếp.
* **Trang chi tiết sản phẩm**: Hiển thị đầy đủ thông tin về một sản phẩm cụ thể.
* **Giao diện giỏ hàng**: Liệt kê các sản phẩm đã được thêm vào và hiển thị bảng tổng hợp đơn hàng.
* **Các trang quản trị**: Giao diện cho phép Quản trị viên (SysAdmin) quản lý sản phẩm, kho hàng và giỏ hàng của người dùng.
* **Form đăng nhập/đăng xuất**.

**Lớp Controller**

**Trách nhiệm chính**:

* Nhận yêu cầu từ lớp View.
* Thực thi các quy tắc nghiệp vụ (ví dụ: kiểm tra tồn kho, tính toán khuyến mãi).
* Tương tác với lớp Database để đọc hoặc ghi dữ liệu.
* Trả kết quả xử lý về lại cho lớp View để hiển thị.

**Các thành phần trong CoolStore**:

* **Xử lý logic giỏ hàng**: Logic thêm sản phẩm, xóa sản phẩm, và tự động cập nhật lại tổng tiền khi giỏ hàng thay đổi.
* **Xử lý thanh toán**: Xác thực thông tin, xử lý giao dịch và gửi email xác nhận cho người mua.
* **Xử lý kiểm soát truy cập**: Xác thực thông tin đăng nhập và phân quyền cho người dùng (Người mua hay Quản trị viên).
* **Xử lý logic quản lý**: Các quy trình thêm/sửa/xóa sản phẩm, gán sản phẩm vào kho do Quản trị viên thực hiện.

**Lớp Database**

**Trách nhiệm chính**:

* Lưu trữ dữ liệu (ví dụ: thông tin sản phẩm, tài khoản người dùng).
* Cung cấp các phương thức để lớp Controller có thể truy vấn, thêm, sửa, xóa dữ liệu (CRUD).
* Đảm bảo tính toàn vẹn và bảo mật của dữ liệu.

**Các thành phần trong CoolStore**:

* **Cơ sở dữ liệu sản phẩm**: Lưu trữ thông tin chi tiết về sản phẩm như tên, mô tả, số lượng tồn kho, đánh giá.
* **Cơ sở dữ liệu người dùng**: Lưu thông tin tài khoản của Người mua và Quản trị viên.
* **Cơ sở dữ liệu giỏ hàng và đơn hàng**: Lưu trạng thái giỏ hàng của từng người dùng và lịch sử các đơn hàng đã thanh toán.
* **Cơ sở dữ liệu kho hàng**: Lưu thông tin về các kho và sản phẩm trong đó.

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.

|  |  |  |  |
| --- | --- | --- | --- |
| Mã TC | Dữ liệu đầu vào | Các bước thực hiện | Kết quả mong đợi |
| TC\_ADD\_01 | Người dùng đã đăng nhập; Sản phẩm "Laptop ABC" (tồn kho: 10); Giỏ hàng trống. | Truy cập trang sản phẩm "Laptop ABC" và nhấn "Thêm vào giỏ hàng". | Hiển thị thông báo thành công; Giỏ hàng cập nhật có 1 sản phẩm "Laptop ABC"; Tổng tiền được tính đúng. |
| TC\_ADD\_02 | Người dùng đã đăng nhập; Sản phẩm "Chuột XYZ" (tồn kho: 0). | Truy cập trang sản phẩm "Chuột XYZ" và nhấn "Thêm vào giỏ hàng". | Hiển thị thông báo lỗi "Sản phẩm đã hết hàng"; Giỏ hàng không thay đổi. |
| TC\_ADD\_03 | Người dùng chưa đăng nhập; Sản phẩm "Laptop ABC" (tồn kho: 10); Giỏ hàng đã có 1 "Laptop ABC". | Truy cập lại trang sản phẩm "Laptop ABC" và nhấn "Thêm vào giỏ hàng". | Hiển thị thông báo lỗi “Vui lòng đăng nhập”; Tự động di chuyển đến trang đăng nhập; Giỏ hàng không thay đổi |