# TRƯỜNG ĐẠI HỌC SÀI GÒN

**KHOA CÔNG NGHỆ THÔNG TIN**



BÀI BÁO CÁO BÀI TẬP LẦN 2 HỌC KỲ I

NĂM HỌC 2025-2026

Học phần: KIỂM THỬ PHẦN MỀM

**Phân tích thiết kế hệ thống**

**GVHD**  TS. Đỗ Như Tài

**Nhóm 7** Nguyễn Phương Vinh 3122411247

Lê Ngọc Hiệp 3122411053

Trần Hảo Điền 3122411042

Ngô Tuấn Anh 3122411007

Thành phố Hồ Chí Minh, Tháng 09/2025

# TRƯỜNG ĐẠI HỌC SÀI GÒN

**KHOA CÔNG NGHỆ THÔNG TIN**



BÀI BÁO CÁO BÀI TẬP LẦN 2 HỌC KỲ I

NĂM HỌC 2025-2026

Học phần: KIỂM THỬ PHẦN MỀM

**Phân tích thiết kế hệ thống**

**GVHD**  TS. Đỗ Như Tài

**Nhóm 7** Nguyễn Phương Vinh 3122411247

Lê Ngọc Hiệp 3122411053

Trần Hảo Điền 3122411042

Ngô Tuấn Anh 3122411007

Thành phố Hồ Chí Minh, Tháng 09/2025

Lời cam đoan

Tôi xin cam đoan đây là công trình báo cáo của cá nhân tôi và nhóm. Các nội dung nghiên cứu trong báo cáo bài tập thứ 2 (Phân tích thiết kế hệ thống) của tôi là trung thực. Nếu có phát hiện bất kỳ sự gian lận nào thì tôi xin hoàn toàn chịu trách nhiệm về bài báo cáo của tôi.

Lời cảm ơn

Lời cảm ơn chân thành đến các đọc giả đã dành thời gian để quan tâm đến bài báo của nhóm.

**BÁO CÁO HOẠT ĐỘNG CỦA NHÓM 7**

|  |  |  |  |
| --- | --- | --- | --- |
| MSSV | Họ và Tên | Nhiệm vụ được giao | Ghi chú |
| 3122411247 | Nguyễn Phương Vinh | - Bài 7, 8 |  |
| 3122411042 | Trần Hảo Điền | - Bài 1, 3, 7, 8 |  |
| 3122411007 | Ngô Tuấn Anh | - Bài 3, 5, 6, 8 |  |
| 3122411053 | Lê Ngọc Hiệp | - Bài 2, 4, 8 |  |

**Mục lục**

[CHƯƠNG I: PHÂN TÍCH THIẾT KẾ HỆ THỐNG 8](#_Toc209128657)

[1. BÀI TẬP MINH HỌA 8](#_Toc209128658)

[**1.1.** **Vẽ lại quy trình nghiệp vụ (Độc giả, NV Tiếp Tân, Thành viên)** 8](#_Toc209128659)

[**1.2.** **Vẽ mô hình khái niệm sau:** 9](#_Toc209128660)

[**1.3.** **Vẽ lại business use case sau:** 10](#_Toc209128661)

[**a)** **Xác định và mô tả các tác nhân (Actors) xuất hiện trong sơ đồ trên.** 10](#_Toc209128662)

[**b)** **Liệt kê và giải thích các trường hợp sử dụng (Use cases) được thể hiện trong sơ đồ.** 11](#_Toc209128663)

[**c)** **Phân tích các mối quan hệ giữa các use case (bao gồm cả mối quan hệ «uses» hoặc «include»).** 12](#_Toc209128664)

[**d)** **Viết kịch bản chi tiết cho một trường hợp sử dụng cụ thể (ví dụ: “Register for Courses” hoặc “Maintain Student Information”).** 12](#_Toc209128665)

[**e)** **Đề xuất cải tiến hoặc bổ sung thêm các use case mà bạn thấy cần thiết.** 18](#_Toc209128666)

[**1.4.** **Vẽ sơ đồ hoạt động và tương tác sau và tóm tắt ý nghĩa sơ đồ** 20](#_Toc209128667)

[**1.5.** **Vẽ lược đồ lớp cho bài toán Quản lý thư viện** 22](#_Toc209128668)

[**1.6.** **Vẽ lược đồ sau:** 23](#_Toc209128669)

[**1.7.** **Phân tích dữ liệu cho ABC Bank** 24](#_Toc209128670)

[**1.** **Thiết kế Cơ sở dữ liệu ở mức Khái niệm, Luận lý và Vật lý** 24](#_Toc209128671)

[**2.** **Query: Insert, Update, Delete, Select** 24](#_Toc209128672)

[2. BÀI TẬP ỨNG DỤNG 37](#_Toc209128673)

[**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.** 37](#_Toc209128674)

[**Yêu cầu chức năng (Functional Requirements)** 37](#_Toc209128675)

[a. Danh mục sản phẩm (Product Catalog) 37](#_Toc209128676)

[b. Giỏ hàng (Shopping Cart) 38](#_Toc209128677)

[c. Quy trình thanh toán (Payment Process) 39](#_Toc209128678)

[d. Tồn kho (Inventory) 39](#_Toc209128679)

[e. Đánh giá sản phẩm (Rating) 39](#_Toc209128680)

[f. Kiểm soát truy cập (Access Control) 40](#_Toc209128681)

[g. Thiết lập dữ liệu ban đầu 40](#_Toc209128682)

[**Yêu cầu phi chức năng (Non-Functional Requirements)** 40](#_Toc209128683)

[**2) Vẽ qui trình nghiệp vụ cho website CoolStore** 43](#_Toc209128684)

[**3) Xây dựng use case diagram cho website CoolStore.** 44](#_Toc209128685)

[**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).** 48](#_Toc209128686)

[**6) Thiết kế sơ đồ Class Diagram cho module Giỏ hàng.** 60](#_Toc209128687)

[**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:** 61](#_Toc209128688)

[**a) Danh sách sản phẩm trong giỏ hàng: tên sản phẩm, hình ảnh, số lượng, giá từng món, tổng phụ.** 61](#_Toc209128689)

[**b) Chức năng thao tác nhanh: Tăng/giảm số lượng sản phẩm; Xóa sản phẩm khỏi giỏ; Lưu sản phẩm để mùa sau.** 61](#_Toc209128690)

[**c) Bảng tổng hợp đơn hàng: Tổng chi phí giỏ hàng; Khuyến mãi tiết kiệm được; Phí vận chuyển; Tổng thanh toán cuối cùng.** 61](#_Toc209128691)

[**d) Hành động chính: Nút “Thanh toán”; Gợi ý sản phẩm liên quan để thêm vào giỏ.** 61](#_Toc209128692)

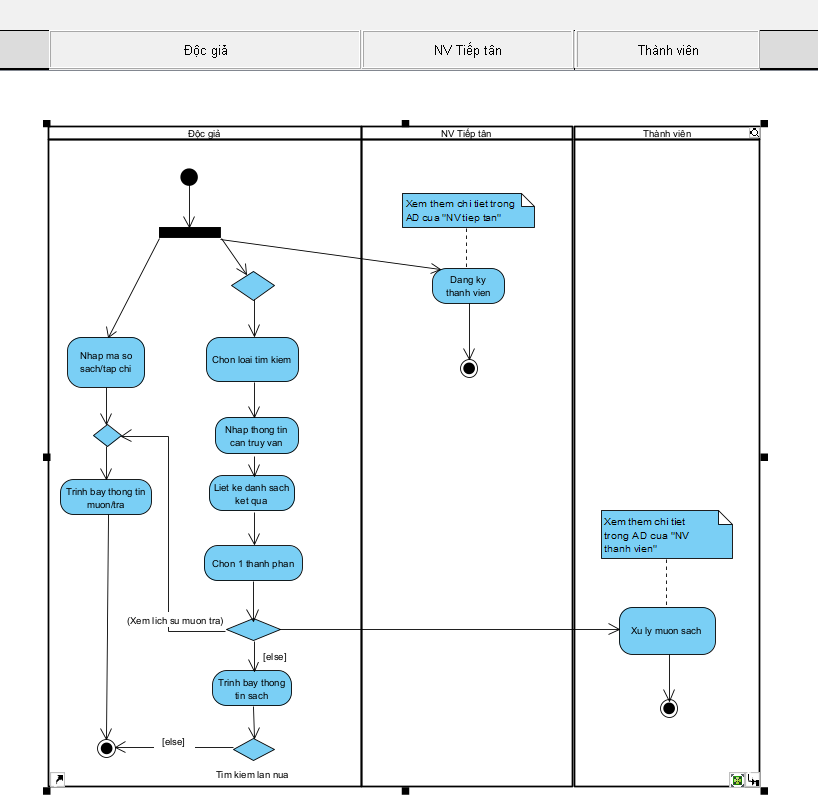
[**8) Đề xuất cách tổ chức kiến trúc phần mềm cho hệ thống CoolStore theo mô hình kiến trúc ba lớp (3-tier architecture).** 63](#_Toc209128693)

[**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.** 65](#_Toc209128694)

# CHƯƠNG I: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

## **BÀI TẬP MINH HỌA**

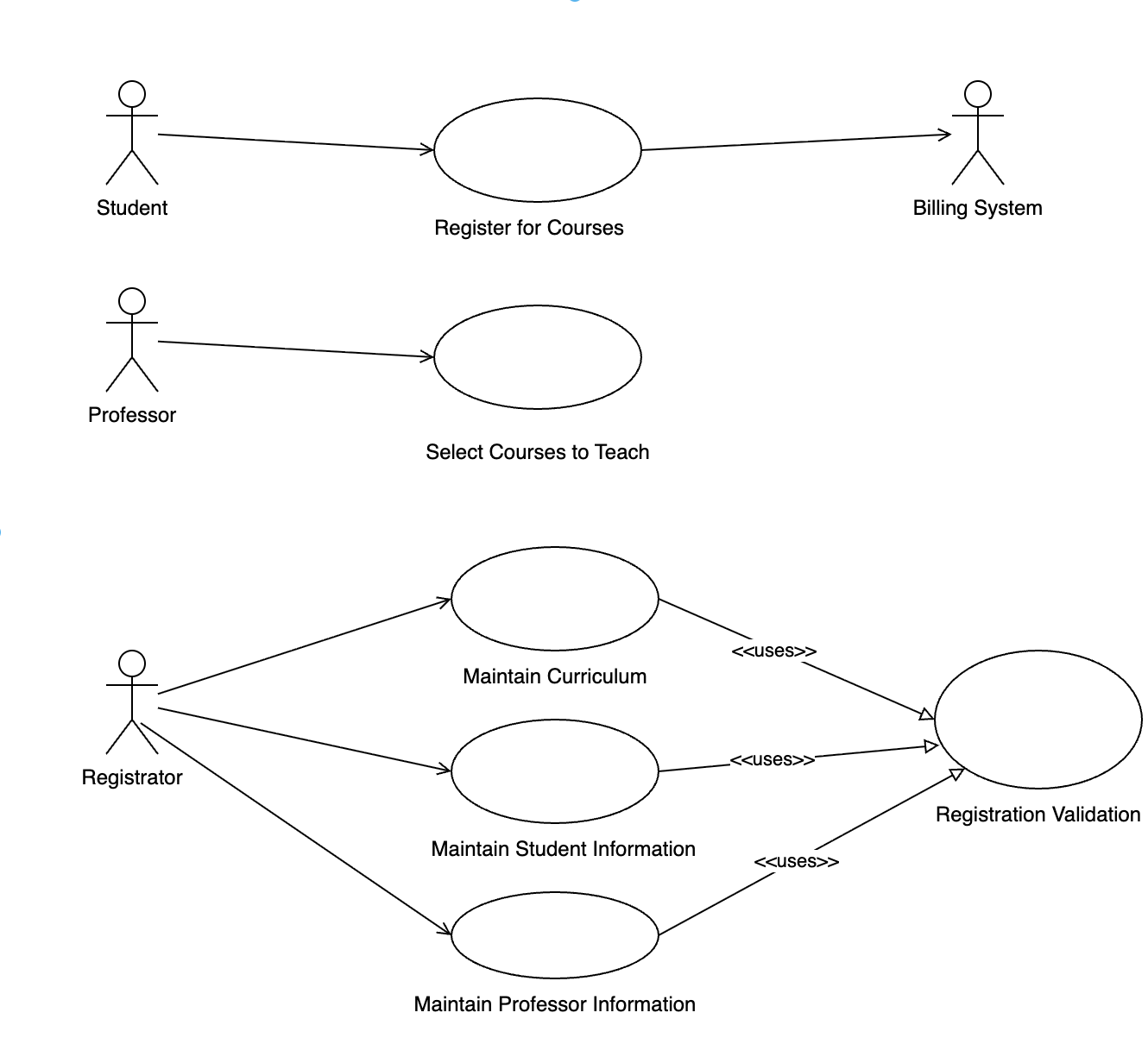
### **Vẽ lại quy trình nghiệp vụ (Độc giả, NV Tiếp Tân, Thành viên)**



### **Vẽ mô hình khái niệm sau:**



### **Vẽ lại business use case sau:**



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

Student (Sinh viên): Người dùng hệ thống để đăng ký các môn học.Có liên quan trực tiếp đến quá trình “Register for Courses”.

Professor (Giảng viên): Người dùng hệ thống để chọn các môn học mà mình sẽ giảng dạy.Có liên quan đến use case “Select Courses to Teach”.

Registrar (Phòng đào tạo/Quản trị viên): Chịu trách nhiệm cập nhật và duy trì dữ liệu trong hệ thống: chương trình học, thông tin sinh viên, thông tin giảng viên. Tham gia vào các use case: “Maintain Curriculum”, “Maintain Student Information”, “Maintain Professor Information”.

Billing System (Hệ thống thanh toán): Là một hệ thống bên ngoài, được tích hợp để xử lý học phí khi sinh viên đăng ký môn học. Liên kết với use case “Register for Courses”.

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

Register for Courses: Sinh viên đăng ký môn học mong muốn. Liên quan đến hệ thống thanh toán để kiểm tra và xử lý học phí.

Select Courses to Teach: Giảng viên chọn môn học mà mình sẽ phụ trách giảng dạy.

Maintain Curriculum: Phòng đào tạo (Registrar) quản lý và cập nhật chương trình đào tạo (môn học, nội dung môn học, kế hoạch giảng dạy).

Maintain Student Information: Quản lý và cập nhật hồ sơ sinh viên (thông tin cá nhân, kết quả học tập, tình trạng đăng ký).

Maintain Professor Information: Quản lý và cập nhật thông tin giảng viên (lý lịch, môn giảng dạy, phân công).

Registrar Validation (use case hỗ trợ): Thực hiện chức năng kiểm tra/đảm bảo tính hợp lệ cho các thao tác của Registrar (ví dụ: nhập dữ liệu đúng định dạng, kiểm tra trùng lặp).

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

Quan hệ «include» (uses):

“Maintain Curriculum” → <<uses>> “Registrar Validation”

“Maintain Student Information” → <<uses>> “Registrar Validation”

“Maintain Professor Information” → <<uses>> “Registrar Validation”

→ Nghĩa là mỗi khi Registrar thực hiện một thao tác quản lý dữ liệu, hệ thống luôn gọi thêm chức năng kiểm tra tính hợp lệ.

Quan hệ giao tiếp giữa Actor và Use Case:

Student <-> Register for Courses <-> Billing System

Professor <-> Select Courses to Teach

Registrar <-> Maintain Curriculum, Maintain Student Information, Maintain Professor Information

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

**Register for Courses**

|  |  |  |
| --- | --- | --- |
|  | | |
| **Use case number** | UC1 | |
| **Use case name** | Register for Courses | |
| **Actor** | Student | |
| **API bên thứ 3, các bên liên quan** | Billing System | |
| **Preconditions:** | 1. Student đã đăng nhập hợp lệ. 2. Kỳ học hiện tại đang ở trạng thái **open for registration**. 3. Hồ sơ sinh viên **không bị hold** (ví dụ: nợ học phí kỳ trước, bị đình chỉ, thiếu hồ sơ…). 4. Hồ sơ, thông tin của sinh viên đa được publish, không cần fill. 5. Dữ liệu Curriculum/Timetable, Course Catalog đã publish. | |
| **Postconditions:** | **Thành công:**   * Bản ghi đăng ký môn của Student được tạo; số lượng đã đăng ký của lớp tăng. * Hóa đơn/học phí cập nhật trong Billing System. * Student nhận xác nhận (email/app notification).   **Thất bại:**   * Không có thay đổi nào với đăng ký/môn; thông báo lỗi rõ nguyên nhân. | |
| **Trigger:** | Student chọn “Register for Courses” | |
| **Maturity** | Focused | |
| **Summary** | Học sinh đăng ký môn học trên hệ thống và hệ thống chuyển yêu cầu tính tiền đến đến hệ thống bên thứ 3 để tính học phí, mục đích để ghi nợ hoặc kiểm tra tình trạng học phí. | |
| **Basic Register for Courses**  **(luồng nghiệp vụ chính)** | **Actor Action** | **System Response** |
| 1. Student đăng nhập |  |
| 2. Student mở màn hình “Register for Courses” |  |
|  | 3. Hệ thống hiển thị danh mục học phần có thể đăng ký (lọc theo chương trình, học kỳ, học phần mở/đủ điều kiện). |
| 4. Student tìm kiếm học phần muốn đăng ký (có thể nhiều môn). |  |
|  | 4. Hệ thống sẽ phản hồi và trả về kết quả tìm kiếm đúng cho Student. **A1** |
| 5. Student sẽ chọn các học phần muốn đăng kí. |  |
|  | 6. Hệ thống sẽ gọi Registration Validation để kiểm tra **A1, A2, A3** (Kiểm tra điều kiện) |
|  | 7. Hệ thống sẽ tạo bảng lưu trữ tạm để giữ chỗ tạm thời cho các lớp học được đăng kí. **A1** |
|  | 8. Hệ thống gửi yêu cầu sang Billing System để tính phí. **E1** |
|  | 9. Billing System trả về tổng phí tạm tính và chi tiết khoản thu cho hệ thống. |
|  | 10. Hệ thống hiển thị **tóm tắt đăng ký**: danh sách môn, lịch học, tín chỉ, tổng phí lên monitor để yêu cầu Student xác nhận. |
| 11. Student theo dõi, kiểm tra thông tin từ phản hồi của hệ thống |  |
| 12. Student xác nhận đăng ký. **A1** |  |
|  | 13. Hệ thống commit |
|  | 13.1. Hệ thống sẽ tạo một bản ghi chính thức cho từng lớp. |
|  | 13.2. Hệ thống xóa đi bảng tạm sau khi commit |
|  | 13.3. Cập nhật sĩ số lớp |
|  | 13.4. Tạo/đồng bộ **invoice** với Billing System (trạng thái “Unpaid” hoặc “Paid” tùy luồng thanh toán). |
|  | 14. Hệ thống gửi thông tin xác nhận và kết quả đăng ký lên monitor. |
| Kết thúc | Kết thúc |
| **Alternative Path**  **(Luồng phụ) (Thay đổi hướng khác từ Action chính, không phải lỗi hệ thống).** | **4.A1. Không tìm thấy môn học đăng ký (Kết quả không tìm thấy hoặc sai).** | |
|  | 1. Hệ thống thông báo lên monitor không có môn học hợp lệ |
|  | 2. Hệ thống trả về bước 3 để Student tìm kiếm lại |
| **6. A1,A2,A3. Thiếu tiên quyết / ràng buộc học vụ** | |
|  | 1. Registration Validation trả về lỗi (E1. Nếu chưa đạt môn này thì không thể đăng ký môn kế tiếp. VD không đạt gt 1 thì không được đăng ký gt 2) hoặc ( E2. Trùng lịch) hoặc (E3. Hết chỗ). |
|  | 2. Hệ thống sẽ hiển thị lỗi theo từng môn mà Registration Validation trả về trên monitor. Không cho đăng ký môn |
| 3. Student theo dõi và chọn bỏ qua. |  |
|  | 4. Quay về bước 3 để Student đăng ký lại. |
| **7. A1. Reserve timeout (Thời gian giữ chỗ)** | |
| 1. Student không xác nhận trong khoảng thời gian giữ. |  |
|  | 2. Hệ thống tự động release chỗ, tức là giải phóng bảng tạm => mất chỗ đặt chỗ. |
|  | 3. Hệ thống thông báo hết thời gian giữ chỗ lớp học lên monitor. |
|  | 4. Quay về bước 3 để Student thao tác lại. |
| **12. A1. Student hủy xác nhận** | |
|  | **1. Hệ thống sẽ release mọi chỗ giữ trong bảng tạm, không làm thay đổi dữ liệu thực.** |
|  | **2. Quay về màn hình chính.** |
| **Exception Paths (Luồng ngoại lệ) (Lỗi hệ thống)** | **8. E1. Lỗi Billing System tạm thời**  - Hệ thống thứ 3 (Billing System) không phản hồi / time-out. | |
| **13. E1. Commit thất bại:**  1. Lỗi khi ghi dữ liệu vào DB.  2. Sau đó hệ thống sẽ rollback transaction toàn bộ đăng ký của Student.  3. Hệ thống hiển thị lỗi ra monitor.  4. Student quay về bước 3 để đăng ký lại môn. | |
| **NFR (Phi chức năng)** | * **Hiệu năng:** tìm lớp < 2 giây; commit đăng ký với n môn < 1–2 giây/tải thường. * **Tính nhất quán:** sử dụng transaction/lock hoặc cơ chế **reserve/commit/release** để tránh over-enroll. * **Khả dụng:** cao trong giai đoạn cao điểm đăng ký (scale-out, queue, cache). | |

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

**Đăng ký khóa học (Register for Courses):**

* Thêm Validate Course Availability: Hệ thống kiểm tra số lượng sinh viên đã đăng ký và số chỗ còn trống của khóa học trước khi cho phép đăng ký.
* Thêm Handle Course Conflicts: Hệ thống tự động kiểm tra và thông báo cho sinh viên nếu có xung đột về thời khóa biểu khi đăng ký nhiều khóa học.

**Chọn khóa học để dạy (Select Courses to Teach):**

* Thêm Verify Professor Qualification: Hệ thống kiểm tra bằng cấp và chuyên môn của giảng viên có phù hợp với khóa học được chọn hay không.
* Thêm Manage Teaching Load: Hệ thống tự động tính toán và cảnh báo khi số lượng khóa học mà giảng viên đăng ký vượt quá giới hạn quy định.

**Quản lý giáo trình (Maintain Curriculum):**

* Thêm Manage Curriculum Version: Hệ thống quản lý các phiên bản giáo trình khác nhau, cho phép truy xuất và so sánh.
* Thêm Link Course Materials: Liên kết trực tiếp tài liệu học tập, giáo trình điện tử với các khóa học cụ thể.

**Quản lý thông tin sinh viên (Maintain Student Information):**

* Thêm Update Student Academic Progress: Cập nhật tiến độ học tập, kết quả học tập của sinh viên theo từng học kỳ/năm học.
* Thêm Generate Student Reports: Tạo các báo cáo về tình hình học tập, chuyên cần, hay các thông tin khác của sinh viên.

**Quản lý thông tin giảng viên (Maintain Professor Information):**

* Thêm Manage Professor Leave/Absence: Quản lý lịch nghỉ phép, vắng mặt của giảng viên và tự động tìm kiếm người thay thế hoặc điều chỉnh lịch dạy.
* Thêm Track Professor Research & Publications: Theo dõi các công trình nghiên cứu và xuất bản của giảng viên để đánh giá hiệu suất.

Bổ sung các use case cần thiết khác

**Quản lý lịch học/lịch dạy (Manage Schedules):**

* Xem lịch học (View Student Schedule): Sinh viên có thể xem lịch học cá nhân của mình.
* Xem lịch dạy (View Professor Schedule): Giảng viên có thể xem lịch giảng dạy của mình.
* Cập nhật lịch (Update Schedule): Quản trị viên/Giáo vụ có thể thay đổi lịch học/dạy khi có sự cố hoặc yêu cầu.

**Quản lý điểm số (Manage Grades):**

* Nhập điểm (Enter Grades): Giảng viên có thể nhập điểm cho các môn học mình phụ trách.
* Xem điểm (View Grades): Sinh viên có thể xem điểm của mình.
* Phê duyệt điểm (Approve Grades): Quản trị viên/Giáo vụ có thể duyệt điểm trước khi công bố.

**Quản lý báo cáo và thống kê (Generate Reports & Statistics):**

* Tạo báo cáo học phí (Generate Tuition Report): Xuất báo cáo về tình hình đóng học phí của sinh viên.
* Tạo báo cáo kết quả học tập (Generate Academic Performance Report): Xuất báo cáo tổng hợp về kết quả học tập của toàn trường, khoa, hoặc từng khóa học.

**Quản lý tài chính (Financial Management):**

* Xem thông tin học phí (View Tuition Information): Sinh viên xem các khoản học phí cần đóng.
* Thanh toán học phí (Pay Tuition): Sinh viên thực hiện thanh toán học phí.
* Quản lý thu/chi (Manage Revenue/Expenses): Quản trị viên theo dõi các khoản thu/chi liên quan đến đào tạo.

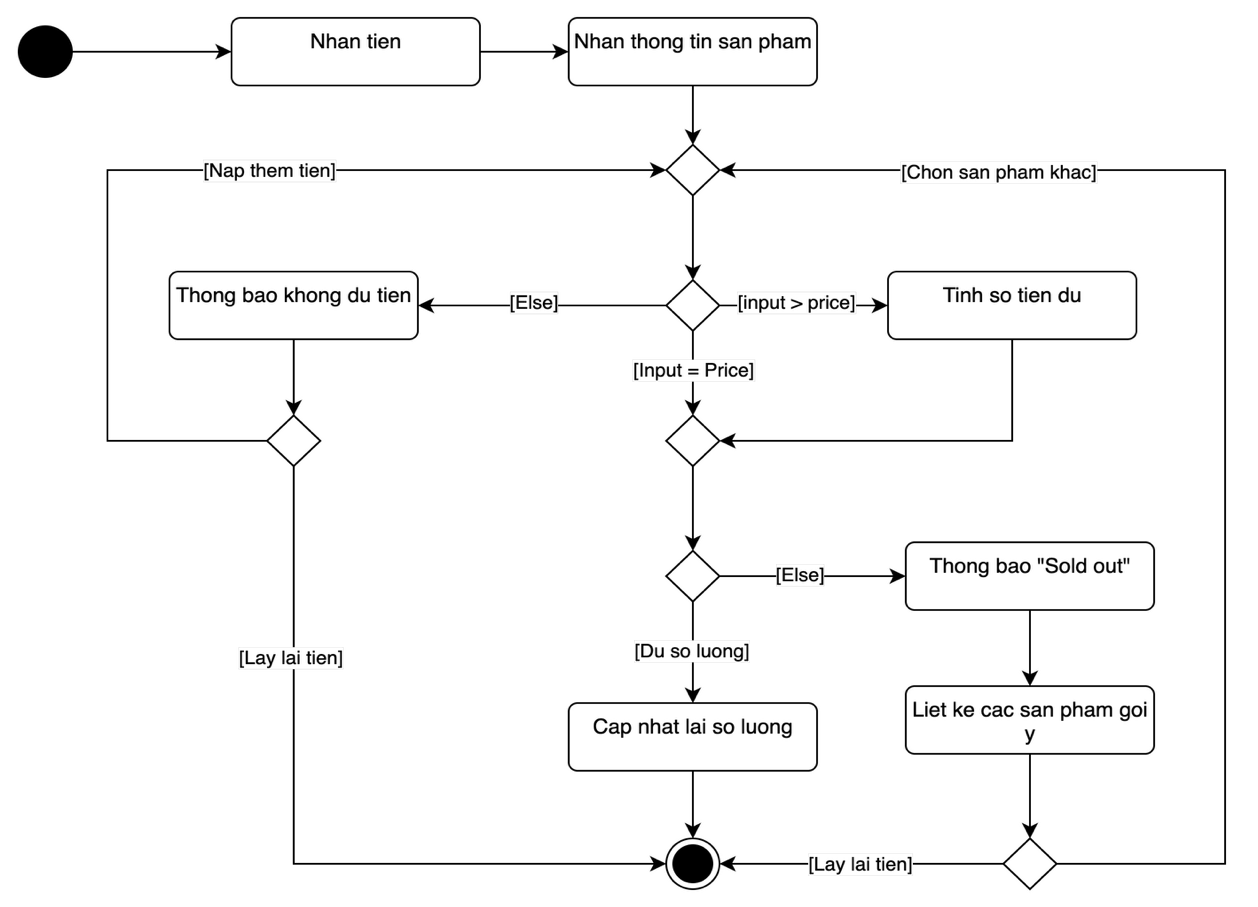
**Hỗ trợ và phản hồi (Support & Feedback):**

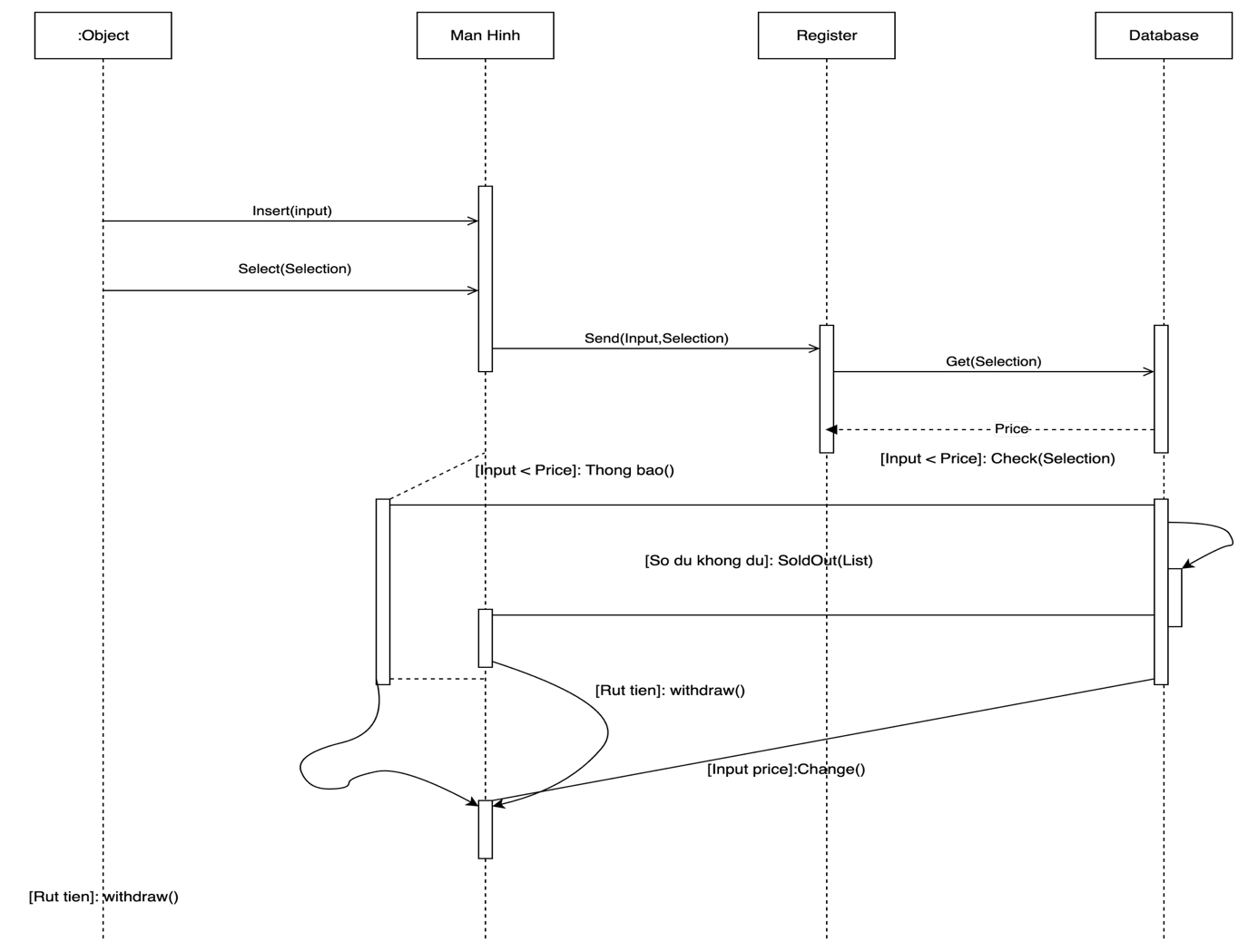
* Gửi phản hồi (Send Feedback): Sinh viên/Giảng viên gửi phản hồi về khóa học, giáo trình, hoặc dịch vụ.
* Quản lý phản hồi (Manage Feedback): Quản trị viên xử lý và phản hồi lại.

**Quản lý thông báo (Notification Management):**

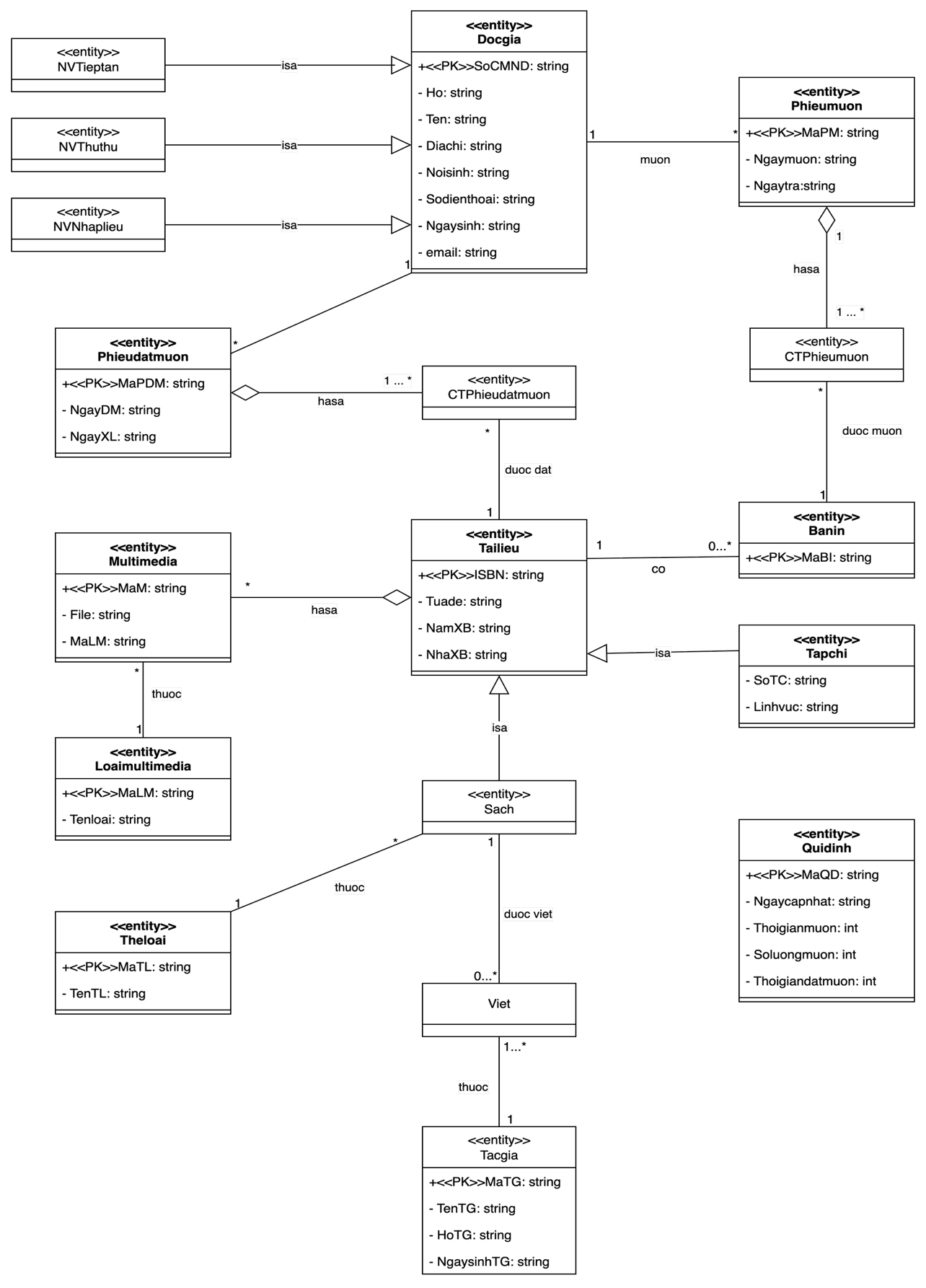
* Gửi thông báo (Send Notifications): Hệ thống gửi thông báo quan trọng đến sinh viên/giảng viên (lịch thi, lịch học thay đổi, thông báo mới).
* Xem thông báo (View Notifications): Người dùng có thể xem các thông báo từ hệ thống.

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



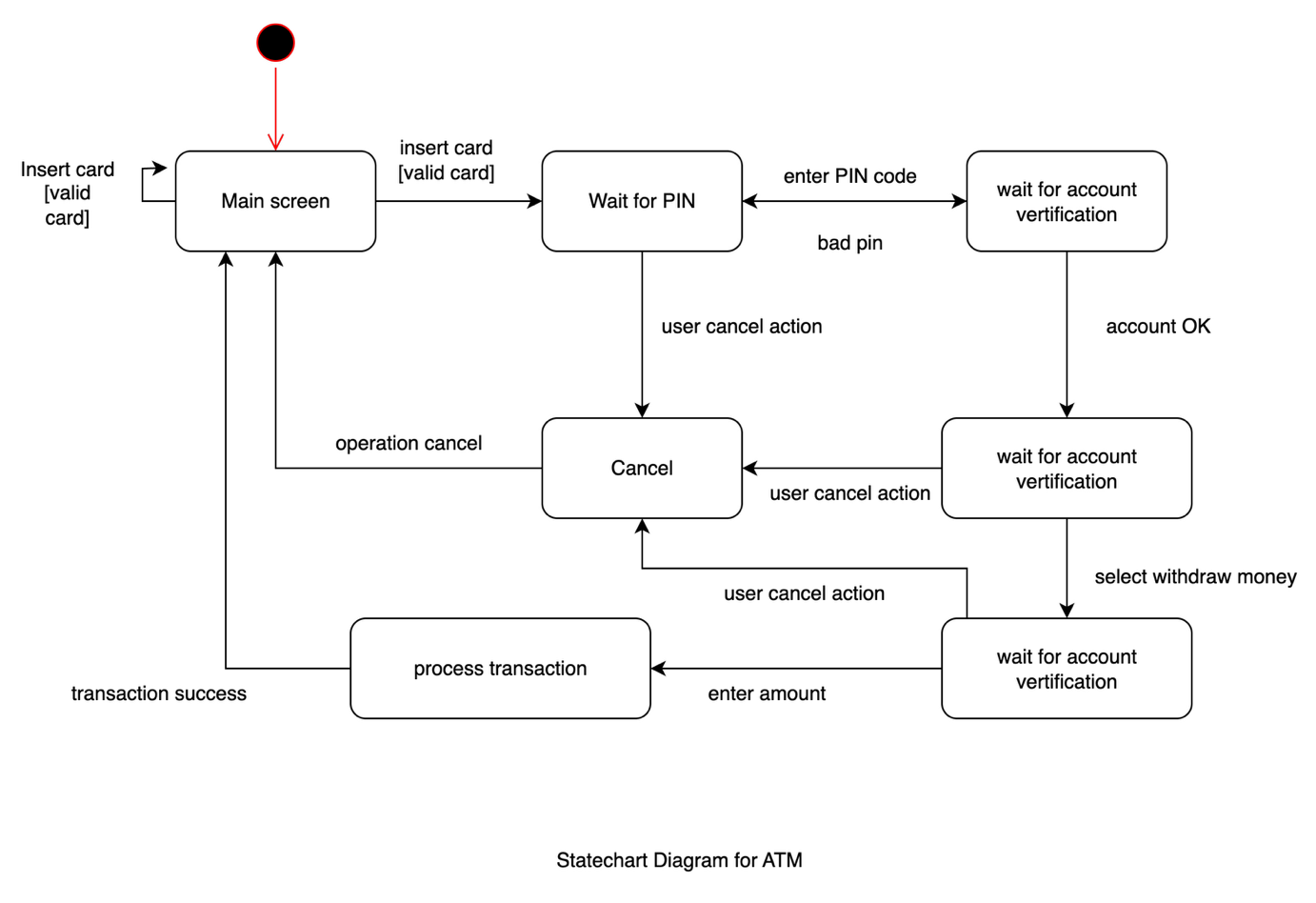


### **Vẽ lược đồ lớp cho bài toán Quản lý thư viện**

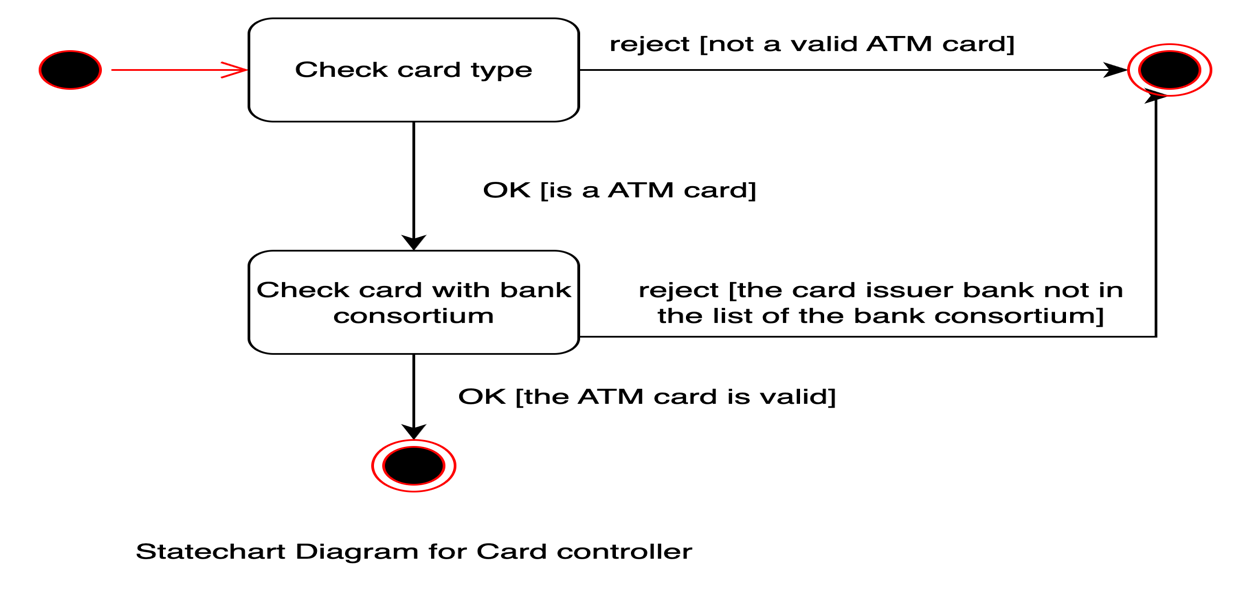


### **Vẽ lược đồ sau:**

Lược đồ 1:

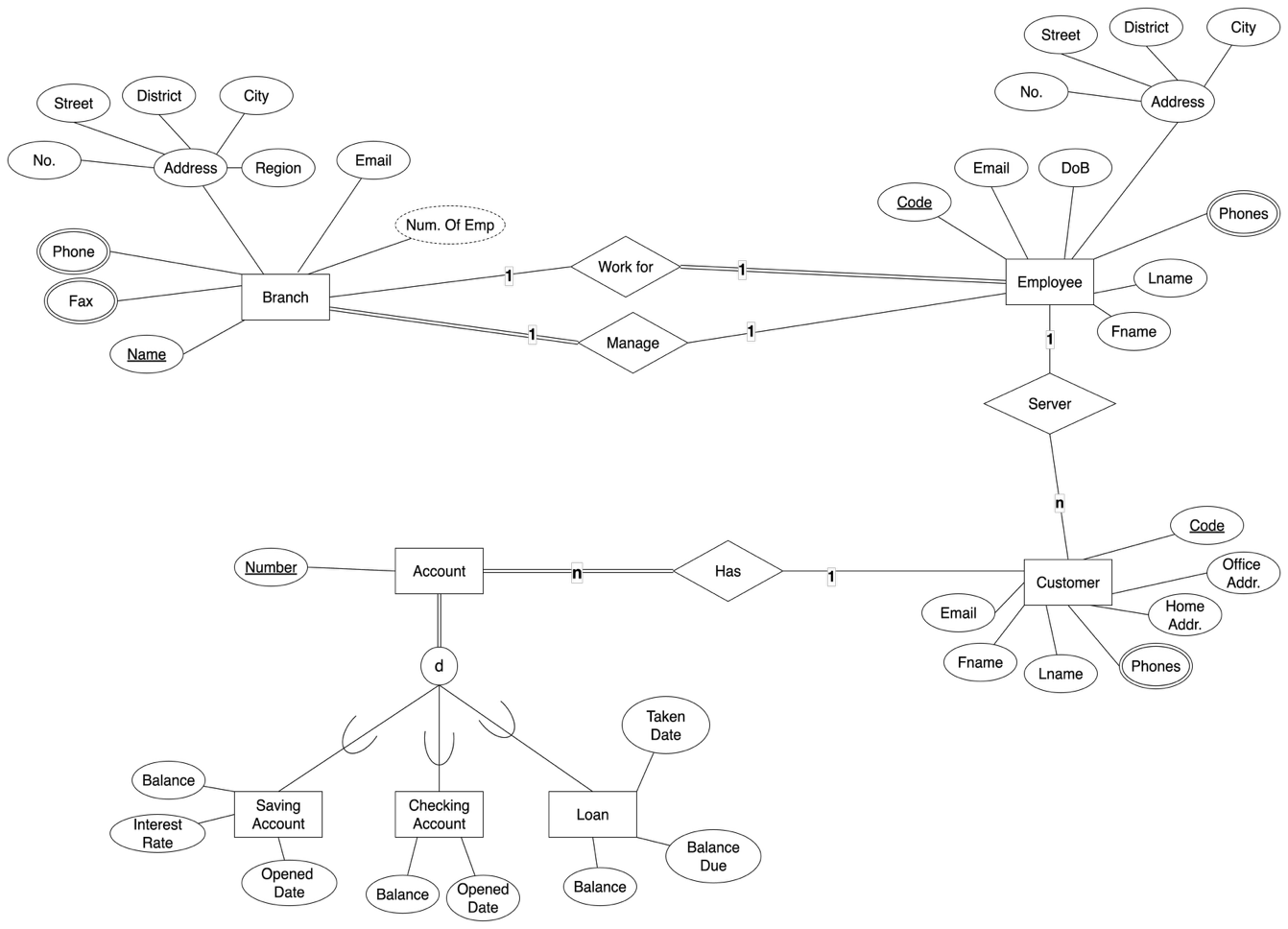


Lược đồ 2:



### **Phân tích dữ liệu cho ABC Bank**

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



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

**TẠO BẢNG CHO DATABASE**

-- Tạo Database

CREATE DATABASE ABCBank;

GO

USE ABCBank;

GO

-- Bảng chi nhánh

CREATE TABLE BRANCH (

Name VARCHAR(100) PRIMARY KEY,

AddressNo VARCHAR(20),

Street VARCHAR(100),

District VARCHAR(100),

City VARCHAR(100),

Region VARCHAR(100),

Email VARCHAR(100),

TotalEmployees INT DEFAULT 0, -- tổng số nhân viên

MngCode INT NULL -- khóa ngoại quản lý

);

-- Bảng điện thoại chi nhánh

CREATE TABLE BRANCHPHONE (

BraName VARCHAR(100),

Phone VARCHAR(20),

PRIMARY KEY (BraName, Phone),

FOREIGN KEY (BraName) REFERENCES BRANCH(Name)

);

-- Bảng fax chi nhánh

CREATE TABLE BRANCHFAX (

BraName VARCHAR(100),

Fax VARCHAR(20),

PRIMARY KEY (BraName, Fax),

FOREIGN KEY (BraName) REFERENCES BRANCH(Name)

);

-- Bảng nhân viên

CREATE TABLE EMPLOYEE (

Code INT PRIMARY KEY,

FName VARCHAR(50),

LName VARCHAR(50),

AddressNo VARCHAR(20),

Street VARCHAR(100),

District VARCHAR(100),

City VARCHAR(100),

DOB DATE,

Email VARCHAR(100),

BraName VARCHAR(100) NOT NULL,

FOREIGN KEY (BraName) REFERENCES BRANCH(Name)

);

-- Bảng điện thoại nhân viên

CREATE TABLE EMPLOYEEPHONE (

EmpCode INT,

Phone VARCHAR(20),

PRIMARY KEY (EmpCode, Phone),

FOREIGN KEY (EmpCode) REFERENCES EMPLOYEE(Code)

);

-- Bảng khách hàng

CREATE TABLE CUSTOMER (

Code INT PRIMARY KEY,

FName VARCHAR(50),

LName VARCHAR(50),

OfficeAddress VARCHAR(200),

HomeAddress VARCHAR(200),

Email VARCHAR(100),

EmpCode INT NOT NULL,

FOREIGN KEY (EmpCode) REFERENCES EMPLOYEE(Code)

);

-- Bảng điện thoại khách hàng

CREATE TABLE CUSTOMERPHONE (

CusCode INT,

Phone VARCHAR(20),

PRIMARY KEY (CusCode, Phone),

FOREIGN KEY (CusCode) REFERENCES CUSTOMER(Code)

);

-- Bảng tài khoản

CREATE TABLE ACCOUNT (

Number INT PRIMARY KEY,

CusCode INT NOT NULL,

FOREIGN KEY (CusCode) REFERENCES CUSTOMER(Code)

);

-- Bảng tài khoản tiết kiệm

CREATE TABLE SAVINGACCOUNT (

AccNumber INT PRIMARY KEY,

Balance DECIMAL(18,2),

InterestRate DECIMAL(5,2),

OpenedDate DATE,

FOREIGN KEY (AccNumber) REFERENCES ACCOUNT(Number)

);

-- Bảng tài khoản vãng lai

CREATE TABLE CHECKINGACCOUNT (

AccNumber INT PRIMARY KEY,

Balance DECIMAL(18,2),

OpenedDate DATE,

FOREIGN KEY (AccNumber) REFERENCES ACCOUNT(Number)

);

-- Bảng khoản vay

CREATE TABLE LOAN (

AccNumber INT PRIMARY KEY,

BalanceDue DECIMAL(18,2),

InterestRate DECIMAL(5,2),

TakenDate DATE,

FOREIGN KEY (AccNumber) REFERENCES ACCOUNT(Number)

);

-- Thêm ràng buộc quản lý chi nhánh

ALTER TABLE BRANCH

ADD CONSTRAINT FK\_Branch\_Manager FOREIGN KEY (MngCode) REFERENCES EMPLOYEE(Code);

CREATE TRIGGER trg\_UpdateBranchTotalEmployees

ON EMPLOYEE

AFTER INSERT, DELETE

AS

BEGIN

-- Khi thêm nhân viên

UPDATE BRANCH

SET TotalEmployees = (

SELECT COUNT(\*) FROM EMPLOYEE WHERE EMPLOYEE.BraName = BRANCH.Name

)

WHERE Name IN (

SELECT DISTINCT BraName FROM INSERTED

UNION

SELECT DISTINCT BraName FROM DELETED

);

END;

**CHÈN DỮ LIỆU VÀ TRUY VẤN**

USE ABCBank;

GO

/\*

a. You are required to insert valid and meaningful data

into the database.Each table has at least 4 rows

\*/

-- 1. Branches

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

VALUES

('Hanoi\_Main', '101', 'Kim Ma', 'Ba Dinh', 'Hanoi', 'North', 'hanoi\_main@abcbank.com', 0, NULL),

('HCM\_Center', '88', 'Le Loi', 'Quan 1', 'HCM', 'South', 'hcm\_center@abcbank.com', 0, NULL),

('Danang\_Office', '55', 'Nguyen Van Linh', 'Hai Chau', 'Da Nang', 'Central', 'danang\_office@abcbank.com', 0, NULL),

('Cantho\_Branch', '22', '30/4', 'Ninh Kieu', 'Can Tho', 'Mekong', 'cantho\_branch@abcbank.com', 0, NULL);

-- 2. BranchPhone Table

INSERT INTO BRANCHPHONE VALUES ('Hanoi\_Main', '024-1111111');

INSERT INTO BRANCHPHONE VALUES ('Hanoi\_Main', '024-2222222');

INSERT INTO BRANCHPHONE VALUES ('HCM\_Center', '028-3333333');

INSERT INTO BRANCHPHONE VALUES ('Danang\_Office', '0236-4444444');

INSERT INTO BRANCHPHONE VALUES ('Cantho\_Branch', '0292-5555555');

-- 3. BranchFax Table

INSERT INTO BRANCHFAX VALUES ('Hanoi\_Main', '024-9999999');

INSERT INTO BRANCHFAX VALUES ('HCM\_Center', '028-8888888');

INSERT INTO BRANCHFAX VALUES ('Danang\_Office', '0236-7777777');

INSERT INTO BRANCHFAX VALUES ('Cantho\_Branch', '0292-6666666');

-- 4. Employees

INSERT INTO EMPLOYEE (Code, FName, LName, AddressNo, Street, District, City, DOB, Email, BraName)

VALUES

(1, 'Nguyen', 'An', '12', 'Tran Phu', 'Ha Dong', 'Hanoi', '1985-04-12', 'an.nguyen@abcbank.com', 'Hanoi\_Main'),

(2, 'Tran', 'Binh', '45', 'Nguyen Trai', 'Thanh Xuan', 'Hanoi', '1990-08-22', 'binh.tran@abcbank.com', 'Hanoi\_Main'),

(3, 'Le', 'Hoa', '20', 'Pasteur', 'Quan 3', 'HCM', '1987-11-05', 'hoa.le@abcbank.com', 'HCM\_Center'),

(4, 'Pham', 'Cuong', '55', 'Vo Thi Sau', 'Quan 1', 'HCM', '1992-02-15', 'cuong.pham@abcbank.com', 'HCM\_Center'),

(5, 'Do', 'Thang', '78', 'Le Duan', 'Hai Chau', 'Da Nang', '1980-03-18', 'thang.do@abcbank.com', 'Danang\_Office'),

(6, 'Nguyen', 'Huong', '33', 'Hoang Dieu', 'Hai Chau', 'Da Nang', '1995-12-07', 'huong.nguyen@abcbank.com', 'Danang\_Office'),

(7, 'Vo', 'Khanh', '9', 'Nguyen Trai', 'Ninh Kieu', 'Can Tho', '1983-06-20', 'khanh.vo@abcbank.com', 'Cantho\_Branch'),

(8, 'Pham', 'Ly', '100', 'Ly Tu Trong', 'Ninh Kieu', 'Can Tho', '1991-09-10', 'ly.pham@abcbank.com', 'Cantho\_Branch'),

(9, 'Nguyen', 'Vinh', '102', 'Hoa Hung', 'Quan 10', 'HCM', '2010-03-10', 'ly.pham@abcbank.com', 'Cantho\_Branch');

-- 5. EmployeePhone Table

INSERT INTO EMPLOYEEPHONE VALUES (1, '0912000001');

INSERT INTO EMPLOYEEPHONE VALUES (2, '0912000002');

INSERT INTO EMPLOYEEPHONE VALUES (3, '0933000003');

INSERT INTO EMPLOYEEPHONE VALUES (4, '0933000004');

INSERT INTO EMPLOYEEPHONE VALUES (5, '0905000005');

INSERT INTO EMPLOYEEPHONE VALUES (6, '0905000006');

INSERT INTO EMPLOYEEPHONE VALUES (7, '0917000007');

INSERT INTO EMPLOYEEPHONE VALUES (8, '0917000008');

-- 6. Customer Table

INSERT INTO CUSTOMER (Code, FName, LName, OfficeAddress, HomeAddress, Email, EmpCode)

VALUES

(1001, 'Hoang', 'Minh', 'FPT Tower, Hanoi', '123 Hoang Quoc Viet, Hanoi', 'minh.hoang@gmail.com', 1),

(1002, 'Vu', 'Lan', 'Viettel HQ, Hanoi', '45 Nguyen Chi Thanh, Hanoi', 'lan.vu@yahoo.com', 2),

(1003, 'Nguyen', 'Phuong', 'Vincom, HCM', '12 Nguyen Hue, HCM', 'phuong.nguyen@gmail.com', 3),

(1004, 'Tran', 'Son', 'Techcombank Tower, HCM', '88 Cach Mang Thang 8, HCM', 'son.tran@outlook.com', 4),

(1005, 'Le', 'Trang', 'Da Nang Software Park', '99 Hoang Dieu, Da Nang', 'trang.le@gmail.com', 5),

(1006, 'Bui', 'Hai', 'Mobifone Da Nang', '12 Ong Ich Khiem, Da Nang', 'hai.bui@yahoo.com', 6),

(1007, 'Pham', 'Quyen', 'Can Tho University', '30/4 Street, Can Tho', 'quyen.pham@gmail.com', 7),

(1008, 'Nguyen', 'Duc', 'VNPT Can Tho', '77 Hung Vuong, Can Tho', 'duc.nguyen@outlook.com', 8);

-- 7. CustomerPhone Table

INSERT INTO CUSTOMERPHONE VALUES (1001, '0987000001');

INSERT INTO CUSTOMERPHONE VALUES (1002, '0987000002');

INSERT INTO CUSTOMERPHONE VALUES (1003, '0978000003');

INSERT INTO CUSTOMERPHONE VALUES (1004, '0978000004');

INSERT INTO CUSTOMERPHONE VALUES (1005, '0969000005');

INSERT INTO CUSTOMERPHONE VALUES (1006, '0969000006');

INSERT INTO CUSTOMERPHONE VALUES (1007, '0958000007');

INSERT INTO CUSTOMERPHONE VALUES (1008, '0958000008');

-- 8. Account Table

INSERT INTO ACCOUNT VALUES (20001, 1001);

INSERT INTO ACCOUNT VALUES (20002, 1002);

INSERT INTO ACCOUNT VALUES (20003, 1003);

INSERT INTO ACCOUNT VALUES (20004, 1004);

INSERT INTO ACCOUNT VALUES (20005, 1005);

INSERT INTO ACCOUNT VALUES (20006, 1006);

INSERT INTO ACCOUNT VALUES (20007, 1007);

INSERT INTO ACCOUNT VALUES (20008, 1008);

INSERT INTO ACCOUNT VALUES (20009, 1001);

INSERT INTO ACCOUNT VALUES (20010, 1003);

-- 9. SavingAccount Table

INSERT INTO SAVINGACCOUNT VALUES (20001, 50000000, 3.5, '2022-01-15');

INSERT INTO SAVINGACCOUNT VALUES (20003, 120000000, 4.0, '2021-11-01');

INSERT INTO SAVINGACCOUNT VALUES (20005, 70000000, 3.2, '2022-05-22');

INSERT INTO SAVINGACCOUNT VALUES (20007, 30000000, 3.8, '2023-03-10');

INSERT INTO SAVINGACCOUNT VALUES (20009, 36360000, 3.6, '2023-06-03');

-- 10. Checking accounts Table

INSERT INTO CHECKINGACCOUNT VALUES (20002, 15000000, '2023-05-10');

INSERT INTO CHECKINGACCOUNT VALUES (20006, 10000000, '2024-01-15');

INSERT INTO CHECKINGACCOUNT VALUES (20009, 8000000, '2024-02-20');

INSERT INTO CHECKINGACCOUNT VALUES (20005, 3636000, '2024-06-03');

-- 11. Loans Table

INSERT INTO LOAN VALUES (20004, 200000000, 8.5, '2023-06-01');

INSERT INTO LOAN VALUES (20008, 100000000, 7.2, '2022-09-15');

INSERT INTO LOAN VALUES (20010, 50000000, 6.8, '2024-04-01');

INSERT INTO LOAN VALUES (20002, 69690000, 6.9, '2024-09-06');

-- 12. Cập nhật Manager cho Branch

------------------------

UPDATE BRANCH SET MngCode = 1 WHERE Name = 'Hanoi\_Main';

UPDATE BRANCH SET MngCode = 3 WHERE Name = 'HCM\_Center';

UPDATE BRANCH SET MngCode = 5 WHERE Name = 'Danang\_Office';

UPDATE BRANCH SET MngCode = 7 WHERE Name = 'Cantho\_Branch';

-------------------------------------------------------------------------------

-- b. Change last name of an employee whose code is “8” to “Brown”.

UPDATE EMPLOYEE SET LName = 'Brown' WHERE Code = 8

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

DELETE EMPLOYEE WHERE Code = 8

/\*

Vì có ràng buộc toàn vẹn, liên kết khoá ngoại thuộc tính ECode trong bảng EMPPhone và bảng Customer nên không thể xoá được

\*/

-- d. Retrieve all account numbers of a customer whose name is Hoang Minh.

SELECT A.Number as AccountNumber, C.Code, C.FName + ' ' + C.LName as FullName

FROM CUSTOMER C

JOIN ACCOUNT A ON C.Code = A.CusCode

WHERE C.FName + ' ' + C.LName = N'Hoang Minh'

-- 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 “Hoang Minh” (Supposing that he has many saving accounts)

SELECT C.Code, C.FName + ' ' + C.LName as FullName, C.Email, SUM(Balance) as TotalBalance

FROM CUSTOMER C

JOIN ACCOUNT A ON C.Code = A.CusCode

JOIN SAVINGACCOUNT S ON S.AccNumber = A.Number

WHERE C.FName + ' ' + C.LName = N'Hoang Minh'

GROUP BY C.Code, C.FName + ' ' + C.LName, C.Email

## **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 (Functional Requirements)**

##### a. Danh mục sản phẩm (Product Catalog)

* Người mua có thể:  
  + Xem danh sách sản phẩm.
  + Lọc sản phẩm theo tên hoặc giá.
  + Sắp xếp sản phẩm theo tên hoặc giá.
  + Nhấp vào sản phẩm để xem chi tiết sản phẩm.
  + Xem chi tiết: tên, mô tả, số lượng tồn kho, địa chỉ kho, trạng thái sản phẩm nổi bật, đánh giá.
* Quản trị viên có thể:  
  + Quản lý sản phẩm (thêm, sửa, xóa).
  + Gán sản phẩm vào kho hiện có.

##### b. Giỏ hàng (Shopping Cart)

* Người mua có thể:  
  + Thêm sản phẩm vào giỏ từ trang danh sách hoặc trang chi tiết thông qua nút “Mua hàng”.
  + Xem giỏ hàng với 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.
  + Xóa sản phẩm khỏi giỏ hàng.
  + Thực hiện thanh toán từ giỏ hàng.
* Hệ thống:  
  + Tự động cập nhật bảng tổng hợp mỗi khi thêm/xóa sản phẩm.
* Quản trị viên có thể:  
  + Xem giỏ hàng của bất kỳ người dùng nào.
  + Kích hoạt hoặc vô hiệu hóa giỏ hàng không hợp lệ.

##### c. Quy trình thanh toán (Payment Process)

* Khi người mua nhấn nút Thanh toán:  
  + Hệ thống xác thực thông tin sản phẩm.
  + Hệ thống xử lý thanh toán.
  + Hệ thống gửi email thông báo cho người mua.

##### d. Tồn kho (Inventory)

* Quản trị viên có thể:  
  + Quản lý kho hàng (thêm, sửa, xóa).
  + Cập nhật số lượng tồn kho của sản phẩm.

##### e. Đánh giá sản phẩm (Rating)

* Người mua có thể:  
  + Đánh giá sản phẩm theo thang điểm 1 → 5 sao.

##### f. Kiểm soát truy cập (Access Control)

* Người dùng có thể:  
  + Đăng nhập/Đăng xuất hệ thống.
  + Nếu là Buyer → chuyển hướng đến trang danh mục sản phẩm.
  + Nếu là SysAdmin → chuyển hướng đến trang quản trị.
* Hệ thống phải:  
  + Xác thực tài khoản đăng nhập.
  + Phân quyền theo vai trò (Buyer hoặc Admin).

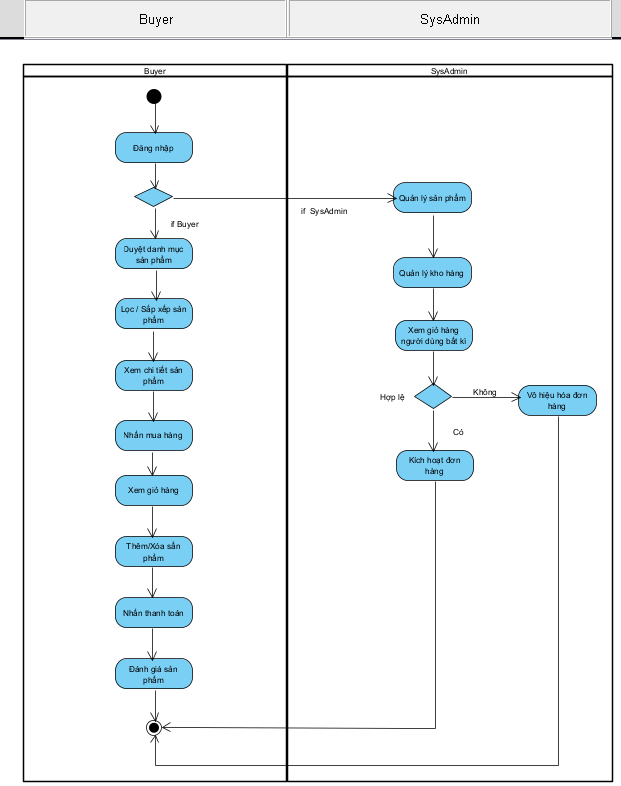
##### g. Thiết lập dữ liệu ban đầu

* Khi hệ thống khởi động:  
  + Tạo người dùng quản trị.
  + Tạo hai người dùng mua hàng.
  + Tạo dữ liệu mẫu cho sản phẩm, kho hàng, đánh giá.

#### **Yêu cầu phi chức năng (Non-Functional Requirements)**

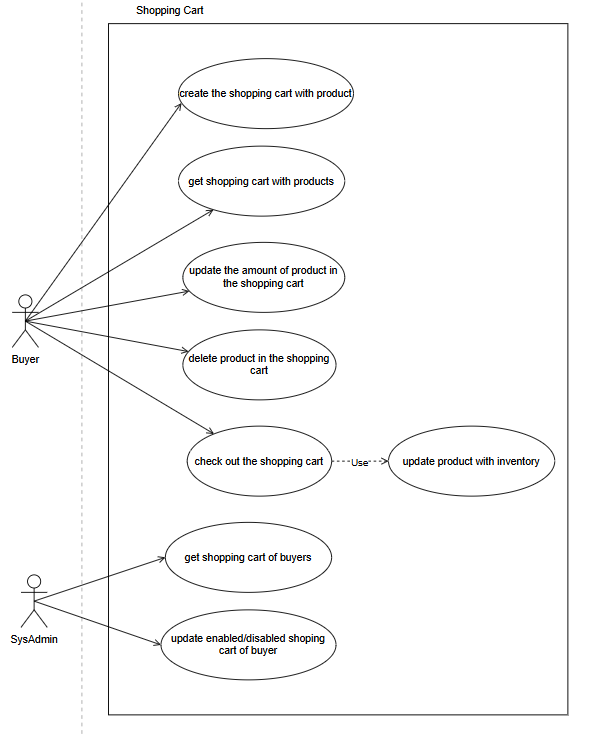
* **Hiệu năng (Performance):**
  + Hệ thống phải tải danh sách sản phẩm nhanh chóng (< 3 giây).
  + Giỏ hàng và bảng tổng hợp phải cập nhật theo thời gian thực sau khi thêm/xóa sản phẩm.
* **Bảo mật (Security):**
  + Thông tin người dùng (mật khẩu) phải được mã hóa.
  + Giao dịch thanh toán cần được bảo vệ bằng HTTPS/SSL.
  + Phân quyền rõ ràng: Buyer không thể truy cập trang quản trị.
* **Khả năng mở rộng (Scalability):**
  + Hệ thống hỗ trợ mở rộng số lượng sản phẩm, kho và người dùng lớn mà không giảm hiệu suất.
* **Tính sẵn sàng (Availability):**
  + Hệ thống cần hoạt động ổn định, tối thiểu 99% uptime.
* **Tính tin cậy (Reliability):**
  + Quá trình thanh toán phải đảm bảo chính xác, không gây mất đơn hàng.
  + Nếu hệ thống gặp lỗi trong quá trình thanh toán, cần rollback và thông báo cho người dùng.
* **Khả năng sử dụng (Usability):**
  + Giao diện trực quan, dễ sử dụng cho cả Buyer và SysAdmin.
  + Có thông báo rõ ràng khi thao tác thành công/thất bại.
* **Khả năng bảo trì (Maintainability):**
  + Hệ thống cần thiết kế theo mô-đun (module-based) để dễ dàng nâng cấp và sửa lỗi.
* **Khả năng tương thích (Compatibility):**
  + Hoạt động trên các trình duyệt phổ biến (Chrome, Edge, Firefox, Safari).
  + Hỗ trợ đa nền tảng: desktop, tablet, mobile.

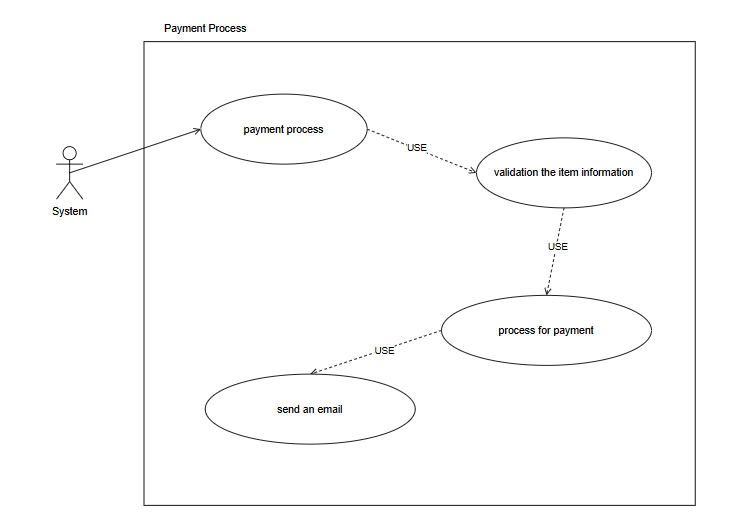
### **2) Vẽ qui trình nghiệp vụ cho website CoolStore**

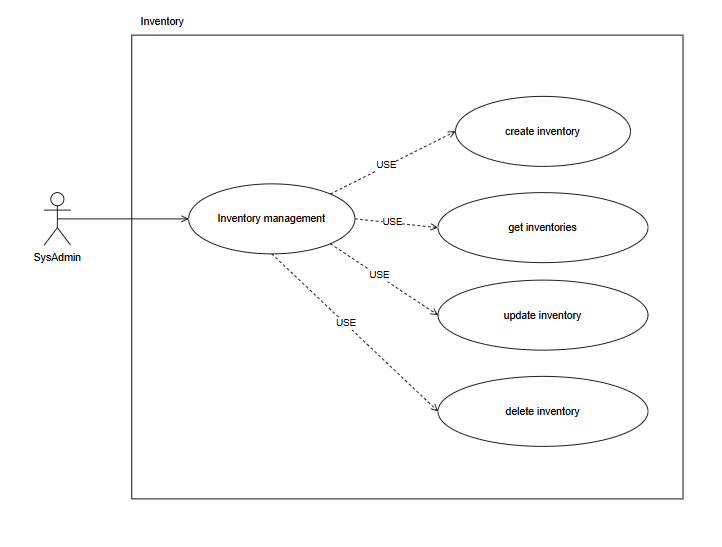


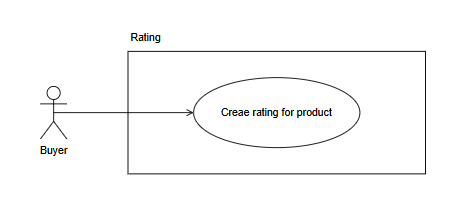
### **3) Xây dựng use case diagram cho website CoolStore.**

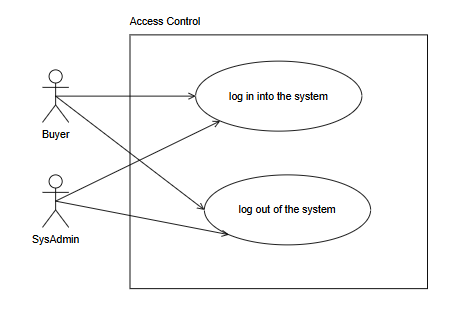












### **4) Mô tả chi tiết một Use Case "Mua sản phẩm từ trang chi tiết sản phẩm". Trình bày theo mẫu chuẩn gồm: Tên use case, Tác nhân chính, Mục tiêu, Tiền điều kiện, Luồng chính, Luồng thay thế, Hậu điều kiện, Ghi chú (nếu có).**

1. **Register for Courses**

Focused Use Case

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
| **Use case number** | UC1 | | |
| **Use case name** | Mua sản phẩm từ trang chi tiết sản phẩm | | |
| **Actor** | Buyer | | |
| **Purpose** | Buyer xem trang chi tiết của một sản phẩm, chọn số lượng (và biến thể nếu có), thêm vào giỏ, tiến hành checkout để tạo đơn hàng; hệ thống trừ tồn kho tương ứng và trả về xác nhận thành công. | | |
| **Preconditions:** | · Sản phẩm tồn tại và đang “enabled/visible”.  · Hệ thống Product Catalog và Shopping Cart hoạt động.  · (Tuỳ chính sách) Buyer đã đăng nhập; nếu cho phép guest thì phải cung cấp thông tin tối thiểu ở bước checkout.  · Tồn kho của sản phẩm ≥ 1. | | |
| **Postconditions:** | Thành công:  · Đơn hàng được tạo (ID đơn hàng, chi tiết mặt hàng, tổng tiền, trạng thái).  · Tồn kho sản phẩm được **giảm tương ứng** (use case “update product with inventory”).  · Giỏ hàng được đánh dấu đã checkout/đóng.  Thất bại/huỷ:  · Không có thay đổi tồn kho; giỏ còn nguyên hoặc ở trạng thái pending theo chính sách. | | |
| **Maturity** | Focused | | |
| **Summary** | Học sinh đăng ký môn học trên hệ thống và hệ thống chuyển yêu cầu tính tiền đến đến hệ thống bên thứ 3 để tính học phí, mục đích để ghi nợ hoặc kiểm tra tình trạng học phí. | | |
| **Basic Register for Courses**  **(luồng nghiệp vụ chính)** | **Actor Action** | **System Response** | |
| 1. Buyer mở trang chi tiết sản phẩm |  | |
|  | 2. Hệ thống hiển thị thông tin: tên, giá, mô tả, hình ảnh, tồn kho hiện tại. **E1** | |
| 3. Buyer chọn số lượng => nhấn Add to Cart **E2** |  | |
|  | 4. Hệ thống sẽ kiểm tra nếu Buyer chưa có giỏ hàng, tạo giỏ (use case “create the shopping cart with product”). **A1** | |
|  | 5. Hệ thống xác nhận thêm giỏ hàng thành công và hiển thị thông báo trên monitor. **A2** | |
| 6. Buyer mở giỏ hàng để kiểm tra (use case “get shopping cart with products”). |  | |
| 7. Buyer bấm Checkout (use case “checkout the shopping cart”). **A3** |  | |
|  | 8. Hệ thống thực hiện và chuyển đến phần thanh toán theo cấu hình (VNPAY, Chuyển khoản, trả sau) và tạo bảng tạm để reserved sản phẩm. **E3** | |
| 9. Buyer xác nhận thanh toán theo lựa chọn được cấu hình từ hệ thống |  | |
|  | 10. Hệ thống commit | |
|  | 11. Hệ thống tạo đơn hàng ở trạng thái “success/paid” (hoặc theo chính sách) và hiển thị **xác nhận đặt hàng** cho Buyer. | |
|  | 12. Hệ thống cập nhật tồn kho (giá trị thực) vào cơ sở dữ liệu. | |
| Kết thúc | Kết thúc | |
| **Alternative Path**  **(Luồng phụ) (Thay đổi hướng khác từ Action chính, không phải lỗi hệ thống).** | **4.A1. Đã có giỏ hàng** | | |
|  | | 1. Hệ thống thêm sản phẩm hoặc cộng dồn số lượng use case “update the amount of product in the shopping cart”). **E2** |
|  | | 2. Hệ thống trả về bước 5 |
| **5. A2. Buyer chỉnh sửa giỏ** | | |
|  | | 1. Buyer thay đổi số lượng (use case “update the amount of product in the shopping cart”) hoặc xoá dòng hàng (use case “delete product in the shopping cart”). |
|  | | 2. Quay lại bước 7. |
| **7. A3. Thông tin yêu cầu thêm** (địa chỉ/điện thoại/đăng nhập) đối với Guest | | |
|  | | 1. Hệ thống yêu cầu bổ sung địa chỉ/đăng nhập. |
| 2. Buyer thực hiện yêu cầu của hệ thống để checkout | |  |
|  | | 3. Hệ thống kiểm tra sau khi hợp lệ sẽ quay về bước 8. |
| **Exception Paths (Luồng ngoại lệ) (Lỗi hệ thống)** | **E1. Sản phẩm bị ẩn/đã xoá**  - Hệ thống báo “Sản phẩm không khả dụng”; kết thúc use case. | | |
| **E2. Số lượng không hợp lệ (≤0 hoặc vượt giới hạn tối đa)**  - Hệ thống báo lỗi validate; yêu cầu Buyer nhập lại hoặc hiển thị “Không đủ hàng” nếu vượt tồn kho tại thời điểm thêm vào giỏ.s | | |
| **E3. Thanh toán thất bại/tạm thời, Tranh chấp tồn kho khi checkout, giá thay đổi trước khi thanh toán hoặc commit một phần thất bại. Hệ thống hiển thị lỗi trên monitor. Quay về bước 6.** | | |
|  |  |  |  |

Đề xuất cải tiến và bổ sung thêm các use case hoặc thêm chức năng cho use case cần thiết.

**Đăng ký khóa học (Register for Courses):**

- Thêm Validate Course Availability: Hệ thống kiểm tra số lượng sinh viên đã đăng ký và số chỗ còn trống của khóa học trước khi cho phép đăng ký.

- Thêm Handle Course Conflicts: Hệ thống tự động kiểm tra và thông báo cho sinh viên nếu có xung đột về thời khóa biểu khi đăng ký nhiều khóa học.

**Chọn khóa học để dạy (Select Courses to Teach):**

- Thêm Verify Professor Qualification: Hệ thống kiểm tra bằng cấp và chuyên môn của giảng viên có phù hợp với khóa học được chọn hay không.

- Thêm Manage Teaching Load: Hệ thống tự động tính toán và cảnh báo khi số lượng khóa học mà giảng viên đăng ký vượt quá giới hạn quy định.

**Quản lý giáo trình (Maintain Curriculum):**

- Thêm Manage Curriculum Version: Hệ thống quản lý các phiên bản giáo trình khác nhau, cho phép truy xuất và so sánh.

- Thêm Link Course Materials: Liên kết trực tiếp tài liệu học tập, giáo trình điện tử với các khóa học cụ thể.

**Quản lý thông tin sinh viên (Maintain Student Information):**

- Thêm Update Student Academic Progress: Cập nhật tiến độ học tập, kết quả học tập của sinh viên theo từng học kỳ/năm học.

- Thêm Generate Student Reports: Tạo các báo cáo về tình hình học tập, chuyên cần, hay các thông tin khác của sinh viên.

**Quản lý thông tin giảng viên (Maintain Professor Information):**

- Thêm Manage Professor Leave/Absence: Quản lý lịch nghỉ phép, vắng mặt của giảng viên và tự động tìm kiếm người thay thế hoặc điều chỉnh lịch dạy.

- Thêm Track Professor Research & Publications: Theo dõi các công trình nghiên cứu và xuất bản của giảng viên để đánh giá hiệu suất.

Bổ sung các use case cần thiết khác

**Quản lý lịch học/lịch dạy (Manage Schedules):**

- Xem lịch học (View Student Schedule): Sinh viên có thể xem lịch học cá nhân của mình.

- Xem lịch dạy (View Professor Schedule): Giảng viên có thể xem lịch giảng dạy của mình.

- Cập nhật lịch (Update Schedule): Quản trị viên/Giáo vụ có thể thay đổi lịch học/dạy khi có sự cố hoặc yêu cầu.

**Quản lý điểm số (Manage Grades):**

- Nhập điểm (Enter Grades): Giảng viên có thể nhập điểm cho các môn học mình phụ trách.

- Xem điểm (View Grades): Sinh viên có thể xem điểm của mình.

- Phê duyệt điểm (Approve Grades): Quản trị viên/Giáo vụ có thể duyệt điểm trước khi công bố.

**Quản lý báo cáo và thống kê (Generate Reports & Statistics):**

- Tạo báo cáo học phí (Generate Tuition Report): Xuất báo cáo về tình hình đóng học phí của sinh viên.

- Tạo báo cáo kết quả học tập (Generate Academic Performance Report): Xuất báo cáo tổng hợp về kết quả học tập của toàn trường, khoa, hoặc từng khóa học.

**Quản lý tài chính (Financial Management):**

- Xem thông tin học phí (View Tuition Information): Sinh viên xem các khoản học phí cần đóng.

- Thanh toán học phí (Pay Tuition): Sinh viên thực hiện thanh toán học phí.

- Quản lý thu/chi (Manage Revenue/Expenses): Quản trị viên theo dõi các khoản thu/chi liên quan đến đào tạo.

**Hỗ trợ và phản hồi (Support & Feedback):**

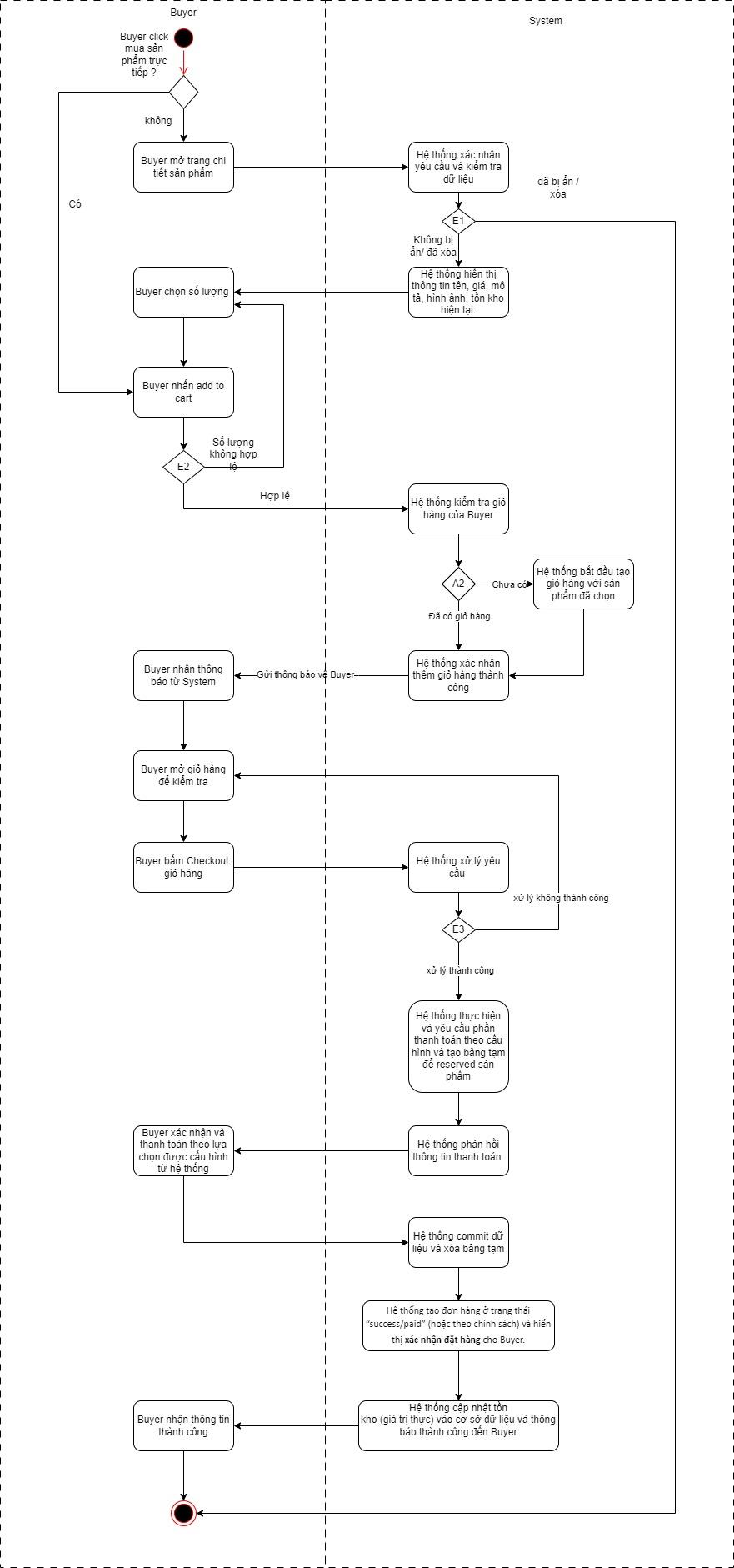
- Gửi phản hồi (Send Feedback): Sinh viên/Giảng viên gửi phản hồi về khóa học, giáo trình, hoặc dịch vụ.

- Quản lý phản hồi (Manage Feedback): Quản trị viên xử lý và phản hồi lại.

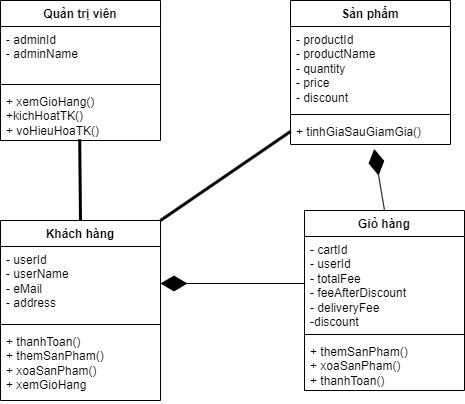
**Quản lý thông báo (Notification Management):**

- Gửi thông báo (Send Notifications): Hệ thống gửi thông báo quan trọng đến sinh viên/giảng viên (lịch thi, lịch học thay đổi, thông báo mới).

- Xem thông báo (View Notifications): Người dùng có thể xem các thông báo từ hệ thống.

**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.**



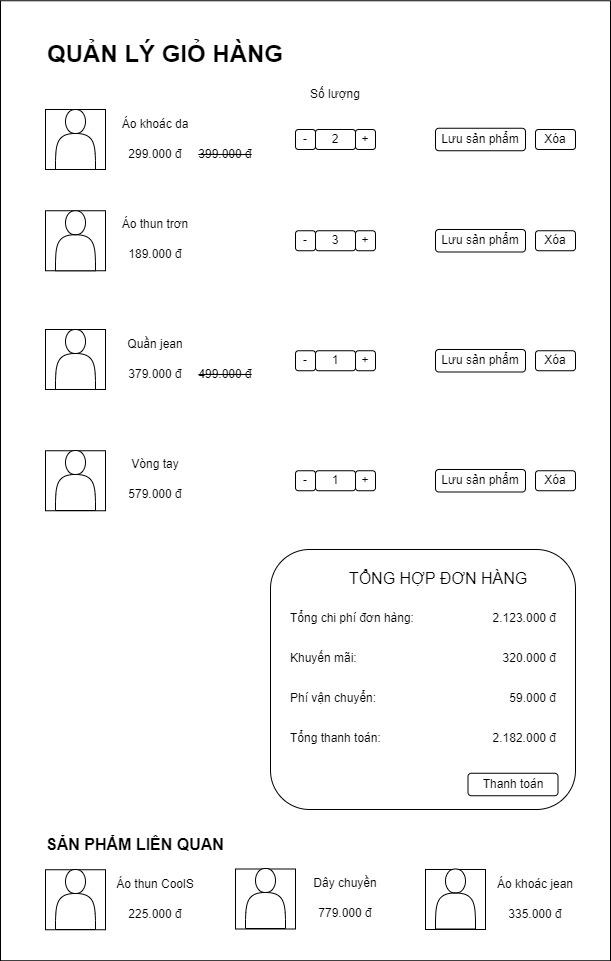
### **7) Thiết kế giao diện màn hình cho chức năng “Quản lý giỏ hàng” (có thể thiết kế một hoặc nhiều màn hình) gồm các thông tin:**

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

#### **b) Chức năng thao tác nhanh: Tăng/giảm số lượng sản phẩm; Xóa sản phẩm khỏi giỏ; Lưu sản phẩm để mùa sau.**

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

#### **d) 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).**

**Kiến trúc 3-tier cho CoolStore**

**1. Presentation Layer (UI/Client Layer)**

* **Chức năng**: Giao diện trực quan để người dùng (Buyer, SysAdmin) tương tác.
* **Thành phần trong CoolStore**:
  + Website CoolStore (React/Angular/Vue hoặc ASP.NET MVC/Razor Pages).
  + Trang:
    - Danh mục sản phẩm (Product Catalog).
    - Trang chi tiết sản phẩm.
    - Giỏ hàng (Shopping Cart).
    - Thanh toán (Payment).
    - Đánh giá (Rating).
    - Trang đăng nhập/đăng ký.
    - Trang quản trị (Admin Dashboard).
* **Nhiệm vụ chính**:
  + Hiển thị dữ liệu lấy từ Business Layer qua API.
  + Gửi yêu cầu thao tác (mua hàng, thêm/xóa sản phẩm, đánh giá, quản trị sản phẩm) xuống Business Layer.

**2. Business Logic Layer (Service Layer / Application Layer)**

* **Chức năng**: Xử lý toàn bộ **nghiệp vụ** và quy tắc kinh doanh của CoolStore.
* **Thành phần trong CoolStore (microservices)**:
  + **Catalog Service**: Quản lý sản phẩm, hiển thị thông tin sản phẩm, lọc/sắp xếp.
  + **Cart Service**: Quản lý giỏ hàng, tính toán tổng tiền, khuyến mãi, phí vận chuyển.
  + **Payment Service**: Xử lý thanh toán, xác thực đơn hàng, gửi email.
  + **Inventory Service**: Quản lý kho, số lượng tồn.
  + **Rating Service**: Lưu và hiển thị đánh giá sản phẩm.
  + **Identity & Access Control Service**: Đăng nhập/đăng xuất, phân quyền Buyer vs Admin.
* **Nhiệm vụ chính**:
  + Xử lý logic nghiệp vụ (ví dụ: cập nhật tồn kho sau khi thanh toán).
  + Điều phối dữ liệu giữa Presentation Layer và Data Layer.
  + Áp dụng quy tắc kinh doanh (business rules) như: giỏ hàng hợp lệ, khuyến mãi, quyền truy cập.

**3. Data Layer (Database Layer / Persistence Layer)**

* **Chức năng**: Quản lý lưu trữ và truy xuất dữ liệu.
* **Thành phần trong CoolStore**:
  + **Product DB**: lưu danh mục sản phẩm.
  + **Cart DB**: lưu giỏ hàng theo người dùng.
  + **Order/Payment DB**: lưu thông tin thanh toán, hóa đơn.
  + **Inventory DB**: lưu thông tin kho, số lượng tồn.
  + **Rating DB**: lưu đánh giá sản phẩm.
  + **User DB (Identity DB)**: lưu thông tin tài khoản, phân quyền.
* **Công nghệ đề xuất**:
  + SQL Database (PostgreSQL, SQL Server, MySQL) cho dữ liệu quan hệ (sản phẩm, đơn hàng, người dùng).
  + Có thể kết hợp NoSQL (MongoDB, Redis) cho cache/giỏ hàng nếu cần hiệu năng cao.

**Luồng xử lý ví dụ**

1. Buyer truy cập **UI → chọn sản phẩm → nhấn “Mua hàng”**.
2. **UI** gửi request đến **Cart Service (Business Layer)**.
3. **Cart Service** kiểm tra thông tin sản phẩm từ **Catalog Service** và tồn kho từ **Inventory Service**.
4. Kết quả được trả về **UI** để cập nhật giỏ hàng cho người mua.
5. Khi thanh toán: **UI → Payment Service** → xử lý nghiệp vụ → cập nhật **Inventory DB, Order DB** → gửi email xác nhận.

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

**Test Cases cho chức năng “Thêm sản phẩm vào giỏ hàng”**

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

* **Dữ liệu đầu vào**:
  + Người mua đã đăng nhập.
  + Sản phẩm "Laptop A" còn tồn kho = 10.
* **Bước thực hiện**:
  1. Buyer đăng nhập vào hệ thống.
  2. Truy cập trang danh mục sản phẩm.
  3. Chọn sản phẩm "Laptop A".
  4. Nhấn nút **“Mua hàng”**.
* **Kết quả mong đợi**:
  1. Sản phẩm "Laptop A" được thêm vào giỏ hàng.
  2. Giỏ hàng hiển thị đúng số lượng sản phẩm, giá tiền và tổng tiền.
  3. Tồn kho của sản phẩm không thay đổi ngay (chỉ trừ sau khi thanh toán).

**Test Case 2: Thêm sản phẩm đã hết hàng**

* **Dữ liệu đầu vào**:
  + Người mua đã đăng nhập.
  + Sản phẩm "Điện thoại B" có tồn kho = 0.
* **Bước thực hiện**:
  1. Buyer đăng nhập vào hệ thống.
  2. Truy cập trang chi tiết sản phẩm "Điện thoại B".
  3. Nhấn nút **“Mua hàng”**.
* **Kết quả mong đợi**:
  1. Hệ thống hiển thị thông báo lỗi: *“Sản phẩm đã hết hàng, không thể thêm vào giỏ hàng”*.
  2. Giỏ hàng không thay đổi.

**Test Case 3: Thêm cùng một sản phẩm nhiều lần**

* **Dữ liệu đầu vào**:
  + Người mua đã đăng nhập.
  + Sản phẩm "Tai nghe C" còn tồn kho = 20.
* **Bước thực hiện**:
  1. Buyer đăng nhập vào hệ thống.
  2. Truy cập trang sản phẩm "Tai nghe C".
  3. Nhấn **“Mua hàng”** 2 lần liên tiếp.
  4. Mở giỏ hàng để kiểm tra.
* **Kết quả mong đợi**:
  1. Giỏ hàng hiển thị sản phẩm "Tai nghe C" với số lượng = 2.
  2. Tổng tiền giỏ hàng được tính đúng = giá sản phẩm × 2.
  3. Không phát sinh lỗi trùng lặp bản ghi trong giỏ hàng.