TRƯỜNG ĐẠI HỌC SÀI GÒN

**KHOA CÔNG NGHỆ THÔNG TIN**

A blue button with white text

Description automatically generated

**KIỂM THỬ PHẦN MỀM**

Lớp: DCT122C3 – Mã học phần: 841408

**ĐỀ TÀI: BÀI TẬP LAB 2**

Sinh viên thực hiện: **TRẦN HOÀNG BẢO– 3122411014**

**THÁI TRÍ VĂN- 3122411242**

**PHÙNG ANH TUẤN-3122411237**

*Giảng viên hướng dẫn: Đỗ Như Tài*

Thành phố Hồ Chí Minh, tháng 9 năm 2025

**BẢNG PHÂN CÔNG**

|  |  |  |
| --- | --- | --- |
| **Tên thành viên** | **Nội dung công việc** | **Tỉ lệ hoàn thành** |
| Thái Trí Văn | Bài tập minh họa (1, 2) và bài tập ứng dụng (câu 1-5) | **100%** |
| Trần Hoàng Bảo | Bài tập minh họa (3, 4) và bài tập ứng dụng (câu 6-9) | **100%** |
| Phùng Anh Tuấn | Bài tập minh họa (5, 6, 7) | **100%** |

**MỤC LỤC**

[**BÀI TẬP MINH HỌA**  4](#_Toc209045844)

[**Bài 1. Vẽ lại qui trình nghiệp vụ sau (Độc giả, NV Tiếp tân, Thành viên)** 4](#_Toc209045845)

[**Bài 2. Vẽ mô hình khái niệm**  5](#_Toc209045846)

[**Bài 3. Vẽ lại business use case**  5](#_Toc209045847)

[**Bài 4. Vẽ sơ đồ hoạt động và tương tác sau và tóm tắt ý nghĩa sơ đồ** 10](#_Toc209045848)

[**Bài 5. Vẽ lược đồ lớp cho bài toán Quản lý thư viện**  12](#_Toc209045849)

[**Bài 6. Vẽ lược đồ** 13](#_Toc209045850)

[**Bài 7. Phân tích dữ liệu cho ABC Bank** 14](#_Toc209045851)

[**1. Thiết kế Cơ sở dữ liệu ở mức Khái niệm, Luận lý và Vật lý** 14](#_Toc209045852)

[**2. Query: Insert, Update, Delete, Select** 15](#_Toc209045853)

[**BÀI TẬP VẬN DỤNG** 20](#_Toc209045854)

[**1. Các yêu cầu chức năng và phi chức năng của hệ thống CoolStore** 20](#_Toc209045855)

[**2. Vẽ qui trình nghiệp vụ cho website CoolStore** 22](#_Toc209045856)

[**3. Xây dựng use case diagram cho website CoolStore.** 23](#_Toc209045857)

[**4. Mô tả chi tiết một Use Case** 27](#_Toc209045858)

[**5. Activity diagram mô tả quy trình mua hàng** 30](#_Toc209045859)

[**6. Class Diagram cho module Giỏ hàng.** 31](#_Toc209045860)

[**7. Giao diện màn hình chức năng “Quản lý giỏ hàng”** 32](#_Toc209045861)

[**8. Mô hình kiến trúc 3 lớp (3-tier architecture) cho hệ thống CoolStore** 32](#_Toc209045862)

[**9. 3 test case cho chức năng “Thêm sản phẩm vào giỏ hàng”** 34](#_Toc209045863)

# **BÀI TẬP MINH HỌA**

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

A diagram of a flowchart

AI-generated content may be incorrect.

## **Bài 2. Vẽ mô hình khái niệm**

A diagram of a car rental

AI-generated content may be incorrect.

## **Bài 3. Vẽ lại business use case**

A black background with white circles

AI-generated content may be incorrect.

a) Sơ đồ trên gồm 4 actor:

* Student: Sinh viên dùng hệ thống để đăng ký học phần.
* Professor: đại diện cho giảng viên đăng ký các học phần sẽ giảng dạy.
* Registrar: đại diện cho bộ phận hành chính chịu trách nhiệm quản lý hệ thống học phần.
* Billing System: hệ thống thanh toán xử lý các giao dịch đăng ký học phần.

b)

* Register for Course: Use case này cho phép sinh viên chọn và đăng ký các khóa học trong kỳ học. Nó liên quan đến việc kiểm tra điều kiện đủ điều kiện, cập nhật lịch học, và kích hoạt thanh toán qua Billing System.
* Select Courses to Teach: Use case dành cho giảng viên, cho phép họ xem danh sách các học phần có sẵn và chọn những khóa mà họ muốn giảng dạy.
* Maintain Curriculum: Use case chính cho Registrar, liên quan đến việc cập nhật, chỉnh sửa và quản lý cấu trúc chương trình học tổng thể, bao gồm các khóa học, yêu cầu tín chỉ, và lịch trình.
* Maintain Student Information: cho phép cập nhật dữ liệu cá nhân của sinh viên như thông tin liên lạc, lịch sử học tập, và điểm số.
* Maintain Professor Information: tập trung vào việc quản lý dữ liệu của giáo sư như bằng cấp, lịch dạy, và thông tin cá nhân.
* Registration Validation : Use case phụ dùng để kiểm tra tính hợp lệ của các đăng ký học phần, chẳng hạn như kiểm tra điều kiện tiên quyết, giới hạn chỗ ngồi, hoặc xung đột lịch.

c)

Sơ đồ use case trên tồn tại mối quan hệ <<uses>> giữa các use case sau:

* Mối quan hệ «uses» từ Maintain Curriculum đến Maintain Student Information và Maintain Professor Information:
  + Khi duy trì chương trình học (Maintain Curriculum), hệ thống cần truy cập và cập nhật thông tin sinh viên (ví dụ: kiểm tra điều kiện đủ điều kiện của sinh viên cho khóa học mới) và thông tin giáo sư (ví dụ: phân công giảng viên phù hợp).
* Mối quan hệ «uses» từ Maintain Student Information đến Registration Validation:
  + Khi duy trì thông tin sinh viên, hệ thống sử dụng chức năng xác thực đăng ký để kiểm tra và cập nhật tình trạng đăng ký hiện tại của sinh viên (ví dụ: xác nhận xem sinh viên có đủ điều kiện cho các khóa học đã đăng ký không).

d) Kịch bản cho use case “Register for Course”

Tên Use Case: Register for Course

Actors chính: Student (Sinh viên)

Mô tả ngắn gọn: Sinh viên đăng ký một hoặc nhiều khóa học cho kỳ học sắp tới, hệ thống kiểm tra điều kiện và kích hoạt quy trình thanh toán.

Preconditions (Điều kiện tiên quyết):

* Sinh viên đã đăng nhập vào hệ thống với tài khoản hợp lệ.
* Kỳ đăng ký khóa học đang mở.
* Thông tin sinh viên (bao gồm tình trạng tài chính) đã được duy trì cập nhật qua use case Maintain Student Information.

Main Flow (Luồng chính):

1. Sinh viên chọn use case "Register for Course" từ giao diện hệ thống.
2. Hệ thống hiển thị danh sách các khóa học khả dụng cho kỳ học (dựa trên chương trình học từ Maintain Curriculum).
3. Sinh viên chọn một hoặc nhiều khóa học và xác nhận (kiểm tra giới hạn tín chỉ tối đa).
4. Hệ thống sử dụng Registration Validation để kiểm tra điều kiện:
   1. Sinh viên có đủ điều kiện tiên quyết (prerequisites) cho khóa học không?
   2. Sinh viên chưa vượt quá giới hạn tín chỉ?
   3. Không có xung đột lịch học?
   4. Nếu hợp lệ, hệ thống cập nhật đăng ký vào hồ sơ sinh viên (tích hợp với Maintain Student Information).
5. Hệ thống gửi yêu cầu đến Billing System để tạo hóa đơn phí khóa học.
6. Hệ thống xác nhận đăng ký thành công và gửi thông báo email cho sinh viên.

Alternative Flows (Luồng thay thế):

* 3a. Nếu sinh viên muốn xem chi tiết khóa học (mô tả, giáo sư từ Maintain Professor Information): Hệ thống hiển thị thông tin bổ sung, sau đó quay lại bước 3.
* 5a. Nếu một khóa học đã đầy chỗ: Hệ thống thông báo và cho phép sinh viên chọn khóa khác, quay lại bước 3.

Exceptions (Ngoại lệ):

* 4a. Nếu Registration Validation thất bại (ví dụ: sinh viên nợ phí từ kỳ trước): Hệ thống hiển thị lỗi và hướng dẫn thanh toán trước khi tiếp tục. Use case bị hủy nếu không khắc phục.
* 6a. Nếu kết nối với Billing System thất bại: Hệ thống lưu đăng ký tạm thời và yêu cầu thử lại sau, gửi thông báo lỗi.

Postconditions (Điều kiện hậu quyết):

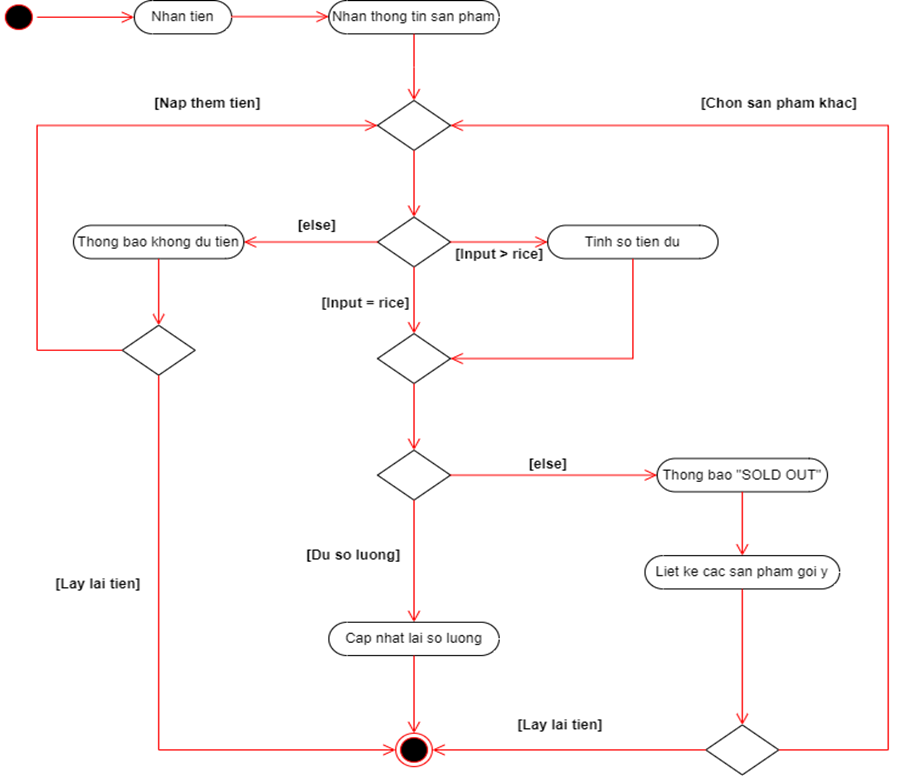
* Đăng ký được cập nhật thành công trong hồ sơ sinh viên.
* Hóa đơn được tạo và gửi đến Billing System.
* Sinh viên nhận được lịch học cập nhật.
* Hệ thống ghi log hoạt động để audit.

e)

Dựa trên sơ đồ hiện tại, hệ thống tập trung vào đăng ký và quản lý dữ liệu cơ bản, nhưng thiếu một số chức năng quan trọng cho hệ thống quản lý trường học toàn diện. Có thể cải tiến như sau:

* Thêm mối quan hệ «include» bắt buộc giữa "Register for Course" và "Registration Validation" để đảm bảo xác thực luôn được thực hiện, tránh trường hợp đăng ký không kiểm tra.
* Tích hợp tự động hóa hơn cho "Select Courses to Teach" bằng cách sử dụng AI gợi ý khóa học dựa trên chuyên môn giáo sư (từ Maintain Professor Information), giảm thủ công và lỗi con người.
* Cải thiện bảo mật: Yêu cầu xác thực hai yếu tố cho tất cả use case liên quan đến dữ liệu cá nhân (Maintain Student/Professor Information).
* Bổ sung thêm các use case mới như:
  + View Grades and Transcripts (Xem điểm số và bảng điểm)
  + Generate Reports (Tạo báo cáo)
  + Course Evaluation (Đánh giá khóa học)

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



Activity diagram trong hình mô tả quy trình mua sắm sản phẩm với các bước cơ bản sau:

1. Nhận tiền: Quá trình bắt đầu khi khách hàng nhận tiền.
2. Nhận thông tin sản phẩm: Khách hàng chọn sản phẩm cần mua.
3. Kiểm tra số dư: Hệ thống kiểm tra xem khách hàng có đủ tiền để thanh toán không.
   1. Nếu không đủ: Thông báo "Không đủ tiền" và yêu cầu nạp thêm tiền hoặc lấy lại tiền.
   2. Nếu đủ: Tiến hành kiểm tra tiếp theo.
4. Chọn sản phẩm: Khách hàng chọn sản phẩm cụ thể (input > rice).
   1. Nếu sản phẩm hết (sold out): Thông báo "SOLD OUT" và liệt kê các sản phẩm khác.
   2. Nếu có sẵn: Cập nhật số lượng và tiếp tục.
5. Cập nhật số lượng: Hệ thống cập nhật số lượng sản phẩm sau khi chọn.
6. Lấy lại tiền: Nếu không mua hoặc chọn sản phẩm khác, khách hàng lấy lại tiền.
7. Lặp lại quy trình: Khách hàng có thể lặp lại quy trình từ bước nhận thông tin sản phẩm.

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

**A screenshot of a computer screen

AI-generated content may be incorrect.**

## **Bài 6. Vẽ lược đồ**

**A screenshot of a chat

AI-generated content may be incorrect.**

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

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

**A diagram of a diagram

AI-generated content may be incorrect.**

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.**

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

('Downtown', '101', 'Main St', 'District 1', 'Metropolis', 'Central', 'contact@downtown.abc.com', NULL), -- MngCode will be updated later

('Uptown', '202', 'Oak Ave', 'District 5', 'Metropolis', 'North', 'contact@uptown.abc.com', NULL);

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

('E1001', 'John', 'Smith', '10', 'First St', 'District 2', 'Metropolis', '1990-05-15', 'j.smith@abc.com', 'Downtown'),

('E1002', 'Jane', 'Doe', '25', 'Second Ave', 'District 3', 'Metropolis', '2010-03-20', 'j.doe@abc.com', 'Downtown'),

('E1003', 'Robert', 'Miller', '50', 'Third Rd', 'District 6', 'Metropolis', '1985-11-01', 'r.miller@abc.com', 'Uptown'),

('N1080', 'Emily', 'White', '75', 'Fourth Blvd', 'District 7', 'Metropolis', '2010-03-10', 'e.white@abc.com', 'Uptown');

UPDATE BRANCH SET MngCode = 'E1001' WHERE Name = 'Downtown';

UPDATE BRANCH SET MngCode = 'E1003' WHERE Name = 'Uptown';

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

('C001', 'Peter', 'Johnson', '200 Business Park', '123 Home Ln', 'p.johnson@email.com', 'E1001'),

('C002', 'Mary', 'Williams', '300 Corp Tower', '456 Residence St', 'm.williams@email.com', 'E1001'),

('C003', 'James', 'Lee', '400 Tech Hub', '789 Dwelling Ave', 'j.lee@email.com', 'E1002'),

('C004', 'Patricia', 'Garcia', '500 Commerce Sq', '101 Condo Cir', 'p.garcia@email.com', 'N1080');

INSERT INTO ACCOUNT (Number, CusCode) VALUES

('ACC01', 'C001'), -- Peter Johnson's Savings Account

('ACC02', 'C001'), -- Peter Johnson's Checking Account

('ACC03', 'C001'), -- Peter Johnson's Second Savings Account

('ACC04', 'C002'), -- Mary Williams' Loan

('ACC05', 'C003'), -- James Lee's Savings Account

('ACC06', 'C004'); -- Patricia Garcia's Checking Account

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

('ACC01', 5000.00, 0.015, '2020-01-20'),

('ACC03', 12000.00, 0.018, '2022-06-10'),

('ACC05', 7500.00, 0.015, '2021-03-15');

INSERT INTO CHECKINGACCOUNT (AccNumber, Balance, OpenedDate) VALUES

('ACC02', 1500.00, '2020-01-20'),

('ACC06', 2200.00, '2023-01-05');

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

('ACC04', 25000.00, 0.05, '2022-08-01');

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

DELETE FROM EMPLOYEE WHERE Code = 'N1080';

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

DELETE FROM EMPLOYEE WHERE Code = 'N1080';

Giải thích việc xảy ra với các employee bị ảnh hướng cho lệnh delete mã N1080:

- Nếu ta sử dụng DELETE RESTRICT hoặc DELETE NO ACTION thì sẽ thất bại vì vi phạm phải foreign key ciolation

- Nếu ta sử dụng DELETE SET NULL thì lệnh Delete sẽ xảy ra thành công và thấy cả các EmpCode served bởi N1080 sẽ được update thành Null

- Nếu ta sử dụng DELETE CASCADE thì lệnh này sẽ thành công và sẽ xóa hết các customer records liên kết với N1080

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

SELECT

a.Number

FROM

ACCOUNT a

JOIN

CUSTOMER c ON a.CusCode = c.Code

WHERE

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

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

SELECT Code,

FName,

LName,

DOB

FROM

EMPLOYEE

WHERE

DOB BETWEEN '2010-03-01' AND '2010-03-31';

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

SELECT

SUM(sa.Balance) AS TotalSavingsBalance

FROM

CUSTOMER c

JOIN

ACCOUNT a ON c.Code = a.CusCode

JOIN

SAVINGACCOUNT sa ON a.Number = sa.AccNumber

WHERE

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

# **BÀI TẬP VẬN DỤNG**

1. **Các yêu cầu chức năng và phi chức năng của hệ thống CoolStore**

**+ Yêu cầu chức năng (Functional Requirements)**

- Danh mục sản phẩm (**Product Catalog**):

* + Người mua (**Buyer**) có thể duyệt danh sách sản phẩm, cho phép lọc và sắp xếp sản phẩm theo tên, giá.
  + Người mua có thể 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” (nếu có) và đánh giá.).
  + Quản trị viên (**SysAdmin**) có thể quản lý sản phẩm và có quyền gán sản phẩm vào kho hiện có.

- Giỏ hàng (**Shopping Cart**):

* + Người mua có thể mua bất kỳ sản phẩm nào trên danh sách sản phẩm thông qua nút “Mua hàng” trên mỗi sản phẩm. 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.
  + Giỏ hàng tự động cập nhật bất cứ khi nào người mua thêm sản phẩm hoặc xóa sản phẩm khỏi giỏ hàng
  + Người mua có thể nhấn "Thanh toán" để sang quy trình thanh toán.
  + Quản trị viên có thể xem giỏ hàng của bất kỳ người dùng nào.
  + Quản trị viên có thể kích hoạt/vô hiệu hóa giỏ hàng không hợp lệ.

- Quy trình thanh toán (**Payment Process**):

* + Khi người mua nhấn nút thanh toán, hệ thống xác thực thông tin sản phẩm trong giỏ.
  + Xử lý thanh toán.
  + Gửi email cho người mua để người mua biết chuyện gì đang xảy ra.

- Tồn kho (**Inventory**):

* + Quản trị viên có thể quản lý kho hàng (thêm, sửa, xóa, cập nhật thông tin kho).

- Đánh giá (**Rating**):

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

- Kiểm soát truy cập (**Access Control**):

* + Người dùng có thể đăng nhập/đăng xuất hệ thống.
  + Nếu đăng nhập là Buyer → đưa đến trang danh mục sản phẩm.
  + Nếu đăng nhập là SysAdmin → đưa đến trang quản trị.

- Nhiệm vụ thiết lập ban đầu:

* + Tạo người dùng quản trị.
  + Tạo hai người dùng mua hàng.
  + Dữ liệu mẫu cho sản phẩm, kho, đánh giá cho một vài sản phẩm.

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

- Hiệu năng tốt.

- Bảo mật:

* + Thông tin đăng nhập phải được mã hóa.
  + Quyền truy cập phân biệt rõ ràng giữa Buyer và SysAdmin.

- Tính khả dụng cao, website phải luôn trong tình trạng sẵn sàng sử dụng.

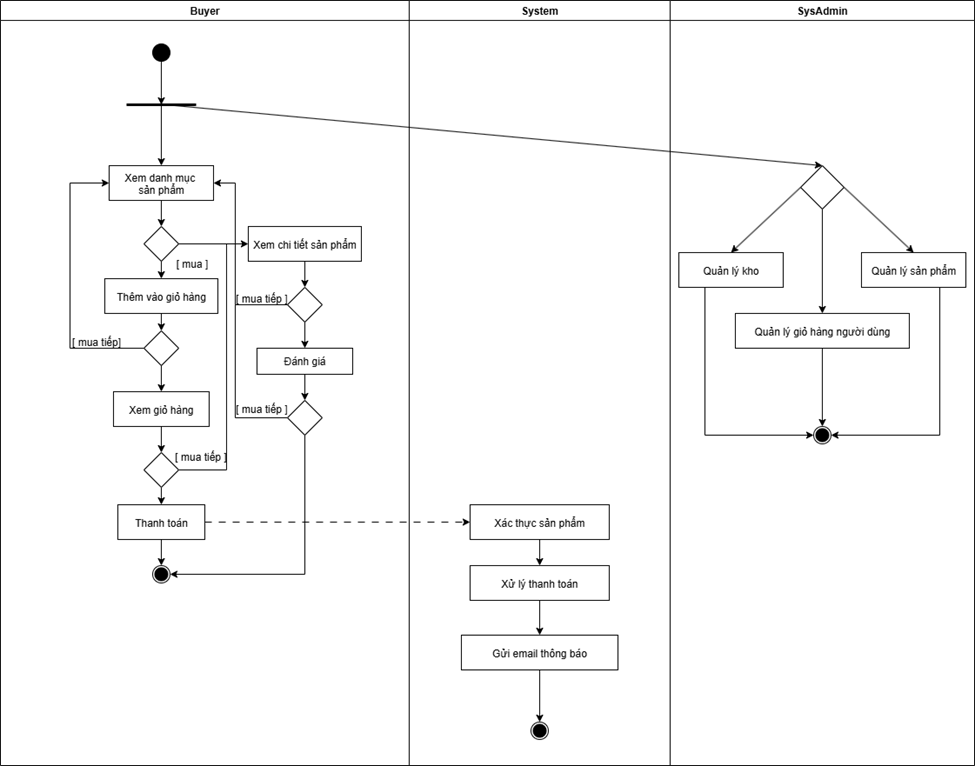
- Khả năng mở rộng mà không ảnh hưởng đến hiệu năng.

- Khả năng bảo trì tốt với mã nguồn dễ chỉnh sửa.

- Khả năng tích hợp với cổng thanh toán bên ngoài.

- Khả năng phục hồi khi lỗi xảy ra trong thanh toán, hệ thống có thể rollback giỏ hàng về trạng thái trước đó.

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



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

+ Danh mục sản phẩm:

A diagram of a diagram

AI-generated content may be incorrect.

+ Giỏ hàng

A diagram of a diagram

AI-generated content may be incorrect.

+ Tồn kho

A diagram of a diagram

AI-generated content may be incorrect.

+ Đánh giá:

A diagram with a circle and a black arrow

AI-generated content may be incorrect.

+ Kiểm soát truy cập:

A diagram of a diagram

AI-generated content may be incorrect.

+ Quy trình thanh toán:

A diagram of a company

AI-generated content may be incorrect.

1. **Mô tả chi tiết một Use Case**

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name** | Mua sản phẩm từ trang chi tiết sản phẩm. | | |
| **Use Case Number** | **UC1** | | |
| **Actor(s):** | Người mua hàng, Hệ Thống | | |
| **Maturity:** | Focused | | |
| **Description:** | Người mua hàng chọn mua sản phẩm từ trang chi tiết sản phẩm thêm vào giỏ hàng. | | |
| **Pre-Condition:** | Người dùng đã đăng nhập vào hệ thống, người dùng đang ở trang sản phẩm và sản phẩm còn tồn kho. | | |
| **Post-Condition:** | Số lượng sản phẩm trong giỏ hàng được cập nhật. | | |
| **Trigger:** | Người dùng chọn mua sản phẩm. | | |
| **Basic Course of Events:** | **Actor Action** | | **System Response** |
| 1. Người dùng chọn mua hàng từ trang chi tiết sản phẩm. | |  |
|  | | 2. Hệ thống kiểm tra số lượng tồn kho, nếu còn hàng, hệ thống thêm sản phẩm vào giỏ hàng người dùng. Thông báo tới người dùng thêm sản phẩm thành công.  **A1, A2** |
| **Alternative paths:** | **A1** | | |
| **Actor Action** | | **System Response** |
|  | | 1. Hệ thống xác nhận không còn sản phẩm tồn kho, thông báo tới người dùng đã hết hàng. |
| **A2** | | |
| **Actor Action** | **System Response** | |
|  | 1. Hệ thống thông báo tới người mua hàng giỏ hàng đã bị vô hiệu hóa bởi Quản Trị Viên | |

1. **Activity diagram mô tả quy trình mua hàng**

A screenshot of a chat

AI-generated content may be incorrect.

1. **Class Diagram cho module Giỏ hàng.**

**A screenshot of a computer

AI-generated content may be incorrect.**

1. **Giao diện màn hình chức năng “Quản lý giỏ hàng”**

**A screenshot of a computer

AI-generated content may be incorrect.**

1. **Mô hình kiến trúc 3 lớp (3-tier architecture) cho hệ thống CoolStore**

1. Presentation Layer (Lớp trình bày / Giao diện người dùng)

* Chức năng: Hiển thị giao diện cho người dùng cuối (Buyer, SysAdmin) và xử lý các thao tác nhập liệu.
* Thành phần trong CoolStore:
  + Website UI (ReactJS/Angular hoặc ASP.NET Razor Pages, Blazor, v.v.)
  + Các màn hình:
    - Danh mục sản phẩm (Product Catalog)
    - Giỏ hàng (Shopping Cart)
    - Thanh toán (Payment)
    - Quản lý kho (Inventory Management)
    - Đánh giá sản phẩm (Rating)
    - Quản trị hệ thống (Access Control)
* Ví dụ: Khi người mua nhấn “Thêm vào giỏ hàng”, yêu cầu được gửi xuống Business Layer để xử lý.

2. Business Logic Layer (Lớp nghiệp vụ / Xử lý trung gian)

* Chức năng:
  + Xử lý các quy tắc nghiệp vụ (business rules).
  + Thực hiện kiểm tra hợp lệ dữ liệu.
  + Tổ chức luồng xử lý giữa Presentation và Data Layer.
* Thành phần trong CoolStore:
  + Product Service: xử lý duyệt, lọc, tìm kiếm sản phẩm.
  + ShoppingCart Service: quản lý thêm/xóa/cập nhật sản phẩm trong giỏ, tính toán chi phí, khuyến mãi.
  + Payment Service: xử lý thanh toán, kết nối cổng thanh toán, gửi email xác nhận.
  + Inventory Service: cập nhật số lượng tồn kho.
  + Rating Service: xử lý thêm đánh giá sản phẩm, tính trung bình điểm.
  + Access Control Service: quản lý đăng nhập, phân quyền Buyer và SysAdmin.
* Ví dụ: Khi Buyer nhấn “Thanh toán”, lớp này sẽ:

1. Kiểm tra tồn kho.
2. Tính tổng đơn hàng + khuyến mãi.
3. Gọi Payment Gateway.
4. Gửi kết quả cho Presentation Layer để hiển thị.

3. Data Layer (Lớp dữ liệu / Kho dữ liệu)

* Chức năng: Lưu trữ và truy xuất dữ liệu.
* Cách tổ chức:
  + Database (SQL/NoSQL):
  + Product Table (thông tin sản phẩm)
  + ShoppingCart Table, CartItem Table (giỏ hàng)
  + Order Table, Payment Table (thanh toán, đơn hàng)
  + Inventory Table (tồn kho)
  + Rating Table (đánh giá)
  + User Table, Role Table (người dùng & phân quyền)
* Data Access Layer (DAL / Repository): trung gian kết nối giữa Business Layer và DB.

1. **3 test case cho chức năng “Thêm sản phẩm vào giỏ hàng”**

Test Case 1: Thêm sản phẩm hợp lệ vào giỏ

* Mã TC: TC\_AddToCart\_01
* Mô tả: Người dùng thêm một sản phẩm có sẵn trong kho vào giỏ hàng.
* Dữ liệu đầu vào:
  + User đã đăng nhập
  + ProductID = 101 (Áo thun)
  + Số lượng = 2 (tồn kho hiện tại = 10)
* Bước thực hiện:
  1. Người dùng truy cập trang chi tiết sản phẩm “Áo thun”.
  2. Nhập số lượng = 2.
  3. Nhấn nút “Thêm vào giỏ hàng”.
* Kết quả mong đợi:
  + Hệ thống hiển thị thông báo “Đã thêm sản phẩm vào giỏ hàng thành công”.
  + Trong giỏ hàng xuất hiện sản phẩm “Áo thun” với số lượng = 2.
  + Tồn kho giảm từ 10 → 8.

Test Case 2: Thêm sản phẩm với số lượng vượt quá tồn kho

* Mã TC: TC\_AddToCart\_02
* Mô tả: Người dùng cố gắng thêm số lượng lớn hơn số lượng còn trong kho.
* Dữ liệu đầu vào:
  + User đã đăng nhập
  + ProductID = 202 (Giày thể thao)
  + Số lượng = 15 (tồn kho hiện tại = 10)
* Bước thực hiện:
  1. Người dùng truy cập trang chi tiết sản phẩm “Giày thể thao”.
  2. Nhập số lượng = 15.
  3. Nhấn nút “Thêm vào giỏ hàng”.
* Kết quả mong đợi:
  + Hệ thống hiển thị thông báo lỗi “Số lượng vượt quá tồn kho, vui lòng giảm số lượng”.
  + Không thêm sản phẩm vào giỏ.
  + Tồn kho không thay đổi (vẫn = 10).

Test Case 3: Thêm sản phẩm khi chưa đăng nhập

* Mã TC: TC\_AddToCart\_03
* Mô tả: Người dùng chưa đăng nhập nhưng thực hiện thao tác thêm sản phẩm.
* Dữ liệu đầu vào:
  + User chưa đăng nhập
  + ProductID = 303 (Balo)
  + Số lượng = 1 (tồn kho hiện tại = 5)
* Bước thực hiện:
  1. Người dùng truy cập trang sản phẩm “Balo”.
  2. Nhấn nút “Thêm vào giỏ hàng”.
* Kết quả mong đợi:
  + Hệ thống chuyển hướng đến trang Đăng nhập hoặc hiển thị popup yêu cầu đăng nhập.
  + Sản phẩm không được thêm vào giỏ hàng cho đến khi đăng nhập thành công.
  + Tồn kho không thay đổi.