TRƯỜNG ĐẠI HỌC THỦY LỢI

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO BÀI TẬP LỚN MÔN HỌC CÁC VẤN ĐỀ TIÊN TIẾN TRONG PHÁT TRIỂN PHẦN MỀM**

**Ứng dụng Design Pattern vào phát triển hệ thống website Quản lý sinh viên TLU**

Nhóm: 7

| Họ và tên Sinh viên: | Ngô Thị Ngọc Ánh - 2251172244 | |
| --- | --- | --- |
| Bùi Viết Hiển - 2251172339 | |
| Trần Hồng Quang - 2251172467 | |
| Lê Sỹ Thắng - 2251172491 | |

Lớp: 64KTPM3

Giảng viên phụ trách môn học**:** TS. Cù Việt Dũng

**Hà Nội - 5/2025**

[**LỜI MỞ ĐẦU 5**](#_x2d2a8jt8sg6)

[**PHẦN 1: GIỚI THIỆU CHUNG 6**](#_tkea5w8uhrmk)

[1.1. Lý do chọn đề tài 6](#_sgirbaps80xx)

[1.2. Mục tiêu đề tài 6](#_adj8b02o5gcc)

[1.3. Phạm vi thực hiện 6](#_8rrpucsi7117)

[1.4. Phương pháp nghiên cứu 6](#_v4x6u8tkcxr8)

[**PHẦN 2: PHÂN TÍCH HỆ THỐNG 8**](#_r9re13zd8zbo)

[2.1. Bài toán được xây dựng 8](#_5b4nevugjolp)

[2.2. Các chức năng chính của hệ thống 8](#_tr3zpqjxwmrc)

[2.3. Biểu đồ Use Case 9](#_396y0s8b30f7)

[2.4. Mô tả Use Case chính 10](#_brpnre28zx33)

[2.4.1. Mô tả Use Case Quản lý sinh viên 10](#_2z9qzyqm3atw)

[2.4.1.1 Use case Thêm sinh viên 10](#_u07nsde7vivu)

[2.4.1.1. Sửa sinh viên 11](#_1q3noruvtnk9)

[2.4.1.3. Xóa sinh viên 12](#_ekyxkk66xuvv)

[2.4.2. Mô tả Use Case Quản lý giảng viên 13](#_99153u5v91m6)

[2.4.3. Mô tả Use Case Quản lý môn học 19](#_f0xfjqt7ghta)

[2.4.3.1. Use case Thêm môn học 19](#_h0hdn9rfitcx)

[2.4.3.2. Sửa môn học 21](#_ckksgxettqeo)

[2.4.3.3. Xem môn học 22](#_767fxykbut3e)

[2.4.3.4. Tìm kiếm môn học 23](#_s6swrt6b1a9j)

[2.4.4. Mô tả Use Case Quản lý điểm 24](#_lm5zo7tbtgkk)

[2.4.4.1. Use Case Nhập điểm 24](#_yarbk8mxm0tb)

[2.4.4.2. Use Case Sửa điểm 25](#_uemxhkhfjwl)

[**PHẦN 3: THIẾT KẾ HỆ THỐNG DESIGN PATTERN 30**](#_47hkiplv24qh)

[3.1. Tổng quan kiến trúc hệ thống 30](#_9leqsatk2yp1)

[3.1.1. Mô hình kiến trúc tổng thể 30](#_j55f9ajwxknb)

[3.1.2. Tương tác giữa các thành phần 30](#_d4i5d6jow73o)

[3.1.3. Ứng dụng vào Design Pattern 30](#_80azvcgutisj)

[3.2. Các Design Pattern được áp dụng 31](#_ibd67rjt2xon)

[3.2.1. Observer 31](#_ink0sr3ufs5t)

[3.2.1.1. Khái niệm 31](#_eii5dlna76j)

[3.2.1.2. Cấu trúc 31](#_bmbs739rl0j9)

[3.2.1.3. Ưu và nhược điểm 31](#_gs58tpblogst)

[3.2.2. Facade 31](#_9x8hpn5qfjgc)

[3.2.2.1. Khái niệm 31](#_b7wnsl10wqqv)

[3.2.2.2. Cấu trúc 31](#_pncwja3zbdsm)

[3.2.2.3. Ưu và nhược điểm 32](#_je0voix1jp1h)

[3.2.3. Decorator 32](#_h749ve4sau7b)

[3.2.3.1. Khái niệm 32](#_coy3ivhoy5ld)

[3.2.3.2. Cấu trúc 32](#_9rwih6itj65u)

[3.2.3.3. Ưu và nhược điểm 32](#_h0zyhfp8et54)

[3.2.4. Repository 32](#_h5jxek47duwq)

[3.2.4.1. Khái niệm 32](#_lpbtuzhylqtu)

[3.2.4.2. Cấu trúc 33](#_89gx3z4ae7uh)

[3.2.4.3. Ưu và nhược điểm 33](#_xamsdf3l5rd7)

[3.2.5. Adapter 33](#_wyyv4eieaq7i)

[3.2.5.1. Khái niệm 33](#_ndkqd1uzx8bl)

[3.2.5.2. Cấu trúc 33](#_chc5eylbtzsn)

[3.2.5.3. Ưu và nhược điểm 34](#_gw0yor8qnaqa)

[3.2.6. Builder 34](#_g1mphvfev8c0)

[3.2.6.1. Khái niệm 34](#_twfhelnx3wk3)

[3.2.6.2. Cấu trúc 34](#_dianlirg6e5z)

[3.2.6.3. Ưu và nhược điểm 34](#_xxzi94u0moqv)

[3.2.7. Abstract 35](#_2mr1xm6o408v)

[3.2.7.1. Khái niệm 35](#_kidbhhe942zs)

[3.2.7.2. Cấu trúc 35](#_7420bznri963)

[3.2.7.3. Ưu và nhược điểm 35](#_1k57qyirynn8)

[3.2.8. Factory Method 36](#_ejxe8ykcpoq5)

[3.2.8.1. Khái niệm 36](#_zc8nxipygevv)

[3.2.8.2. Cấu trúc 36](#_19yb394c8m21)

[3.2.8.3. Ưu và nhược điểm 36](#_3l6nzwz22uw)

[3.2.9. Singleton 36](#_1tas56tbsg1r)

[3.2.9.1. Khái niệm 36](#_488k3znad4cg)

[3.2.9.2. Cấu trúc 36](#_cqbqq35o76w3)

[3.2.9.3. Ưu và nhược điểm 37](#_2ruzejp602c)

[3.3. Giải thích lý do chọn mỗi mẫu 37](#_h7f538lbn2h8)

[3.4. Biểu đồ minh họa cho mỗi Design Pattern 39](#_gun04ursu88o)

[3.4.1. Observer 39](#_o8sa45bir0wl)

[3.4.2. Facade 39](#_jvqravbpg8o7)

[3.4.3. Decorator 40](#_lniv49ofvvxb)

[3.4.4. Repository 41](#_aqet1fz9zjov)

[3.4.5. Adapter 42](#_5hmkjk1e8whs)

[3.4.6. Builder 43](#_e88o7xxi46gy)

[3.4.7. Abstract 44](#_l41quv9o3uwh)

[3.4.8. Factory Method 45](#_fshal9bvdqwq)

[3.4.9. Singleton 46](#_w7kg6du3y0h6)

[3.5. Mối quan hệ giữa các pattern trong hệ thống 46](#_qszj42mae1tj)

[3.5.1. Mối quan hệ giữa Singleton, Decorator và Repository 46](#_co8tkbn28o42)

[3.5.2. Mối quan hệ giữa Facade, Service và Builder 47](#_1trw9vpek4p)

[3.5.3. Mối quan hệ giữa Observer và Grade Service 47](#_bhaowprcxul4)

[3.5.4. Mối quan hệ giữa Factory Method và Abstract Factory 47](#_vzed073u5ph6)

[3.5.5. Mối quan hệ giữa Adapter và Builder 47](#_i4eppsgx0jcy)

[**KẾT LUẬN 48**](#_q5al8v9yni98)

# LỜI MỞ ĐẦU

Trong bối cảnh công nghệ thông tin ngày càng phát triển mạnh mẽ, việc ứng dụng các phương pháp và kỹ thuật hiện đại vào quá trình phát triển phần mềm là điều tất yếu nhằm nâng cao chất lượng, tính linh hoạt và khả năng bảo trì của hệ thống. Đặc biệt, trong lĩnh vực giáo dục, nhu cầu xây dựng các hệ thống quản lý trực tuyến nhằm tối ưu hóa công tác quản lý sinh viên, giảng viên, môn học và kết quả học tập là vô cùng cần thiết.

Môn học “Các vấn đề tiên tiến trong phát triển phần mềm” đã trang bị cho sinh viên những kiến thức chuyên sâu về thiết kế và phát triển hệ thống phần mềm theo hướng hiện đại, trong đó có việc sử dụng các mẫu thiết kế phần mềm (Design Pattern). Đây là một hướng tiếp cận hiệu quả nhằm chuẩn hóa quy trình lập trình, tăng tính tái sử dụng mã nguồn và đảm bảo cấu trúc hệ thống rõ ràng.

Trong khuôn khổ môn học, nhóm chúng em đã thực hiện đề tài: **“Áp dụng Design Pattern vào phát triển hệ thống website Quản lý sinh viên TLU”**. Báo cáo này trình bày toàn bộ quá trình thực hiện đề tài từ khâu phân tích yêu cầu, thiết kế hệ thống đến triển khai và đánh giá kết quả. Thông qua đề tài, nhóm mong muốn áp dụng hiệu quả kiến thức đã học để xây dựng một hệ thống thực tế, đồng thời nâng cao kỹ năng làm việc nhóm, tư duy thiết kế và lập trình hướng đối tượng.

Nhóm xin chân thành cảm ơn thầy **TS. Cù Việt Dũng** đã tận tình hướng dẫn trong suốt quá trình thực hiện đề tài. Mặc dù đã cố gắng hoàn thiện, nhưng do giới hạn về thời gian và kinh nghiệm, báo cáo không tránh khỏi những thiếu sót. Nhóm rất mong nhận được sự góp ý, nhận xét từ thầy để hoàn thiện hơn trong các dự án tương lai.

# 

# PHẦN 1: GIỚI THIỆU CHUNG

## 1.1. Lý do chọn đề tài

Trong thời đại công nghệ thông tin phát triển mạnh mẽ hiện nay, việc quản lý sinh viên bằng phương pháp truyền thống (sổ sách, Excel rời rạc...) không còn phù hợp và dễ dẫn đến sai sót, khó bảo trì. Việc xây dựng một hệ thống quản lý sinh viên theo hướng hiện đại, sử dụng các mẫu thiết kế (Design Pattern) trong phát triển phần mềm là một hướng đi đúng đắn và hiệu quả.

Design Pattern không chỉ giúp phần mềm có cấu trúc rõ ràng, dễ bảo trì và mở rộng, mà còn nâng cao khả năng tái sử dụng mã nguồn. Do đó, nhóm chúng em lựa chọn đề tài “Áp dụng Design Pattern vào phát triển hệ thống website Quản lý sinh viên TLU” để nghiên cứu và thực hành những kỹ thuật tiên tiến trong phát triển phần mềm.

## 1.2. Mục tiêu đề tài

Xây dựng một hệ thống quản lý sinh viên có giao diện web, giúp nhà trường và giảng viên dễ dàng quản lý thông tin sinh viên, giảng viên, môn học và điểm số.

Áp dụng các mẫu thiết kế phần mềm (Design Pattern) nhằm tăng tính hiệu quả, khả năng mở rộng và bảo trì của hệ thống.

Nâng cao kỹ năng lập trình, tư duy thiết kế phần mềm cho các thành viên trong nhóm thông qua việc áp dụng lý thuyết vào thực tế.

## 1.3. Phạm vi thực hiện

Đề tài tập trung vào các chức năng chính của hệ thống quản lý sinh viên, bao gồm:

* Quản lý thông tin sinh viên
* Quản lý giảng viên
* Quản lý môn học
* Quản lý điểm

Website được xây dựng với quy mô dành cho nội bộ trường Đại học Thủy Lợi, không triển khai các tính năng nâng cao như hệ thống xác thực nhiều lớp, phân tích học lực, hay tích hợp API bên ngoài.

## 1.4. Phương pháp nghiên cứu

**Tìm hiểu lý thuyết**: Nghiên cứu tài liệu về Design Pattern và các kiến trúc phần mềm hiện đại.

**Phân tích hệ thống**: Phân tích yêu cầu nghiệp vụ và chức năng cần có trong một hệ thống quản lý sinh viên.

**Thiết kế hệ thống**: Áp dụng các mẫu thiết kế phù hợp để xây dựng kiến trúc hệ thống.

**Lập trình và triển khai**: Phát triển hệ thống dựa trên ngôn ngữ lập trình và framework phù hợp.

**Kiểm thử và đánh giá**: Thử nghiệm hệ thống và đánh giá hiệu quả áp dụng Design Pattern.

# PHẦN 2: PHÂN TÍCH HỆ THỐNG

## 2.1. Bài toán được xây dựng

Hiện nay, nhiều trường đại học vẫn đang quản lý thông tin sinh viên, giảng viên và kết quả học tập bằng các công cụ truyền thống như Excel hoặc phần mềm đơn giản, thiếu tính kết nối và khó mở rộng. Điều này gây ra nhiều khó khăn trong việc cập nhật, tra cứu và đảm bảo tính nhất quán của dữ liệu.

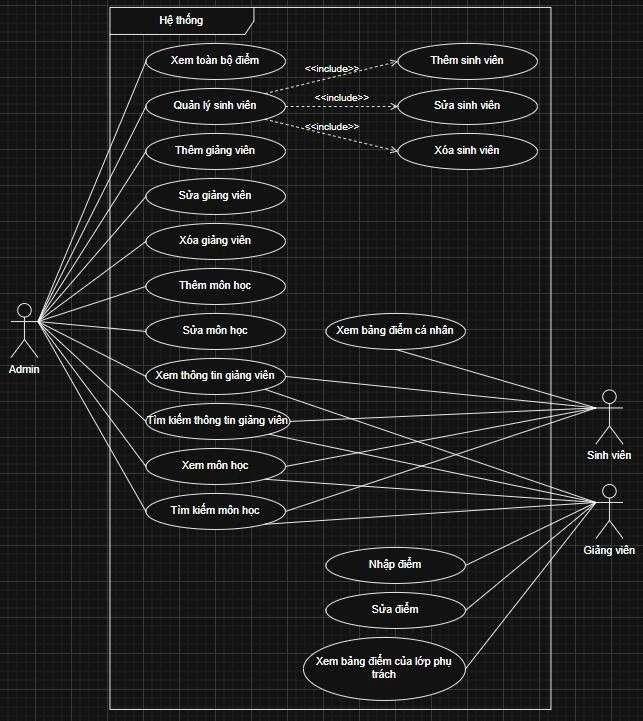
Bài toán đặt ra là cần xây dựng một hệ thống quản lý sinh viên dưới dạng website, cho phép thực hiện các thao tác cơ bản như thêm/sửa/xóa sinh viên, giảng viên, môn học, nhập điểm và xem kết quả học tập. Hệ thống phải được thiết kế theo hướng module hóa, áp dụng Design Pattern để đảm bảo dễ mở rộng, dễ bảo trì và có thể tích hợp thêm các tính năng khác trong tương lai.

## 2.2. Các chức năng chính của hệ thống

Hệ thống website quản lý sinh viên bao gồm các chức năng chính sau:

* **Quản lý sinh viên:** Thêm mới, chỉnh sửa, xóa thông tin sinh viên; tìm kiếm và liệt kê danh sách sinh viên.
* **Quản lý giảng viên:** Quản lý thông tin giảng viên giảng dạy, bao gồm thông tin cá nhân và môn học phụ trách.
* **Quản lý môn học:** Thêm môn học, sửa môn học, xem môn học, tìm kiếm môn học
* **Quản lý điểm:** Nhập điểm cho sinh viên theo từng môn học; cập nhật lại điểm cho sinh viên trong trường hợp sai sót, xem bảng điểm theo lớp hoặc sinh viên

## 2.3. Biểu đồ Use Case



## 2.4. Mô tả Use Case chính

### 2.4.1. Mô tả Use Case Quản lý sinh viên

#### 2.4.1.1 Use case Thêm sinh viên

| Use Case ID | 01 | | |
| --- | --- | --- | --- |
| Use Case Name | Thêm sinh viên | | |
| Created By: | Ngô Thị Ngọc Ánh | Last Updated By: | Ngô Thị Ngọc Ánh |
| Date Created: | 16/05/2025 | Date Last Updated: |  |
| Actor () | Người dùng (Admin) | | |
| Description (mô tả) | Cho phép quản trị viên thêm mới một sinh viên vào hệ thống. | | |
| Trigger (kích hoạt) | Người dùng phải có tài khoản đăng nhập vào hệ thống.  Người dùng phải có quyền quản trị Thêm sinh viên vào hệ thống.  Người dùng nhấn vào nút “Thêm sinh viên” trên giao diện quản lý sinh viên | | |
| Pre-conditions (điều kiện trước) | Đã đăng nhập với vai trò quản trị viên.  Trang quản lý sinh viên đã sẵn sàng. | | |
| Post conditions (điều kiện sau) | Sinh viên mới được lưu vào cơ sở dữ liệu.  Danh sách sinh viên được cập nhật. | | |
| Yêu cầu dữ liệu đầu vào | * Mã sinh viên * Họ tên * Ngày sinh * Giới tính * Lớp * Ngành học * Quê quán * Email, số điện thoại (nếu có) | | |
| Flow of Events (luồng sự kiện) | 1. Người dùng đăng nhập thành công vào trang chủ hệ thống website quản lý sinh viên TLU. Hệ thống hiển thị giao diện trang chủ. 2. Người dùng truy cập vào trang quản lý sinh viên TLU. Hệ thống hiển thị giao diện trang quản lý sinh viên 3. Người dùng nhấn chọn nút “Thêm sinh viên”. Hệ thống hiển thị form sẵn sàng nhập liệu bao gồm các trường nhập. 4. Người dùng nhập đầy đủ thông tin bắt buộc để thêm sinh viên mới. 5. Người dùng nhấn nút “Lưu”. Hệ thống kiểm tra tính hợp lệ, nếu có lỗi xảy ra luồng ngoại lệ **E-1** sẽ thực thi. 6. Nếu hệ thống kiểm tra dữ liệu hợp lệ hệ thống thông báo “Thêm sinh viên mới thành công”. 7. Sinh viên mới được lưu trữ dữ liệu vào hệ thông và hiển thị trên danh sách sinh viên. | | |
| Luồng con | **A-1: Xác nhận thêm sinh viên mới.**  - Người dùng nhấn nút Lưu. Luồng sự kiện sẽ sang bước tiếp theo. Nếu nhấn nút Hủy hệ thống hủy thao tác thêm sinh viên và dữ liệu sẽ không được lưu. | | |
| Exceptions (Ngoại lệ) | **E-1: Người dùng bỏ trống các trường nhập liệu bắt buộc**  Người dùng bỏ trống 1 trường bắt buộc hệ thống sẽ hiển thị thông báo lỗi “Vui lòng nhập đầy đủ thông tin.”. | | |

#### **2.4.1.1. Sửa sinh viên**

| Use Case ID | 02 | | |
| --- | --- | --- | --- |
| Use Case Name | Sửa sinh viên | | |
| Created By: | Ngô Thị Ngọc Ánh | Last Updated By: | Ngô Thị Ngọc Ánh |
| Date Created: | 16/05/2025 | Date Last Updated: |  |
| Actor () | Người dùng (Admin) | | |
| Description (mô tả) | Cho phép quản trị viên chỉnh sửa thông tin một sinh viên trên hệ thống quản lý. (Không được sửa mã sinh viên) | | |
| Trigger (kích hoạt) | Người dùng phải có tài khoản đăng nhập vào hệ thống.  Người dùng phải có quyền quản trị sửa sinh viên vào hệ thống.  Người dùng nhấn vào nút “sửa sinh viên” trên giao diện quản lý sinh viên | | |
| Pre-conditions (điều kiện trước) | Đã đăng nhập với vai trò quản trị viên.  Trang quản lý sinh viên đã sẵn sàng. | | |
| Post conditions (điều kiện sau) | Sinh viên mới được lưu vào cơ sở dữ liệu.  Danh sách sinh viên được cập nhật. | | |
| Yêu cầu dữ liệu đầu vào | * Họ tên * Ngày sinh * Giới tính * Lớp * Ngành học * Quê quán * Email, số điện thoại (nếu có) | | |
| Flow of Events (luồng sự kiện) | 1. Người dùng đăng nhập thành công vào trang chủ hệ thống website quản lý sinh viên TLU. Hệ thống hiển thị giao diện trang chủ. 2. Người dùng truy cập vào trang quản lý sinh viên TLU. Hệ thống hiển thị giao diện trang quản lý sinh viên. 3. Người dùng nhấn chọn nút “sửa sinh viên”. Hệ thống hiển thị form sẵn sàng nhập liệu bao gồm các trường nhập. 4. Người dùng nhập đầy đủ thông tin bắt buộc để sửa sinh viên mới. 5. Người dùng nhấn nút “Lưu”. Hệ thống kiểm tra tính hợp lệ, nếu có lỗi xảy ra luồng ngoại lệ **E-1** sẽ thực thi. 6. Nếu hệ thống kiểm tra dữ liệu hợp lệ hệ thống thông báo xác nhận “Xác nhận lưu và chỉnh sửa sinh viên”. 7. Người dùng nhấn “xác nhận”. Hệ thống thông báo “Sửa sinh viên thành công” 8. Sinh viên mới được lưu trữ dữ liệu vào hệ thông và hiển thị trên danh sách sinh viên. | | |
| Luồng con | **A-1: Xác nhận sửa sinh viên mới.**  - Người dùng nhấn nút Lưu. Luồng sự kiện sẽ sang bước tiếp theo. Nếu nhấn nút Hủy hệ thống hủy thao tác sửa sinh viên và dữ liệu sẽ không được lưu. | | |
| Exceptions (Ngoại lệ) | **E-1: Người dùng bỏ trống các trường nhập liệu bắt buộc**  Người dùng bỏ trống 1 trường bắt buộc hệ thống sẽ hiển thị thông báo lỗi “Vui lòng nhập đầy đủ thông tin.”. | | |

### 2.4.1.3. Xóa sinh viên

| Use Case ID | 03 | | |
| --- | --- | --- | --- |
| Use Case Name | Xóa sinh viên | | |
| Created By: | Ngô Thị Ngọc Ánh | Last Updated By: | Ngô Thị Ngọc Ánh |
| Date Created: | 16/05/2025 | Date Last Updated: |  |
| Actor () | Người dùng (Admin) | | |
| Description (mô tả) | Chức năng cho phép người dùng xóa thông tin sinh viên khỏi hệ thống quản lý. | | |
| Trigger (kích hoạt) | Người dùng nhấn nút “Xóa sinh viên” trên giao diện quản lý. | | |
| Pre-conditions (điều kiện trước) | Người dùng phải có tài khoản đăng nhập vào hệ thống quản lý và có quyền quản trị Xóa sinh viên | | |
| Post conditions (điều kiện sau) | thông tin sinh viên được xóa khỏi hệ thống và danh sách sinh vi | | |
| Yêu cầu dữ liệu đầu vào | Không có | | |
| Flow of Events (luồng sự kiện) | 1. Người dùng đăng nhập thành công vào trang chủ hệ thống website quản lý sinh viên TLU. Hệ thống hiển thị giao diện trang chủ. 2. Người dùng truy cập vào trang quản lý sinh viên. Hệ thống hiển thị giao diện trang quản lý sinh viên. 3. Người dùng nhấn nút “Xóa thông tin sinh viên” trên giao diện quản lý sinh viên. 4. Hệ thống kiểm tra dữ liệu, hệ thống thông báo “Xác nhận xóa thông tin sinh viên?”. 5. Người dùng nhấn nút “Xác nhận” để xóa thông tin sinh viên 6. Hệ thống thông báo “Thông sinh viên đã được xóa.” 7. Hệ thống xóa thông tin sinh viên khỏi danh sách và dữ liệu hệ thống quản lý. | | |
| Luồng con | **A-1: Xác nhận xóa thông tin sinh viên mới.**  - Người dùng nhấn nút “Xác nhận”. Luồng sự kiện sẽ sang bước tiếp theo. Nếu nhấn nút Hủy hệ thống hủy thao tác xóa thông tin sinh viên và dữ liệu sẽ không bị xóa. | | |
| Exceptions (Ngoại lệ) | Không có. | | |

### 2.4.2. Mô tả Use Case Quản lý giảng viên

***2.4.2.1. Use Case Thêm thông tin giảng viên***

| Use Case ID | 04 | | |
| --- | --- | --- | --- |
| Use Case Name | Thêm thông tin giảng viên | | |
| Created By: | Lê Sỹ Thắng | Last Updated By: | Lê Sỹ Thắng |
| Date Created: | 16/05/2025 | Date Last Updated: | 16/05/2025 |
| Actor () | Người dùng (Admin) | | |
| Description (mô tả) | Chức năng cho phép người dùng thêm một giảng viên mới vào hệ thống quản lý. | | |
| Trigger (kích hoạt) | Người dùng nhấn nút “Thêm giảng viên” trên giao diện quản lý. | | |
| Pre-conditions (điều kiện trước) | Người dùng phải có tài khoản đăng nhập vào hệ thống quản lý và có quyền quản trị thêm giảng viên | | |
| Post conditions (điều kiện sau) | Một giảng viên mới được thêm vào hệ thống quản lý và hiển thị thông tin giảng viên lên danh sách | | |
| Yêu cầu dữ liệu đầu vào | Thông tin giảng viên   * Họ tên giảng viên * Ngày sinh * Giới tính * Email * Sô điện thoại * Khoa phụ trách | | |
| Flow of Events (luồng sự kiện) | 1. Người dùng đăng nhập thành công vào trang chủ hệ thống website quản lý sinh viên TLU. Hệ thống hiển thị giao diện trang chủ. 2. Người dùng truy cập vào trang quản lý giảng viên. Hệ thống hiển thị giao diện trang quản lý giảng viên. 3. Người dùng nhấn nút “Thêm giảng viên” trên giao diện quản lý giảng viên. Hệ thống hiển form sẵn sàng nhập liệu. 4. Người dùng nhập đầy đủ thông tin bắt buộc để thêm giảng viên mới. 5. Người dùng nhấn nút “Lưu”. Hệ thống kiểm tra tính hợp lệ, nếu có lỗi xảy ra luồng ngoại lệ **E-1** sẽ thực thi. 6. Nếu hệ thống kiểm tra dữ liệu hợp lệ hệ thống thông báo “Thêm giảng viên thành công”. 7. Giảng viên mới được lưu trữ dữ liệu vào hệ thống và hiển thị trên danh sách giảng viên | | |
| Luồng con | **A-1: Xác nhận thêm giảng viên mới.**  - Người dùng nhấn nút Lưu. Luồng sự kiện sẽ sang bước tiếp theo. Nếu nhấn nút Hủy hệ thống hủy thao tác thêm giảng viên và dữ liệu sẽ không được lưu. | | |
| Exceptions (Ngoại lệ) | **E-1: Người dùng bỏ trống các trường nhập liệu bắt buộc**  Người dùng bỏ trống 1 trường bắt buộc hệ thống sẽ hiển thị thông báo lỗi “Vui lòng nhập đầy đủ thông tin.”.  **E-2: Người dùng nhập thông tin không hợp lệ**  Người dùng nhập thông tin không hợp như Email, Số điện thoại… hệ thống hiển thị cảnh báo cụ thể vào các trường nhập liệu “Dữ liệu không hợp lệ”. | | |

***2.4.2.2. Use Case Sửa thông tin giảng viên***

| Use Case ID | 05 | | |
| --- | --- | --- | --- |
| Use Case Name | Sửa thông tin giảng viên | | |
| Created By: | Lê Sỹ Thắng | Last Updated By: | Lê Sỹ Thắng |
| Date Created: | 16/05/2025 | Date Last Updated: | 16/05/2025 |
| Actor () | Người dùng (Admin) | | |
| Description (mô tả) | Chức năng cho phép người dùng chỉnh sửa thông tin giảng viên trên hệ thống quản lý. ( Không được phép sửa địa chỉ Email) | | |
| Trigger (kích hoạt) | Người dùng nhấn nút “Sửa thông tin giảng viên” trên giao diện quản lý. | | |
| Pre-conditions (điều kiện trước) | Người dùng phải có tài khoản đăng nhập vào hệ thống quản lý và có quyền quản trị sửa thông tin giảng viên | | |
| Post conditions (điều kiện sau) | Thông tin giảng viên được chỉnh sửa lưu vào hệ thống và hiển thị trên danh sách giảng viên | | |
| Yêu cầu dữ liệu đầu vào | Thông tin giảng viên   * Họ tên giảng viên * Ngày sinh * Giới tính * Sô điện thoại * Khoa phụ trách | | |
| Flow of Events (luồng sự kiện) | 1. Người dùng đăng nhập thành công vào trang chủ hệ thống website quản lý sinh viên TLU. Hệ thống hiển thị giao diện trang chủ. 2. Người dùng truy cập vào trang quản lý giảng viên. Hệ thống hiển thị giao diện trang quản lý giảng viên. 3. Người dùng nhấn nút “Sửa thông tin giảng viên” trên giao diện quản lý giảng viên. Hệ thống hiển form sẵn sàng nhập liệu. 4. Người dùng nhập đầy đủ thông tin bắt buộc để sửa thông tin giảng viên mới. 5. Người dùng nhấn nút “Lưu”. Hệ thống kiểm tra tính hợp lệ, nếu có lỗi xảy ra luồng ngoại lệ **E-1** sẽ thực thi. 6. Nếu hệ thống kiểm tra dữ liệu hợp lệ hệ thống thông báo “Xác nhận lưu và chỉnh sửa thông tin giảng viên”. 7. Người dùng nhấn nút “Xác nhận” để sửa thông tin giảng viên 8. Thông tin giảng viên mới được lưu trữ dữ liệu vào hệ thống và hiển thị trên danh sách giảng viên | | |
| Luồng con | **A-1: Xác nhận sửa thông tin giảng viên mới.**  - Người dùng nhấn nút Lưu. Luồng sự kiện sẽ sang bước tiếp theo. Nếu nhấn nút Hủy hệ thống hủy thao tác sửa thông tin giảng viên và dữ liệu sẽ không được lưu. | | |
| Exceptions (Ngoại lệ) | **E-1: Người dùng bỏ trống các trường nhập liệu bắt buộc**  Người dùng bỏ trống 1 trường bắt buộc hệ thống sẽ hiển thị thông báo lỗi “Vui lòng nhập đầy đủ thông tin.”.  **E-2: Người dùng nhập thông tin không hợp lệ**  Người dùng nhập thông tin không hợp như Email, Số điện thoại… hệ thống hiển thị cảnh báo cụ thể vào các trường nhập liệu “Dữ liệu không hợp lệ”. | | |

***2.4.2.3. Use Case Xóa thông tin giảng viên***

| Use Case ID | 06 | | |
| --- | --- | --- | --- |
| Use Case Name | Xóa thông tin giảng viên | | |
| Created By: | Lê Sỹ Thắng | Last Updated By: | Lê Sỹ Thắng |
| Date Created: | 16/05/2025 | Date Last Updated: | 16/05/2025 |
| Actor () | Người dùng (Admin) | | |
| Description (mô tả) | Chức năng cho phép người dùng xóa thông tin giảng viên khỏi hệ thống quản lý. | | |
| Trigger (kích hoạt) | Người dùng nhấn nút “Xóa thông tin giảng viên” trên giao diện quản lý. | | |
| Pre-conditions (điều kiện trước) | Người dùng phải có tài khoản đăng nhập vào hệ thống quản lý và có quyền quản trị Xóa thông tin giảng viên | | |
| Post conditions (điều kiện sau) | Thông tin giảng viên được xóa khỏi hệ thống và danh sách giảng viên. | | |
| Yêu cầu dữ liệu đầu vào | Không có | | |
| Flow of Events (luồng sự kiện) | 1. Người dùng đăng nhập thành công vào trang chủ hệ thống website quản lý sinh viên TLU. Hệ thống hiển thị giao diện trang chủ. 2. Người dùng truy cập vào trang quản lý giảng viên. Hệ thống hiển thị giao diện trang quản lý giảng viên. 3. Người dùng nhấn nút “Xóa thông tin giảng viên” trên giao diện quản lý giảng viên. 4. Hệ thống kiểm tra dữ liệu, hệ thống thông báo “Xác nhận xóa thông tin giảng viên?”. 5. Người dùng nhấn nút “Xác nhận” để xóa thông tin giảng viên 6. Hệ thống thông báo “Thông tin giảng viên đã được xóa.” 7. Hệ thống xóa thông tin giảng viên khỏi danh sách và dữ liệu hệ thống quản lý. | | |
| Luồng con | **A-1: Xác nhận xóa thông tin giảng viên mới.**  - Người dùng nhấn nút “Xác nhận”. Luồng sự kiện sẽ sang bước tiếp theo. Nếu nhấn nút Hủy hệ thống hủy thao tác xóa thông tin giảng viên và dữ liệu sẽ không bị xóa. | | |
| Exceptions (Ngoại lệ) | Không có. | | |

***2.4.2.4. Use Case Xem thông tin giảng viên***

| Use Case ID | 07 | | |
| --- | --- | --- | --- |
| Use Case Name | Xem thông tin giảng viên | | |
| Created By: | Lê Sỹ Thắng | Last Updated By: | Lê Sỹ Thắng |
| Date Created: | 16/05/2025 | Date Last Updated: | 16/05/2025 |
| Actor () | Người dùng (Admin, Giảng viên, Sinh viên) | | |
| Description (mô tả) | Chức năng cho phép người dùng xem chi tiết thông tin giảng viên trên hệ thống quản lý. | | |
| Trigger (kích hoạt) | Người dùng nhấn nút “Xem thông tin giảng viên” trên giao diện quản lý. | | |
| Pre-conditions (điều kiện trước) | Người dùng phải có tài khoản đăng nhập vào hệ thống quản lý và có quyền xem thông tin giảng viên | | |
| Post conditions (điều kiện sau) | Hệ thống hiển thị giao diện xem thông tin giảng viên. | | |
| Yêu cầu dữ liệu đầu vào | Không có | | |
| Flow of Events (luồng sự kiện) | 1. Người dùng đăng nhập thành công vào trang chủ hệ thống website quản lý sinh viên TLU. Hệ thống hiển thị giao diện trang chủ. 2. Người dùng truy cập vào trang quản lý giảng viên. Hệ thống hiển thị giao diện trang quản lý giảng viên. 3. Người dùng nhấn nút “Xem thông tin giảng viên” trên giao diện quản lý giảng viên. 4. Hệ thống hiển thị giao diện chi tiết thông tin giảng viên cần được xem. | | |
| Luồng con | Không có | | |
| Exceptions (Ngoại lệ) | Không có | | |

***2.4.2.5. Use Case Tìm kiếm thông tin giảng viên***

| Use Case ID | 08 | | |
| --- | --- | --- | --- |
| Use Case Name | Tìm kiếm thông tin giảng viên | | |
| Created By: | Lê Sỹ Thắng | Last Updated By: | Lê Sỹ Thắng |
| Date Created: | 16/05/2025 | Date Last Updated: | 16/05/2025 |
| Actor () | Người dùng (Admin, Giảng viên, Sinh viên) | | |
| Description (mô tả) | Chức năng cho phép người dùng thực hiện tìm kiếm giảng viên trên hệ thống quản lý. | | |
| Trigger (kích hoạt) | Người dùng nhập từ khóa điều kiện cần tìm kiếm thông tin giảng viên và thực hiện thao tác tìm kiếm thông tin giảng viên trên hệ thống quản lý. | | |
| Pre-conditions (điều kiện trước) | Người dùng phải có tài khoản đăng nhập vào hệ thống quản lý và có quyền tìm kiếm thông tin giảng viên | | |
| Post conditions (điều kiện sau) | Hệ thống hiển thị giao diện xem thông tin giảng viên. | | |
| Yêu cầu dữ liệu đầu vào | Từ khóa điều kiện nhập phù hợp với thông tin giảng viên cần tìm kiếm:   * Tên giảng viên * Mã giảng viên * Email * Số điện thoại * Khoa phụ trách | | |
| Flow of Events (luồng sự kiện) | 1. Người dùng đăng nhập thành công vào trang chủ hệ thống website quản lý sinh viên TLU. Hệ thống hiển thị giao diện trang chủ. 2. Người dùng truy cập vào trang quản lý giảng viên. Hệ thống hiển thị giao diện trang quản lý giảng viên. 3. Người dùng nhập từ khóa điều kiện phù hợp với thông tin giảng viên cần tìm kiếm trên công cụ tìm kiếm. 4. Hệ thống hiển thị thông tin giảng viên phù hợp với từ khóa điều kiện cần tìm kiếm của người dùng. | | |
| Luồng con | **A-1: Người dùng có thể thay đổi từ khóa tìm kiếm** Người dùng có thể thay đổi từ khóa tìm kiếm nếu chưa tìm thấy thông tin giảng viên phù hợp với nhu cầu để thực hiện lại thao tác tìm kiếm. | | |
| Exceptions (Ngoại lệ) | **E-1: Không tìm thấy thông tin giảng viên:** Người dùng nhập từ khóa tìm kiếm không phù hợp với thông tin giảng viên cần tìm kiếm trong danh sách giảng viên. Hệ thống không thể hiển thị giảng viên và hiển thị thông báo “Không tìm thấy thông tin giảng viên phù hợp”. | | |

### 2.4.3. Mô tả Use Case Quản lý môn học

#### 2.4.3.1. Use case Thêm môn học

| Use Case ID | 09 | | |
| --- | --- | --- | --- |
| Use Case Name | Thêm môn học | | |
| Created By: | Trần Hồng Quang | Last Updated By: | Trần Hồng Quang |
| Date Created: | 16/05/2025 | Date Last Updated: |  |
| Actor () | Người dùng (Admin) | | |
| Description (mô tả) | Chức năng cho phép người dùng có quyền quản trị thêm mới một môn học vào hệ thống. | | |
| Trigger (kích hoạt) | Người dùng phải có tài khoản đăng nhập vào hệ thống.  Người dùng phải có quyền quản trị Thêm môn học vào hệ thống.  Người dùng nhấn vào nút “Thêm môn học” trên giao diện quản lý môn học. | | |
| Pre-conditions (điều kiện trước) | Người dùng đã đăng nhập vào hệ thống thành công với vai trò là quản trị viên có quyền Thêm môn học vào hệ thông.  Hệ thống đã tải trang danh sách môn học. | | |
| Post conditions (điều kiện sau) | Dữ liệu môn học mới được lưu vào cơ sở dữ liệu.  Giao diện được cập nhật và hiển thị môn học vừa thêm vào. | | |
| Yêu cầu dữ liệu đầu vào | * Mã môn học * Tên môn học * Số tín chỉ * Khoa phụ trách * Học kỳ * Loại môn học * Số tiết * Ghi chú (nếu có) | | |
| Flow of Events (luồng sự kiện) | 1. Người dùng đăng nhập thành công vào trang chủ hệ thống website quản lý sinh viên TLU. Hệ thống hiển thị giao diện trang chủ. 2. Người dùng truy cập vào trang quản lý sinh viên TLU. Hệ thống hiển thị giao diện trang quản lý sinh viên. 3. Người dùng nhấn chọn nút “Thêm môn học”. Hệ thống hiển thị form sẵn sàng nhập liệu bao gồm các trường nhập. 4. Người dùng nhập đầy đủ thông tin bắt buộc để thêm môn học mới. 5. Người dùng nhấn nút “Lưu”. Hệ thống kiểm tra tính hợp lệ, nếu có lỗi xảy ra luồng ngoại lệ **E-1** sẽ thực thi. 6. Nếu hệ thống kiểm tra dữ liệu hợp lệ hệ thống thông báo “Thêm môn học mới thành công”. 7. Môn học mới được lưu trữ dữ liệu vào hệ thông và hiển thị trên danh sách môn học. | | |
| Luồng con | **A-1: Xác nhận thêm môn học mới.**  - Người dùng nhấn nút Lưu. Luồng sự kiện sẽ sang bước tiếp theo. Nếu nhấn nút Hủy hệ thống hủy thao tác thêm môn học và dữ liệu sẽ không được lưu. | | |
| Exceptions (Ngoại lệ) | **E-1: Người dùng bỏ trống các trường nhập liệu bắt buộc**  Người dùng bỏ trống 1 trường bắt buộc hệ thống sẽ hiển thị thông báo lỗi “Vui lòng nhập đầy đủ thông tin.”. | | |

#### 2.4.3.2. Sửa môn học

| Use Case ID | 10 | | |
| --- | --- | --- | --- |
| Use Case Name | Sửa môn học | | |
| Created By: | Trần Hồng Quang | Last Updated By: | Trần Hồng Quang |
| Date Created: | 16/05/2025 | Date Last Updated: |  |
| Actor () | Người dùng (Admin) | | |
| Description (mô tả) | Chức năng cho phép người dùng có quyền quản trị chỉnh sửa thông tin một môn học trong danh sách hệ thống. | | |
| Trigger (kích hoạt) | Người dùng phải có tài khoản đăng nhập vào hệ thống.  Người dùng phải có quyền quản trị Thêm môn học vào hệ thống.  Người dùng nhấn vào nút “Sửa môn học” trên giao diện quản lý môn học. | | |
| Pre-conditions (điều kiện trước) | Người dùng đã đăng nhập vào hệ thống thành công với vai trò là quản trị viên có quyền Sửa môn học vào hệ thông.  Hệ thống đã tải trang danh sách môn học. | | |
| Post conditions (điều kiện sau) | Dữ liệu môn học mới được lưu vào cơ sở dữ liệu.  Giao diện được cập nhật và hiển thị môn học vừa được chỉnh sửa. | | |
| Yêu cầu dữ liệu đầu vào | * Tên môn học * Số tín chỉ * Khoa phụ trách * Học kỳ * Loại môn học * Số tiết * Ghi chú (nếu có) | | |
| Flow of Events (luồng sự kiện) | 1. Người dùng đăng nhập thành công vào trang chủ hệ thống website quản lý sinh viên TLU. Hệ thống hiển thị giao diện trang chủ. 2. Người dùng truy cập vào trang quản lý môn học TLU. Hệ thống hiển thị giao diện trang quản lý môn học. 3. Người dùng nhấn chọn nút “Sửa môn học Hệ thống hiển thị form sẵn sàng nhập liệu bao gồm các trường nhập. 4. Người dùng nhập đầy đủ thông tin bắt buộc để chỉnh sửa môn học. 5. Người dùng nhấn nút “Lưu”. Hệ thống kiểm tra tính hợp lệ, nếu có lỗi xảy ra luồng ngoại lệ **E-1** sẽ thực thi. 6. Nếu hệ thống kiểm tra dữ liệu hợp lệ hệ thống hiên thị thông báo xác nhận “Xác nhận lưu và chỉnh sửa môn học”. 7. Người dùng nhấn “Xác nhận” Hệ thống thông báo “Sửa môn học thành công”. 8. Môn học được lưu trữ dữ liệu vào hệ thông và hiển thị trên danh sách môn học. | | |
| Luồng con | **A-1: Xác nhận sửa môn học.**  - Người dùng nhấn nút Lưu. Luồng sự kiện sẽ sang bước tiếp theo. Nếu nhấn nút Hủy hệ thống hủy thao tác thêm môn học và dữ liệu mới sẽ không được lưu. | | |
| Exceptions (Ngoại lệ) | **E-1: Người dùng bỏ trống các trường nhập liệu bắt buộc**  Người dùng bỏ trống 1 trường bắt buộc hệ thống sẽ hiển thị thông báo lỗi “Vui lòng nhập đầy đủ thông tin.”. | | |

#### 2.4.3.3. Xem môn học

| Use Case ID | 11 | | |
| --- | --- | --- | --- |
| Use Case Name | Xem môn học | | |
| Created By: | Trần Hồng Quang | Last Updated By: | Trần Hồng Quang |
| Date Created: | 16/05/2025 | Date Last Updated: |  |
| Actor () | Admin, Giảng viên, Sinh viên | | |
| Description (mô tả) | Chức năng cho phép người dùng có quyền quản trị xem thông tin một môn học trong danh sách hệ thống. | | |
| Trigger (kích hoạt) | Người dùng phải có tài khoản đăng nhập vào hệ thống.  Người dùng phải có quyền quản trị Thêm môn học vào hệ thống.  Người dùng nhấn vào nút “Xem môn học” trên giao diện quản lý môn học. | | |
| Pre-conditions (điều kiện trước) | Người dùng đã đăng nhập vào hệ thống thành công vào hệ thống.  Hệ thống đã tải trang danh sách môn học. | | |
| Post conditions (điều kiện sau) | Hệ thống hiển thị giao diện xem chi tiết thông tin môn học. | | |
| Yêu cầu dữ liệu đầu vào | Không có | | |
| Flow of Events (luồng sự kiện) | 1. Người dùng đăng nhập thành công vào trang chủ hệ thống website quản lý sinh viên TLU. Hệ thống hiển thị giao diện trang chủ. 2. Người dùng truy cập vào trang quản lý môn học TLU. Hệ thống hiển thị giao diện trang quản lý môn học. 3. Người dùng nhấn chọn nút “Xem môn học”. 4. Hệ thống hiển thị chi tiết thông tin môn học cần xem. | | |
| Luồng con | Không có | | |
| Exceptions (Ngoại lệ) | Không có | | |

#### 2.4.3.4. Tìm kiếm môn học

| Use Case ID | 12 | | |
| --- | --- | --- | --- |
| Use Case Name | Tìm kiếm môn học | | |
| Created By: | Trần Hồng Quang | Last Updated By: | Trần Hồng Quang |
| Date Created: | 16/05/2025 | Date Last Updated: |  |
| Actor () | Admin, Giảng viên, Sinh viên | | |
| Description (mô tả) | Cho phép người dùng thực hiện tìm kiếm môn học theo các tiêu chí như: mã môn, tên môn học, khoa phụ trách, học kỳ, số tín chỉ,… | | |
| Trigger (kích hoạt) | Người dùng phải có tài khoản đăng nhập vào hệ thống.  Người dùng thực hiện tìm kiếm môn học. | | |
| Pre-conditions (điều kiện trước) | Người dùng đã đăng nhập vào hệ thống thành công.  Hệ thống đã tải trang danh sách môn học. | | |
| Post conditions (điều kiện sau) | Hệ thống hiển thị thông tin phù hợp với điều kiện người dùng cần tìm kiếm. | | |
| Yêu cầu dữ liệu đầu vào | Từ khóa người dùng nhập vào công cụ tìm kiếm có thể là:   * Mã môn học * Tên môn học * Số tín chỉ * Khoa phụ trách * Học kỳ * Loại môn học | | |
| Flow of Events (luồng sự kiện) | 1. Người dùng đăng nhập thành công vào trang chủ hệ thống website quản lý sinh viên TLU. Hệ thống hiển thị giao diện trang chủ. 2. Người dùng truy cập vào trang quản lý môn học TLU. Hệ thống hiển thị giao diện trang quản lý môn học. 3. Người dùng thực hiện thao tác tìm kiếm môn học. 4. Người dùng nhập từ khóa phù hợp với môn học cần tìm lên công cụ tìm kiếm. 5. Hệ thống hiển thị danh sách thông tin môn học phù hợp với điều kiện tìm kiếm của người dùng. | | |
| Luồng con | **A-1: Người dùng có thay đổi từ khóa tìm kiếm**  Người dùng có thể thay đổi từ khóa tìm kiếm nếu chưa tìm thấy môn học phù hợp với nhu cầu để thực hiện lại thao tác tìm kiếm. | | |
| Exceptions (Ngoại lệ) | **E-1: Không tìm thấy môn học:** Người dùng nhập từ khóa tìm kiếm không phù hợp với môn học trong danh sách môn học. Hệ thống không thể hiển thị môn học và hiển thị thông báo “Không tìm thấy môn học phù hợp”. | | |

### 2.4.4. Mô tả Use Case Quản lý điểm

#### 2.4.4.1. Use Case Nhập điểm

| Use Case ID | 13 | | |
| --- | --- | --- | --- |
| Use Case Name | Nhập điểm | | |
| Created By: | Bùi Viết Hiển | Last Updated By: | Bùi Viết Hiển |
| Date Created: | 16/06/2025 | Date Last Updated: | 19/06/2025 |
| Actor () | Giảng viên | | |
| Description (mô tả) | Giảng viên thực hiện nhập điểm cho sinh viên theo từng môn học. Dữ liệu điểm sẽ được lưu vào hệ thống. | | |
| Trigger (kích hoạt) | Giảng viên chọn chức năng "Nhập điểm" trong phần quản lý môn học. | | |
| Pre-conditions (điều kiện trước) | * Giảng viên đã được phân công giảng dạy môn học. * Danh sách sinh viên đã tồn tại trong lớp tương ứng. | | |
| Post conditions (điều kiện sau) | * Điểm được lưu thành công vào hệ thống. * Dữ liệu có thể được xem hoặc chỉnh sửa sau đó. | | |
| Yêu cầu dữ liệu đầu vào | * Mã sinh viên * Mã môn học * Điểm số | | |
| Flow of Events (luồng sự kiện) | 1. Giảng viên chọn môn học cần nhập điểm. 2. Hệ thống hiển thị danh sách sinh viên. 3. Giảng viên nhập điểm cho từng sinh viên. 4. Giảng viên dùng nhấn nút “Lưu điểm”. 5. Hệ thống kiểm tra tính hợp lệ và lưu dữ liệu. 6. Hiển thị thông báo nhập thành công. | | |
| Luồng con | Không có | | |
| Exceptions (Ngoại lệ) | **E-1: Định dạng điểm không hợp lệ**  Khi giảng viên nhập điểm không đúng định dạng, hệ thống sẽ hiển thị thông báo lỗi và yêu cầu nhập lại đúng định dạng điểm số.  **E-2: Sinh viên không tồn tại trong lớp**  Nếu mã sinh viên được nhập không thuộc danh sách lớp đã chọn, hệ thống sẽ thông báo lỗi và yêu cầu người dùng kiểm tra lại thông tin sinh viên.  **E-3: Lỗi kết nối cơ sở dữ liệu**  Trong trường hợp xảy ra sự cố khi lưu dữ liệu, hệ thống sẽ hiển thị thông báo lỗi và đề nghị người dùng thử lại sau hoặc liên hệ quản trị hệ thống. | | |

#### 2.4.4.2. Use Case Sửa điểm

| Use Case ID | 14 | | |
| --- | --- | --- | --- |
| Use Case Name | Sửa điểm | | |
| Created By: | Bùi Viết Hiển | Last Updated By: | Bùi Viết Hiển |
| Date Created: | 16/06/2025 | Date Last Updated: | 19/06/2025 |
| Actor () | Giảng viên | | |
| Description (mô tả) | Cho phép giảng viên cập nhật lại điểm đã nhập cho sinh viên trong trường hợp sai sót. | | |
| Trigger (kích hoạt) | Giảng viên chọn chức năng “Sửa điểm” trong giao diện môn học. | | |
| Pre-conditions (điều kiện trước) | * Điểm của sinh viên đã được nhập trước đó. * Giảng viên có quyền sửa điểm. | | |
| Post conditions (điều kiện sau) | * Dữ liệu điểm được cập nhật trong hệ thống. | | |
| Yêu cầu dữ liệu đầu vào | * Mã sinh viên * Mã môn học * Điểm mới cần cập nhật | | |
| Flow of Events (luồng sự kiện) | 1. Giảng viên chọn lớp học → môn học. 2. Chọn sinh viên cần sửa điểm. 3. Nhập điểm mới. 4. Nhấn “Cập nhật”. 5. Hệ thống kiểm tra và lưu điểm mới. | | |
| Luồng con | Không có | | |
| Exceptions (Ngoại lệ) | **E-1: Lỗi khi cập nhật cơ sở dữ liệu**  Nếu có sự cố xảy ra trong quá trình lưu dữ liệu, hệ thống sẽ hiển thị thông báo lỗi và yêu cầu thực hiện lại thao tác sau khi hệ thống ổn định. | | |

2.4.4.3. **Use Case Xem bảng điểm**

**2.4.4.3a. Use Case Quản trị viên xem bảng điểm**

| Use Case ID | 15 | | |
| --- | --- | --- | --- |
| Use Case Name | Admin xem bảng điểm | | |
| Created By: | Bùi Viết Hiển | Last Updated By: | Bùi Viết Hiển |
| Date Created: | 16/06/2025 | Date Last Updated: | 19/06/2025 |
| Actor () | Người dùng (Admin) | | |
| Description (mô tả) | Admin có toàn quyền truy cập và xem bảng điểm của tất cả sinh viên, tất cả lớp học, và tất cả môn học. | | |
| Trigger (kích hoạt) | Admin đăng nhập thành công và chọn chức năng “Xem bảng điểm” từ giao diện điều khiển hệ thống. | | |
| Pre-conditions (điều kiện trước) | Đăng nhập bằng tài khoản có quyền quản trị. | | |
| Post conditions (điều kiện sau) | Hiển thị bảng điểm tương ứng, có thể lọc và xuất file. | | |
| Yêu cầu dữ liệu đầu vào | * Mã lớp hoặc mã sinh viên * Tên môn học * Học kỳ/năm học | | |
| Flow of Events (luồng sự kiện) | 1. Admin đăng nhập vào hệ thống. 2. Chọn “Xem bảng điểm”. 3. Nhập hoặc chọn bộ lọc. (A-1) 4. Hệ thống hiển thị kết quả phù hợp. 5. Có thể xuất file nếu cần. (A-2) | | |
| Luồng con | **A-1:** Lọc theo từng tiêu chí riêng lẻ.  **A-2:** Xuất bảng điểm ra file PDF/Excel. | | |
| Exceptions (Ngoại lệ) | **E-1:** Không có kết quả → Thông báo “Không tìm thấy dữ liệu”.  **E-2:** Lỗi hệ thống → Hiển thị lỗi truy xuất, đề nghị thử lại. | | |

**2.4.4.3b. Use Case Giảng viên xem bảng điểm lớp phụ trách**

| Use Case ID | 16 | | |
| --- | --- | --- | --- |
| Use Case Name | Giảng viên xem bảng điểm lớp phụ trách | | |
| Created By: | Bùi Viết Hiển | Last Updated By: | Bùi Viết Hiển |
| Date Created: | 16/06/2025 | Date Last Updated: | 19/06/2025 |
| Actor () | Giảng viên | | |
| Description (mô tả) | Giảng viên có thể xem bảng điểm của các lớp/môn mình được phân công giảng dạy. | | |
| Trigger (kích hoạt) | Giảng viên đăng nhập và chọn chức năng “Xem bảng điểm”. | | |
| Pre-conditions (điều kiện trước) | Tài khoản giảng viên đã được phân công môn học. | | |
| Post conditions (điều kiện sau) | Bảng điểm lớp tương ứng hiển thị đúng. | | |
| Yêu cầu dữ liệu đầu vào | * Môn học được phân công * Lớp giảng dạy * Học kỳ/năm học | | |
| Flow of Events (luồng sự kiện) | 1. Giảng viên đăng nhập. 2. Chọn “Xem bảng điểm”. 3. Hệ thống liệt kê các môn phụ trách. 4. Giảng viên chọn môn/lớp cụ thể. 5. Hệ thống hiển thị bảng điểm. 6. Giảng viên có thể lc theo kỳ nếu cần (A-1) 7. Có thể xuất file nếu cần (A-2) | | |
| Luồng con | **A-1:** Lọc theo học kỳ.  **A-2:** Xuất bảng điểm lớp phụ trách. | | |
| Exceptions (Ngoại lệ) | **E-1:** Truy cập sai lớp → “Bạn không được phân công lớp này.”  **E-2:** Chưa có điểm → “Chưa có dữ liệu để hiển thị.”  **E-3:** Lỗi hệ thống → Thông báo lỗi truy vấn. | | |

**2.4.4.3c. Use Case Sinh viên xem điểm cá nhân**

| Use Case ID | 17 | | |
| --- | --- | --- | --- |
| Use Case Name | Sinh viên xem điểm cá nhân | | |
| Created By: | Bùi Viết Hiển | Last Updated By: | Bùi Viết Hiển |
| Date Created: | 16/06/2025 | Date Last Updated: | 19/06/2025 |
| Actor () | Sinh viên | | |
| Description (mô tả) | Sinh viên có thể xem điểm các môn học mình đã đăng ký và hoàn thành. | | |
| Trigger (kích hoạt) | Sinh viên đăng nhập và chọn chức năng “Xem điểm”. | | |
| Pre-conditions (điều kiện trước) | Đã có tài khoản sinh viên và dữ liệu điểm. | | |
| Post conditions (điều kiện sau) | Hệ thống hiển thị điểm các môn đã học. | | |
| Yêu cầu dữ liệu đầu vào | * Mã sinh viên (được hệ thống tự xác định sau đăng nhập) * Lọc theo học kỳ | | |
| Flow of Events (luồng sự kiện) | 1. Sinh viên đăng nhập hệ thống. 2. Chọn chức năng “Xem điểm”. 3. Hệ thống hiển thị danh sách điểm các môn đã học. 4. Sinh viên có thể lọc theo học kỳ/năm học. (A-1) 5. Hệ thống hiển thị điểm của sinh viên | | |
| Luồng con | **A-1:** Lọc theo học kỳ. | | |
| Exceptions (Ngoại lệ) | **E-1:** Lỗi hệ thống → Hiển thị lỗi và đề nghị thử lại sau. | | |

# 

# PHẦN 3: THIẾT KẾ HỆ THỐNG DESIGN PATTERN

## 3.1. Tổng quan kiến trúc hệ thống

### 3.1.1. Mô hình kiến trúc tổng thể

Hệ thống quản lý sinh viên được xây dựng theo mô hình 3 lớp (Three-tier Architecture) bao gồm:

* Lớp Giao diện người dùng (Presentation Layer):
  + Giao diện website cho phép người dùng thao tác với hệ thống như: thêm sinh viên, giảng viên, nhập điểm, xem bảng điểm,...
  + Công nghệ sử dụng: HTML, CSS, JavaScript, Razor Pages hoặc Blazor (nếu dùng ASP.NET Core).
* Lớp Xử lý nghiệp vụ (Business Logic Layer):
  + Chứa toàn bộ logic xử lý, bao gồm các dịch vụ xử lý dữ liệu, tính toán điểm, xác thực vai trò người dùng, gọi các mẫu thiết kế (Design Pattern).
  + Ở lớp này, các Design Pattern như Facade, Builder, Observer, Decorator,... được áp dụng nhằm tổ chức hệ thống rõ ràng và dễ mở rộng.
* Lớp Dữ liệu (Data Access Layer):
  + Giao tiếp với cơ sở dữ liệu SQL Server thông qua các repository.
  + Áp dụng các mẫu thiết kế Repository, Decorator để quản lý dữ liệu một cách linh hoạt và có thể thêm tính năng như caching, logging.

### 3.1.2. Tương tác giữa các thành phần

Khi người dùng thao tác (ví dụ: Thêm sinh viên), yêu cầu được gửi từ giao diện đến lớp Business thông qua Controller/Service.

Lớp Business sử dụng các dịch vụ và mẫu thiết kế như:

* Builder: tạo đối tượng sinh viên.
* Facade: nhập học sinh viên (bao gồm tạo tài khoản, gán môn học,...).
* Observer: gửi thông báo khi điểm được nhập.

Cuối cùng, lớp Business gọi Repository để thao tác với cơ sở dữ liệu.

### 3.1.3. Ứng dụng vào Design Pattern

Hệ thống áp dụng nhiều mẫu thiết kế phần mềm (Design Patterns) để giải quyết các vấn đề khác nhau:

**Tính đơn nhiệm:** mỗi class có một vai trò rõ ràng.

**Tái sử dụng mã**: các thành phần như Builder, Adapter có thể dùng lại ở nhiều nơi.

**Mở rộng dễ dàng**: Decorator cho phép thêm chức năng (như cache) mà không thay đổi logic gốc.

**Giao tiếp đơn giản**: Facade giúp client tương tác dễ dàng với hệ thống con phức tạp.

## 3.2. Các Design Pattern được áp dụng

### 3.2.1. Observer

#### 3.2.1.1. Khái niệm

Observer là một Behavioral Design Pattern, cho phép một đối tượng (Subject) thông báo cho nhiều đối tượng phụ thuộc (Observers) khi có sự thay đổi trạng thái.

Dùng trong các tình huống cần đồng bộ nhiều thành phần khi một đối tượng thay đổi.

#### 3.2.1.2. Cấu trúc

Các thành phần:

**Subject:** Giao diện quản lý Observer (thêm/xóa/thông báo).

**ConcreteSubject:** Thực thi Subject, giữ trạng thái và thông báo khi có thay đổi.

Observer: Giao diện nhận thông báo.

Concrete Observer: Thực thi Observer, cập nhật trạng thái khi được thông báo.

#### 3.2.1.3. Ưu và nhược điểm

Ưu điểm:

Tăng tính linh hoạt và mở rộng.

Dễ quản lý khi có nhiều đối tượng phụ thuộc.

Giảm sự phụ thuộc chặt giữa các lớp (loose coupling).  
Nhược điểm:

Khó debug nếu có quá nhiều observer.

Dễ gây hiệu suất kém nếu thông báo đến nhiều observer không cần thiết.

### 3.2.2. Facade

#### 3.2.2.1. Khái niệm

Facade là một **Structural Pattern**, cung cấp **giao diện đơn giản** để truy cập vào hệ thống con phức tạp, **ẩn đi sự phức tạp** của các lớp bên trong hệ thống.

#### 3.2.2.2. Cấu trúc

Facade: Lớp giao tiếp chính với client.

Subsystem classes: Các lớp hệ thống con với logic xử lý riêng.

#### 3.2.2.3. Ưu và nhược điểm

Ưu điểm:

Giảm độ phức tạp của hệ thống.

Che giấu sự phức tạp bên trong hệ thống con.

Tăng tính đóng gói, dễ bảo trì.

Nhược điểm:

Nếu không thiết kế cẩn thận, Facade có thể trở thành lớp "God Object".

Giảm tính linh hoạt khi cần truy cập trực tiếp đến các lớp con.

### 3.2.3. Decorator

#### 3.2.3.1. Khái niệm

Decorator Pattern là một mẫu thiết kế thuộc nhóm Structural Design Pattern. Mẫu thiết kế này cho phép mở rộng chức năng của một đối tượng một cách linh hoạt bằng cách bao bọc đối tượng đó trong một lớp khác mà không làm thay đổi cấu trúc gốc của đối tượng.

#### 3.2.3.2. Cấu trúc

Component: Giao diện chung cho đối tượng cần mở rộng.

ConcreteComponent: Đối tượng cụ thể cần được thêm chức năng.

Decorator: Lớp trừu tượng thực thi giao diện Component và chứa tham chiếu đến một đối tượng Component.

ConcreteDecorator: Lớp mở rộng chức năng cụ thể cho đối tượng được bao bọc.

#### 3.2.3.3. Ưu và nhược điểm

Ưu điểm:

Cho phép mở rộng linh hoạt mà không làm thay đổi lớp gốc.

Hỗ trợ kết hợp nhiều decorator để mở rộng theo yêu cầu.

Tuân thủ nguyên tắc mở rộng – đóng (Open/Closed Principle).

Nhược điểm:

Tăng số lượng lớp, gây khó khăn trong việc quản lý mã nguồn.

Có thể gây rối nếu sử dụng lồng ghép quá nhiều decorator.

### 3.2.4. Repository

#### 3.2.4.1. Khái niệm

Repository Pattern là một mẫu thiết kế thuộc nhóm Architectural Pattern. Mục đích chính là tách biệt logic truy xuất dữ liệu khỏi logic nghiệp vụ, đóng vai trò như một cầu nối giữa tầng nghiệp vụ và tầng lưu trữ dữ liệu (như cơ sở dữ liệu, API,…). Repository cung cấp một giao diện trừu tượng cho phép thao tác với dữ liệu một cách thống nhất và dễ kiểm soát.

#### 3.2.4.2. Cấu trúc

**Entity**: Lớp mô tả đối tượng dữ liệu (model).

**Repository Interface**: Giao diện định nghĩa các phương thức CRUD cơ bản.  
**Concrete Repository**: Hiện thực các phương thức tương tác với nguồn dữ liệu thực tế.

**Service/Controller**: Thành phần sử dụng repository để thao tác dữ liệu mà không biết chi tiết cách dữ liệu được lưu trữ.

#### 3.2.4.3. Ưu và nhược điểm

**Ưu điểm:**

Tăng khả năng kiểm thử và tái sử dụng mã nguồn.

Dễ dàng thay đổi tầng dữ liệu mà không ảnh hưởng đến logic nghiệp vụ.

Tăng khả năng bảo trì và mở rộng hệ thống.

**Nhược điểm:**

Tăng độ phức tạp trong các ứng dụng nhỏ hoặc đơn giản.

Có thể trùng lặp logic với các công cụ ORM nếu không tối ưu tốt.

### 3.2.5. Adapter

#### 3.2.5.1. Khái niệm

***Adapter Pattern*** (Mẫu bộ chuyển đổi) là một mẫu thiết kế thuộc nhóm Structural Design Pattern (Mẫu thiết kế cấu trúc), cho phép chuyển đổi giao diện (interface) của một lớp thành một giao diện khác mà hệ thống cần.

Hiểu một cách đơn giản, Adapter giống như một “cầu nối” giúp kết nối giữa hai thành phần không tương thích về mặt giao tiếp. Adapter không làm thay đổi bản chất của đối tượng gốc mà chỉ đóng gói (wrap) lại để có thể tương tác được với phần còn lại của hệ thống.

#### 3.2.5.2. Cấu trúc

**Target**: Giao diện mà client mong muốn sử dụng.

**Adaptee**: Lớp có sẵn, nhưng không tương thích với Target.

**Adapter**: Lớp trung gian, chuyển đổi từ Adaptee sang Target.

**Client**: Gọi các phương thức từ Target mà không biết đến Adaptee.

#### 3.2.5.3. Ưu và nhược điểm

**Ưu điểm:**

Tái sử dụng mã: Có thể tận dụng lại các lớp cũ mà không cần sửa đổi mã gốc.

Tăng tính linh hoạt: Hệ thống có thể tích hợp thêm các thành phần mới một cách dễ dàng.

Tách biệt rõ ràng giữa logic xử lý và lớp chuyển đổi → dễ bảo trì, mở rộng.

Tuân thủ nguyên tắc SOLID – đặc biệt là OCP (Open/Closed Principle).

**Nhược điểm:**

Tăng thêm lớp trung gian, có thể làm phức tạp hệ thống nếu dùng không hợp lý.

Nếu lạm dụng, Adapter sẽ làm ẩn đi sự không tương thích ban đầu → khó phát hiện lỗi logic.

Đôi khi Adapter chỉ có thể cung cấp chuyển đổi giới hạn, không phải đầy đủ chức năng như client mong muốn.

### 3.2.6. Builder

#### 3.2.6.1. Khái niệm

Builder Pattern là mẫu thiết kế **tạo dựng đối tượng phức tạp** bằng cách chia nhỏ quá trình khởi tạo thành nhiều bước. Mỗi bước thiết lập một thuộc tính của đối tượng, cho phép tùy chỉnh linh hoạt và cải thiện khả năng đọc mã.

#### 3.2.6.2. Cấu trúc

**Director (tuỳ chọn):** Quản lý quá trình xây dựng.

**Builder Interface:** Khai báo các phương thức bước xây dựng.

**Concrete Builder:** Cài đặt cụ thể từng bước xây dựng và tạo đối tượng.

**Product:** Đối tượng cuối cùng được xây dựng.

#### 3.2.6.3. Ưu và nhược điểm

**Ưu điểm:**

Hữu ích khi đối tượng có nhiều thuộc tính, đặc biệt có thuộc tính tùy chọn.

Giảm rối rắm khi dùng constructor dài.

Tăng tính linh hoạt và khả năng đọc hiểu mã nguồn.

**Nhược điểm:**

Làm tăng số lượng class.

Phức tạp không cần thiết với đối tượng đơn giản.

### 3.2.7. Abstract

#### 3.2.7.1. Khái niệm

Abstract Factory là mẫu thiết kế tạo ra một họ các đối tượng liên quan mà không cần biết lớp cụ thể. Nó cung cấp một interface cho việc khởi tạo các đối tượng thuộc cùng một nhóm.

#### 3.2.7.2. Cấu trúc

**AbstractFactory:** Giao diện chung để tạo các sản phẩm liên quan.

**ConcreteFactory:** Cài đặt cụ thể để tạo từng sản phẩm.

**AbstractProduct:** Giao diện sản phẩm chung.

**ConcreteProduct:** Sản phẩm cụ thể.

#### 3.2.7.3. Ưu và nhược điểm

**Ưu điểm:**

Tạo nhóm đối tượng đồng bộ một cách dễ dàng.

Đảm bảo các đối tượng được sử dụng là tương thích với nhau.

Dễ dàng mở rộng thêm định dạng đầu ra mới (ví dụ: Word, CSV...).

**Nhược điểm:**

Mỗi nhóm sản phẩm cần tạo thêm nhiều class.

Khó linh hoạt nếu muốn thay đổi cấu trúc đối tượng đã tạo.

### 3.2.8. Factory Method

#### 3.2.8.1. Khái niệm

Factory Method là mẫu thiết kế cho phép lớp con quyết định loại đối tượng cần tạo. Nó định nghĩa một phương thức để khởi tạo, nhưng để lớp kế thừa quyết định lớp cụ thể.

#### 3.2.8.2. Cấu trúc

**Creator (Abstract class):** Khai báo phương thức FactoryMethod().

**ConcreteCreator:** Cài đặt cụ thể phương thức FactoryMethod().

**Product:** Giao diện chung cho các đối tượng được tạo.

**ConcreteProduct:** Lớp cụ thể được tạo ra.

#### 3.2.8.3. Ưu và nhược điểm

**Ưu điểm:**

Tăng tính mở rộng hệ thống khi cần thêm loại đối tượng mới.

Tuân thủ nguyên lý “đóng cho sửa đổi, mở cho mở rộng”.

**Nhược điểm:**

Gây phức tạp với các hệ thống nhỏ, không cần nhiều loại sản phẩm.

Cần tạo nhiều lớp kế thừa để triển khai sản phẩm khác nhau.

### 3.2.9. Singleton

#### 3.2.9.1. Khái niệm

Singleton là mẫu thiết kế nhằm đảm bảo chỉ có duy nhất một đối tượng của một lớp được tạo ra trong suốt vòng đời ứng dụng. Thường dùng cho các tài nguyên dùng chung như log, config, kết nối CSDL...

#### 3.2.9.2. Cấu trúc

**Private Constructor:** Ngăn việc tạo đối tượng từ bên ngoài.

**Static Field:** Lưu thể hiện duy nhất.

**Static Property/Method:** Trả về thể hiện singleton.

#### 3.2.9.3. Ưu và nhược điểm

**Ưu điểm:**

Quản lý tài nguyên dùng chung hiệu quả (ghi log, cache, cấu hình).

Đảm bảo nhất quán dữ liệu trong toàn hệ thống.

Tích hợp tốt với Dependency Injection (ASP.NET Core).

**Nhược điểm:**

Khó kiểm thử nếu không kết hợp DI.

Dễ bị lạm dụng → trở thành biến toàn cục gây phụ thuộc.

Không an toàn trong môi trường đa luồng nếu không cài đặt đúng.

## 3.3. Giải thích lý do chọn mỗi mẫu

**Singleton Pattern**

Lý do chọn: Trong hệ thống, việc ghi log là một nhiệm vụ xuyên suốt, cần được thực hiện nhất quán và không lặp lại nhiều lần thể hiện. Singleton đảm bảo chỉ có một thể hiện FileLogger tồn tại, giúp quản lý tài nguyên như file log hiệu quả và đồng nhất. Đồng thời, phù hợp với kiến trúc ASP.NET Core khi tích hợp vào DI container qua AddSingleton().

**Factory Method & Abstract Factory Pattern**

Lý do chọn: Việc xuất báo cáo dưới nhiều định dạng (PDF, Excel...) yêu cầu khởi tạo các thành phần khác nhau như Header, Body. Abstract Factory giúp tạo ra một nhóm các thành phần tương ứng (đồng bộ về định dạng) mà không cần biết cụ thể lớp nào được sử dụng. Factory Method hỗ trợ mở rộng khi có thêm định dạng báo cáo mới mà không sửa đổi logic cũ.

**Builder Pattern**

Lý do chọn: Khi thêm mới sinh viên, đối tượng Student có nhiều thuộc tính, trong đó có nhiều trường không bắt buộc. Builder Pattern giúp tách biệt việc khởi tạo đối tượng thành từng bước riêng biệt, nâng cao tính linh hoạt và dễ đọc mã nguồn hơn so với constructor dài.

**Adapter Pattern**

Lý do chọn: Dữ liệu sinh viên có thể đến từ các nguồn bên ngoài như API hoặc file XML không tương thích với cấu trúc Student của hệ thống. Adapter Pattern đóng vai trò là cầu nối, chuyển đổi dữ liệu từ định dạng không tương thích sang định dạng hệ thống yêu cầu mà không làm thay đổi logic lõi.

**Repository Pattern**

Lý do chọn: Hệ thống cần tách biệt logic truy xuất dữ liệu khỏi logic nghiệp vụ, giúp dễ bảo trì và kiểm thử. Repository Pattern giúp chuẩn hóa cách thao tác với cơ sở dữ liệu, tạo ra lớp trung gian giữa dữ liệu và logic xử lý.

**Decorator Pattern**

Lý do chọn: Nhằm mở rộng chức năng repository như thêm cache mà không chỉnh sửa repository gốc. Decorator giúp bọc thêm logic caching xung quanh repository gốc một cách linh hoạt mà không làm ảnh hưởng đến cấu trúc hiện tại.

**Facade Pattern**

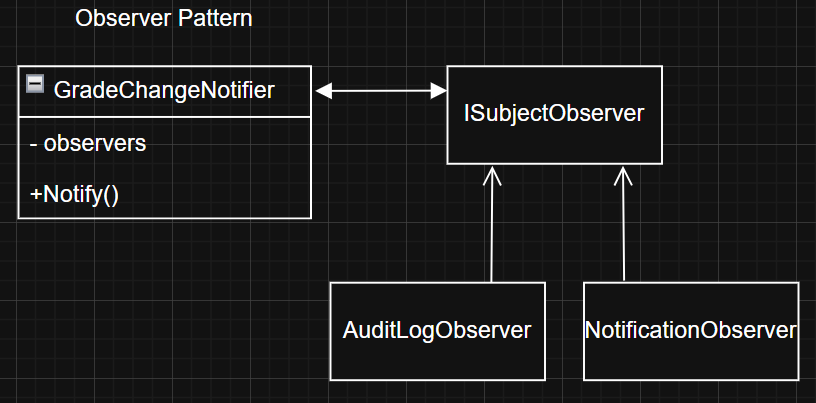
Lý do chọn: Quy trình nhập học sinh viên bao gồm nhiều bước nhỏ (thêm sinh viên, gán môn học, tạo tài khoản, gửi thông báo...). Facade Pattern cung cấp một giao diện đơn giản hóa, giúp controller hoặc client không cần biết logic chi tiết mà vẫn thực hiện đầy đủ quy trình.

**Observer Pattern**

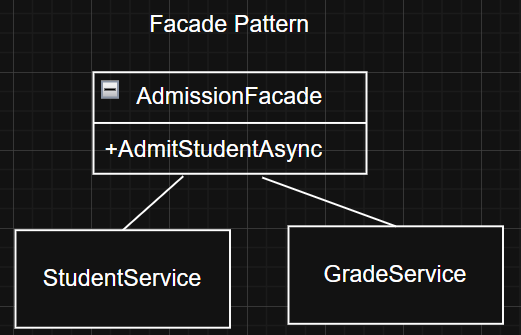
Lý do chọn: Khi giảng viên cập nhật điểm, cần tự động thông báo đến nhiều thành phần như hệ thống ghi log, hệ thống gửi thông báo. Observer Pattern cho phép mở rộng số lượng tác vụ phản ứng với sự kiện mà không thay đổi logic cập nhật điểm.

## 3.4. Biểu đồ minh họa cho mỗi Design Pattern

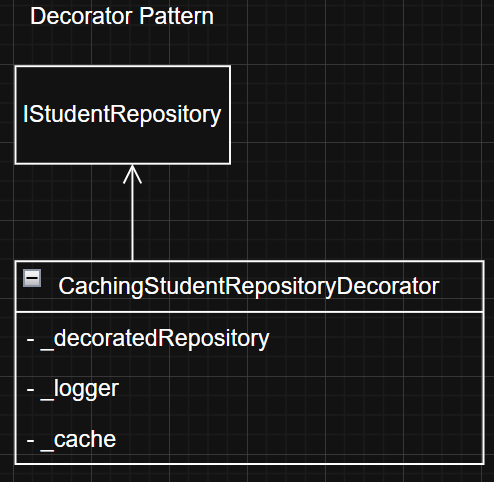
### 3.4.1. Observer



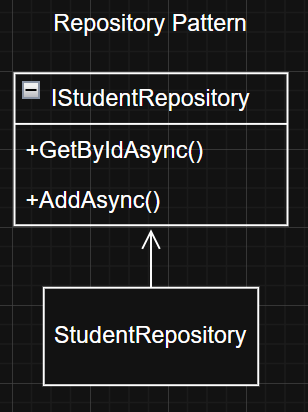
### 3.4.2. Facade



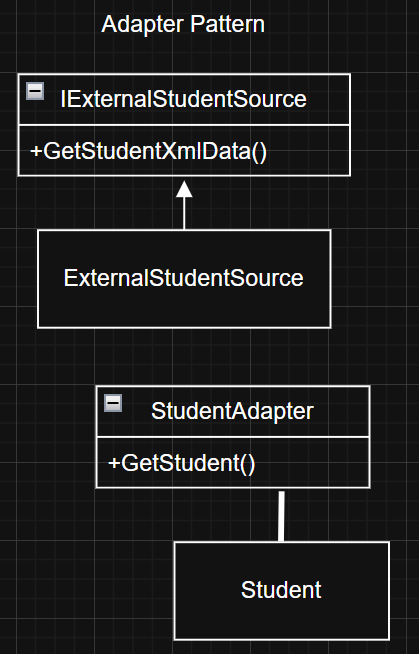
### 3.4.3. Decorator



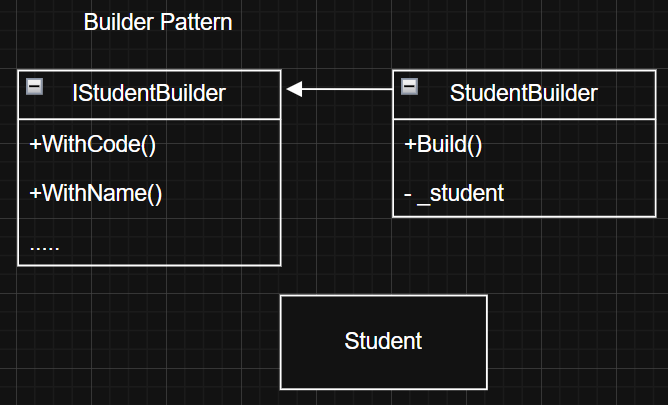
### 3.4.4. Repository



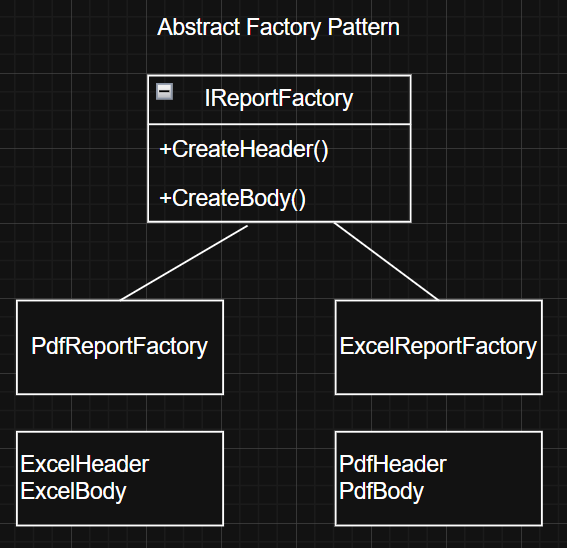
### 3.4.5. Adapter



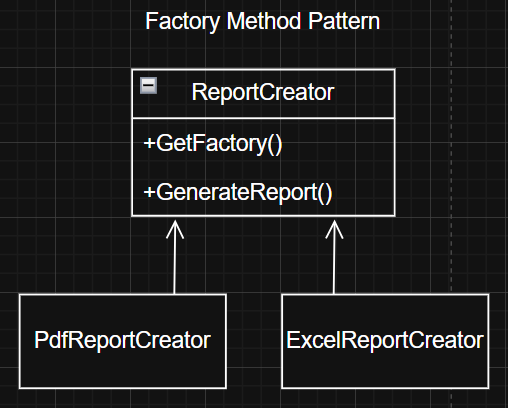
### 3.4.6. Builder



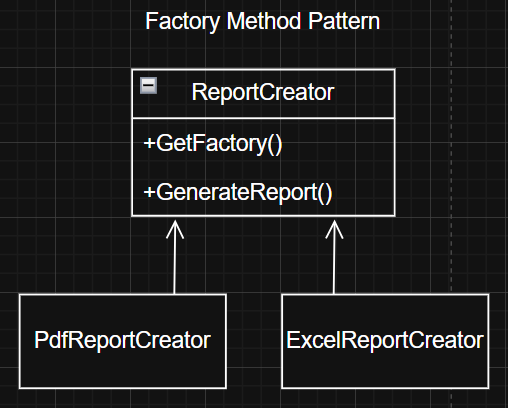
### 3.4.7. Abstract



### 3.4.8. Factory Method



### 3.4.9. Singleton



## 3.5. Mối quan hệ giữa các pattern trong hệ thống

Trong quá trình thiết kế và phát triển hệ thống quản lý sinh viên, các mẫu thiết kế (Design Pattern) không được sử dụng đơn lẻ mà được kết hợp với nhau một cách hợp lý nhằm tận dụng tối đa điểm mạnh của từng mẫu, đồng thời tạo nên một kiến trúc phần mềm rõ ràng, linh hoạt và dễ mở rộng. Dưới đây là các mối quan hệ giữa các Design Pattern trong hệ thống:

### 3.5.1. Mối quan hệ giữa Singleton, Decorator và Repository

* Mẫu **Singleton** được sử dụng để đảm bảo chỉ tồn tại một đối tượng FileLogger ghi log hệ thống.
* Mẫu **Decorator** được sử dụng để mở rộng chức năng cho IStudentRepository, điển hình là thêm tính năng cache thông qua lớp CachingStudentRepositoryDecorator.
* Mẫu **Repository** đóng vai trò giao tiếp với cơ sở dữ liệu.
* **Mối quan hệ:** Singleton hỗ trợ Decorator bằng cách cung cấp một logger dùng chung cho tất cả repository decorator. Decorator giúp mở rộng Repository mà không cần sửa đổi logic cốt lõi.

### 3.5.2. Mối quan hệ giữa Facade, Service và Builder

* Mẫu **Facade** (AdmissionFacade) đóng vai trò là giao diện đơn giản giúp client (controller) dễ dàng gọi các thao tác phức tạp như nhập học sinh viên.
* Trong quá trình nhập học, **Builder** được sử dụng để khởi tạo đối tượng Student với nhiều thuộc tính linh hoạt.
* **Mối quan hệ:** Facade ẩn đi sự phức tạp của các bước nghiệp vụ bên trong và gọi đến các dịch vụ như StudentService, nơi Builder được sử dụng để tạo đối tượng Student.

### 3.5.3. Mối quan hệ giữa Observer và Grade Service

* Khi giảng viên cập nhật điểm, hệ thống cần đồng thời thực hiện nhiều tác vụ như ghi log, gửi thông báo,...
* Mẫu **Observer** được sử dụng để theo dõi sự thay đổi điểm và thông báo đến các observer như AuditLogObserver và NotificationObserver.
* **Mối quan hệ:** GradeService tương tác với GradeChangeNotifier để kích hoạt cơ chế Observer, đảm bảo các thành phần liên quan được đồng bộ hóa khi điểm số thay đổi.

### 3.5.4. Mối quan hệ giữa Factory Method và Abstract Factory

* Mẫu **Factory Method** (qua ReportCreator) xác định kiểu báo cáo cần sinh ra (PDF hoặc Excel).
* Mẫu **Abstract Factory** cung cấp khả năng tạo các thành phần đồng bộ của báo cáo (Header và Body) phù hợp với định dạng tương ứng.
* **Mối quan hệ:** Factory Method sử dụng Abstract Factory như một phần trong logic tạo sản phẩm, từ đó sinh ra các thành phần báo cáo một cách đồng nhất.

### 3.5.5. Mối quan hệ giữa Adapter và Builder

* Adapter giúp chuyển đổi dữ liệu từ định dạng bên ngoài (ví dụ: XML) thành đối tượng nội bộ Student.
* Sau đó, Builder có thể được sử dụng để hoàn tất việc tạo đối tượng Student với các thuộc tính cần thiết.
* **Mối quan hệ:** Adapter xử lý phần tương thích dữ liệu, còn Builder xử lý phần khởi tạo đối tượng phức tạp theo chuẩn nội bộ hệ thống.

# KẾT LUẬN

Trong khuôn khổ môn học *“Các vấn đề tiên tiến trong phát triển phần mềm”*, nhóm chúng em đã nghiên cứu, thiết kế và xây dựng hệ thống website **Quản lý sinh viên TLU** với việc áp dụng hiệu quả các **Design Pattern** trong kiến trúc phần mềm. Việc sử dụng các mẫu thiết kế như **Singleton, Factory Method, Abstract Factory, Builder, Adapter, Repository, Decorator, Facade** và **Observer** không chỉ giúp hệ thống có **cấu trúc rõ ràng**, **tính mở rộng cao**, mà còn đảm bảo **tái sử dụng mã nguồn**, **dễ bảo trì** và **dễ kiểm thử** trong tương lai.

Qua quá trình thực hiện đề tài, nhóm đã rèn luyện và nâng cao tư duy thiết kế phần mềm theo hướng chuyên nghiệp, hiểu rõ hơn về **kiến trúc ba lớp**, cũng như cách kết hợp các pattern với nhau để giải quyết các bài toán phức tạp trong thực tế. Các chức năng chính như quản lý sinh viên, giảng viên, môn học và điểm đều được triển khai với sự hỗ trợ của Design Pattern, góp phần nâng cao chất lượng phần mềm và trải nghiệm người dùng.

Mặc dù đề tài đã đạt được những mục tiêu cơ bản, nhưng do hạn chế về thời gian và kinh nghiệm thực tế, hệ thống vẫn còn một số điểm cần cải thiện. Trong tương lai, nhóm mong muốn tiếp tục phát triển thêm các tính năng nâng cao như phân quyền người dùng, phân tích học lực, tích hợp API và tối ưu hiệu suất hệ thống.

Nhóm xin chân thành cảm ơn thầy **TS. Cù Việt Dũng** đã tận tình hướng dẫn và góp ý trong suốt quá trình thực hiện báo cáo. Những kiến thức và kỹ năng thu được từ đề tài này sẽ là nền tảng quý báu cho chúng em trong các dự án phần mềm chuyên nghiệp sau này.

Link code: [Link](https://github.com/thanglele/QLSinhvien_DesignPattern.git)

Link video: [Link](https://drive.google.com/drive/folders/1zw6kU7ltyysvyrL9qJUabNrdrtgBEV_n?usp=drive_link)