

TRƯỜNG ĐẠI HỌC SÀI GÒN

KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO

ĐỒ ÁN XÂY DỰNG PHẦN MỀM THEO MÔ HÌNH PHÂN LỚP

ĐỀ TÀI: Xây dựng phần mềm quản lý khóa học theo 3-Layer

GIẢNG VIÊN HƯỚNG DẪN

GV. Cao Minh Thành

SINH VIÊN THỰC HIỆN

Họ tên	MSSV
1. Nguyễn Trần Huỳnh Long	3119410231
2. Phạm Tuấn Khanh	3119410183
3. Hà Khang Kỳ	3119410217
4. Võ Hoàng Kiệt	3119410215
5. Nguyễn Hữu An	3119410003

TP. Hồ Chí Minh, 9/2022

LỜI CẢM ƠN

Lời đầu tiên em xin chân thành cảm ơn các thầy cô trong khoa CNTT của trường Đại Học Sài Gòn, những người đã trực tiếp giảng dạy cung cấp kiến thức và phương pháp trong 3 năm qua, đó là những nền tảng cơ bản, là những hành trang vô cùng quý giá để em có thể bước vào sự nghiệp trong tương lai.

Để có được kết quả này chúng em xin đặc biệt gửi lời cảm ơn chân thành nhất tới **thầy Cao Minh Thành** đã quan tâm giúp đỡ, vạch kế hoạch hướng dẫn em hoàn thành một cách tốt nhất đồ án ngành trong thời gian qua.

Cuối cùng em xin chân thành cảm ơn gia đình, bạn bè đã động viên chia sẻ, giúp đỡ nhiệt tình và đóng góp nhiều ý kiến quý báu để em có thể hoàn thành đồ án ngành này.

Trong quá trình hoàn thành đồ án, vì chưa có kinh nghiệm thực tế chỉ dựa vào lý thuyết đã học, cùng với thời gian có hạn nên đồ án sẽ không tránh khỏi những sai sót.

Kính mong nhận được sự góp ý, nhận xét từ các thầy cô để kiến thức của em ngày càng hoàn thiện hơn và rút ra được nhiều kinh nghiệm bổ ích có thể áp dụng vào thực tiễn một cách hiệu quả trong tương lai.

Chúng em xin chân thành cảm ơn!

This image shows a full page of white paper with horizontal dotted lines, typical of primary school writing paper. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

Cao Minh Thành

LỜI NÓI ĐẦU

Những năm gần đây, vai trò của các phần mềm đang càng ngày lớn mạnh. Từ chỗ chỉ sử dụng để hỗ trợ một số hoạt động trong công nghệ, phần mềm đã trở nên có vai trò thiết yếu trong công nghệ thông tin. Đặc biệt những thành tựu về phần mềm trong công nghệ thông tin đã khiến thế giới ngày càng chú ý hơn tới việc áp dụng nó để gia tăng ưu thế cạnh tranh và tạo cơ hội cho sự phát triển. Hiện nay trào lưu ứng dụng thành tựu công nghệ thông tin không chỉ giới hạn trong chỉ doanh nghiệp lớn tầm cỡ đa quốc gia, mà còn lan rộng trong tất cả doanh nghiệp, kể cả những doanh nghiệp vừa và nhỏ ở những nước đang phát triển.

Tuy nhiên việc xây dựng phần mềm dựa trên thành tựu công nghệ thông tin không phải đơn giản, ngay với cả những doanh nghiệp lớn, đòi hỏi nguồn tài lực, nhân lực và kinh nghiệm. Một phần mềm thành công trong doanh nghiệp này cũng chưa chắc thành công trong doanh nghiệp khác. Vì vậy nhóm em chọn đề tài là Quản lý khóa học để có cái nhìn sâu hơn, thực tế hơn về việc ứng dụng hệ thống thông tin trong hoạt động nghiệp vụ trong giáo dục. Từ đó nhóm rút ra được mặt ưu, khuyết của phần mềm, kinh nghiệm triển khai ứng dụng theo mô hình 3 Layer và các mặt về đạo đức, xã hội trong quản lý hệ thống thông tin. Vì thời gian thực hiện không nhiều, nên chắc chắn còn nhiều thiếu sót, mong được nhận được sự góp ý từ thầy/cô.

Trân trọng cảm ơn

Mục lục

LỜI NÓI ĐẦU	4
BẢNG PHÂN CHIA CÔNG VIỆC.....	6
I. GIỚI THIỆU CHUNG	7
1. Giới thiệu đồ án.....	7
1.1. Giới thiệu chung về đồ án.....	7
1.2. Mô hình 3 Layer	7
Ưu điểm.....	8
II. GIAO DIỆN PHẦN MỀM.....	9
1. Quản lý học viên	9
2. Quản lý giảng viên	13
3. Quản lý khóa học	15
4. Quản lý phân công	17
5. Quản lý kết quả.....	18
III. Code và Sơ đồ tuần tự	21
1. Quản lý sinh viên, giáo viên	21
1.1 Sinh viên.....	21
1.2 Giáo viên	31
2. Quản lý khóa học	41
2.1 Onsite	43
2.2 Online	53
3. Quản lý phân công	61
4. Quản lý kết quả.....	61
IV. Cài đặt phần mềm.....	72
1. Phương án cài đặt	72
2. Source code.....	72
V. Tài liệu tham khảo.....	72

BẢNG PHÂN CHIA CÔNG VIỆC

Thành viên	Công việc
Nguyễn Trần Huỳnh Long	Quản lý sinh viên, giáo viên
Phạm Tuấn Khanh	Quản lý kết quả
Hà Khang Kỳ	Quản lý khóa học onsite, online
Võ Hoàng Kiệt	Quản lý phân công
Nguyễn Hữu An	Tổng hợp và viết báo cáo

I. GIỚI THIỆU CHUNG

1. Giới thiệu đồ án

1.1. Giới thiệu chung về đồ án

Trong thời đại công nghệ phát triển, việc đào tạo và giáo dục luôn đặt lên hàng đầu để cho giới trẻ ngày nay có thể bắt kịp xu hướng phát triển của công nghệ. Do đó nhu cầu về các phần mềm quản lý về việc giáo dục và đào tạo ngày càng có nhu cầu thiết yếu. Trong hoàn cảnh đó, nhóm chúng em quyết định thực hiện đồ án xây dựng phần mềm quản lý khóa học theo mô hình 3 Layer. Phần mềm được xây dựng theo ngôn ngữ lập trình java và cơ sở dữ liệu MySQL.

1.2. Mô hình 3 Layer

3-tiers là một kiến trúc kiểu **client/server** mà trong đó giao diện người dùng (UI-user interface), các quy tắc xử lý (BR-business rule hay BL-business logic), và việc lưu trữ dữ liệu được phát triển như những module độc lập, và hầu hết là được duy trì trên các nền tảng độc lập, và mô hình 3 tầng (3-tiers) được coi là một kiến trúc phần mềm và là một mẫu thiết kế.” (dịch lại từ wikipedia tiếng Anh).

Đây là kiến trúc triển khai ứng dụng ở mức vật lý. Kiến trúc gồm 3 module chính và riêng biệt:

- Tầng Presentation:** hiển thị các thành phần giao diện để tương tác với người dùng như tiếp nhận thông tin, thông báo lỗi, ...
- Tầng Business Logic:** thực hiện các hành động nghiệp vụ của phần mềm như tính toán, đánh giá tính hợp lệ của thông tin, ... Tầng này còn di chuyển, xử lý thông tin giữa 2 tầng trên dưới.
- Tầng Data:** nơi lưu trữ và trích xuất dữ liệu từ các hệ quản trị CSDL hay các file trong hệ thống. Cho phép tầng Business logic thực hiện các truy vấn dữ liệu .

Mọi người vẫn hay nhầm lẫn giữa tier và layer vì cấu trúc phân chia giống nhau (presentation, bussiness , data). Tuy nhiên, thực tế chúng hoàn toàn khác nhau. Nếu 3 tiers có tính vật lý thì 3 layer có tính logic. Nghĩa là ta phân chia ứng dụng thành các phần (các lớp) theo chức năng hoặc vai trò một cách logic. Các layer khác nhau được thực thi trong 1 phân vùng bộ nhớ của process. Vì thế nên một tier có thể có nhiều layer.

Mô hình 3-layer gồm có 3 phần chính:

– **Presentation Layer (GUI)** : Lớp này có nhiệm vụ chính giao tiếp với người dùng. Nó gồm các thành phần giao diện (win form, web form,...) và thực hiện các công việc như nhập liệu, hiển thị dữ liệu, kiểm tra tính đúng đắn dữ liệu trước khi gọi lớp Business Logic Layer (BLL).

– **Business Logic Layer (BLL)** : Layer này phân ra 2 thành nhiệm vụ :

- Đây là nơi đáp ứng các yêu cầu thao tác dữ liệu của GUI layer, xử lý chính nguồn dữ liệu từ Presentation Layer trước khi truyền xuống Data Access Layer và lưu xuống hệ quản trị CSDL.
- Đây còn là nơi kiểm tra các ràng buộc, tính toàn vẹn và hợp lệ dữ liệu, thực hiện tính toán và xử lý các yêu cầu nghiệp vụ, trước khi trả kết quả về Presentation Layer.

– **Data Access Layer (DAL)** : Lớp này có chức năng giao tiếp với hệ quản trị CSDL như thực hiện các công việc liên quan đến lưu trữ và truy vấn dữ liệu (tìm kiếm, thêm, xóa, sửa,...).

Ưu điểm

- Việc phân chia thành từng lớp giúp cho code được tường minh hơn. Nhờ vào việc chia ra từng lớp đảm nhận các chức năng khác nhau và riêng biệt như giao diện, xử lý, truy vấn thay vì để tất cả lại một chỗ. Nhằm giảm sự kết dính.
- Dễ bảo trì khi được phân chia, thì một thành phần của hệ thống sẽ dễ thay đổi. Việc thay đổi này có thể được cô lập trong 1 lớp, hoặc ảnh hưởng đến lớp gần nhất mà không ảnh hưởng đến cả chương trình.
- Dễ phát triển, tái sử dụng: khi chúng ta muốn thêm một chức năng nào đó thì việc lập trình theo một mô hình sẽ dễ dàng hơn vì chúng ta đã có chuẩn để tuân theo. Và việc sử dụng lại khi có sự thay đổi giữa hai môi trường (Winform sang Webfrom) thì chỉ việc thay đổi lại lớp GUI.
- Dễ bàn giao. Nếu mọi người đều theo một quy chuẩn đã được định sẵn, thì công việc bàn giao, tương tác với nhau sẽ dễ dàng hơn và tiết kiệm được nhiều thời gian.
- Dễ phân phối khối lượng công việc. Mỗi một nhóm, một bộ phận sẽ nhận một nhiệm vụ trong mô hình 3 lớp. Việc phân chia rõ ràng như thế sẽ giúp các lập trình viên kiểm soát được khối lượng công việc của mình.

II. GIAO DIỆN PHẦN MỀM

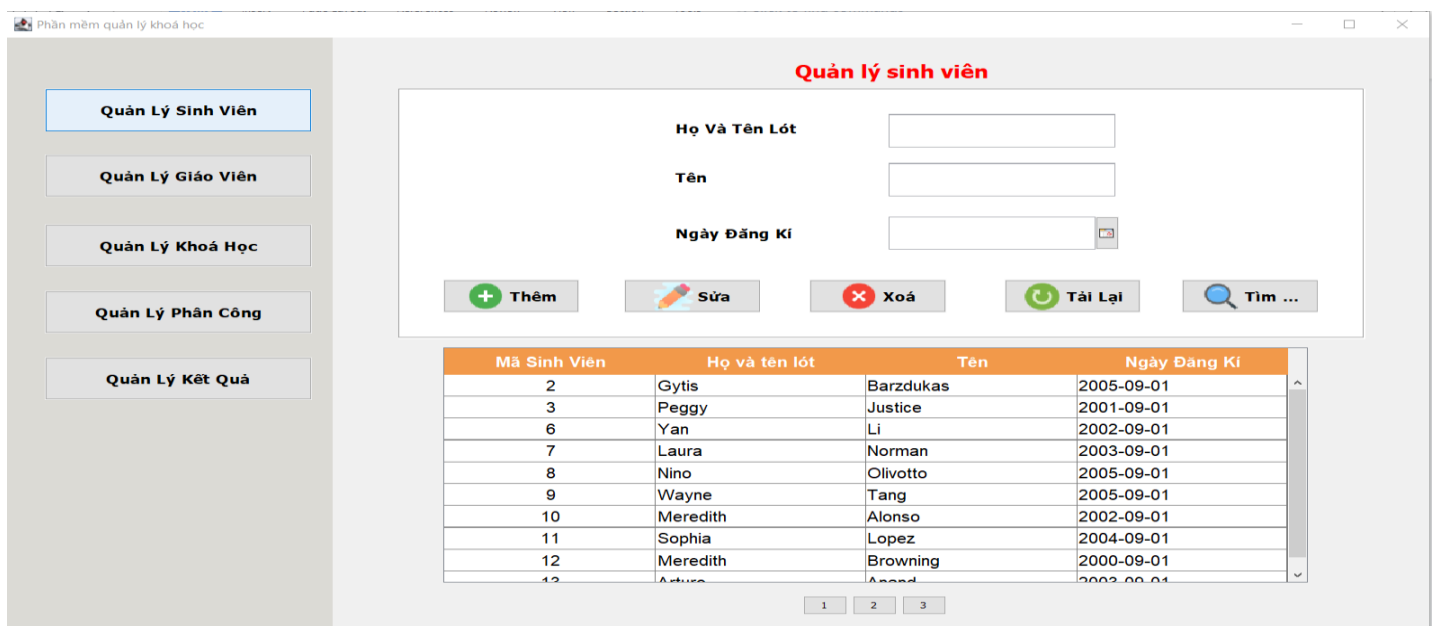
Do không yêu cầu về việc đăng nhập nên khi chạy chương trình người dùng sẽ thấy màn hình hiển thị giao diện dưới đây.

Trong Giao diện dưới gồm có 3 phần chính:

- Bên phải là các chức năng quản lý từng yêu cầu.
- Bên dưới là bảng hiển thị tất cả thông tin có trong cơ sở dữ liệu.
- Bên trên là các hành động có thể thực hiện của từng chức năng.

Tất cả các chức năng còn lại đều có chung một kiểu thiết kế như hình bên dưới.

1. Quản lý học viên








Phần mềm quản lý khoá học

Quản lý sinh viên

Họ Và Tên Lót

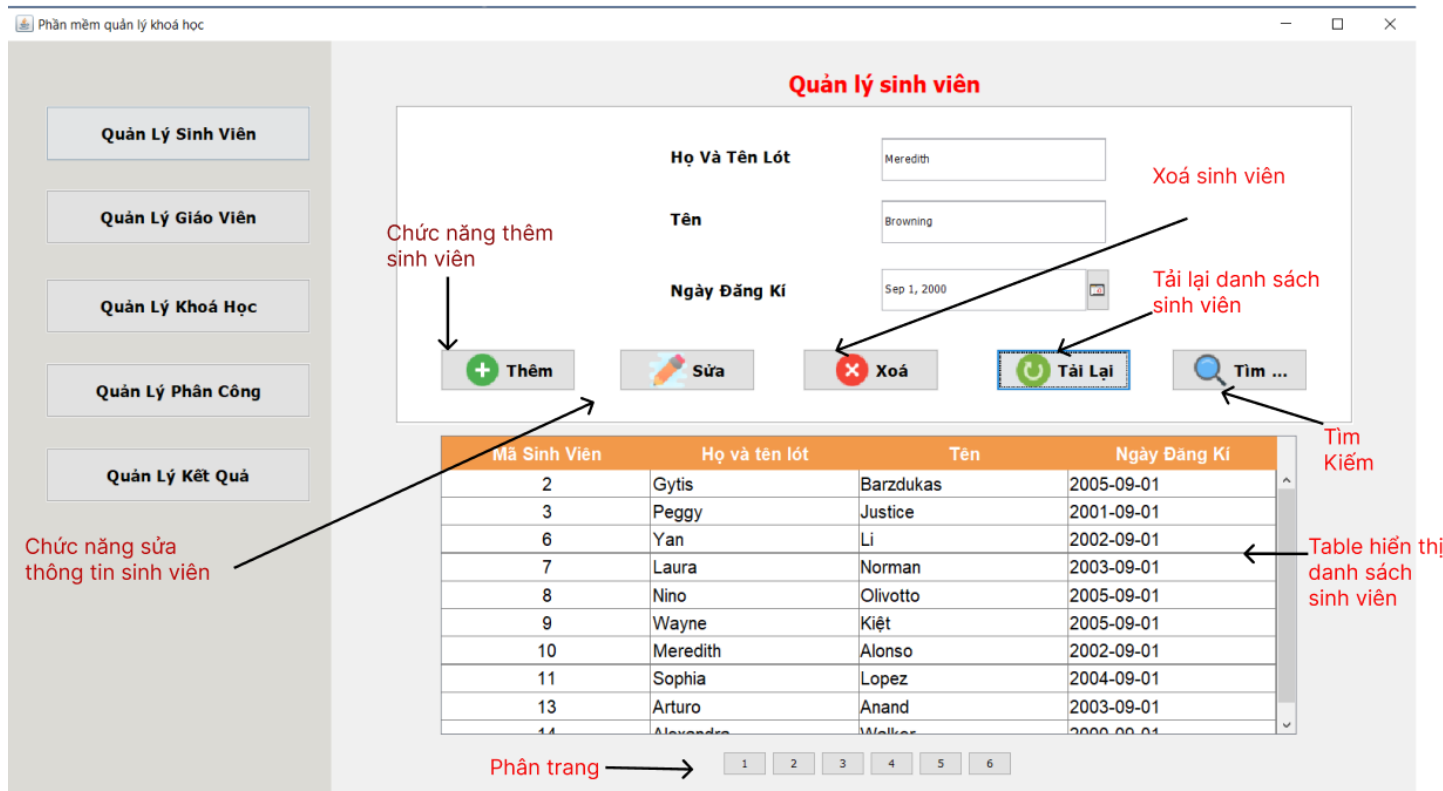
Tên

Ngày Đăng Kí

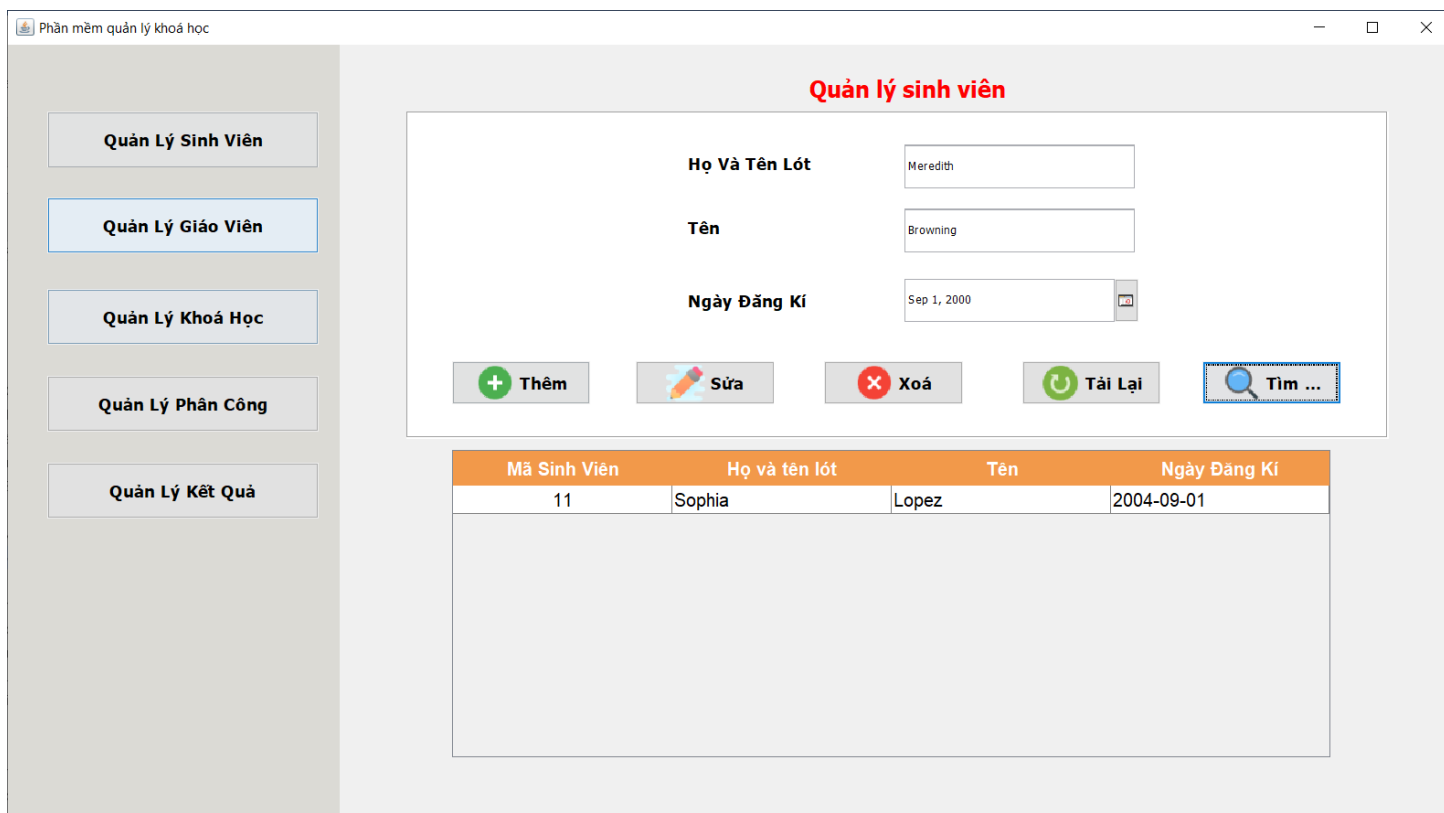
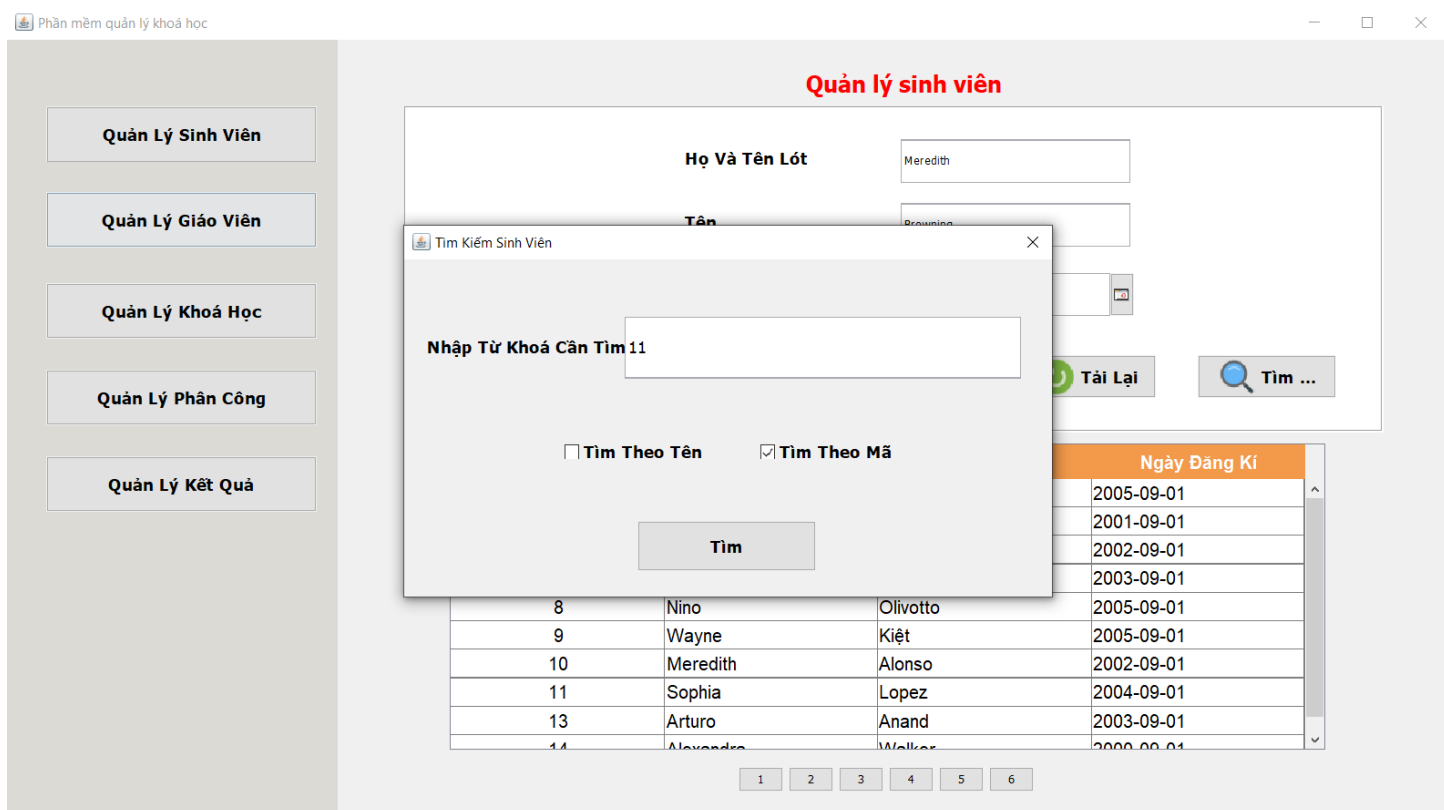
 Thêm  Sửa  Xóa  Tải Lại  Tìm ...

Mã Sinh Viên	Họ và tên lót	Tên	Ngày Đăng Kí
2	Gytis	Barzdukas	2005-09-01
3	Peggy	Justice	2001-09-01
6	Yan	Li	2002-09-01
7	Laura	Norman	2003-09-01
8	Nino	Olivotto	2005-09-01
9	Wayne	Tang	2005-09-01
10	Meredith	Alonso	2002-09-01
11	Sophia	Lopez	2004-09-01
12	Meredith	Browning	2000-09-01
13	Arturo	Arce	2002-09-01

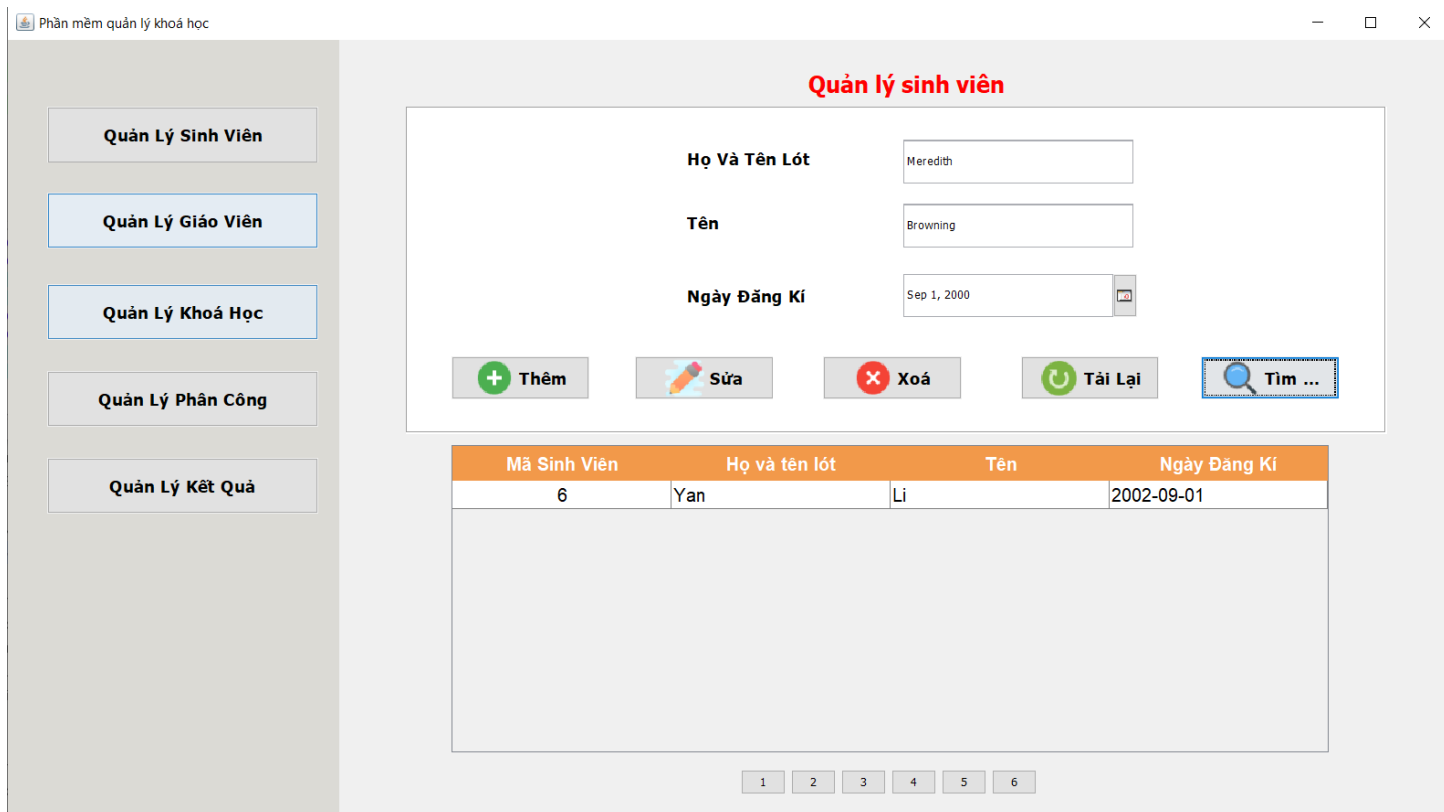
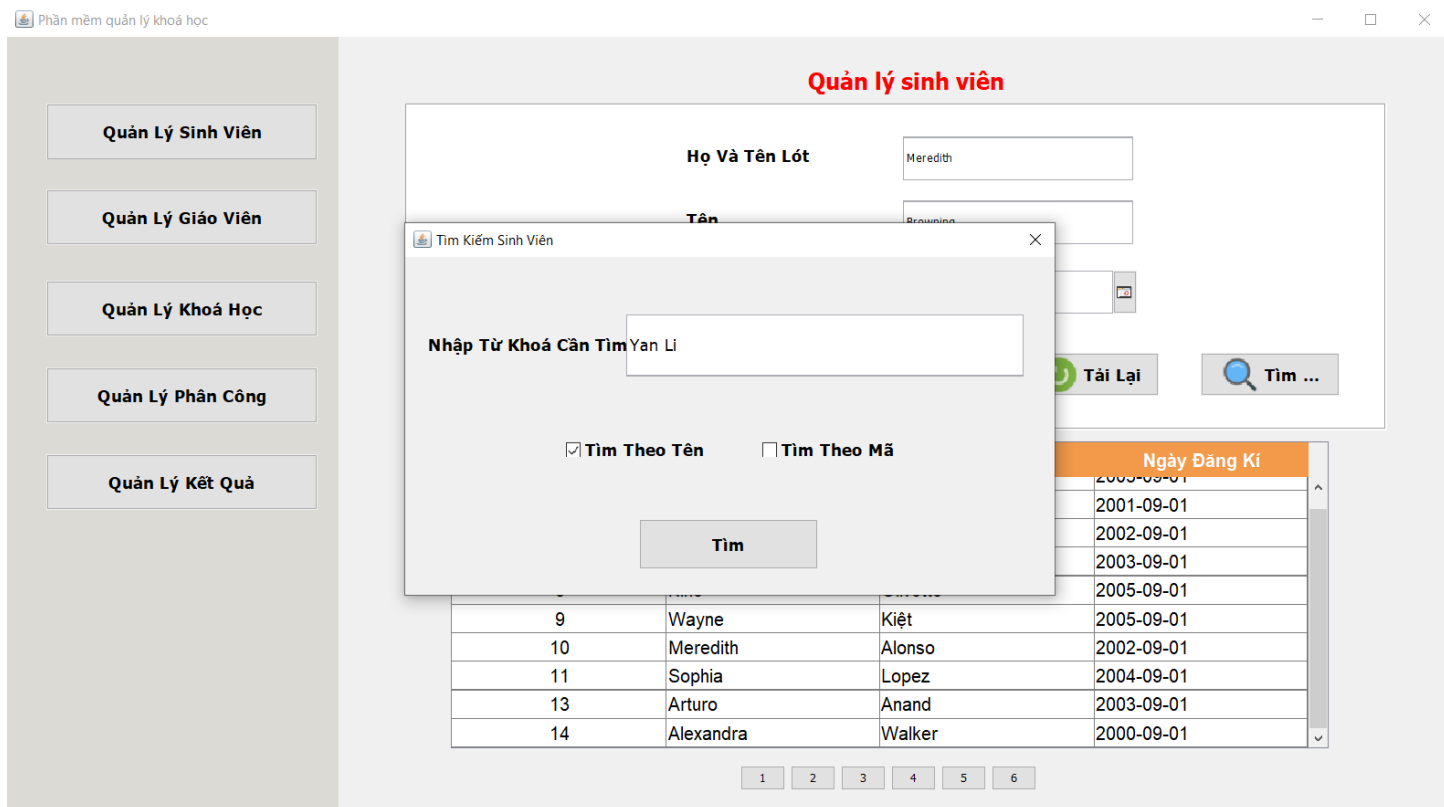
1 2 3



Giao diện quản lý sinh viên



Giao diện quản lý sinh viên – Chức năng tìm sinh viên theo mã



Giao diện quản lý sinh viên – Chức năng tìm sinh viên theo tên

2. Quản lý giảng viên

Phần mềm quản lý khoa học

Quản Lý Sinh Viên

Quản Lý Giáo Viên

Quản Lý Khoá Học

Quản Lý Phân Công

Quản Lý Kết Quả

Quản lý giáo viên

Họ Và Tên Lót

Tên

Ngày Thuê

Thêm

Sửa

Xoá

Tải Lại

Tìm ...

Mã Giáo Viên	Họ và Chữ Lót	Tên	Ngày Thuê
1	Kim	Abercrombie	1995-03-11
4	Fadi	Fakhouri	2002-08-06
5	Roger	Harui	1998-07-01
18	Roger	Zheng	2004-02-12
25	Candace	Kapoor	2001-01-15
27	Stacy	Serrano	1999-06-01
31	Jasmine	Stewart	1997-10-12
32	Kristen	Xu	2001-07-23
34	Roger	Van Houten	2000-12-07

1

Phần mềm quản lý khoa học

Quản Lý Sinh Viên

Quản Lý Giáo Viên

Quản Lý Khoá Học

Quản Lý Phân Công

Quản Lý Kết Quả

Quản lý giáo viên

Họ Và Tên Lót

Tên

Ngày Thuê

Thêm

Sửa

Xoá

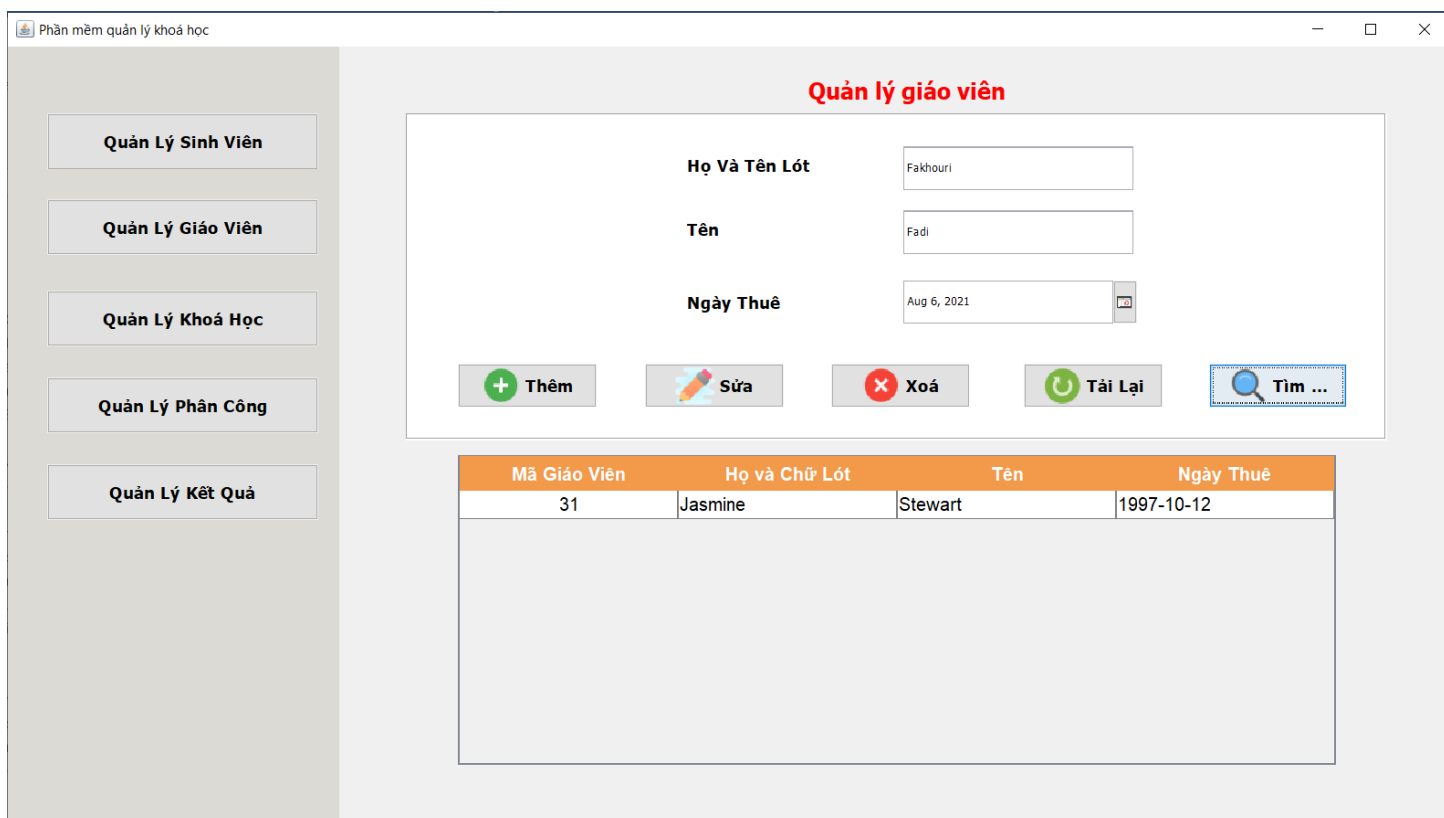
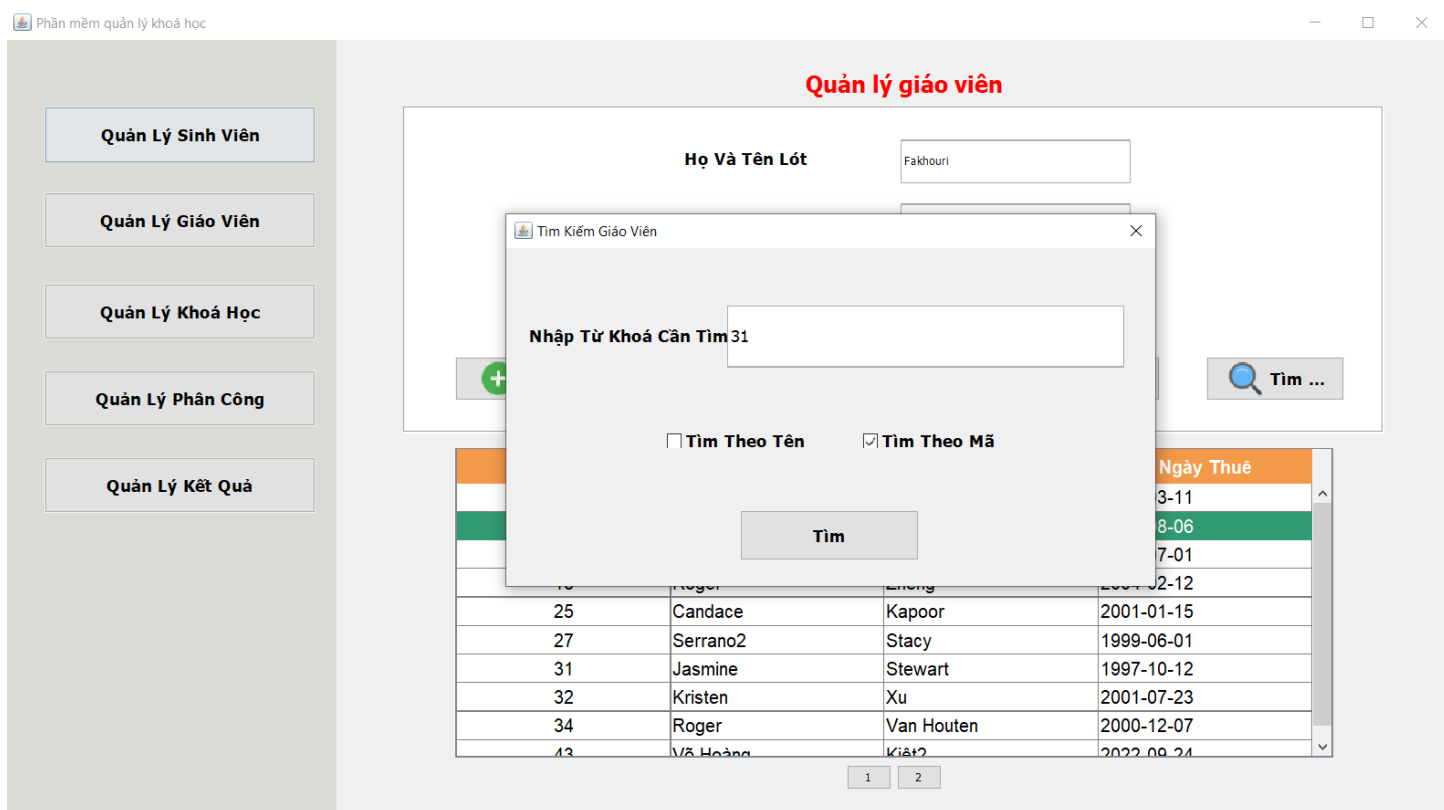
Tải Lại

Tìm ...

Mã Giáo Viên	Họ và Chữ Lót	Tên	Ngày Thuê
1	Abercrombie	Kim	1995-03-11
4	Fakhouri	Fadi	2021-08-06
5	Roger	Harui	1998-07-01
18	Roger	Zheng	2004-02-12
25	Candace	Kapoor	2001-01-15
27	Serrano2	Stacy	1999-06-01
31	Jasmine	Stewart	1997-10-12
32	Kristen	Xu	2001-07-23
34	Roger	Van Houten	2000-12-07
43	Vũ Hoàng	Kiệt?	2022-09-24

1 2

Giao diện quản lý giáo viên



Giao diện quản lý giáo viên – Chức năng tìm kiếm

3. Quản lý khóa học

Phần mềm quản lý khóa học

Quản Lý Khóa Học

Quản Lý Sinh Viên

Quản Lý Giáo Viên

Quản Lý Khóa Học

Quản Lý Phân Công

Quản Lý Kết Quả

Tên KH

Khoa: 1 - Engineering

URL

Địa Điểm

Ngày

Giờ: 00:00:00

Thêm

Sửa

Xoá

Tải ...

Tim

Khoá Học Onsite

STT	Mã K...	Tên K...	Khoa	Phòng	Ngày	Giờ
1	1045	Calculus	Mathe...	121 Sm...	MWHF	15:30:00
2	1050	Chemis...	Econo...	123 Sm...	MTWH	11:30:00
3	1061	Physics	Econo...	234 Sm...	TWHF	13:15:00
4	2042	Literature	Mathe...	225 Ad...	MTWH	11:00:00
5	4022	Microe...	Econo...	23 Willi...	MWF	09:00:00

Khoá Học Online

STT	Mã Kh...	Tên K...	Khoa	Url
1	2021	Compo...	English	http://w...
2	2030	Poetry	English	http://w...
3	3141	Trigono...	Mathem...	http://w...
4	4041	Macro...	Econom...	http://w...

Chức năng thêm khoá học (Thêm)

Chức năng sửa thông tin khoá học (Sửa)

Chức năng xóa khoá học (Xoá)

Tải lại danh sách khoá học (Tải ...)

Chức năng tìm kiếm (Tim)

Table hiển thị danh sách khoá học onsite

Table hiển thị danh sách khoá học online

Phân trang

Giao diện quản lý khóa học

Phần mềm quản lý khóa học

Quản Lý Khóa Học

Quản Lý Sinh Viên

Quản Lý Giáo Viên

Quản Lý Khóa Học

Quản Lý Phân Công

Quản Lý Kết Quả

Tên KH

Khoa

URL

Địa Điểm

Ngày

Giờ

Thêm

Tìm

Khoá Học Onsite

STT	Mã K...	Tên K...	Khoa	Phòng	Ngày	Giờ
1	1045	Calculus	Mathe...	121 Sm...	MWHF	15:30:00
2	1050	Chemis...	Econo...	123 Sm...	MTWH	11:30:00
3	1061	Physics	Econo...	234 Sm...	TWHF	13:15:00
4	2042	Literature	Mathe...	225 Ad...	MTWH	11:00:00
5	4022	Microe...	Econo...	23 Willi...	MWF	09:00:00

Khoá Học Online

STT	Mã Kh...	Tên K...	Khoa	Url
1	2021	Compo...	English	http://w...
2	2030	Poetry	English	http://w...
3	3141	Trigono...	Mathem...	http://w...
4	4041	Macro...	Econom...	http://w...

Tim Kiếm Khoá Học

Nhập Mã Khoá Học

☐ Tim Khoa Hoc Onsite ☒ Tim Khoa Hoc Online

Tim

Phản mềm quản lý khoá học

Quản Lý Sinh Viên

Quản Lý Giáo Viên

Quản Lý Khoá Học

Quản Lý Phân Công

Quản Lý Kết Quả

Quản Lý Khoá Học

Tên KH

Composition

Địa Điểm

Khoa

2 - English

Ngày

URL

http://www.fineartschool.net/Composition

Giờ

00:00:00

Thêm

Sửa

Xoá

Tải ...

Tim

Khoá Học Onsite

STT	Mã K...	Tên Khoá...	Khoa	Phòng	Ngày	Giờ
1	1045	Calculus	Mathe...	121 S...	MWHF	15:30:00
2	1050	Chemistry	Econo...	123 S...	MTWH	11:30:00
3	1061	Physics	Econo...	234 S...	TWHF	13:15:00
4	2042	Literature	Mathe...	225 A...	MTWH	11:00:00
5	4022	Microecon...	Econo...	23 Will...	MWF	09:00:00

1

Khoá Học Online

STT	Mã Kh...	Tên K...	Khoa	Url
1	2021	Compo...	English	http://w...
2	2021	Compo...	English	http://w...
3	2021	Compo...	English	http://w...
4	2021	Compo...	English	http://w...

Danh sách sau khi tìm kiếm

1

Giao diện quản lý khoá học – Chức năng tìm kiếm

16

4. Quản lý phân công

Phần mềm quản lý khoá học

Quản Lý Khoá Học

Tên KH: Địa Điểm:

Khoa: 1 - Engineering Ngày:

URL: Giờ: 00:00:00

Chức năng thêm khoá học

Thêm Sửa Xóa Tải ... Tìm

Chức năng sửa thông tin khoá học

Table hiển thị danh sách khoá học onsite

Phân trang 1

Table hiển thị danh sách khoá học online

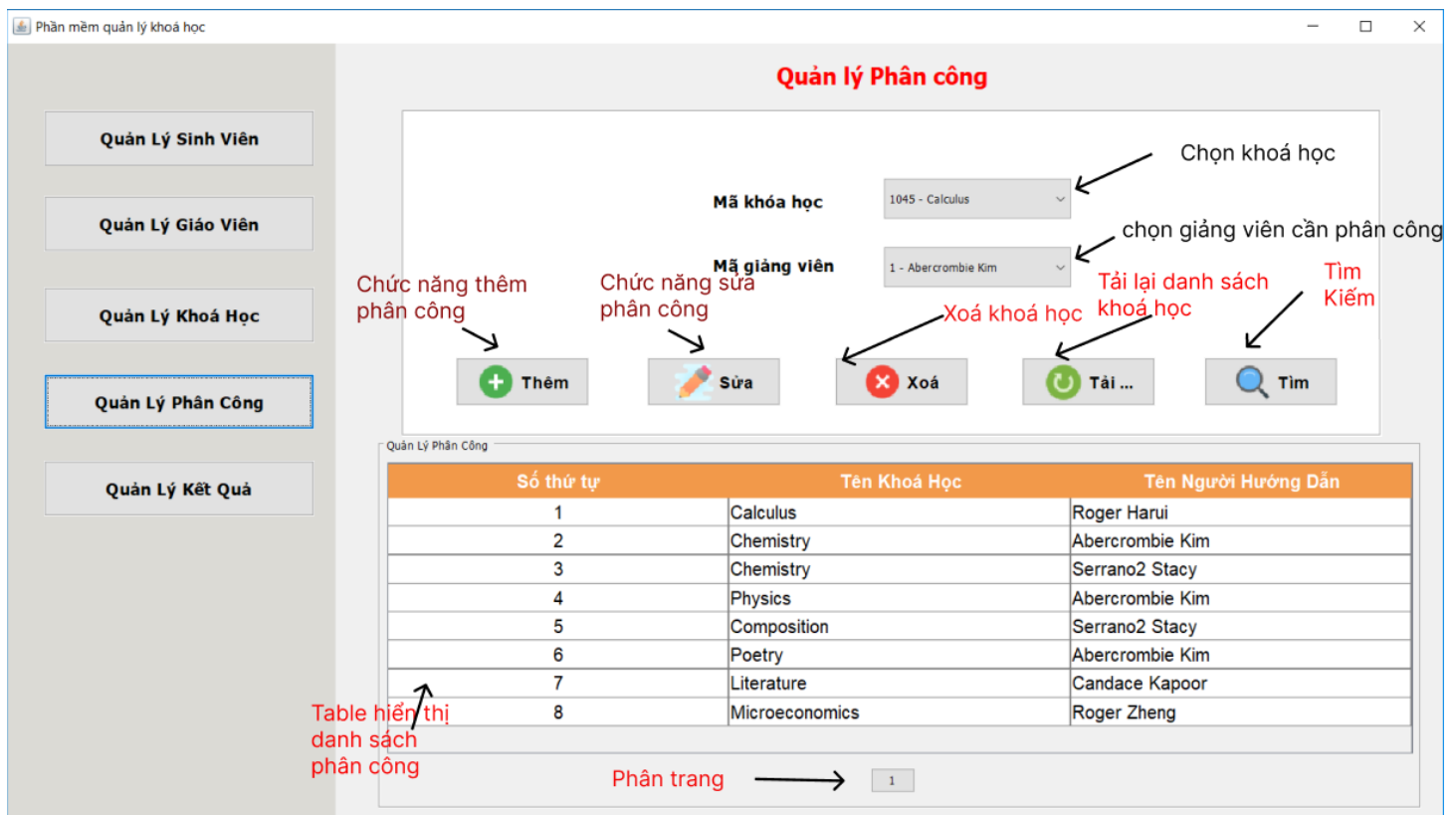
Chức năng xóa khoá học

Tải lại danh sách khoá học

Tìm Kiếm

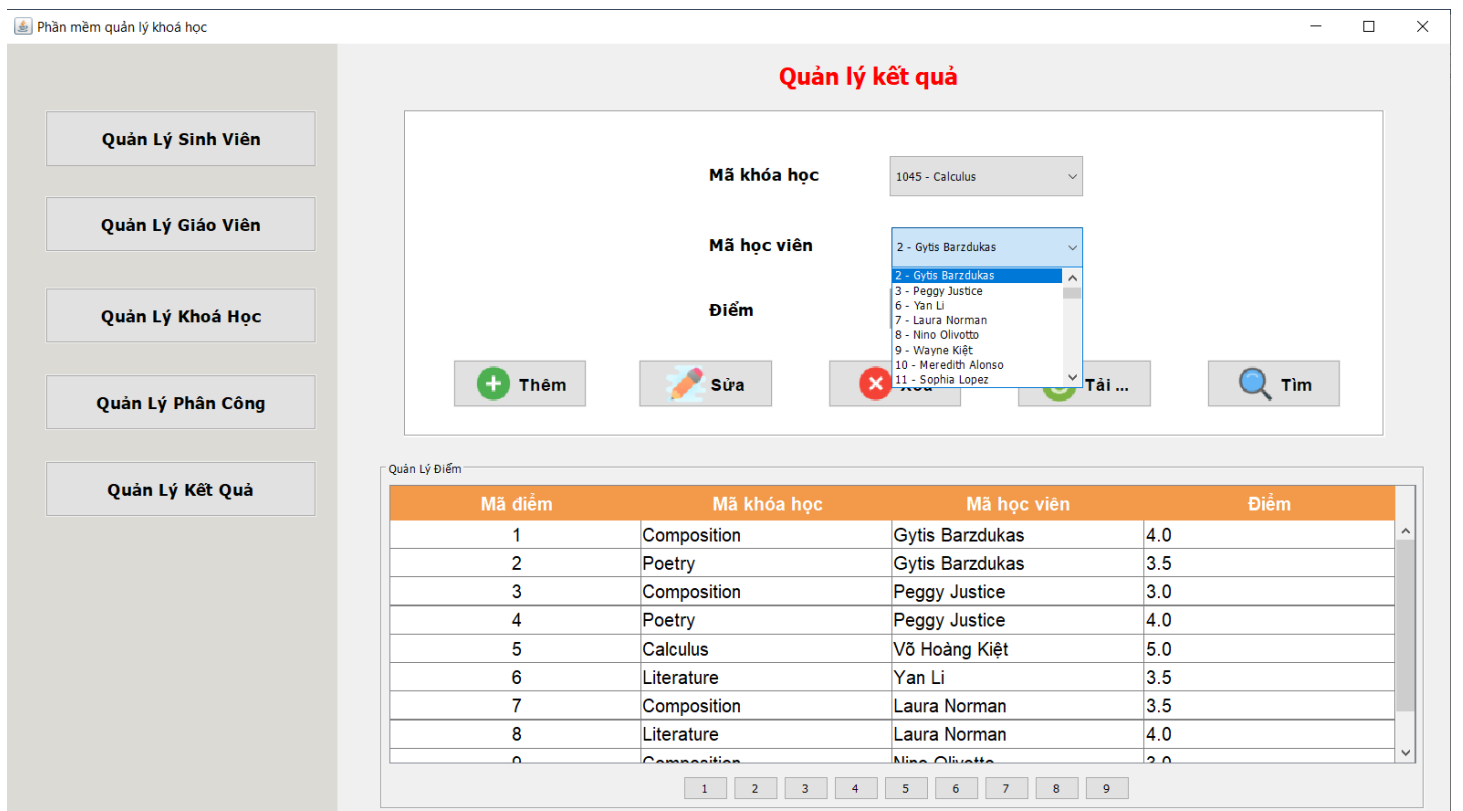
STT	Mã K...	Tên K...	Khoa	Phòng	Ngày	Giờ
1	1045	Calculus	Mathe...	121 Sm...	MWHF	15:30:00
2	1050	Chemis...	Econo...	123 Sm...	MTWH	11:30:00
3	1061	Physics	Econo...	234 Sm...	TWHF	13:15:00
4	2042	Literature	Mathe...	225 Ad...	MTWH	11:00:00
5	4022	Microe...	Econo...	23 Willi...	MWF	09:00:00

STT	Mã Kh...	Tên K...	Khoa	Url
1	2021	Compo...	English	http://w...
2	2030	Poetry	English	http://w...
3	3141	Trigono...	Mathem...	http://w...
4	4041	Macro...	Econom...	http://w...



Giao diện quản lý phân công

5. Quản lý kết quả



Phần mềm quản lý khoá học

Quản lý kết quả

Quản Lý Sinh Viên

Quản Lý Giáo Viên

Quản Lý Khoá Học

Quản Lý Phân Công

Quản Lý Kết Quả

Mã khoá học 1045 - Calculus ← Chọn khoá học

Mã học viên 2 - Gytis Barzdukas ← chọn học viên

Điểm ← Nhập điểm

Thêm điểm Sửa điểm Xóa điểm Tải lại danh sách điểm Tìm Kiểm

Thêm Sửa Xóa Tải ... Tìm

Table hiển thị danh sách điểm

Phân trang

Mã điểm	Mã khoá học	Mã học viên	Điểm
1	Composition	Gytis Barzdukas	4.0
2	Poetry	Gytis Barzdukas	3.5
3	Composition	Peggy Justice	3.0
4	Poetry	Peggy Justice	4.0
5	Calculus	Võ Hoàng Kiệt	5.0
6	Literature	Yan Li	3.5
7	Composition	Laura Norman	3.5
8	Literature	Laura Norman	4.0
9	Composition	Nico Olivette	3.0

Giao diện quản lý kết quả

Phần mềm quản lý khoá học

Quản lý kết quả

Quản Lý Sinh Viên

Quản Lý Giáo Viên

Quản Lý Khoá Học

Quản Lý Phân Công

Quản Lý Kết Quả

Mã khoá học 1045 - Calculus

Mã học viên 2 - Gytis Barzdukas

Điểm

Tìm Kiểm

Tìm

Tìm Kiểm Phân Công

Nhập Từ Khoá Cần Tìm 2

☒ Tìm Theo Mã Học Viên

Tìm

Table hiển thị danh sách điểm

Mã điểm	Mã khoá học	Mã học viên	Điểm
4	Poetry	Peggy Justice	4.0
5	Calculus	Võ Hoàng Kiệt	5.0
6	Literature	Yan Li	3.5
7	Composition	Laura Norman	3.5
8	Literature	Laura Norman	4.0
9	Composition	Nico Olivette	3.0

Phần mềm quản lý khoá học

Quản Lý Sinh Viên

Quản Lý Giáo Viên

Quản Lý Khoá Học

Quản Lý Phân Công

Quản Lý Kết Quả

kết quả sau khi
lọc

Quản lý kết quả

Mã khóa học

1045 - Calculus

Mã học viên

2 - Gytis Barzdukas

Điểm

+

Thêm

Sửa

×

Xoá

↺

Tải ...

Tìm

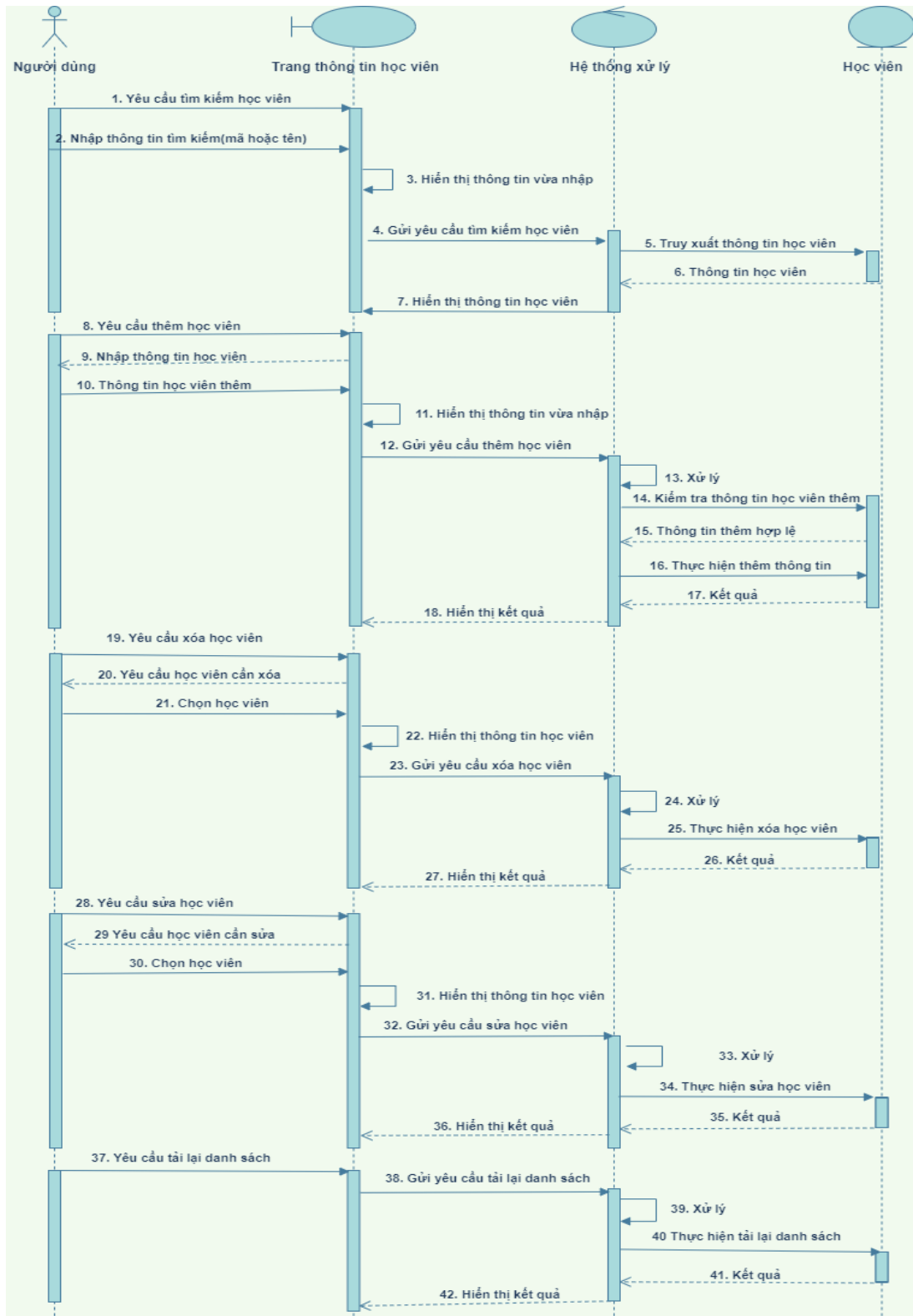
Quản Lý Điểm

Mã điểm	Mã khóa học	Mã học viên	Điểm
1	Composition	Gytis Barzdukas	4.0
2	Poetry	Gytis Barzdukas	3.5
48	Calculus	Gytis Barzdukas	7.0
49	Calculus	Gytis Barzdukas	1.0
52	Calculus	Gytis Barzdukas	1.0
53	Calculus	Gytis Barzdukas	1.0
54	Calculus	Gytis Barzdukas	1.0
55	Calculus	Gytis Barzdukas	1.0
56	Calculus	Gytis Barzdukas	1.0

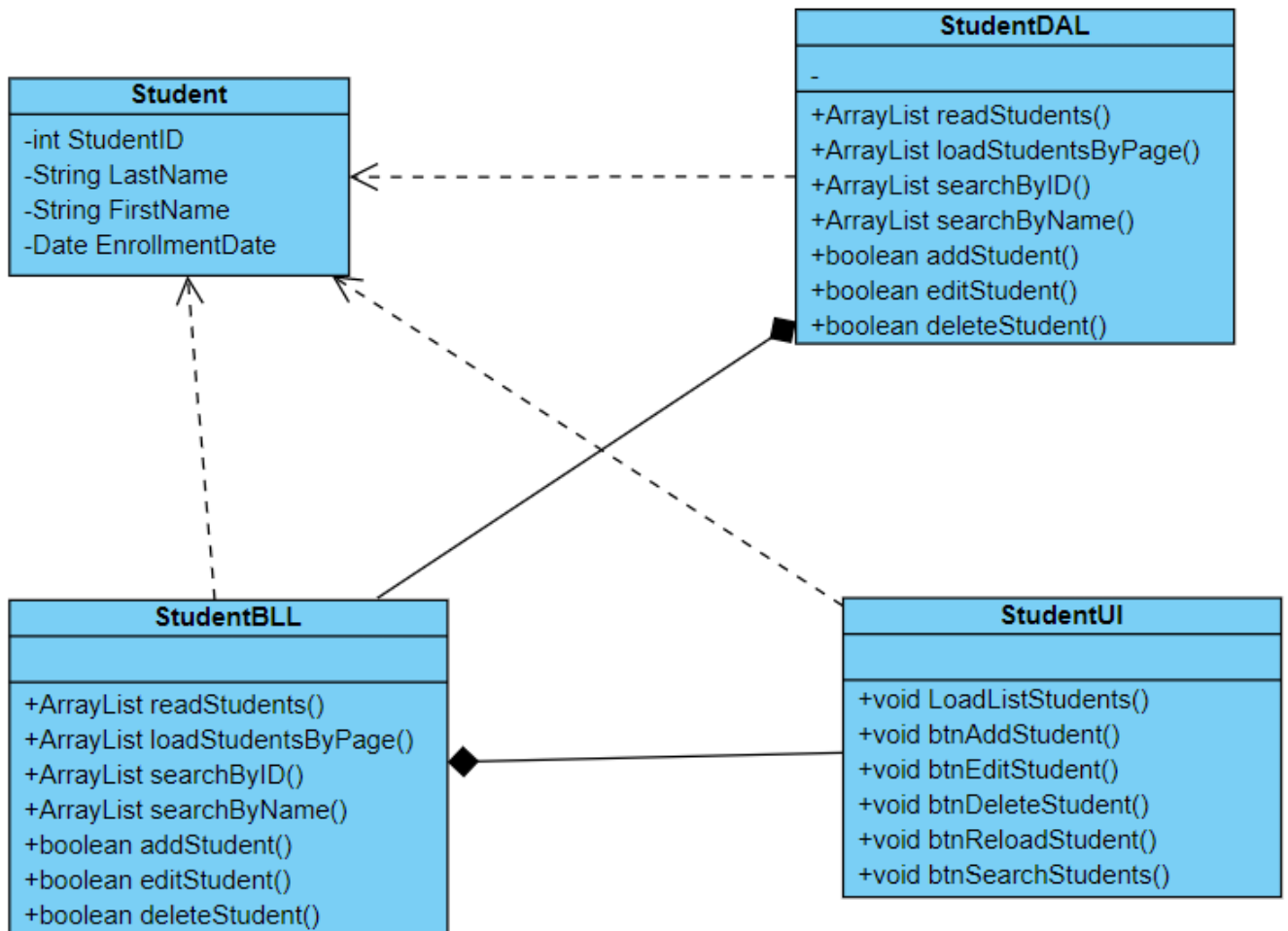
III. Code và Sơ đồ tuần tự

1. Quản lý sinh viên, giáo viên

1.1 Sinh viên



Sơ đồ tuần tự Quản lý sinh viên



Sơ đồ lớp Quản lý sinh viên

Code:

- Xử lý 1: Hiện thị danh sách sinh viên

DAL

```
// Đọc tất cả thông tin học viên
```

```

public ArrayList<Person> readStudents() {
    try {

        Statement stmt = conn.createStatement();
        ArrayList<Person> listStudents = new ArrayList<>();
        String query = "SELECT * FROM person WHERE HireDate IS NULL ";
        ResultSet rs = stmt.executeQuery(query);
        if (rs != null) {
            int i = 1;
            while (rs.next()) {
                int id = rs.getInt("PersonID");
                String lastName = rs.getString("Lastname");
                String FirstName = rs.getString("Firstname");
                Date HireDate = rs.getDate("HireDate");
                Date EnrollmentDate = rs.getDate("EnrollmentDate");
                Person person = new Person(id, lastName, FirstName, HireDate, EnrollmentDate);
                listStudents.add(person);
            }
        }

        return listStudents;
    } catch (Exception e) {
    }
    return null;
}

```

BLL

// Đọc tất cả thông tin học viên

```

public ArrayList<Person> readStudents() {
    return StudentDAL.gI().readStudents();
}

```

UI

// Hiển thị danh sách học viên

```

public void LoadListStudents() {
    studentsList = null;
    studentsList = BLL.StudentBLL.gI().readStudents();
    ArrayList<Person> arrayList = BLL.StudentBLL.gI().loadStudentsByPage(1);
    dtmStudent.setRowCount(0);
    cmbStudent.removeAllItems();
    for (Person person : arrayList) {
        Vector<Object> vec = new Vector<Object>();
        vec.add(person.getID());
        vec.add(person.getFirstname());
        vec.add(person.getLastname());
        vec.add(person.getEnrollmentDate());
        dtmStudent.addRow(vec);
    }
    // add combobox students
    for (Person person : studentsList) {
        cmbStudent.addItem(person);
    }
}

```

- Xử lý 2: Thêm sinh viên

DAL

// Thêm một học viên

```

public boolean addStudent(Person s) {
    try {
        String sql = "INSERT INTO `person`(`Lastname`, `Firstname`, `HireDate`, `EnrollmentDate`) VALUES (?, ?, ?, ?)";

        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, s.getLastname());
        pstmt.setString(2, s.getFirstname());
        // pstmt.setDate(3, new java.sql.Date(s.getHireDate().getTime()));
        pstmt.setDate(3, null);
        pstmt.setDate(4, new java.sql.Date(s.getEnrollmentDate().getTime()));
        int i = pstmt.executeUpdate();

        return i > 0;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return false;
}

```

BLL

// Thêm học viên

```

public boolean addStudents(Person person) {
    return StudentDAL.gI().addStudent(person);
}

```


UI

// Thêm học viên

```
// add Students
btnAddStudent.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String lastName = txtStudentLastName.getText();
        String firstName = txtStudentsFirstName.getText();
        Date dateenrollment = dateEnrollment.getDate();
        Person person = new Person(0, lastName, firstName, null, dateenrollment);
        if (person.getFirstname().isEmpty() || person.getLastname().isBlank()) {
            JOptionPane.showMessageDialog(null, "Bạn không được để trống tên");
            return;
        }
        if (person.getLastname().isEmpty() || person.getLastname().isBlank()) {
            JOptionPane.showMessageDialog(null, "Bạn không được để trống họ");
            return;
        }
        if (person.getEnrollmentDate() == null) {
            JOptionPane.showMessageDialog(null, "Bạn không được để trống ngày đăng kí");
            return;
        }
        if (StudentBLL.gI().addStudents(person)) {
            LoadListStudents();
            AddPageStudent();
            JOptionPane.showMessageDialog(null, "Đã thêm thành công");
        } else {
            JOptionPane.showMessageDialog(null, "Đã có lỗi xảy ra khi thêm giáo viên");
        }
    }
});
```

- Xử lý 3: Sửa sinh viên

DAL

// Sửa một học viên

```
public boolean editStudent(Person s) {
    try {
        String sql = "UPDATE `person` SET `Lastname`= ? ,`Firstname`= ? ,`HireDate`= ?,`EnrollmentDate`= ? WHERE PersonID = ?";

        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, s.getLastname());
        pstmt.setString(2, s.getFirstname());
        // pstmt.setDate(3, new java.sql.Date(s.getHireDate().getTime()));
        pstmt.setDate(3, null);
        pstmt.setDate(4, new java.sql.Date(s.getEnrollmentDate().getTime()));
        pstmt.setInt(5, s.getID());
        int i = pstmt.executeUpdate();

        return i > 0;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return false;
}
```

BLL

// Sửa học viên

```
public boolean editStudents(Person person) {

    return StudentDAL.gI().editStudent(person);
}
```

UI

```

btnEditStudent.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String lastName = txtStudentLastName.getText();
        String firstName = txtStudentsFirstName.getText();
        Date dateenrollment = dateEnrollment.getDate();

        int i = tblStudents.getSelectedRow();
        try {
            if (i >= 0) {
                int ID = Integer.parseInt(dtmStudent.getValueAt(i, 0).toString());
                Person person = new Person(ID, lastName, firstName, null, dateenrollment);
                if (person.getFirstname().isEmpty() || person.getLastname().isBlank()) {
                    JOptionPane.showMessageDialog(null, "Bạn không được để trống tên");
                    return;
                }
                if (person.getLastname().isEmpty() || person.getLastname().isBlank()) {
                    JOptionPane.showMessageDialog(null, "Bạn không được để trống họ");
                    return;
                }
                if (person.getEnrollmentDate() == null) {
                    JOptionPane.showMessageDialog(null, "Bạn không được để trống ngày đăng kí");
                    return;
                }
                if (StudentBLL.gI().editStudents(person)) {
                    LoadListStudents();
                    AddPageStudent();
                    JOptionPane.showMessageDialog(null, "Đã sửa thành công");
                } else {
                    JOptionPane.showMessageDialog(null, "Đã có lỗi xảy ra");
                }
            } else {
                JOptionPane.showMessageDialog(null, "Bạn chưa chọn học sinh cần sửa");
            }
        } catch (Exception e2) {
            // TODO: handle exception
            e2.printStackTrace();
        }
    }
});

```

- Xử lý 4: Xóa sinh viên

DAL

```
// Xóa một học viên
public boolean deleteStudent(int id) {
    try {
        String sql = "DELETE FROM `person` WHERE PersonID = ? ";

        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, id);
        int i = pstmt.executeUpdate();

        return i > 0;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return false;
}
```

BLL

```
// Xóa học viên

public boolean deleteLecture(int id) {
    return StudentDAL.gI().deleteStudent(id);
}
```

UI

```
btnDeleteStudent.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int i = tblStudents.getSelectedRow();
        try {
            if (i >= 0) {
                int ID = Integer.parseInt(dtmStudent.getValueAt(i, 0).toString());
                int dialogButton = JOptionPane.YES_NO_OPTION;
                int dialogResult = JOptionPane.showConfirmDialog(null, "Bạn có muốn xóa không?", "Warning",
                    dialogButton);
                if (dialogResult == JOptionPane.YES_OPTION) {
                    if (StudentBLL.gI().deleteLecture(ID)) {
                        LoadListStudents();
                        AddPageStudent();
                        JOptionPane.showMessageDialog(null, "Đã xóa thành công");
                    } else {
                        JOptionPane.showMessageDialog(null, "Đã có lỗi xảy ra khi xóa sinh viên");
                    }
                }
            }
            else {
                JOptionPane.showMessageDialog(null, "Bạn chưa chọn sinh viên cần xóa");
            }
        } catch (Exception e2) {
            // TODO: handle exception
            e2.printStackTrace();
        }
    }
});
```

- Xử lý 5: Tải lại danh sách sinh viên

DAL (giống với Xử lý 1)

BLL (giống với Xử lý 1)

UI

```
btnReloadStudents.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {
            LoadListStudents();
            AddPageStudent(); // xử lý phân trang
            JOptionPane.showMessageDialog(null, "Đã tải lại danh sách thành công");
        } catch (Exception e2) {
            e2.printStackTrace();
            JOptionPane.showMessageDialog(null, "Có lỗi khi tải danh sách giáo viên");
        }
    }
});
```

- Xử lý 6: Tìm kiếm sinh viên

DAL

```
// Tìm kiếm bằng mã
public ArrayList<Person> searchByID(int ID) {
    try {
        Statement stmt = conn.createStatement();
        ArrayList<Person> listStudents = new ArrayList<>();
        String query = "SELECT * FROM person WHERE HireDate IS NULL AND PersonID =?";
        PreparedStatement pstmt = conn.prepareStatement(query);
        pstmt.setInt(1, ID);
        ResultSet rs = pstmt.executeQuery();
        if (rs != null) {
            while (rs.next()) {
                int id = rs.getInt("PersonID");
                String lastName = rs.getString("Lastname");
                String Firstname = rs.getString("Firstname");
                Date HireDate = rs.getDate("HireDate");
                Date EnrollmentDate = rs.getDate("EnrollmentDate");
                Person person = new Person(id, lastName, Firstname, HireDate, EnrollmentDate);
                listStudents.add(person);
            }
        }
        return listStudents;
    } catch (Exception e) {
    }
    return null;
}
```

```
// Tìm kiếm bằng tên
public ArrayList<Person> searchByName(String name) {
    try {

        Statement stmt = conn.createStatement();
        ArrayList<Person> listStudents = new ArrayList<>();
        String query = "SELECT * FROM person WHERE HireDate IS NULL AND concat(FirstName, ' ', LastName) like ?";
        PreparedStatement pstmt = conn.prepareStatement(query);
        pstmt.setString(1, "%" + name + "%");
        ResultSet rs = pstmt.executeQuery();
        if (rs != null) {
            while (rs.next()) {
                int id = rs.getInt("PersonID");
                String lastName = rs.getString("Lastname");
                String FirstName = rs.getString("Firstname");
                Date HireDate = rs.getDate("HireDate");
                Date EnrollmentDate = rs.getDate("EnrollmentDate");
                Person person = new Person(id, lastName, FirstName, HireDate, EnrollmentDate);
                listStudents.add(person);
            }
        }

        return listStudents;
    } catch (Exception e) {
    }
    return null;
}
}
```

BLL

```
// Tìm kiếm theo mã
public ArrayList<Person> searchByID(int ID) {
    return StudentDAL.gI().searchByID(ID);
}

// Tìm kiếm theo tên
public ArrayList<Person> searchByName(String name) {
    return StudentDAL.gI().searchByName(name);
}
}
```

UI

dlgSearchStudent.java

```

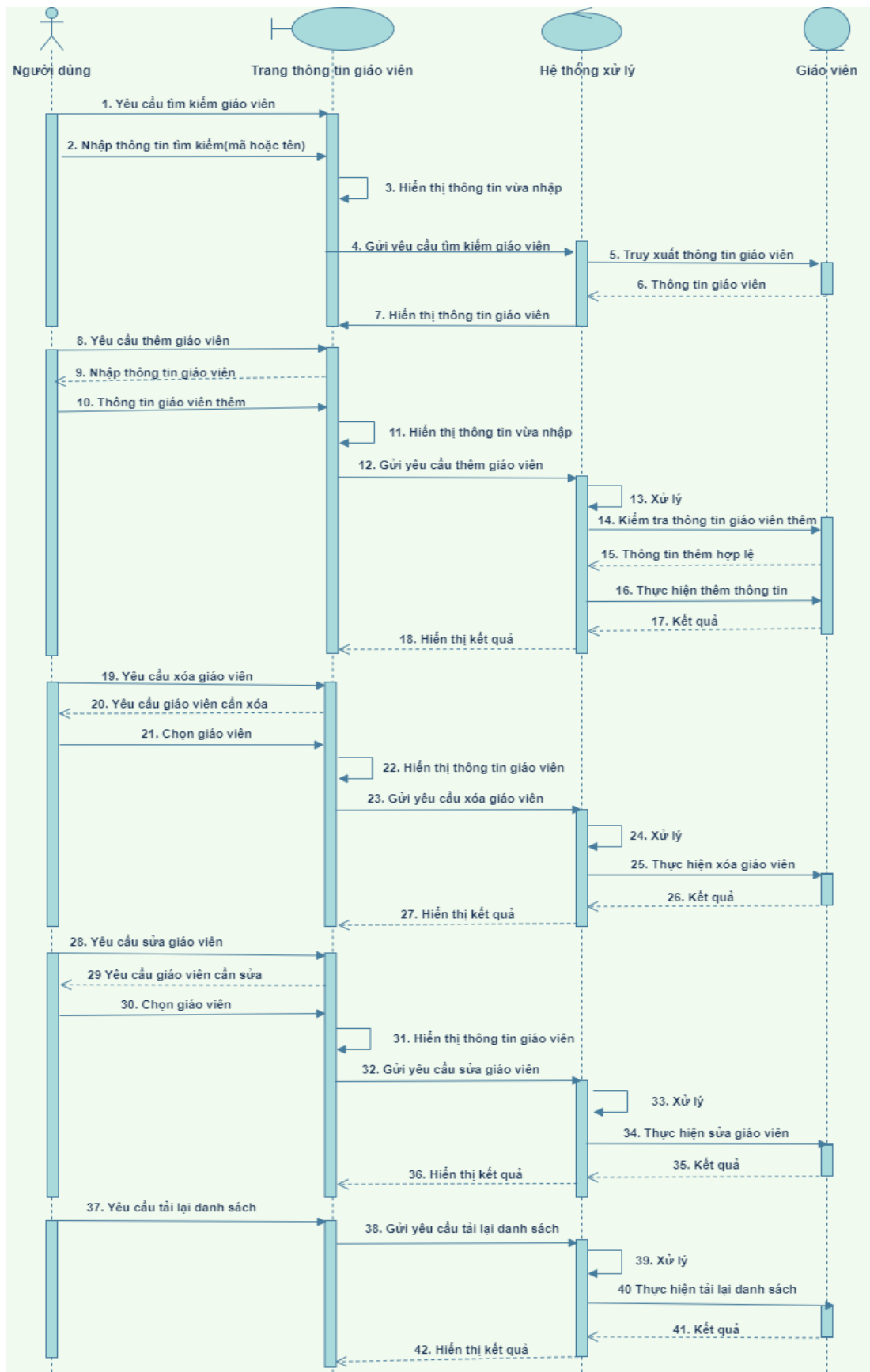
protected void SearchByName() {
    String Name = txtKeyWord.getText().trim();
    if (!isNumber(Name)) {

        ArrayList<Person> listStudents = new BLL.StudentBLL().searchByName(Name);
        if (listStudents != null) {
            MainFrame.dtmStudent.setRowCount(0);
            for (Person person : listStudents) {
                Vector<Object> vec = new Vector<Object>();
                vec.add(person.getID());
                vec.add(person.getFirstname());
                vec.add(person.getLastname());
                vec.add(person.getEnrollmentDate());
                MainFrame.dtmStudent.addRow(vec);
                this.setVisible(false);
            }
        }
        else {
            JOptionPane.showMessageDialog(null, "Không tìm thấy");
        }
    }
}

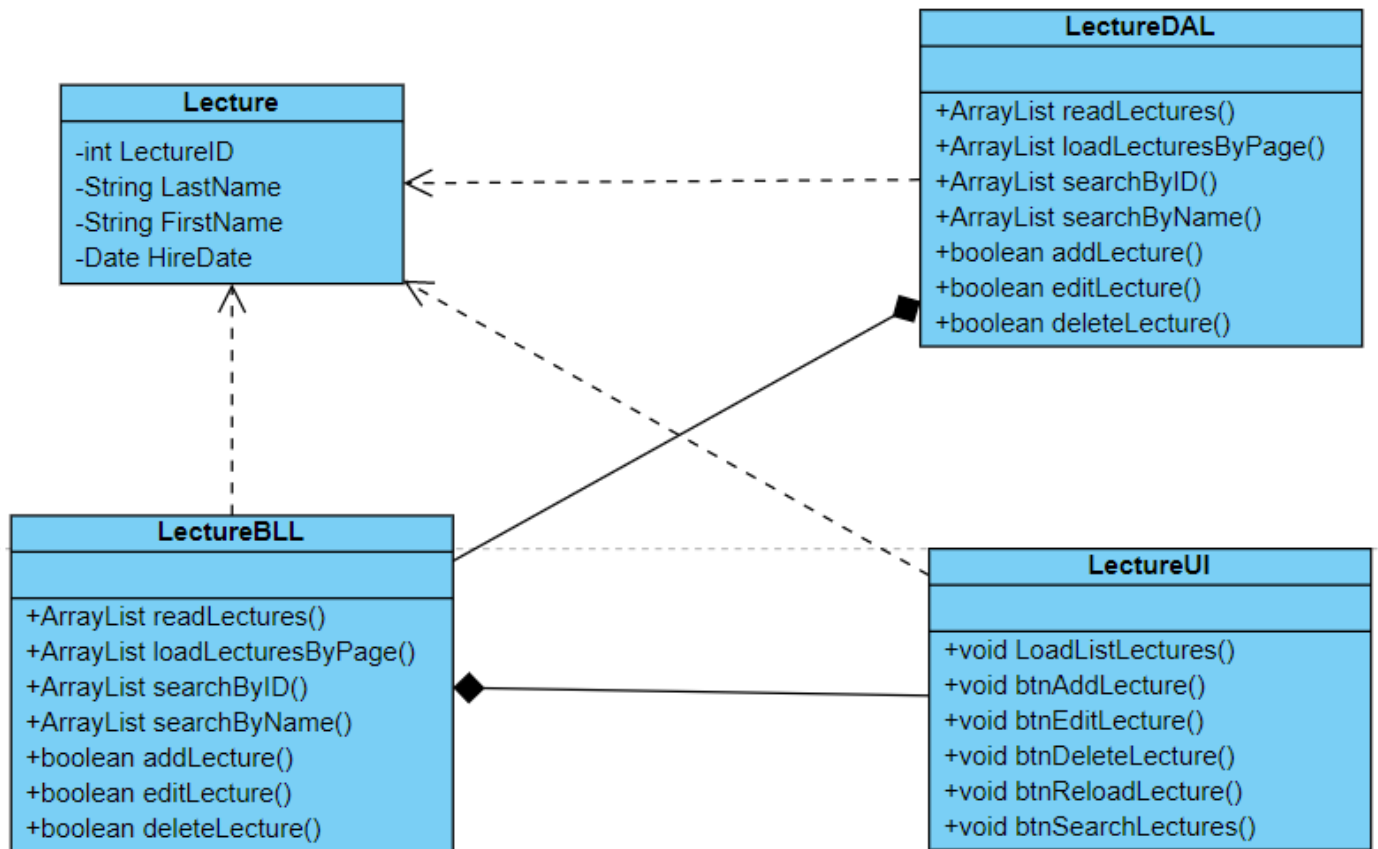
private void SearchByID() {
    String ID = txtKeyWord.getText().trim();
    if (isNumber(ID)) {
        System.out.println("Ok");
        ArrayList<Person> listStudents = new BLL.StudentBLL().searchByID(Integer.parseInt(ID));
        System.out.println(listStudents);
        if (listStudents != null) {
            MainFrame.dtmStudent.setRowCount(0);
            repaintPage();
            for (Person person : listStudents) {
                Vector<Object> vec = new Vector<Object>();
                vec.add(person.getID());
                vec.add(person.getFirstname());
                vec.add(person.getLastname());
                vec.add(person.getEnrollmentDate());
                MainFrame.dtmStudent.addRow(vec);
                this.setVisible(false);
            }
        }
        else {
            JOptionPane.showMessageDialog(null, "Không tìm thấy");
        }
    }
    else {
        JOptionPane.showMessageDialog(null, "ID phải là số");
    }
}
}

```

1.2 Giáo viên



Sơ đồ tuần tự Quản lý giáo viên



Sơ đồ lớp Quản lý giáo viên

Code:

- Xử lý 1: Hiện thị danh sách giáo viên

DAL

```
// Đọc tất cả thông tin giáo viên
```

```

public ArrayList<Person> readLecture() {
    try {

        Statement stmt = conn.createStatement();
        ArrayList<Person> listStudents = new ArrayList<>();
        String query = "SELECT * FROM person WHERE HireDate IS NOT NULL ";
        ResultSet rs = stmt.executeQuery(query);
        if (rs != null) {
            int i = 1;
            while (rs.next()) {
                int id = rs.getInt("PersonID");
                String lastName = rs.getString("Lastname");
                String Firstname = rs.getString("Firstname");
                Date HireDate = rs.getDate("HireDate");
                Date EnrollmentDate = rs.getDate("EnrollmentDate");
                Person person = new Person(id, lastName, Firstname, HireDate, EnrollmentDate);
                listStudents.add(person);
            }
        }

        return listStudents;
    } catch (Exception e) {
    }
    return null;
}

```

BLL

// Đọc tất cả thông tin giáo viên

```

public ArrayList<Person> readLectures() {
    return LectureDAL.gI().readLecture();
}

```

UI

```

public void LoadListLecture() {
    lecturesList = null;
    lecturesList = BLL.LectureBLL.gI().readLectures();
    ArrayList<Person> arrayList = LectureBLL.gI().loadLecturesByPage(1);
    dtmLecture.setRowCount(0);
    for (Person person : arrayList) {
        Vector<Object> vec = new Vector<Object>();
        vec.add(person.getID());
        vec.add(person.getFirstname());
        vec.add(person.getLastname());
        vec.add(person.getHireDate());
        dtmLecture.addRow(vec);
        cmbLectureInstructor.addItem(person);
    }
    for (Person person : lecturesList) {
        cmbLectureInstructor.addItem(person);
    }
}

```

- Xử lý 2: Thêm giáo viên

DAL

```
// Thêm một giáo viên
public boolean addLecture(Person s) {
    try {
        String sql = "INSERT INTO `person`(`Lastname`, `Firstname`, `HireDate`, `EnrollmentDate`) VALUES (?, ?, ?, ?)";

        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, s.getLastname());
        pstmt.setString(2, s.getFirstname());
        pstmt.setDate(3, new java.sql.Date(s.getHireDate().getTime()));
        // pstmt.setDate(4, new java.sql.Date(s.getEnrollmentDate().getTime()));
        pstmt.setDate(4, null);
        int i = pstmt.executeUpdate();

        return i > 0;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return false;
}
```

BLL

```
// Thêm giáo viên
public boolean addLecture(Person person) {

    return LectureDAL.gI().addLecture(person);
}
```

UI

```

btnLectureAdd.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String lastName = txtLectureLastName.getText();
        String firstName = txtLectureFirstName.getText();
        Date dateLecture = dateLectures.getDate();
        // ID tự động tăng nên chèn số 0 hoặc null vào đầu của Constructor
        Person person = new Person(0, lastName, firstName, dateLecture, null);
        if (person.getFirstname().isEmpty() || person.getLastname().isBlank()) {
            JOptionPane.showMessageDialog(null, "Bạn không được để trống tên");
            return;
        }
        if (person.getLastname().isEmpty() || person.getLastname().isBlank()) {
            JOptionPane.showMessageDialog(null, "Bạn không được để trống họ");
            return;
        }
        if (person.getHireDate() == null) {
            JOptionPane.showMessageDialog(null, "Bạn không được để trống ngày thuê");
            return;
        }
        if (LectureBLL.gI().addLecture(person)) {
            LoadListLecture();
            addPageLecture();
            JOptionPane.showMessageDialog(null, "Đã thêm thành công");
        } else {
            JOptionPane.showMessageDialog(null, "Đã có lỗi xảy ra khi thêm giáo viên");
        }
    }
});

```

- Xử lý 3: Sửa giáo viên

DAL

// Sửa một giáo viên

```

public boolean editLecture(Person s) {
    try {
        String sql = "UPDATE `person` SET `Lastname`= ? ,`Firstname`= ? ,`HireDate`= ?,`EnrollmentDate`= ? WHERE PersonID = ?";

        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, s.getLastname());
        pstmt.setString(2, s.getFirstname());
        pstmt.setDate(3, new java.sql.Date(s.getHireDate().getTime()));
        // pstmt.setDate(4, new java.sql.Date(s.getEnrollmentDate().getTime()));
        pstmt.setDate(4, null);
        pstmt.setInt(5, s.getID());
        int i = pstmt.executeUpdate();

        return i > 0;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return false;
}

```

BLL

// Sửa giáo viên

```

public boolean editLecture(Person person) {

    return LectureDAL.gI().editLecture(person);
}

```

UI

```

btnLectureEdit.addActionListener(new ActionListener() {
    public JButton btnLectureEdit;
    String firstName = txtLectureFirstName.getText();
    Date dateLecture = dateLectures.getDate();
    int i = tblLectures.getSelectedRow();
    try {
        if (i >= 0) {
            int ID = Integer.parseInt(dtmLecture.getValueAt(i, 0).toString());
            Person person = new Person(ID, lastName, firstName, dateLecture, null);
            if (person.getFirstname().isEmpty() || person.getLastname().isBlank()) {
                JOptionPane.showMessageDialog(null, "Bạn không được để trống tên");
                return;
            }
            if (person.getLastname().isEmpty() || person.getLastname().isBlank()) {
                JOptionPane.showMessageDialog(null, "Bạn không được để trống họ");
                return;
            }
            if (person.getHireDate() == null) {
                JOptionPane.showMessageDialog(null, "Bạn không được để trống ngày thuê");
                return;
            }
            if (LectureBLL.gI().editLecture(person)) {
                LoadListLecture();
                addPageLecture();
                JOptionPane.showMessageDialog(null, "Đã sửa thành công");
            } else {
                JOptionPane.showMessageDialog(null, "Đã có lỗi xảy ra");
            }
        } else {
            JOptionPane.showMessageDialog(null, "Bạn chưa chọn giáo viên cần sửa");
        }
    } catch (Exception e2) {
        // TODO: handle exception
        e2.printStackTrace();
    }
});

```

- Xử lý 4: Xóa sinh viên

DAL

// Xóa một giáo viên

```

public boolean deleteLecture(int id) {
    try {
        String sql = "DELETE FROM `person` WHERE PersonID = ? ";

        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, id);
        int i = pstmt.executeUpdate();

        return i > 0;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return false;
}

```

BLL

// Xóa giáo viên

```

public boolean deleteLecture(int id) {

    return LectureDAL.gI().deleteLecture(id);
}

```

UI

// delete Lecture

```

btnLectureDelete.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int i = tblLectures.getSelectedRow();
        try {
            if (i >= 0) {
                int ID = Integer.parseInt(dtmLecture.getValueAt(i, 0).toString());
                int dialogButton = JOptionPane.YES_NO_OPTION;
                int dialogResult = JOptionPane.showConfirmDialog(null, "Bạn có muốn xóa không?", "Warning",
                    dialogButton);
                if (dialogResult == JOptionPane.YES_OPTION) {
                    if (LectureBLL.gI().deleteLecture(ID)) {
                        LoadListLecture();
                        addPageLecture();
                        JOptionPane.showMessageDialog(null, "Đã xóa thành công");
                    } else {
                        JOptionPane.showMessageDialog(null, "Đã có lỗi xảy ra");
                    }
                }
            }
            else {
                JOptionPane.showMessageDialog(null, "Bạn chưa chọn giáo viên cần xóa");
            }
        } catch (Exception e2) {
            // TODO: handle exception
            e2.printStackTrace();
        }
    }
});

```

- Xử lý 5: Tải lại danh sách giáo viên

DAL (giống Xử lý 1)

BLL (giống Xử lý 1)

UI

```
// reload Lecture
btnLectureReload.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {
            LoadListLecture();
            addPageLecture();
            JOptionPane.showMessageDialog(null, "Đã tải lại danh sách thành công");
        } catch (Exception e2) {
            e2.printStackTrace();
            JOptionPane.showMessageDialog(null, "Có lỗi khi tải danh sách giáo viên");
        }
    }
});
```

- Xử lý 6:

Xử lý 6 : Tìm kiếm giáo viên

DAL

```
// Tìm kiếm bằng mã
public ArrayList<Person> searchByID(int ID) {
    try {
        Statement stmt = conn.createStatement();
        ArrayList<Person> listStudents = new ArrayList<>();
        String query = "SELECT * FROM person WHERE HireDate IS NOT NULL AND PersonID =?";
        PreparedStatement pstmt = conn.prepareStatement(query);
        pstmt.setInt(1, ID);
        ResultSet rs = pstmt.executeQuery();
        if (rs != null) {
            while (rs.next()) {
                int id = rs.getInt("PersonID");
                String lastName = rs.getString("Lastname");
                String Firstname = rs.getString("Firstname");
                Date HireDate = rs.getDate("HireDate");
                Date EnrollmentDate = rs.getDate("EnrollmentDate");
                Person person = new Person(id, lastName, Firstname, HireDate, EnrollmentDate);
                listStudents.add(person);
            }
        }
        return listStudents;
    } catch (Exception e) {
    }
    return null;
}
```

```
// Tìm kiếm bằng tên
public ArrayList<Person> searchByName(String name) {
    try {
        Statement stmt = conn.createStatement();
        ArrayList<Person> listStudents = new ArrayList<>();
        String query = "SELECT * FROM person WHERE HireDate IS NOT NULL AND concat(FirstName, ' ', LastName) like ?";
        PreparedStatement pstmt = conn.prepareStatement(query);
        pstmt.setString(1, "%" + name + "%");
        ResultSet rs = pstmt.executeQuery();
        if (rs != null) {
            while (rs.next()) {
                int id = rs.getInt("PersonID");
                String lastName = rs.getString("Lastname");
                String FirstName = rs.getString("Firstname");
                Date HireDate = rs.getDate("HireDate");
                Date EnrollmentDate = rs.getDate("EnrollmentDate");
                Person person = new Person(id, lastName, FirstName, HireDate, EnrollmentDate);
                listStudents.add(person);
            }
        }
        return listStudents;
    } catch (Exception e) {
    }
    return null;
}
}
```

BLL

```
public ArrayList<Person> searchByID(int ID) {
    return LectureDAL.gI().searchByID(ID);
}

public ArrayList<Person> searchByName(String name) {
    return LectureDAL.gI().searchByName(name);
}
}
```

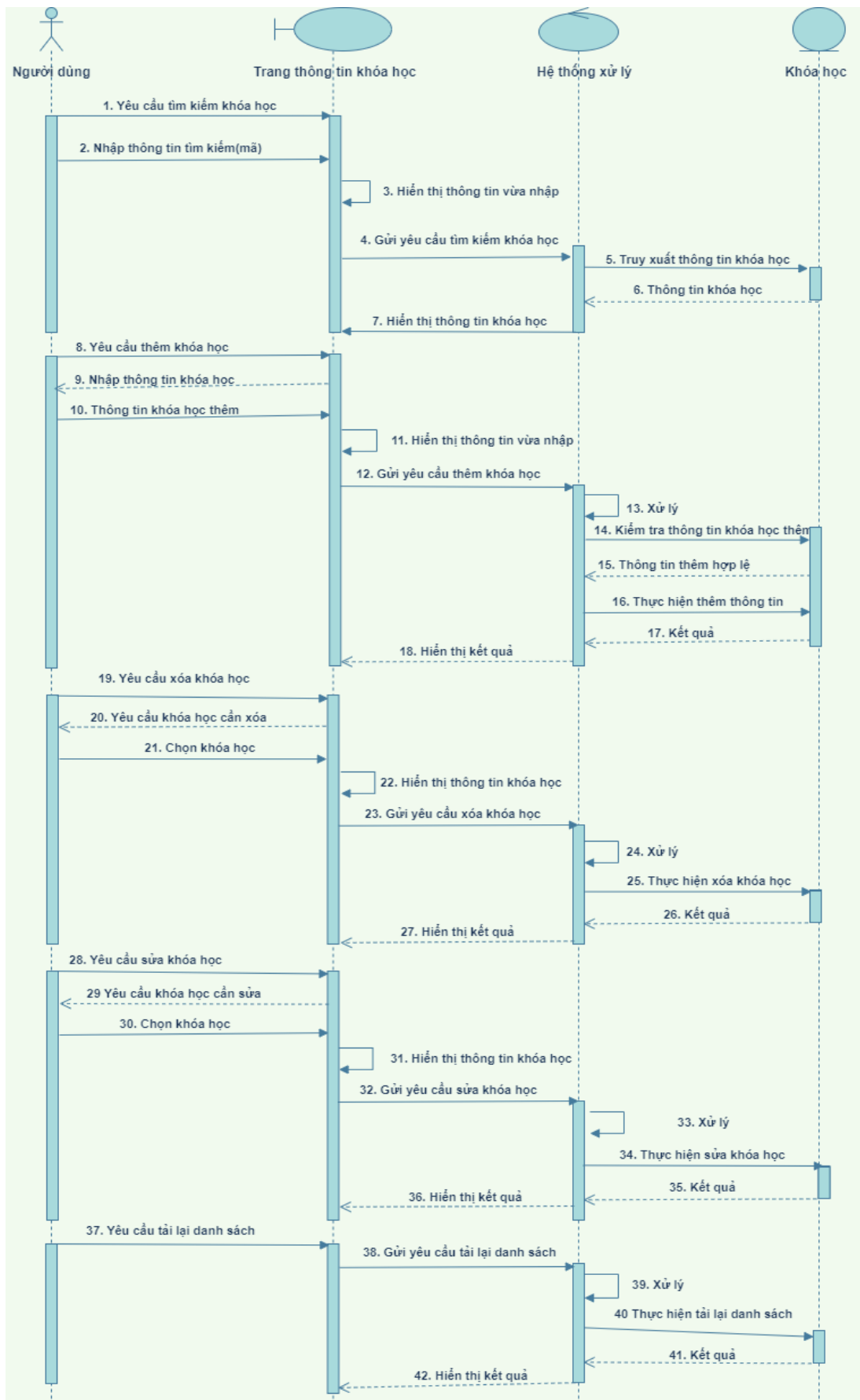

UI

```
protected void SearchByName() {
    String Name = txtKeyWord.getText().trim();
    if (!isNumber(Name)) {

        ArrayList<Person> list = new BLL.LectureBLL().searchByName(Name);
        if (list != null && list.size() > 0) {
            MainFrame.dtmLecture.setRowCount(0);
            repaintPage();
            for (Person person : list) {
                Vector<Object> vec = new Vector<Object>();
                vec.add(person.getID());
                vec.add(person.getFirstname());
                vec.add(person.getLastname());
                vec.add(person.getHireDate());
                MainFrame.dtmLecture.addRow(vec);
                this.setVisible(false);
            }
        } else {
            JOptionPane.showMessageDialog(null, "Không tìm thấy");
        }
    }
}

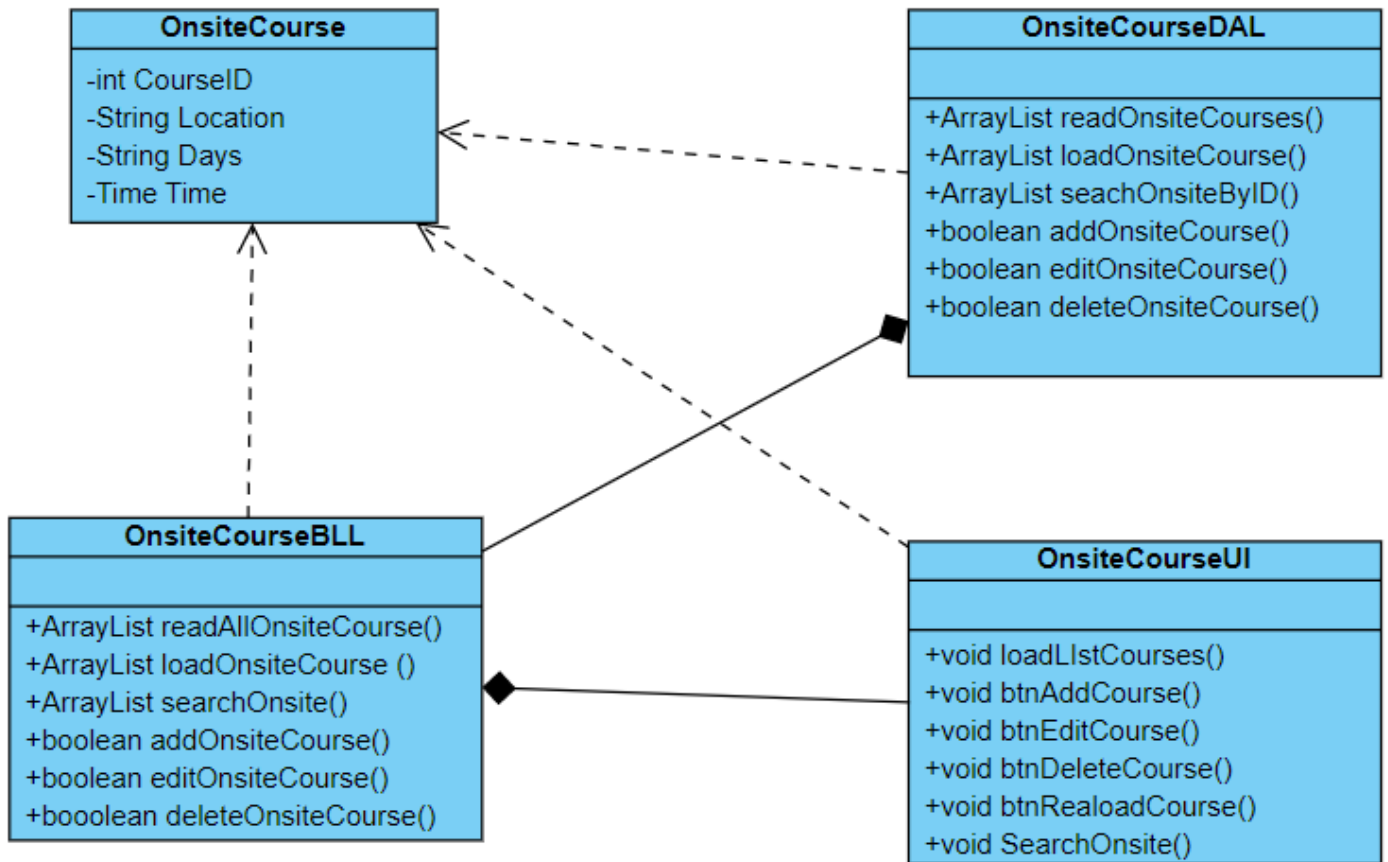
private void SearchByID() {
    String ID = txtKeyWord.getText().trim();
    if (isNumber(ID)) {
        System.out.println("Ok");
        ArrayList<Person> list = new BLL.LectureBLL().searchByID(Integer.parseInt(ID));
        System.out.println(list);
        if (list != null && list.size() > 0) {
            MainFrame.dtmLecture.setRowCount(0);
            repaintPage();
            for (Person person : list) {
                Vector<Object> vec = new Vector<Object>();
                vec.add(person.getID());
                vec.add(person.getFirstname());
                vec.add(person.getLastname());
                vec.add(person.getHireDate());
                MainFrame.dtmLecture.addRow(vec);
                this.setVisible(false);
            }
        }
        else {
            JOptionPane.showMessageDialog(null, "Không tìm thấy");
        }
    } else {
        JOptionPane.showMessageDialog(null, "ID phải là số");
    }
}
```

2. Quản lý khóa học



Sơ đồ tuần tự Quản lý khóa học

2.1 Onsite



Sơ đồ lớp Quản lý khóa học Onsite

Code:

- Xử lý 1: Hiện thị danh sách khóa học onsite

DAL

```

public ArrayList<Course> readOnsiteCourse() {
    try {

        Statement stmt = conn.createStatement();
        ArrayList<Course> listCourses = new ArrayList<>();
        String query = "SELECT course.CourseID, course.Title, course.Credits, course.DepartmentID,"
            + " onsitecourse.Location, onsitecourse.Days, onsitecourse.Time "
            + "FROM `course` , onsitecourse "
            + "WHERE course.CourseID = onsitecourse.CourseID";
        ResultSet rs = stmt.executeQuery(query);
        if (rs != null) {
            int i = 1;
            while (rs.next()) {
                int idCourse = rs.getInt("CourseID");
                String title = rs.getString("Title");
                int credit = rs.getInt("Credits");
                int department = rs.getInt("DepartmentID");
                String location = rs.getString("Location");
                String Days = rs.getString("Days");
                Time time = rs.getTime("Time");
                OnsiteCourse onsiteCourse = new OnsiteCourse(idCourse, location, Days, time);
                Course course = new Course(idCourse, title, credit, department, onsiteCourse);
                listCourses.add(course);
            }
        }

        return listCourses;
    } catch (Exception e) {
    }
    return null;
}

```

BLL

```

public ArrayList<DTO.Course> readOnsiteCourse() {
    return new CourseDAL().readOnsiteCourse();
}

```

UI

```

public void LoadListOnsiteCourse() {
    coursesOnsiteList = null;
    coursesOnsiteList = new BLL.CourseBLL().readOnsiteCourse();
    ArrayList<Course> arrayList = new BLL.CourseBLL().readOnsiteCoursePage(1);
    dtmcourseSite.setRowCount(0);
    int i = 0;
    for (Course course : arrayList) {
        i++;
        Vector<Object> vec = new Vector<Object>();
        vec.add(String.valueOf(i));
        vec.add(course.getCourseID());
        vec.add(course.getTitle());
        for (Department department : departmentsList) {
            if (department.getDepartmentID() == course.getDepartmentID()) {
                vec.add(department.getName());
                break;
            }
        }
        vec.add(course.getOnsiteCourse().getLocation());
        vec.add(course.getOnsiteCourse().getDays());
        vec.add(course.getOnsiteCourse().getTime());
        dtmcourseSite.addRow(vec);
    }
}

```

- Xử lý 2: Thêm khóa học onsite

DAL

```

public boolean addOnsiteCourse(OnsiteCourse o) {
    try {
        String sql = "INSERT INTO `onsitecourse` ( `CourseID`, `Location`, `Days`, `Time`) VALUES (?, ?, ?, ?)";

        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, o.getCourseID());
        pstmt.setString(2, o.getLocation());
        pstmt.setString(3, o.getDays());
        pstmt.setTime(4, o.getTime());
        int i = pstmt.executeUpdate();

        return i > 0 ;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return false;
}

```

BLL

```

public boolean addOnSiteCourse(OnsiteCourse course) {

    return new OnsiteCourseDAL().addOnsiteCourse(course);

}

```

UI

```
btnAddCourse.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String url = txtUrl.getText();
        String TenKH;
        int Khoa;

        TenKH = txtCourseName.getText();
        String department = cmbDepartment.getSelectedItem().toString();
        Khoa = Integer.parseInt(department.substring(0, department.indexOf("-")).trim());
        Course course = new Course(0, TenKH, 1, Khoa);
        int id = BLL.CourseBLL.addCourse(course);
        if (id != -1) {
            JOptionPane.showMessageDialog(null, "Đã thêm thành công");
            if (url.isBlank() || url.isEmpty()) {
                String location = txtLocation.getText();
                String days = txtdateCourse.getText();
                String timestr = cmbTimeCourse.getSelectedItem().toString();
                Time time = null;
                try {
                    time = java.sql.Time.valueOf(timestr);
                    OnsiteCourse onsiteCourse = new OnsiteCourse(id, location, days, time);
                    if (onsiteCourse.getDays().isEmpty() || onsiteCourse.getDays().isBlank()) {
                        JOptionPane.showMessageDialog(null, "Không được để trống ngày");
                        return;
                    }
                    if (onsiteCourse.getLocation().isEmpty()) {
                        JOptionPane.showMessageDialog(null, "Không được để trống phòng học");
                        return;
                    }
                    if (onsiteCourse.getTime() == null) {
                        JOptionPane.showMessageDialog(null, "Thời gian sai");
                        return;
                    }
                    if (new BLL.OnsiteCourseBLL().addOnSiteCourse(onsiteCourse)) {
                        JOptionPane.showMessageDialog(null, "Đã thêm thành công khoá học Onsite");
                        LoadListOnsiteCourse();
                        addPageOnsite();
                    }
                } catch (IllegalArgumentException e2) {
                    e2.printStackTrace();
                }
            }
        } else {
```

```

        OnlineCourse onlineCourse = new OnlineCourse(id, url);
        if (onlineCourse.getUrl().isEmpty()) {
            JOptionPane.showMessageDialog(null, "Không được để trống URL");
            return;
        }
        if (onlineCourse.getCourseID() < 0) {
            JOptionPane.showMessageDialog(null, "Lỗi không có ID");
            return;
        }
        if (new BLL.OnlineCourseBLL().addOnlineCourse(onlineCourse)) {
            JOptionPane.showMessageDialog(null, "Đã thêm thành công khoá học Online");
            LoadListOnlineCourse();
            addPageOnline();
        }
    }
} else if (id == -1) {
    JOptionPane.showMessageDialog(null, "Tên khoá học không được để trống");
} else if (id == -2) {
    JOptionPane.showMessageDialog(null, "Tên khoa không được để trống");
}
}
});

```

- Xử lý 3: Sửa khóa học onsite

DAL

```

public boolean editOnsiteCourse(OnsiteCourse o) {
    try {
        String sql = "UPDATE `onsitecourse` SET `Location`= ? ,`Days`= ? ,`Time`= ? WHERE CourseID = ?";

        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, o.getLocation());
        pstmt.setString(2, o.getDays());
        pstmt.setTime(3, o.getTime());
        pstmt.setInt(4, o.getCourseID());
        int i = pstmt.executeUpdate();

        return i > 0 ;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return false;
}

```

BLL

```
public boolean editOnSiteCourse(OnsiteCourse course) {  
    return new OnsiteCourseDAL().editOnsiteCourse(course);  
}
```

UI

```
btnEditCourse.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        int i = tblCourseOnsite.getSelectedRow();  
        int j = tblCourseOnline.getSelectedRow();  
        if (i >= 0) {  
            j = -1;  
            String url = txtUrl.getText();  
            String TenKH;  
            int Khoa;  
            TenKH = txtCourseName.getText();  
            String department = cmbDepartment.getSelectedItem().toString();  
            Khoa = Integer.parseInt(department.substring(0, department.indexOf("-")).trim());  
            int id = Integer.parseInt(dtmcourseSite.getValueAt(i, 1).toString());  
            Course course = new Course(id, TenKH, 1, Khoa);  
            if (BLL.CourseBLL.editCourse(course)) {  
                // JOptionPane.showMessageDialog(null, "Đã sửa thành công");  
                if (url.isBlank() || url.isEmpty()) {  
                    String location = txtLocation.getText();  
                    String days = txtdateCourse.getText();  
                    String timestr = cmbTimeCourse.getSelectedItem().toString();  
                    Time time = java.sql.Time.valueOf(timestr);  
                    OnsiteCourse onsiteCourse = new OnsiteCourse(id, location, days, time);  
                    if (onsiteCourse.getDays().isEmpty() || onsiteCourse.getDays().isBlank()) {  
                        JOptionPane.showMessageDialog(null, "Không được để trống ngày");  
                        return;  
                    }  
                    if (onsiteCourse.getLocation().isEmpty()) {  
                        JOptionPane.showMessageDialog(null, "Không được để trống phòng học");  
                        return;  
                    }  
                    if (onsiteCourse.getTime() == null) {  
                        JOptionPane.showMessageDialog(null, "Thời gian sai");  
                        return;  
                    }  
                    if (new BLL.OnsiteCourseBLL().editOnSiteCourse(onsiteCourse)) {  
                        JOptionPane.showMessageDialog(null, "Đã sửa thành công khoá học Onsite");  
                        LoadListOnsiteCourse();  
                    }  
                }  
            } else {  
                JOptionPane.showMessageDialog(null, "Sửa khoá học onsite không thành công");  
            }  
        }  
    }  
});
```


- Xử lý 4: Xóa khóa học onsite

DAL

```
public boolean deleteOnsiteCourse(int id) {  
    try {  
        String sql = "DELETE FROM `onsitecourse` WHERE CourseID = ? ";  
        PreparedStatement pstmt = conn.prepareStatement(sql);  
        pstmt.setInt(1, id);  
        int i = pstmt.executeUpdate();  
  
        return i > 0 ;  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
    return false;  
}
```

BLL

```
public boolean deleteSiteCourse(int id) {  
    return new OnsiteCourseDAL().deleteOnsiteCourse(id);  
}
```

UI

```

btnDeleteCourse.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int ID;
        //.... xử lý xoá khoá học online ///
        int i = tblCourseOnline.getSelectedRow();
        if (i >= 0) {
            ID = Integer.parseInt(dtmcourseOnline.getValueAt(i, 1).toString());
            if (new BLL.OnlineCourseBLL().deleteOnlineCourse(ID)) {
                if (new BLL.CourseBLL().deleteCourse(ID)) {
                    JOptionPane.showMessageDialog(null, "Đã xoá khoá học online thành công");
                    LoadListOnlineCourse();
                    addPageOnline();
                } else {
                    JOptionPane.showMessageDialog(null, "Đã xoá khoá học online thất bại");
                }
            } else {
                JOptionPane.showMessageDialog(null, "Đã xoá khoá học thất bại");
            }
        }
        int j = tblCourseOnsite.getSelectedRow();
        // xử lý xoá khoá học onsite
        if (j >= 0) {
            ID = Integer.parseInt(dtmcourseSite.getValueAt(j, 1).toString());
            if (new BLL.OnsiteCourseBLL().deleteSiteCourse(ID)) {
                if (new BLL.CourseBLL().deleteCourse(ID)) {
                    JOptionPane.showMessageDialog(null, "Đã xoá khoá học onsite thành công");
                    LoadListOnsiteCourse();
                    addPageOnsite();
                } else {
                    JOptionPane.showMessageDialog(null, "Đã xoá khoá học onsite thất bại");
                }
            } else {
                JOptionPane.showMessageDialog(null, "Đã xoá khoá học thất bại");
            }
        }
    }
});

```

- Xử lý 5: Tải lại danh sách khóa học onsite

DAL (giống Xử lý 1)

BLL (giống Xử lý 1)

UI

```

btnReloadCourse.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        JOptionPane.showMessageDialog(null, "Đã tải lại danh sách khóa học thành công");
        LoadListOnlineCourse();
        LoadListOnsiteCourse();
    }
});

```

- Xử lý 6: Tìm kiếm khóa học onsite theo mã khoá học

DAL

```
public ArrayList<Course> searchOnsiteByID(int ID) {
    try {
        Statement stmt = conn.createStatement();
        ArrayList<Course> listCourses = new ArrayList<>();
        String query = "SELECT course.CourseID, course.Title, course.Credits, course.DepartmentID, onsitecourse.Location,"
            + " onsitecourse.Days, onsitecourse.Time "
            + "FROM `course` , onsitecourse"
            + " WHERE course.CourseID = ?";
        PreparedStatement preparedStatement = conn.prepareStatement(query);
        preparedStatement.setInt(1, ID);
        ResultSet rs = preparedStatement.executeQuery();
        if (rs != null) {
            int i = 1;
            while (rs.next()) {
                int idCourse = rs.getInt("CourseID");
                String title = rs.getString("Title");
                int credit = rs.getInt("Credits");
                int department = rs.getInt("DepartmentID");
                String location = rs.getString("Location");
                String Days = rs.getString("Days");
                Time time = rs.getTime("Time");
                OnsiteCourse onsiteCourse = new OnsiteCourse(idCourse, location, Days, time);
                Course course = new Course(idCourse, title, credit, department, onsiteCourse);
                listCourses.add(course);
            }
        }
        return listCourses;
    } catch (Exception e) {
    }
    return null;
}
```

BLL

```
public static ArrayList<DTO.Course> searchOnsite(int ID) {
    return new CourseDAL().searchOnsiteByID(ID);
}
```

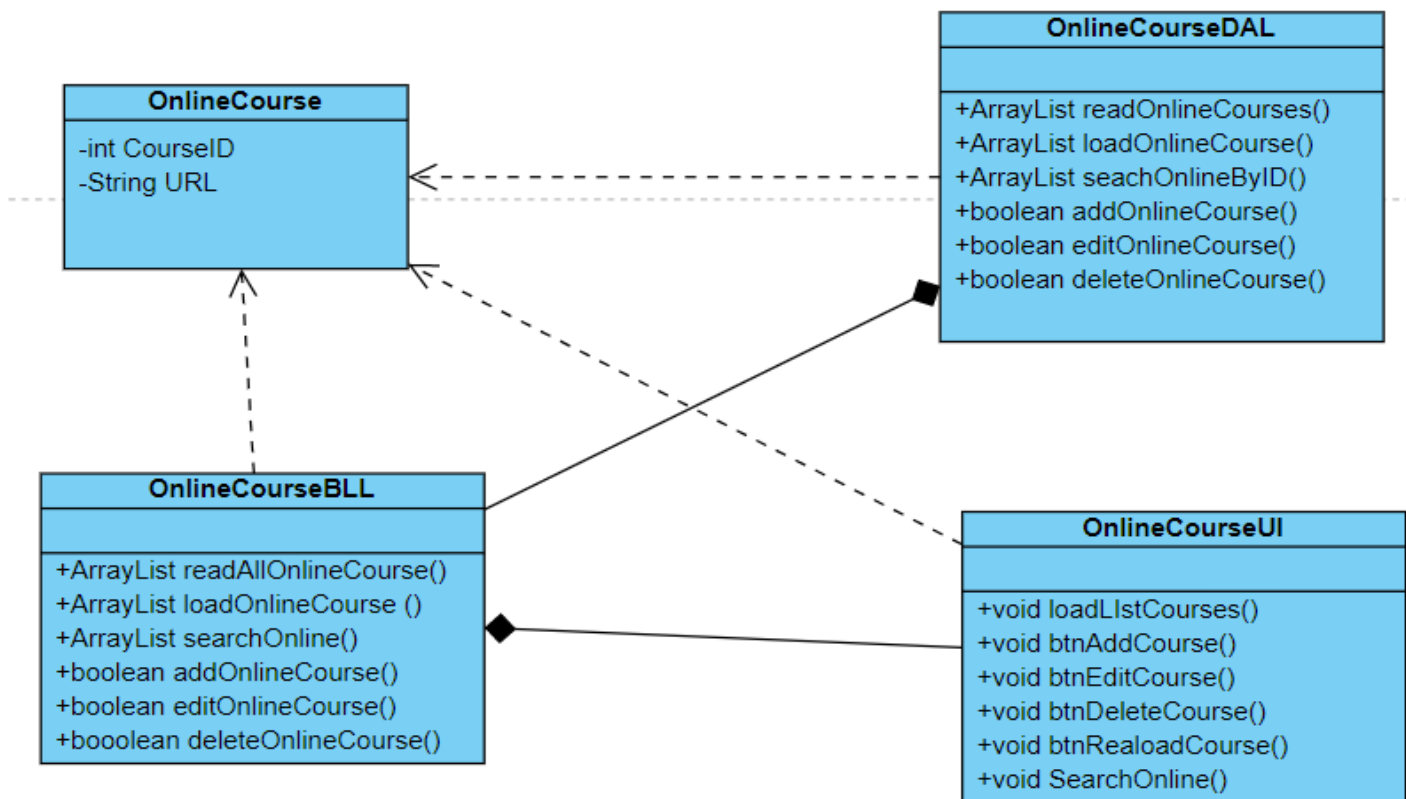
UI

```

protected void SearchOnsite(int id) {
    ArrayList<Course> arrayList = new BLL.CourseBLL().searchOnsite(id);
    if( arrayList != null && arrayList.size() > 0) {
        MainFrame.dtmcourseSite.setRowCount(0);
        int i = 0;
        for (Course course : arrayList) {
            i++;
            Vector<Object> vec = new Vector<Object>();
            vec.add(String.valueOf(i));
            vec.add(course.getCourseID());
            vec.add(course.getTitle());
            for (Department department : MainFrame.departmentsList) {
                if (department.getDepartmentID() == course.getDepartmentID()) {
                    vec.add(department.getName());
                    break;
                }
            }
            vec.add(course.getOnsiteCourse().getLocation());
            vec.add(course.getOnsiteCourse().getDays());
            vec.add(course.getOnsiteCourse().getTime());
            MainFrame.dtmcourseSite.addRow(vec);
        }
    }
    else {
        JOptionPane.showMessageDialog(null, "Không tìm thấy");
    }
}

```

2.2 Online



Sơ đồ lớp Quản lý khóa học Online

Code:

- Xử lý 1: Hiện thị danh sách khóa học online

DAL

```
public ArrayList<Course> readOnlineCourse() {
    try {
        Statement stmt = conn.createStatement();
        ArrayList<Course> listCourses = new ArrayList<>();
        String query = "SELECT course.CourseID, course.Title, course.Credits, course.DepartmentID, onlinecourse.url FROM `course` , "
            + "onlinecourse WHERE course.CourseID = onlinecourse.CourseID;";
        ResultSet rs = stmt.executeQuery(query);
        if (rs != null) {
            int i = 1;
            while (rs.next()) {
                int idCourse = rs.getInt("CourseID");
                String title = rs.getString("Title");
                int credit = rs.getInt("Credits");
                int department = rs.getInt("DepartmentID");
                String url = rs.getString("url");
                OnlineCourse onlineCourse = new OnlineCourse(idCourse, url);
                Course course = new Course(idCourse, title, credit, department, onlineCourse);
                listCourses.add(course);
            }
        }
        return listCourses;
    } catch (Exception e) {
    }
    return null;
}
```

BLL

```
public static ArrayList<DTO.Course> readOnlineCourse() {
    return new CourseDAL().readOnlineCourse();
}
```

UI

```
public void LoadListOnlineCourse() {
    coursesOnlineList = null;
    coursesOnlineList = BLL.CourseBLL.readOnlineCourse();
    dtmcourseOnline.setRowCount(0);
    ArrayList<Course> arrayList = BLL.CourseBLL.readOnlineCoursePage(1);
    int i = 0;
    for (Course course : arrayList) {
        i++;
        Vector<Object> vec = new Vector<Object>();
        vec.add(String.valueOf(i));
        vec.add(course.getCourseID());
        vec.add(course.getTitle());
        for (Department department : departmentsList) {
            if (department.getDepartmentID() == course.getDepartmentID()) {
                vec.add(department.getName());
                break;
            }
        }
        vec.add(course.getOnlineCourse().getUrl());
        dtmcourseOnline.addRow(vec);
    }
}
```

- Xử lý 2: Thêm khóa học online

DAL

```

public boolean addOnlineCourse(OnlineCourse o) {
    try {
        String sql = "INSERT INTO `onlinecourse`(`CourseID`, `url`) VALUES (?,?)";

        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, o.getCourseID());
        pstmt.setString(2, o.getUrl());
        int i = pstmt.executeUpdate();

        return i > 0;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return false;
}

```

BLL

```

public boolean addOnlineCourse(OnlineCourse course) {

    return new OnlineCourseDAL().addOnlineCourse(course);
}

```

UI

```

btnAddCourse.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String url = txtUrl.getText();
        String TenKH;
        int Khoa;

        TenKH = txtCourseName.getText();
        String department = cmbDepartment.getSelectedItem().toString();
        Khoa = Integer.parseInt(department.substring(0, department.indexOf("-")).trim());
        Course course = new Course(0, TenKH, 1, Khoa);
        int id = BLL.CourseBLL.addCourse(course);
        if (id != -1) {
            JOptionPane.showMessageDialog(null, "Đã thêm thành công");
            if (url.isBlank() || url.isEmpty()) {
                String location = txtLocation.getText();
                String days = txtdateCourse.getText();
                String timestr = cmbTimeCourse.getSelectedItem().toString();
                Time time = null;
                try {
                    time = java.sql.Time.valueOf(timestr);
                    OnsiteCourse onsiteCourse = new OnsiteCourse(id, location, days, time);
                    if (onsiteCourse.getDays().isEmpty() || onsiteCourse.getDays().isBlank()) {
                        JOptionPane.showMessageDialog(null, "Không được để trống ngày");
                        return;
                    }
                    if (onsiteCourse.getLocation().isEmpty()) {
                        JOptionPane.showMessageDialog(null, "Không được để trống phòng học");
                        return;
                    }
                    if (onsiteCourse.getTime() == null) {
                        JOptionPane.showMessageDialog(null, "Thời gian sai");
                        return;
                    }
                    if (new BLL.OnsiteCourseBLL().addOnSiteCourse(onsiteCourse)) {
                        JOptionPane.showMessageDialog(null, "Đã thêm thành công khoá học Onsite");
                        LoadListOnsiteCourse();
                        addPageOnsite();
                    }
                } catch (IllegalArgumentException e2) {
                    e2.printStackTrace();
                }
            }
        } else {

```



```

, ---- {
    OnlineCourse onlineCourse = new OnlineCourse(id, url);
    if (onlineCourse.getUrl().isEmpty()) {
        JOptionPane.showMessageDialog(null, "Không được để trống URL");
        return;
    }
    if (onlineCourse.getCourseID() < 0) {
        JOptionPane.showMessageDialog(null, "Lỗi không có ID");
        return;
    }
    if (new BLL.OnlineCourseBLL().addOnlineCourse(onlineCourse)) {
        JOptionPane.showMessageDialog(null, "Đã thêm thành công khoá học Online");
        LoadListOnlineCourse();
        addPageOnline();
    }
}
} else if (id == -1) {
    JOptionPane.showMessageDialog(null, "Tên khoá học không được để trống");
} else if (id == -2) {
    JOptionPane.showMessageDialog(null, "Tên khoa không được để trống");
}
}

```

- Xử lý 3: Sửa khóa học online

DAL

```

public boolean editOnlineCourse(OnlineCourse o) {
    try {
        String sql = "UPDATE `onlinecourse` SET `url`= ? WHERE CourseID = ?";

        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, o.getUrl());
        pstmt.setInt(2, o.getCourseID());
        int i = pstmt.executeUpdate();

        return i > 0;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return false;
}

```

BLL

```

public boolean editOnlineCourse(OnlineCourse course) {

    return new OnlineCourseDAL().editOnlineCourse(course);
}

```

UI

```

btnEditCourse.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int i = tblCourseOnsite.getSelectedRow();
        int j = tblCourseOnline.getSelectedRow();
        if (j >= 0) {
            i = -1;
            String url = txtUrl.getText();
            String TenKH;
            int Khoa;
            TenKH = txtCourseName.getText();
            String department = cmbDepartment.getSelectedItem().toString();
            Khoa = Integer.parseInt(department.substring(0, department.indexOf("-")).trim());
            int id = Integer.parseInt(dtmcourseOnline.getValueAt(j, 1).toString());
            Course course = new Course(id, TenKH, 1, Khoa);
            if (BLL.CourseBLL.editCourse(course)) {
                OnlineCourse onlineCourse = new OnlineCourse(id, url);
                if (onlineCourse.getUrl().isEmpty()) {
                    JOptionPane.showMessageDialog(null, "Không được để trống URL");
                    return;
                }
                if (onlineCourse.getCourseID() < 0) {
                    JOptionPane.showMessageDialog(null, "Lỗi không có ID");
                    return;
                }
                if (new BLL.OnlineCourseBLL().editOnlineCourse(onlineCourse)) {
                    JOptionPane.showMessageDialog(null, "Đã sửa thành công khoá học Online");
                    LoadListOnlineCourse();
                } else {
                    JOptionPane.showMessageDialog(null, "Đã có lỗi xảy ra");
                }
                // }
            } else {
                JOptionPane.showMessageDialog(null, "Có lỗi xảy ra");
            }
        }
    }
});

```

- Xử lý 4: Xóa khóa học online

DAL

```

public boolean deleteOnlineCourse(int id) {
    try {
        String sql = "DELETE FROM `onlinecourse` WHERE CourseID = ? ";

        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, id);
        int i = pstmt.executeUpdate();

        return i > 0;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return false;
}

```

BLL

```

public boolean deleteOnlineCourse(int id) {
    return new OnlineCourseDAL().deleteOnlineCourse(id);
}

```

UI

```

btnDeleteCourse.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int ID;
        //.... xử lý xoá khoá học online ///
        int i = tblCourseOnline.getSelectedRow();
        if (i >= 0) {
            ID = Integer.parseInt(dtmcourseOnline.getValueAt(i, 1).toString());
            if (new BLL.OnlineCourseBLL().deleteOnlineCourse(ID)) {
                if (new BLL.CourseBLL().deleteCourse(ID)) {
                    JOptionPane.showMessageDialog(null, "Đã xoá khoá học online thành công");
                    LoadListOnlineCourse();
                    addPageOnline();
                } else {
                    JOptionPane.showMessageDialog(null, "Đã xoá khoá học online thất bại");
                }
            } else {
                JOptionPane.showMessageDialog(null, "Đã xoá khoá học thất bại");
            }
        }
    }
});

```

- Xử lý 5: Tải lại danh sách khóa học online

DAL (giống Xử lý 1)

BLL (giống Xử lý 1)

UI

```

btnReloadCourse.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        JOptionPane.showMessageDialog(null, "Đã tải lại danh sách khoá học thành công");
        LoadListOnlineCourse();
        LoadListOnsiteCourse();
    }
});

```

- Xử lý 6: Tìm kiếm khóa học online theo mã khóa học

DAL

```

public ArrayList<Course> searchOnlineByID(int ID) {
    try {

        ArrayList<Course> listCourses = new ArrayList<>();
        String query = "SELECT course.CourseID, course.Title, course.Credits, course.DepartmentID,"
            + " onlinecourse.url FROM `course` , onlinecourse WHERE course.CourseID = ?";
        PreparedStatement preparedStatement = conn.prepareStatement(query);
        preparedStatement.setInt(1, ID);
        ResultSet rs = preparedStatement.executeQuery();
        if (rs != null) {
            int i = 1;
            while (rs.next()) {
                int idCourse = rs.getInt("CourseID");
                String title = rs.getString("Title");
                int credit = rs.getInt("Credits");
                int department = rs.getInt("DepartmentID");
                String url = rs.getString("url");
                OnlineCourse onlineCourse = new OnlineCourse(idCourse, url);
                Course course = new Course(idCourse, title, credit, department, onlineCourse);
                listCourses.add(course);
            }
        }

        return listCourses;
    } catch (Exception e) {
    }
    return null;
}

```

BLL

```

public static ArrayList<DTO.Course> searchOnline(int ID) {
    return new CourseDAL().searchOnlineByID(ID);
}

```

UI

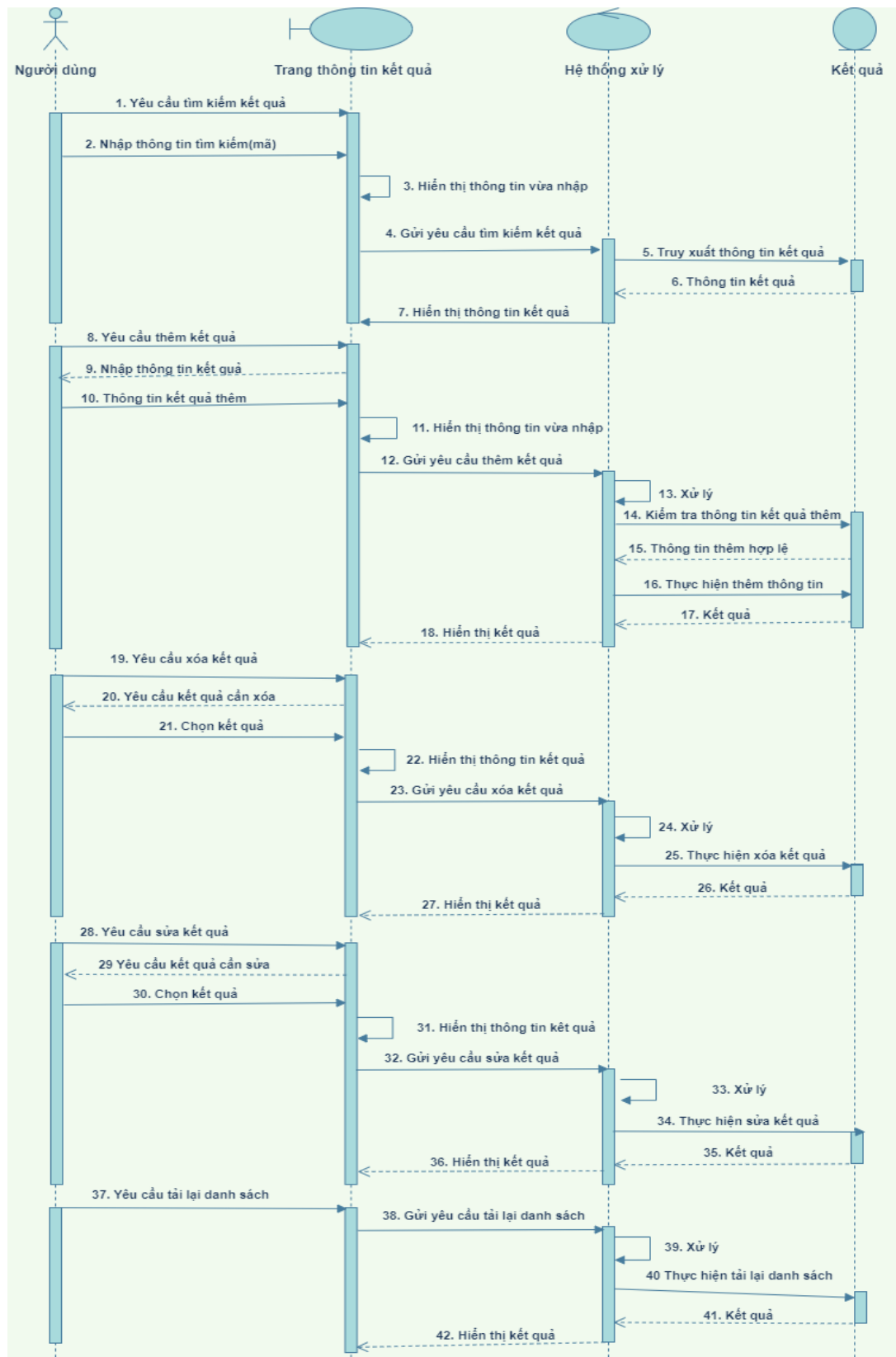
```

protected void SearchOnline(int iD) {
    ArrayList<Course> arrayList = BLL.CourseBLL.searchOnline(iD);
    if(arrayList != null && arrayList.size() > 0) {
        int i = 0;
        MainFrame.dtmcourseOnline.setRowCount(0);
        for (Course course : arrayList) {
            i++;
            Vector<Object> vec = new Vector<Object>();
            vec.add(String.valueOf(i));
            vec.add(course.getCourseID());
            vec.add(course.getTitle());
            for (Department department : MainFrame.departmentsList) {
                if (department.getDepartmentID() == course.getDepartmentID()) {
                    vec.add(department.getName());
                    break;
                }
            }
            vec.add(course.getOnlineCourse().getUrl());
            MainFrame.dtmcourseOnline.addRow(vec);
        }
    }
    else {
        JOptionPane.showMessageDialog(null, "Không tìm thấy");
    }
}

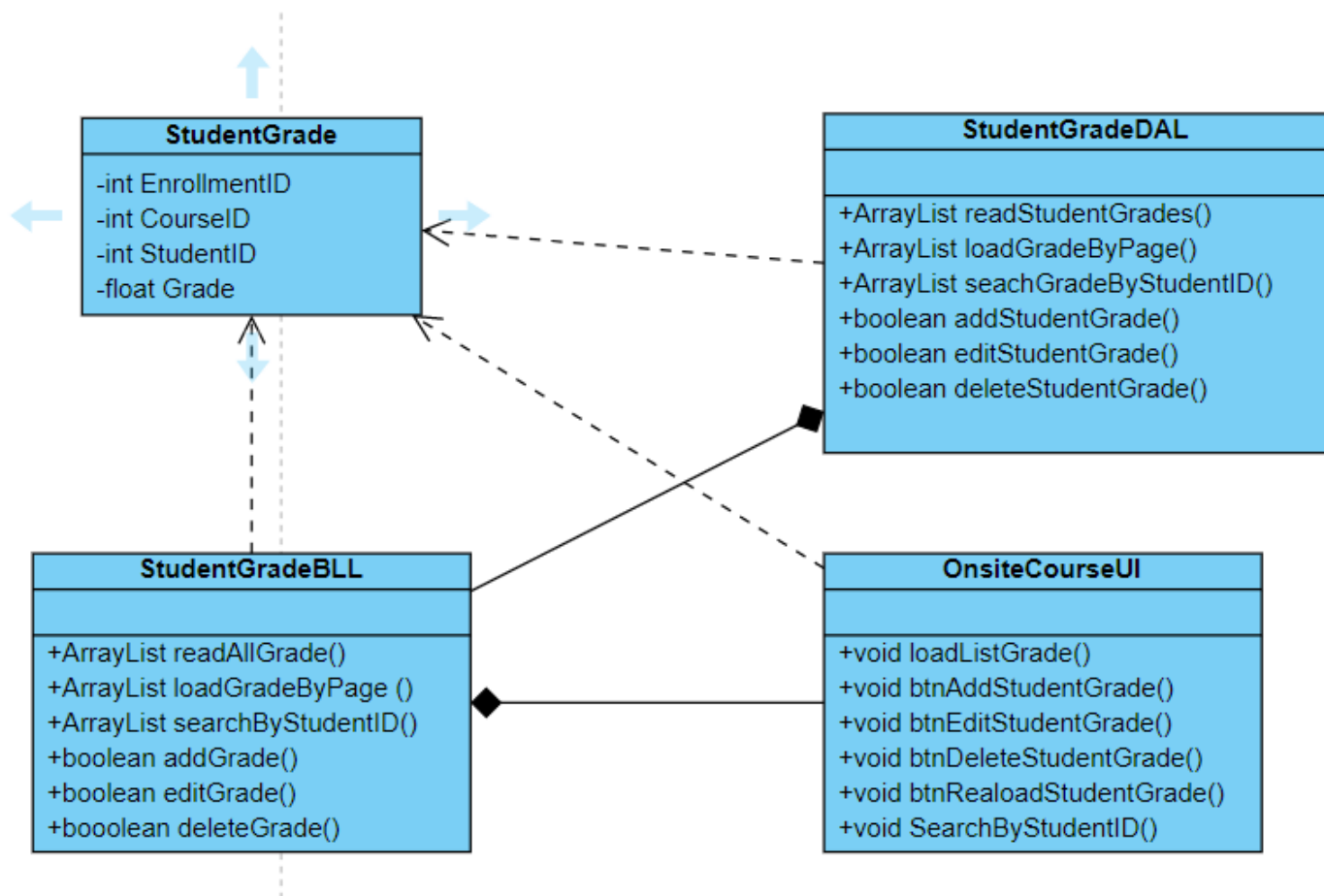
```

3. Quản lý phân công

4. Quản lý kết quả



Sơ đồ tuần tự Quản lý kết quả



Sơ đồ lớp Quản lý kết quả

Code:

- Xử lý 1: Hiển thị danh sách kết quả

DAL

// Đọc tất cả thông tin kết quả

```

public ArrayList<StudentGrade> readStudentGrades() {
    try {

        Statement stmt = conn.createStatement();
        ArrayList<StudentGrade> listStudentGrades = new ArrayList<>();
        String query = "SELECT * FROM `studentgrade` ";
        ResultSet rs = stmt.executeQuery(query);
        if (rs != null) {
            int i = 1;
            while (rs.next()) {
                int idEn = rs.getInt("EnrollmentID");
                int idCo = rs.getInt("CourseID");
                int idSt = rs.getInt("StudentID");
                float grade = rs.getFloat("Grade");
                StudentGrade stugrade = new StudentGrade(idEn, idCo, idSt, grade);
                listStudentGrades.add(stugrade);
            }
        }

        return listStudentGrades;
    } catch (Exception e) {
    }
    return null;
}

```

BLL

```

public static ArrayList<DT0.StudentGrade> readAllGrade() {
    return new StudentGradeDAL().GI().readStudentGrades();
}
// Đọc tất cả thông tin kết quả

```

UI


```

public void LoadListGrade() {
    studentGradeList = null;
    studentGradeList = BLL.StudentGradeBLL.gI().readAllGrade();
    ArrayList<DT0.StudentGrade> arrayList = BLL.StudentGradeBLL.gI().LoadGradeByPage(1);
    dtmStudentGrade.setRowCount(0);
    LoadListStudents();
    LoadListCourseToComboBox();
    for (StudentGrade studentGrade : arrayList) {
        Vector<Object> vec = new Vector<Object>();
        vec.add(studentGrade.getEnrollmentID());
        for (DT0.Course course : coursesList) {
            if (course.getCourseID() == studentGrade.getCourseID()) {
                vec.add(course.getTitle());
                break;
            }
        }
        for (DT0.Person per : studentsList) {
            if (per.getID() == studentGrade.getStudentID()) {
                vec.add(per.getFirstname() + " " + per.getLastname());
                break;
            }
        }
        vec.add(studentGrade.getGrade());
        dtmStudentGrade.addRow(vec);
    }
}

```

- Xử lý 2: Thêm kết quả

DAL

```

public boolean addStudentGrade(StudentGrade s) {
    try {
        String sql = "INSERT INTO `studentgrade` ( `CourseID`, `StudentID`, `Grade`) VALUES (?, ?, ?)";

        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, s.getCourseID());
        pstmt.setInt(2, s.getStudentID());
        pstmt.setFloat(3, s.getGrade());
        int i = pstmt.executeUpdate();

        return i > 0;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return false;
}

```

BLL

```

public static boolean addGrade(StudentGrade grade) {
    return new StudentGradeDAL().gI().addStudentGrade(grade);
}

```

UI

```

// add Student Grade
btnAddStudentGrade.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {

        if (courseGradeSelected != null && studentGradeSelected != null) {
            Person student = studentGradeSelected;
            Course course = courseGradeSelected;
            String gradestr = txtGrade.getText();
            float grade = 0;
            try {
                grade = Float.parseFloat(gradestr);
            } catch (NumberFormatException e2) {
                JOptionPane.showMessageDialog(null, "Điểm phải là số");
                return;
            }
            StudentGrade studentGrade = new StudentGrade(0, course.getCourseID(), student.getID(), grade);
            if (studentGrade.getGrade() > 10 || studentGrade.getGrade() < 0) {
                JOptionPane.showMessageDialog(null, "Điểm không hợp lệ");
                return;
            }
            if (studentGrade.getCourseID() < 0) {
                JOptionPane.showMessageDialog(null, "Tên khóa học không được để trống");
                return;
            }
            if (studentGrade.getStudentID() < 0) {
                JOptionPane.showMessageDialog(null, "Tên học viên không được để trống");
                return;
            }
            if (StudentGradeBLL.gI().addGrade(studentGrade)) {
                addPageCourseGrade();
                LoadListGrade();
                LoadListStudents();
                JOptionPane.showMessageDialog(null, "Đã thêm thành công");
            } else {
                JOptionPane.showMessageDialog(null, "Đã có lỗi xảy ra khi thêm điểm khóa học");
            }
        }
    }
});

```

- Xử lý 3: Sửa kết quả

DAL

```
public boolean editStudentGrade(StudentGrade s) {
    try {
        String sql = "UPDATE `studentgrade` SET `CourseID`= ? ,`StudentID`= ? ,`Grade`= ? WHERE EnrollmentID = ?";

        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, s.getCourseID());
        pstmt.setInt(2, s.getStudentID());
        pstmt.setFloat(3, s.getGrade());
        pstmt.setInt(4, s.getEnrollmentID());
        int i = pstmt.executeUpdate();

        return i > 0;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return false;
}
// Sửa một kết quả
```

BLL

```
// Sửa kết quả
public static boolean editGrade(StudentGrade grade) {
    return new StudentGradeDAL().GI().editStudentGrade(grade);
}
```

UI

```
// edit Student Grade
```

```

btnEditStudentGrade.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int i = tblStudentGrade.getSelectedRow();
        try {
            float grade = Float.parseFloat(txtGrade.getText());
            if (i >= 0) {
                if (courseGradeSelected != null && studentGradeSelected != null) {
                    Person student = studentGradeSelected;
                    Course course = courseGradeSelected;
                    String gradeStr = txtGrade.getText();
                    Float grade1 = Float.parseFloat(gradeStr);
                    String IDstr = dtmStudentGrade.getValueAt(i, 0).toString();
                    int ID = Integer.parseInt(IDstr);
                    StudentGrade studentGrade = new StudentGrade(ID, course.getCourseID(), student.getID(),
                        grade1);
                    if (studentGrade.getGrade() > 10 || studentGrade.getGrade() < 0) {
                        JOptionPane.showMessageDialog(null, "Điểm không hợp lệ");
                        return;
                    }
                    if (studentGrade.getCourseID() < 0) {
                        JOptionPane.showMessageDialog(null, "Tên khóa học không được để trống");
                        return;
                    }
                    if (studentGrade.getStudentID() < 0) {
                        JOptionPane.showMessageDialog(null, "Tên học viên không được để trống");
                        return;
                    }
                    if (StudentGradeBLL.gI().editGrade(studentGrade)) {
                        LoadListGrade();
                        LoadListStudents();
                        JOptionPane.showMessageDialog(null, "Đã sửa thành công");
                    } else {
                        JOptionPane.showMessageDialog(null, "Đã có lỗi xảy ra khi sửa điểm khóa học");
                    }
                }
            } else {
                JOptionPane.showMessageDialog(null, "Bạn chưa chọn điểm cần sửa");
            }
        } catch (NumberFormatException e2) {
            JOptionPane.showMessageDialog(null, "Điểm không hợp lệ");
            e2.printStackTrace();
        }
    }
}

```

- Xử lý 4: Xóa kết quả

DAL

// Xóa một kết quả

```
public boolean deleteStudentGrade(int id) {
    try {
        String sql = "DELETE FROM `studentgrade` WHERE EnrollmentID = ? ";

        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, id);
        int i = pstmt.executeUpdate();

        return i > 0;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return false;
}
```

BLL

```
public static boolean deleteGrade (int id) {
    return StudentGradeDAL.gI().deleteStudentGrade(id);
}
// Xóa kết quả
```

UI

```
// delete Student Grade
btnDeleteStudentGrade.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int i = tblStudentGrade.getSelectedRow();
        try {
            if (i >= 0) {
                int idEnrollment = Integer.parseInt(dtmStudentGrade.getValueAt(i, 0).toString());
                int dialogButton = JOptionPane.YES_NO_OPTION;
                int dialogResult = JOptionPane.showConfirmDialog(null, "Bạn có muốn xóa không?", "Warning",
                    dialogButton);
                if (dialogResult == JOptionPane.YES_OPTION) {
                    if (StudentGradeBLL.gI().deleteGrade(idEnrollment)) {
                        LoadListGrade();
                        JOptionPane.showMessageDialog(null, "Đã xóa thành công");
                    } else {
                        JOptionPane.showMessageDialog(null, "Đã có lỗi xảy ra khi xóa điểm");
                    }
                }
            }
            } else {
                JOptionPane.showMessageDialog(null, "Bạn chưa chọn điểm cần xóa");
            }
        } catch (Exception e2) {
            // TODO: handle exception
            e2.printStackTrace();
        }
    }
});
```

- Xử lý 5: Tải lại danh sách kết quả

DAL (giống Xử lý 1)

BLL (giống Xử lý 1)

UI

```
...
// reload student grade
btnReloadStudentGrade.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {
            LoadListGrade();
            addPageCourseGrade();
            JOptionPane.showMessageDialog(null, "Đã tải lại danh sách thành công");
        } catch (Exception e2) {
            e2.printStackTrace();
            JOptionPane.showMessageDialog(null, "Có lỗi khi tải danh sách điểm");
        }
    }
});
...

```

- Xử lý 6: Tìm kiếm kết quả theo mã sinh viên

DAL

```
public ArrayList<StudentGrade> searchGradeByStudentID(int studentID) {
    try {

        Statement stmt = conn.createStatement();
        ArrayList<StudentGrade> listStudentGrades = new ArrayList<>();
        String query = "SELECT * FROM `studentgrade` where StudentID = ? ";
        PreparedStatement pstmt = conn.prepareStatement(query);
        pstmt.setInt(1, studentID);
        ResultSet rs = pstmt.executeQuery();
        if (rs != null) {
            int i = 1;
            while (rs.next()) {
                int idEn = rs.getInt("EnrollmentID");
                int idCo = rs.getInt("CourseID");
                int idSt = rs.getInt("StudentID");
                float Grade = rs.getFloat("Grade");
                StudentGrade stugrade = new StudentGrade(idEn, idCo, idSt, Grade);
                listStudentGrades.add(stugrade);
            }
        }

        return listStudentGrades;
    } catch (Exception e) {
    }
    return null;
}

```

BLL

```

public static ArrayList<DTO.StudentGrade> searchByStudentID(int ID) {
    return new StudentGradeDAL().gI().searchGradeByStudentID(ID);
}

```

UI

```

protected void SearchByStudentID(int ID) {

    ArrayList<StudentGrade> arrayList = BLL.StudentGradeBLL.searchByStudentID(ID);
    MainFrame.dtmCourseInstructor.setRowCount(0);
    int i = 0;
    if (arrayList != null && arrayList.size() > 0) {
        MainFrame.pnGradeCard.removeAll();
        MainFrame.pnGradeCard.revalidate();
        MainFrame.pnGradeCard.repaint();
        MainFrame.dtmStudentGrade.setRowCount(0);
        for (StudentGrade studentGrade : arrayList) {
            Vector<Object> vec = new Vector<Object>();
            vec.add(studentGrade.getEnrollmentID());
            for (DTO.Course course : MainFrame.coursesList) {
                if (course.getCourseID() == studentGrade.getCourseID()) {
                    vec.add(course.getTitle());
                    break;
                }
            }
            for (DTO.Person per : MainFrame.studentsList) {
                if (per.getID() == studentGrade.getStudentID()) {
                    vec.add(per.getFirstname() + " " + per.getLastname());
                    break;
                }
            }
            vec.add(studentGrade.getGrade());
            MainFrame.dtmStudentGrade.addRow(vec);
        }
    } else {
        JOptionPane.showMessageDialog(null, "Không tìm thấy sinh viên / Mã sinh viên không tồn tại");
    }
}

```

IV. Cài đặt phần mềm

1. Phương án cài đặt

Ngôn ngữ lập trình: Java

Cơ sở dữ liệu: PhpMyAdmin, Xampp

Phần mềm chạy chương trình: Netbean hoặc Eclipse

- Bước 1: import CSDL vào xampp, đặt tên CSDL là school2
- Bước 2: import project vào Netbean hoặc Eclipse
- Bước 3: Thêm cái file thư viện cần thiết vào project
- Bước 4: Khởi động project

2. Source code

Link github: <https://github.com/thekids1002/MoHinhPhanLop.git>

V. Tài liệu tham khảo

- Học phần Xây dựng phần mềm theo mô hình phân lớp của giảng viên Cao Minh Thành:

<https://sites.google.com/view/caominhthanh/x%C3%A2y-d%E1%BB%B1ng-ph%E1%BA%A7n-m%E1%BB%81m-theo-m%C3%B4-h%C3%ACnh-ph%C3%A2n-l%E1%BB%9Bp?authuser=0>